Studies in Computational Intelligence 287

Dan Schonfeld
Caifeng Shan
Dacheng Tao
Liang Wang (Eds.)

Video Search and Mining

Springer

Studies in Computational Intelligence, Volume 287

Editor-in-Chief
Prof. Janusz Kacprzyk
Systems Research Institute
Polish Academy of Sciences
ul. Newelska 6
01-447 Warsaw
Poland
E-mail: kacprzyk@ibspan.waw.pl

Further volumes of this series can be found on our homepage: springer.com

Vol. 266. Akitoshi Hanazawa, Tsutom Miki, and Keiichi Horio (Eds.)
Brain-Inspired Information Technology, 2009
ISBN 978-3-642-04024-5

Vol. 267. Ivan Zelinka, Sergej Celikovský, Hendrik Richter, and Guanrong Chen (Eds.)
Evolutionary Algorithms and Chaotic Systems, 2009
ISBN 978-3-642-10706-1

Vol. 268. Johann M.Ph. Schumann and Yan Liu (Eds.)
Applications of Neural Networks in High Assurance Systems, 2009
ISBN 978-3-642-10689-7

Vol. 269. Francisco Fernández de de Vega and Erick Cantú-Paz (Eds.)
Parallel and Distributed Computational Intelligence, 2009
ISBN 978-3-642-10674-3

Vol. 270. Zong Woo Geem
Recent Advances In Harmony Search Algorithm, 2009
ISBN 978-3-642-04316-1

Vol. 271. Janusz Kacprzyk, Frederick E. Petry, and Adnan Yazici (Eds.)
Uncertainty Approaches for Spatial Data Modeling and Processing, 2009
ISBN 978-3-642-10662-0

Vol. 272. Carlos A. Coello Coello, Clarisse Dhaenens, and Laetitia Jourdan (Eds.)
Advances in Multi-Objective Nature Inspired Computing, 2009
ISBN 978-3-642-11217-1

Vol. 273. Fatos Xhafa, Santi Caballé, Ajith Abraham, Thanasis Daradoumis, and Angel Alejandro Juan Perez (Eds.)
Computational Intelligence for Technology Enhanced Learning, 2010
ISBN 978-3-642-11223-2

Vol. 274. Zbigniew W. Raś and Alicja Wieczorkowska (Eds.)
Advances in Music Information Retrieval, 2010
ISBN 978-3-642-11673-5

Vol. 275. Dilip Kumar Pratihar and Lakhmi C. Jain (Eds.)
Intelligent Autonomous Systems, 2010
ISBN 978-3-642-11675-9

Vol. 276. Jacek Mańdziuk
Knowledge-Free and Learning-Based Methods in Intelligent Game Playing, 2010
ISBN 978-3-642-11677-3

Vol. 277. Filippo Spagnolo and Benedetto Di Paola (Eds.)
European and Chinese Cognitive Styles and their Impact on Teaching Mathematics, 2010
ISBN 978-3-642-11679-7

Vol. 278. Radomir S. Stankovic and Jaakko Astola
From Boolean Logic to Switching Circuits and Automata, 2010
ISBN 978-3-642-11681-0

Vol. 279. Manolis Wallace, Ioannis E. Anagnostopoulos, Phivos Mylonas, and Maria Bielikova (Eds.)
Semantics in Adaptive and Personalized Services, 2010
ISBN 978-3-642-11683-4

Vol. 280. Chang Wen Chen, Zhu Li, and Shiguo Lian (Eds.)
Intelligent Multimedia Communication: Techniques and Applications, 2010
ISBN 978-3-642-11685-8

Vol. 281. Robert Babuska and Frans C.A. Groen (Eds.)
Interactive Collaborative Information Systems, 2010
ISBN 978-3-642-11687-2

Vol. 282. Husrev Taha Sencar, Sergio Velastin, Nikolaos Nikolaidis, and Shiguo Lian (Eds.)
Intelligent Multimedia Analysis for Security Applications, 2010
ISBN 978-3-642-11754-1

Vol. 283. Ngoc Thanh Nguyen, Radoslaw Katarzyniak, and Shyi-Ming Chen (Eds.)
Advances in Intelligent Information and Database Systems, 2010
ISBN 978-3-642-12089-3

Vol. 284. Juan R. González, David Alejandro Pelta, Carlos Cruz, Germán Terrazas, and Natalio Krasnogor (Eds.)
Nature Inspired Cooperative Strategies for Optimization (NICSO 2010), 2010
ISBN 978-3-642-12537-9

Vol. 285. Roberto Cipolla, Sebastiano Battiato, and Giovanni Maria Farinella (Eds.)
Computer Vision, 2010
ISBN 978-3-642-12847-9

Vol. 286. Zeev Volkovich, Alexander Bolshoy, Valery Kirzhner, and Zeev Barzily
Genome Clustering, 2010
ISBN 978-3-642-12951-3

Vol. 287. Dan Schonfeld, Caifeng Shan, Dacheng Tao, and Liang Wang (Eds.)
Video Search and Mining, 2010
ISBN 978-3-642-12899-8

Dan Schonfeld, Caifeng Shan, Dacheng Tao,
and Liang Wang (Eds.)

Video Search and Mining

Springer

Dan Schonfeld
Multimedia Communications Laboratory
Department of Electrical & Computer Engineering
University of Illinois at Chicago
Room 1020 SEO (M/C 154)
851 South Morgan Street
Chicago, IL 60607-7053
USA
E-mail: dans@uic.edu

Caifeng Shan
Philips Research
High-Tech Campus 36
5656 AE, Eindhoven
The Netherlands
E-mail: caifeng.shan@philips.com

Dacheng Tao
Department of Computing
Hong Kong Polytechnic University
PQ704, 7/F, Building P
Hung Hom, Kowloon, Hong Kong
China
E-mail: dacheng.tao@gmail.com

Liang Wang
Department of Computer Science
University of Bath, BA2 7AY
United Kingdom
Email: lw356@cs.bath.ac.uk

ISBN 978-3-642-12899-8 e-ISBN 978-3-642-12900-1

DOI 10.1007/978-3-642-12900-1

Studies in Computational Intelligence ISSN 1860-949X

Library of Congress Control Number: Applied for

Typeset & *Cover Design:* Scientific Publishing Services Pvt. Ltd., Chennai, India.

Printed on acid-free paper

9 8 7 6 5 4 3 2 1

springer.com

Preface

As cameras become more pervasive in our daily life, vast amounts of video data are generated. The popularity of YouTube and similar websites such as Tudou and Youku provides strong evidence for the increasing role of video in society. One of the main challenges confronting us in the era of information technology is to effectively rely on the huge and rapidly growing video data accumulating in large multimedia archives. Innovative video processing and analysis techniques will play an increasingly important role in resolving the difficult task of video search and retrieval. A wide range of video-based applications have benefited from advances in video search and mining including multimedia information management, human-computer interaction, security and surveillance, copyright protection, and personal entertainment, to name a few.

This book provides an overview of emerging new approaches to video search and mining based on promising methods being developed in the computer vision and image analysis community. Video search and mining is a rapidly evolving discipline whose aim is to capture interesting patterns in video data. It has become one of the core areas in the data mining research community. In comparison to other types of data mining (e.g. text), video mining is still in its infancy. Many challenging research problems are facing video mining researchers. For example, how to extract knowledge from spatio-temporal data? How to infer high-level semantic concepts from low-level features in videos? How to exploit unlabeled and untagged video data? The use of classical data mining techniques for video data is impractical due to the massive volume of high-dimensional video data. To address these difficult challenges, it is necessary to develop search and mining techniques and methods that are suitable for video data.

The objective of this book is to present the latest advances in video search and mining covering both theoretical approaches and practical applications. The book provides researchers and practitioners a comprehensive understanding of the start-of-the-art in video search and mining techniques and a resource for potential applications and successful practice. This book can also serve as an important reference tool and handbook for researchers and practitioners in video search and mining.

The target audience of this book is mainly engineers and students working on video analysis in various disciplines, e.g. computer vision, pattern recognition, information technology, image processing, and artificial intelligence. The book is intended to be accessible to a broader audience including researchers and practicing professionals working in video applications such as video surveillance, video retrieval, etc.

The origin of this book stems from the immense success of the First International Workshop on Video Mining (VM'08), held in conjunction with the IEEE International Conference on Data Mining 2008. This workshop gathered experts from different fields working on video search and mining.

Organization and Themes

The book comprises both theoretical advances and applications in video search and mining. The organization of the book reflects the combination of analytical and practical topics addressed throughout the book. We have divided the book chapters into five parts; each addresses a specific theme in video search and mining. The five themes presented include motion trajectory analysis, high-dimensional video representation, semantic video analysis, personalized video, and video mining.

Part I. Motion Trajectory Analysis: Object motion trajectories describe the rich dynamic content of video data. Motion trajectory analysis plays an important role in video mining and has been exploited for various applications, e.g. video retrieval, video summarization, video surveillance, traffic monitoring, and sports analysis.

Chapter 1 is focused on the latest research in motion trajectory analysis for video search and mining. The main methodologies for the description of motion trajectories, as well as the indexing techniques and similarity metrics used in the retrieval process are introduced and examined through a comparative analysis, application examples, and discussion on future trends. In Chapter 2, trajectory clustering methods for learning activity models are introduced with a focus on parametric and non-parametric partition algorithms. A soft partition algorithm based on non-parametric mean-shift clustering is proposed and validated. The use of one- and multi-dimensional hidden Markov models (HMMs) for video classification and recognition is also introduced.

Several aspects of motion trajectory analysis including their representation, indexing, similarity, invariance, and application are addressed in Chapter 3. For instance, a representation of motion trajectories that is invariant to camera view is introduced based on null-space invariants. The representation of motion trajectories includes both isolated motion trajectories describing the dynamics of a single object as well as multiple motion trajectories characterizing the interaction among a group of objects in complex events. Several methods for efficient indexing based on matrix and tensor decomposition and various similarity measures for efficient storage, retrieval and classification of motion trajectories are reviewed.

Examples of the use of motion trajectory analysis in real world applications provide a vivid illustration and help the reader gain a deeper insight into the potential of motion trajectory analysis in video search and mining.

Part II. High-Dimensional Video Representation: Because of the innate abundance of information in video sequences, efficient representation of video data has long been a subject of intense interest in video analysis. Video representations are invariably high-dimensional, and thus a great deal of effort is required for processing video data. The representation of video in high-dimensional spaces is the topic of this part of the book.

Chapter 4 explores extraction of features that reflect three-dimensional structural information embedded in videos. Three-dimensional features venture beyond traditional video features, e.g. *two-dimensional appearance-based* features or spatio-temporal features. These three-dimensional features expand the kind of information captured from videos, which is essential in tasks such as video mining and retrieval. The main difficulty posed by this approach is the need to infer accurate three-dimensional information. This limitation is due to the fact that typical video footage does not convey prior knowledge about the scene configuration or camera calibration. Several recent methods that address the challenge posed by three-dimensional feature-based video analysis are reviewed including simultaneous localization and mapping, structure-from-motion, and 3D reconstruction. Illustrative use of tensor-based representation in practical applications including shot boundary detection, object recognition, content-based video retrieval as well as human activity recognition are presented, and limitations of the state-of-the-art techniques are discussed.

Models designed to understand, organize and utilize acquired high-dimensional video representation are considered. The goal is to extract semantically meaningful information from a large collection of samples in a high-dimensional space. Specifically, in video analysis, one of the fundamental aims is to recognize actions or persons. Statistical inference has been very successful in performing these tasks. Extension of statistical inference tools to manifolds in high-dimensional spaces is discussed in Chapter 5. Manifold is critical to the representation of video data, which is distributed over a small fraction of the high-dimensional space. Once the distribution of video data on manifolds has been captured, efficient statistical inference and learning methods can be developed. Empirical evaluation of various manifold-based video analysis techniques for applications such as video understanding is presented.

Part III. Semantic Video Analysis: The topic of understanding videos by semantic learning has been an important research area over the past decade. The formalism of semantic activity analysis including ontological representation of domain knowledge, statistical models and logical languages to describe activities, and the integration of semantic schemes will be discussed in Chapter 6. The focus is on techniques for addressing challenges in automatic activity recognition and abnormality detection such as robust primitive action detection, multi-agent interaction, higher-/lower-level action mapping, etc.

Video genre inference, a technique that automatically clusters and tags video data and thus facilitates user browsing, search and retrieval, provides an important practical illustration of semantic inference in video analysis. A comprehensive introduction of the necessary background for video genre inference as well as recent developments in the use of task-specific features, reflecting the video capture and camera configuration, is presented in Chapter 7.

Chapter 8 explores the problem of semantic visual learning from weakly labeled web video. A general tenet of learning theory is that an infallible supervisor always provides useful information that yields improved results. In the context of learning from videos, however, it is generally too expensive, and often not possible, to attain complete and accurate information by a supervisor. A technique for robust learning from weakly tagged video data (as available from the web) is described. The interesting feature of the technique illustrated is the mechanism used for automatic identification and filtering of irrelevant information. Another important component of the method described is the use of active learning to allow users to intervene in order to further improve the performance at a controllable cost.

Part IV. Personalized Video: Personalization is an increasingly important trait in recent multimedia applications. The theme in this part of the book will investigate the role of personalization in video analysis. In particular, the use of video-based face recognition for personalized video as well as a personalized recommendation system for broadcast news will be described.

Automatic face recognition is one of the most active research areas in computer vision. It allows automatic identification and verification of a person from a static image or video sequences. In contrast to traditional static image-based approaches, video-based face recognition technology utilizes the abundant video information, leading to more accurate and robust face recognition methods, and thus provides an invaluable tool for personalized video applications. A comprehensive survey of video-based face recognition and retrieval techniques for personalized video applications is presented in Chapter 9.

Because of the large number of broadcast channels and TV news programs, finding news videos of interest can be a difficult task. Chapter 10 presents an interactive framework for personalized news video recommendation. The personalized system allows news seekers to access programs of interest from large-scale news video archives more effectively. In this framework, multiple media sources (i.e. audio, video, and closed captions) are integrated to capture news topics, and their inter-topic contextual relationships are visualized to enable news seekers to interactively find news topics of interest.

Part V. Video Mining: The final theme of the book is focused on empirical tasks in video mining. The scope of the discussion ranges from low-level tasks, such as motion detection and re-occurrence identification to semantic-level tasks.

Motion is an important feature in content-based video parsing for high-level understanding. Mining motion information from video data is a critical task in video analysis. In Chapter 11, a case study is illustrated to introduce a unique technique of independent motion segmentation. A holistic, in-compression approach is presented,

thus attaining high-efficiency while providing very good performance in mining motion information from video data.

Another critical facet of video data is represented by pattern re-occurrence in video sequences. The problem of detection of (almost)-repetitive pattern occurrence in video footage is an important task in video mining. Examples of recurring video sequences include commercials, channel advertisements, channel intros, and newscast intros. Video occurrence identification requires integration of various fundamental video analysis techniques, including feature extraction, classifier training, and efficient search. Several technologies for recognizing predefined and unknown recurring video sequences are discussed for mining general videos and broadcast television in Chapters 12 and 13, respectively. A real-time recurring video sequence recognition system is also presented in Chapter 13.

Automatic video annotation is an important tool for video indexing and search. It has been well studied for several years. However, most existing efforts in video annotation have been conducted on small video corpuses and the size of concept vocabulary has been limited to tens or hundreds. The exponential increase in video data on the Web presents a great potential as well as new challenges for video annotation. Chapter 14 introduces the use of video mining techniques for automatic web-scale video annotation as well as annotating videos using large-sized natural language keywords. State-of-the-art techniques for video analysis, feature extraction and classification are presented. The focus is on the use of suitable techniques, key algorithms, and data structures, for extremely large-scale automatic video annotation. A new web-scale automatic video annotation method is presented in which the keywords are augmented based on previous annotations, by leveraging the large collection of video data. The effectiveness and efficiency of the method described are validated using experiments conducted on a corpus of over 1.2 million videos crawled from YouTube.

Acknowledgements

The preparation of this book has been an arduous task and required the devotion of many people. We would like to thank all the authors for their tremendous effort and dedication in preparing the book chapters. We would also like to thank all of the reviewers as well as the members of the editorial board of the Video Mining Workshop at ICDM'08. The invaluable comments and suggestions provided by the reviewers have helped shape the final book. Finally, we would like to thank Thomas Ditzinger, Senior Editor at Springer-Verlag Heidelberg, for his constant dedication and support throughout the preparation of this book.

Editors

Prof. Dan Schonfeld
Department of Electrical & Computer Engineering
University of Illinois at Chicago, Chicago, IL 60607-7053, USA
Email: dans@uic.edu
URL: http://www.ece.uic.edu/~ds

Dr. Caifeng Shan
Philips Research
High-Tech Campus 36, 5656 AE, Eindhoven, The Netherlands
Email: caifeng.shan@philips.com
URL: http://www.dcs.qmul.ac.uk/~cfshan

Dr. Dacheng Tao
School of Computer Engineering
Nanyang Technological University, Singapore 639798
Email: dacheng.tao@ieee.org
URL: http://www.ntu.edu.sg/home/dctao

Dr. Liang Wang
Department of Computer Science
University of Bath, BA2 7AY, United Kingdom
Email: lw356@cs.bath.ac.uk
URL: http://www.cs.bath.ac.uk/~lw356/

Contents

Part I
Motion Trajectory Analysis

Object Trajectory Analysis in Video Indexing and Retrieval Applications

Mattia Broilo, Nicola Piotto, Giulia Boato,
Nicola Conci, and Francesco G.B. De Natale

Abstract. The focus of this chapter is to present a survey on the most recent advances in representation and analysis of video object trajectories, with application to indexing and retrieval systems. We will review the main methodologies for the description of motion trajectories, as well as the indexing techniques and similarity metrics used in the retrieval process. Strengths and weaknesses of different solutions will be discussed through a comparative analysis, taking into account performance and implementation issues. In order to provide a deeper insight on the exploitation of these technologies in real world products, a selection of examples will be introduced and examined. The set of possible applications is very wide and includes (but it is not limited to) generic browsing of video databases, as well as more specific and context-dependent scenarios such as indexing and retrieval in visual surveillance, traffic monitoring, sport events analysis, video-on-demand, and video broadcasting.

1 Introduction and Motivations, and Requirements

The concept of indexing and retrieval of videos is rather mature and various consumer systems are already available and widely used in web browsing and related applications (Google Video, Yahoo!, YouTube). Nevertheless, content-based video retrieval is a very complex task and commercial systems still largely rely on pure textual search. The most challenging issue consists in capturing users' semantic, which is still an unsolved problem, except for very specific or professional application domains where video analysis and interpretation is less affected by subjectivity. Unambiguous descriptions are difficult to be extrapolated unless a common dictionary or specific rules for annotation are shared a priori. To bridge the semantic gap between video content and its interpretation given by humans, motion information

Mattia Broilo, Nicola Piotto, Giulia Boato, Nicola Conci, Francesco G.B. De Natale
DISI - University of Trento, Via Sommarive, 14, 38100 Trento (Italy)
e-mail: {broilo,piotto,boato,conci}@disi.unitn.it,
denatale@ing.unitn.it

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 3–32.
springerlink.com

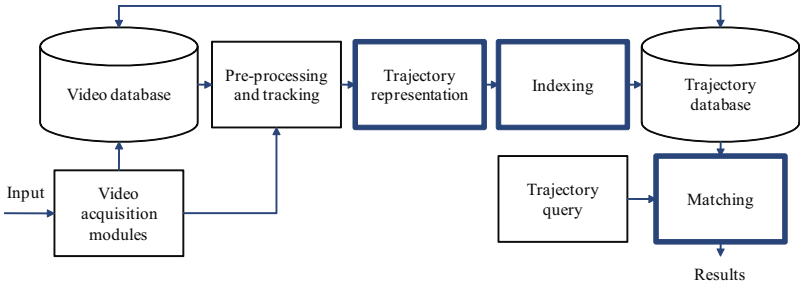


Fig. 1 General scheme for a content-based video retrieval (CBVR) system.

plays a crucial role [1]. In fact, suitable motion analysis tools may allow identifying activities, behaviors, and interactions that are fundamental to understand the video content. The effectiveness of such features has been already proven in different application domains, including video surveillance and monitoring, human-machine interaction, automatic target recognition, automotive applications, and sport analysis.

Regardless the application scenario, a proper, compact, and meaningful description of the video sequence is crucial to achieve effective indexing. The identification and extraction of significant features is a complex task, since video data convey huge amount of information in both static (e.g., color statistics, appearance models, shapes, textures) and dynamic (e.g., object or camera motion, inter-frame statistics, scene activity) form. These features are widely used to capture the visual contents of the sequence [2]. For instance, through moving object segmentation and trajectory analysis it is possible to achieve a spatio-temporal description of an object that would support activity classification [3] [4], behavior analysis [5] [6], and interaction description [7] [8] [9].

Fig. 1 presents a high-level block diagram of a trajectory-based retrieval system. First, the video sequence is pre-processed (on-line or off-line) by appropriate filtering and tracking algorithms to extract object trajectories. Such trajectories are usually noisy and need to be properly processed to ease the analysis, clustering, and classification. After that, an indexing module uses the trajectory description to find the best match within a set of models or a repository to be browsed. Several factors may affect the performance of the whole process. In particular, effective motion representation cannot be achieved without taking into consideration the issues related to the acquisition process. For example, the trajectory extracted from a video acquired using a camera on the ceiling of a room is significantly different from the one extracted by another camera along the walls, even though they refer to the same scene. The corresponding motion description will be different and only an a priori knowledge about the geometry of the room allows to associate them to the same event. More in general, we can summarize the characteristics of the analyzed video on the basis of the following categories: (i) the geometrical features, (ii) the appearance of objects, (iii) the presence and characteristics of motion in the scene, (iv) the dimensionality of the analysis domain, and (v) the impact of noise and compression artifacts.

In this chapter a thorough overview of the most recent developments in trajectory analysis and matching is given, providing a critical analysis of the presented methods, paying particular attention to their possible application to media indexing and retrieval. The focus is mainly put on the modules of trajectory representation, indexing, and matching (highlighted in Fig. 1).

The chapter is organized as follows. In Section 2 we present and compare state of the art methodologies for trajectory representation. In Section 3 techniques for trajectory indexing are introduced, while corresponding matching strategies are outlined and compared in Section 4. Section 5 provides some concrete examples of trajectory-based video retrieval applications. Finally, future research trends and challenges are drawn in Section 6.

2 Trajectory Representation

The problem of trajectory representation mainly consists of achieving an approximation of the raw path through a parametric curve. The simplest model consists in the use of chain codes [10], or piecewise linear approximations [11]. More accurate representations may use curvilinear approximations such as polynomials [12] or splines [13]. The above methods consider the trajectory as a 2D projection of the spatial displacement of the point inside the scene, even though 3D representations are gaining considerable ground in the research community. In the following sections we will mainly focus on 2D representations, since they are more widespread. However, some of the methods introduced for 2D have a straightforward extension to 3D, also considering that in most cases the multidimensional analysis can be carried out by combining different 2D views. Even though other features such as velocity, motion direction, temporal offset, or view invariant tensor null-space representations [14] have been considered in the literature, for the sake of conciseness we will focus on spatial approximation, making it clear that the methods described hereafter could be extended to the other trajectory features.

2.1 Polygonal Approximation

Polygonal approximation has been used in many pattern recognition problems, including shape and contour representation. The idea is to interpolate the raw trajectory with a piecewise linear curve with limited number of vertices. Since most of the information is connected to the points of maximum curvature, this representation turns out to be significant as well as compact. The resulting polygon should fulfill two requirements: (i) best fit of the original curve, and (ii) minimum number of segments. The two conditions are conflicting, then a trade-off has to be found by concurrently solving two optimization problems:

- *Error $\varepsilon()$ minimization problem:* given a set of N points representing a raw trajectory $T = \{t_i\} = \{x_i, y_i\}_{i=1}^{N}$, find the polygonal curve $P = \{p_j\} = \{x_j, y_j\}_{j=1}^{M}$

with a number of line segments M, so that the approximation error $\varepsilon(T,P)$ is minimized.

- *Number of dominant points minimization problem:* given a set of N points representing a raw trajectory T, find the polygonal curve P with the minimum number of segments M so that the approximation error $\varepsilon(T,P)$ does not exceed a maximum tolerance ε_{tol}.

The most common criteria used for optimization are the compression ratio $CR = N/M$ and the integral square error [15] between vertices of T and linear segments of P. Fig. 2 shows a real trajectory and its approximation. The final number of segments varies according the reconstruction error threshold imposed. Many polygonal approximation techniques have been proposed in the literature. Jointly optimal algorithms are quite slow, with a complexity in the order of $O(N^2)$ or even $O(N^3)$. It is possible to reduce the complexity to $O(Nlog(N))$ by focusing on one of the two minimization problems [16]. Heuristics are also widely used.

Approximation methods differ upon specific requirements, such as the application context (e.g., pedestrians vs vehicles), and the error metrics used for evaluation. The most important approximation methods can be roughly classified into four categories: (1) sequential, (2) split & merge, (3) dominant point-detection, and (4) optimization algorithms approaches. In the next paragraphs we will provide a general overview of them.

2.1.1 Sequential Tracing Approaches

In sequential algorithms, the trajectory is progressively scanned and a mismatch condition evaluated, when the error exceeds a threshold a new segment is started. Algorithms are usually fast and can be applied in real-time, thus making them attractive in trajectory representation even though the accuracy of the approximation is quite limited. Among these methods, Sklansky and Gonzalez [17] proposed a scan-along procedure for digitized curves, which starts from a point chosen randomly and tries to find the longest line segments sequentially. Kurozumi and Davis [18] proposed instead a minimax method, which determines the segments by minimizing the maximum distance between a given set of points and the corresponding

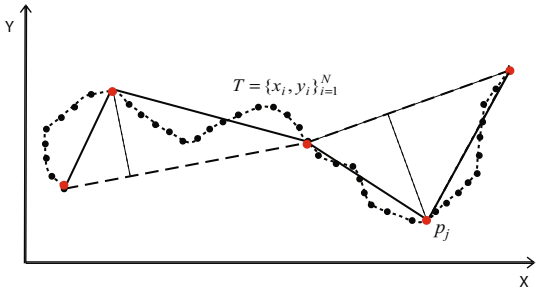


Fig. 2 Polygonal approximation.

segment. Wall and Danielsson [19] proposed a sequential method, which scans the points and outputs a new segment when the area deviation per length unit of the current segment exceeds a pre-defined error. Ray and Ray [20] determine the longest possible line segments with the minimum possible error. All the aforementioned algorithms are designed to solve one of the minimization problems and reach the solution in $O(logN)$ or $O(N)$ steps of binary search [16].

2.1.2 Split and Merge Methods

Split-based methods use a top-down approach where the coarsest approximation is the segment connecting the first and last point of the path. If the approximation is not satisfactory, it is refined by recursively splitting the segments until the accumulated error reaches a predefined threshold or the maximum number of segments is exceeded (see Fig. 3). Depending on the split procedure, the number of pieces at the end of the process may be higher than needed. In this case, a merging may occur to re-connect adjacent segments with similar direction. The depth of the split is driven by the application requirements and can be adjusted by varying the split criterion or the thresholds. The algorithm requires the availability of the whole path.

The most popular algorithm in this field is a heuristic method known as "Douglas-Peucker" [21], adopted in both [22] and [23]. The iterative procedure splits the curve into smaller elements and, at each iteration, calculates the distance of each vertex from the original curve. The stop condition is fulfilled when the cumulative distance is smaller than a given tolerance ε. The complexity of the method is $O(N^2)$ in the worst case, and $O(NlogN)$ on average.

Leu and Chen [24] presented a hierarchical merging method, which considers the trajectory as composed by a number of consecutive arcs. They are replaced by their chord only if the arc-to-chord deviation results lower than a given threshold. Ansari and Delp [25] proposed a technique that first uses Gaussian smoothing to reduce the noise and then selects the points with maximal curvature as break points. The split & merge process is then applied to the obtained samples. Ray and Ray [26] proposed an orientation-invariant and scale-invariant method by introducing the use of ranks

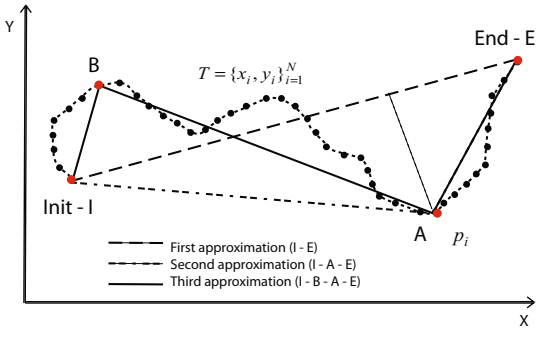


Fig. 3 Polygonal approximation using iterative splitting.

of points and normalized distances. In this case, the approximation returned by the split-and-merge may be far from the optimal one if the initial segmentation is not accurate.

2.1.3 Dominant Point-Detection Methods

The core idea of this class of algorithms is that a shape is well represented by its high-curvature points [27]. Then, a contour can be described by using such points as the vertices of a piecewise linear interpolation. Several heuristics have been designed to this purpose. Teh and Chin [28] determine the curvature at each point based on a support region, and detect the dominant points through a non-maxima suppression process. Other approaches rely on the detection of salient points. Held et al. [29] first apply a coarse-to-fine smoothing to identify dominant points, and then define a hierarchical approximation based on perceptual significance. Zhu and Chirlian [30] determine the importance of each point by transforming the curve into polar coordinates and then calculating the relevant derivatives. In this class of algorithms it is also possible to identify methods that search for the most significant points using relaxation labeling [31]. The paper focuses on the contour extraction of shapes, but the extension of the work to trajectory analysis is straightforward. In this approach, the left and right slopes and the curvature are evaluated and associated with an attribute list to each point of the input curve. This information determines the initial probability of the current point to be a *side* (a linear piece in the case of a trajectory), or an *angle* (a point with strong curvature). The relaxation process iteratively updates the probabilities until convergence. The obtained *angle* points can therefore be used as a meaningful representation of the whole trajectory.

Similarly to split&merge, the above methods require the availability of the whole trajectory. Moreover, their performance is bounded by the accuracy achieved in the evaluation of the curvature.

2.1.4 Optimization Algorithms

The approximation problem is here considered as an optimization task where the global error is the cost to be minimized. The search of the solution that provides the minimum error can be performed by stochastic optimization methods (e.g., genetic algorithms [32], ant colony [33], particle swarm optimizations [34]), or by local optimization methods (e.g., tabu search [35] and vertex adjustment [36]). The initialization is obtained based on some heuristics, and the approximation is progressively improved towards the minimum of the global error. The final solution can be considered nearly-optimal, although the global minimum is usually not guaranteed. In these algorithms, the trajectory points are typically examined in sequential order. The computational cost of these algorithms is pretty high, but the achievable results have a higher fidelity since they are specifically designed to climb local minima associated to suboptimal representations.

2.2 Spline Approximation

A completely different approach consists in the use of splines [13]. Splines are smooth curves (typically polynomials) that interpolate a set of points in a plane. Among the many different spline types, B-Splines (a generalization of the Bèzier curves) are very commonly adopted:

$$A(u) = \sum_{i=0}^{n} N_{i,d}(u) p_i \tag{1}$$

where $p_i, i = 1,2,\ldots,n$ are the control points and $N_{i,d}(u)$ are the B-spline basis functions of order d. Control points represent the points of the original trajectory. To create the spline approximation $A(u)$, a vector of *knots* $U = \{u_0, u_1, \ldots u_{m-1}\}$ is needed. Given the degree of the polynomial d and n control points, the number of *knots* m should be equal to $n+d+1$. U is a set of non-decreasing values in $[u_0, u_{m-1}]$ and all basis functions lie in this interval. The i-th basis function $N_{i,d}$ is calculated using the Cox de Boor formula:

$$N_{i,0}(u) = \begin{cases} 1 \ if \ u_i \leq u < u_{i+1} \\ \\ 0 \quad otherwise \end{cases}$$

$$N_{i,d}(u) = \frac{u - u_i}{u_{i+d} - u_i} \cdot N_{i,d-1}(u) + \frac{u_{i+d+1} - u}{u_{i+d+1} - u_{i+1}} \cdot N_{i+1,d-1}(u). \tag{2}$$

The spline approximation consists of determining the optimal coefficients of the polynomial model to fit the trajectory. If the trajectories to be described are complex, higher degree polynomials may be used, or more often the curve is segmented and represented through piecewise polynomial fitting. In this case, trajectory segmentation problems arise similar to those discussed in polygonal approximation. Usually, the degree of the polynomial is kept low (e.g., third degree) to ensure smoothness and avoid oscillations.

Spline approximation can be considered as a subset of the polynomial approximations used in many areas of computer graphics. In the specific case of video indexing and retrieval it is mainly used to smooth the noisy path of a moving object in tracking. An important property of the approximated curve is that it is invariant to affine transformations. Another advantage of the B-splines is that a local change in the raw trajectory value does not affect the whole approximated curve, because each control vertex influences $\pm k/2$ segments of the polygon in u. A sample curve approximated using a spline is shown in Fig. 4. Even though splines are usually computed directly on the raw trajectory for computational reasons, it may be desirable to adopt a curvature detection algorithm to extract dominant points to be used as control points. The above problem can be formulated as a nonlinear optimization problem as follows. Given:

- a set of points of the raw trajectory t_k, $k = 1,2,\ldots,K$ in the plane
- a B-spline curve $A(u) = \sum_{i=0}^{n} N_{i,d}(u) p_i$ with control points p_i

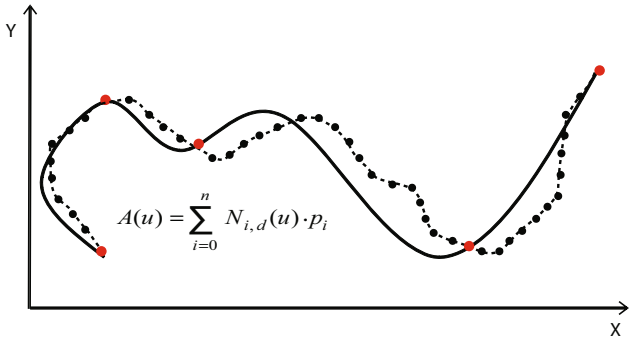


Fig. 4 Spline approximation.

- the order and the knots of the B-spline curve (not subject to optimization)

find the control points $p_i, i = 1, 2, \ldots, n$ such that f, defined in (3), is minimized

$$f = \frac{1}{2} \sum_{k=1}^{K} ||A(u) - t_k||^2 + \lambda f_s. \tag{3}$$

where f_s is a regularization term to ensure a smooth curve, and λ is a positive constant that determines the weight of f_s.

2.3 PCA Coefficients

Principal Components Analysis (PCA) is a method that reduces data dimensionality based on the analysis of samples covariance. As such, it is suitable for data sets in multiple dimensions, such as trajectory points [37]. The idea is to characterize a raw trajectory only with its principal components, getting rid of all features that do not convey significant information. Given a vector trajectory T made by a set of random variables and known correlation matrix C, the $k-$th principal component (PC) is given by an orthonormal linear transformation:

$$PC_k = a_k T \tag{4}$$

where a_k is an eigenvector of C corresponding to its $k-$th largest eigenvalue λ_k. Algorithms for the computation of the PCs are usually based on correlation or covariance [38]. Another approach that exploits PCA, is presented by Bashir et al. in [39], where the authors first segment trajectories using a curvature zero-crossing approach, followed by a clustering routine. The method applies then a two-level PCA analysis, in which the principal components are first extracted from the whole object trajectory, and successively analyzed to determine the corresponding sub-trajectories.

2.4 String-Based (Syntactical) Representation

Syntax-based approaches convert the analytic representation of a trajectory into strings of symbols, to provide a higher-level description of the path. The main idea

Table 1 Trajectory representation: a comparative analysis.

Category	Description	Strengths and Weaknesses
Polygonal approximation	Interpolate the raw trajectories with piecewise linear curves with the minimum number of vertices.	**Pros:** simple and fast. **Cons:** depends on thresholds.
Sequential tracing [17] [18] [20]	Scan the raw trajectory to evaluate error conditions. Start a new segment when the condition is not satisfied.	**Pros:** minimization of the delay. **Cons:** segmentation is not optimal.
Split and merge [22] [23] [24] [25] [26]	Iterative split until the accumulated error is below a threshold.	**Pros:** simple and fast. **Cons:** requires the full trajectory.
Dominant points detection [28] [29] [30] [31]	Find high-curvature points, and connect these key points via linear segments.	**Pros:** the trajectory curvature is preserved. **Cons:** entire trajectory needed.
Optimization algorithms [32] [33] [34] [35] [36]	Starting from an initial approximation, iterate to find the global approximation error.	**Pros:** near-optimal solution is found. **Cons:** high computational cost.
Spline approximation	Interpolate points with a smooth curve.	**Pros:** approximation as a smooth curve. **Cons:** entire trajectory needed.
Polynomial interpolation [12]	Numerical analysis to represent an interpolated curve with a polynomial.	**Pros:** optimal solution according numerical analisys. **Cons:** high computational cost; order of the polynomial dependent on the number of key-points.
B-spline approximation [46] [47] [48]	Piecewise polynomial approximation.	**Pros:** keypoint choice affects only a small part of the approximated curve. **Cons:** non linear approximation.
PCA [38] [39] [49]	Characterize a raw trajectory only with its principal components.	**Pros:** compact representation. **Cons:** part of the information is lost.
String-based representation	High level description using symbols.	**Pros:** add semantic information. **Cons:** requires preprocessing.
Points labeling [43]	Attach to a point/ set of points a text label describing motion primitives.	**Pros:** it accepts textual queries. **Cons:** machine learning needed.
Characters coding [44] [50]	Translate key points information into symbols and then code the sequence as a single string.	**Pros:** easy local and global matching. **Cons:** quantization necessary; key points detection is required.

is inspired by the alignment strategies used in bioinformatics to match genomic sequences [40]. String-based representation may be applied to raw trajectory samples, or to the approximated representation achieved by one of the methods described above. Once the key points of the trajectory are extracted, they are translated into symbols according to the associated spatio-temporal information, and aligned into strings.

String-based algorithms have been used in computer vision especially for shape classification [41] and they have been introduced more recently in object trajectory representation. For instance, in [42] the authors assume that each trajectory segment is labeled with a semantic symbol. Using the chain of successive symbols, a support vector machine is trained to classify different events. In this case, the clustering scheme requires a lot of training samples and each fragment has to be labeled in advance. In the work of Chen [43], key points are labeled using characters to map the moving direction. In this way, strings can be compared for both matching and clustering purposes. This representation can easily tackle the problem of partial matching, making it possible to detect sub-strings within the whole sequence of symbols. Furthermore, this representation allows to easily achieve the invariance to spatial shift and scaling. [44] provides a description of the trajectory using syntactical elements. Here, key trajectory points are extracted and represented by three characters, corresponding to angle, speed and temporal offset with respect to the previous point, to achieve a full spatio-temporal representation.

The advantage of syntactical approaches is that the matching phase can be simply implemented as a matching of ”words”, like in text processing tools, typically using similarity metrics such as the *edit distance* [45]. This part will be better described in the corresponding paragraph in Section 4.

3 Trajectory Indexing

In a retrieval system, indexing is a fundamental step to effectively structure large data sets, and to ease the matching process in successive search operations [51]. In content-based media retrieval this issue becomes even more evident. In fact, data are extremely heterogeneous and search is made more complex by the joint use of visual and textual features. Indexing methods taking into account object motion have been already investigated [52]. Two main classes of methodologies are considered in the following sections: tree-based approaches, where the data are organized into indexing structures, and transform-based approaches, in which the dimensionality of the problem is reduced by exploiting the characteristics of different digital transforms.

3.1 Tree-Based Indexing

The capability of storing data and track their changes over time is a considerably demanding task that required the implementation of efficient data structures that can be browsed easily and which content can be updated with minimum effort. A possibility is to adopt trees, thanks to the capability of performing insertion and

deletion operation in avery intuitive manner. In this paragraph we want therefore to present a number of works that address this issue in the specific case of handling the spatiotemporal evolution of a trajectory. In tree-based indexing the trajectories are pre-processed to extract a set of feature vectors, which are inserted into a multidimensional index tree, corresponding to the domain in which users express queries to retrieve the desired content. Tree-based indexing is mainly used to index video streams in real time monitoring systems (e.g., traffic monitoring, video surveillance), where the content and structure of the database is continuously evolving. Most of the proposed spatio-temporal techniques are based on variations of the R-tree [53] (where "R" stands for *rectangle*), which are common data structures used to organize data in multi-dimensional spaces. Here, the data is split into nested and possibly overlapping rectangles. Each non-leaf rectangle contains the information about its child nodes and their corresponding bounding box, making it possible to access data in a hierarchical manner (see Fig. 5).

3D R-trees [54] introduce time as an additional dimension by approximating trajectory segments with the relevant Minimum Bounding Boxes (MBBs). The weakness of 3D R-trees lies in the fact that spatial and temporal dimensions are treated in the same manner. Since the bounding box includes also the time dimension, the box overlap increases, thus reducing the discrimination capability of the tree. Looking at Fig. 6 it can be observed that bounding boxes cover large parts of space, whereas the actual space occupied by the trajectory is small. Accordingly, many trajectory segments could be represented by the same leaf node, even if they carry completely different information. Due to this small discrimination capability, the performance of 3D R-trees degrades rapidly, as soon as the data set size increases. Another category of trees is the so-called Historical R-trees (HR-trees) [55]. A separate R-tree is maintained for each time stamp, so that duplication of unchanged trajectory nodes in consecutive R-trees is avoided. This allows maintaining a history of the spatial

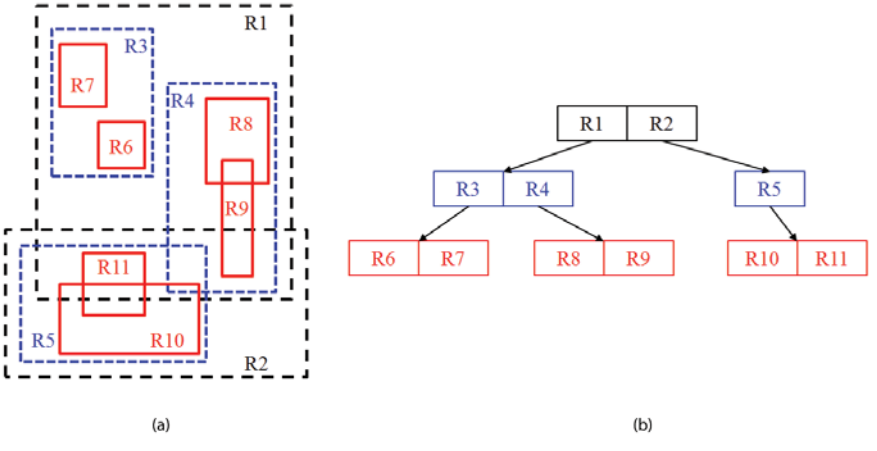


Fig. 5 R-Tree indexing example: 2D visualization (a), hierarchical dependencies (b).

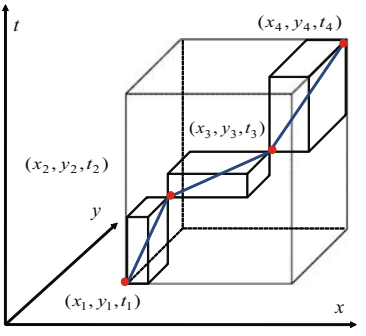


Fig. 6 3D R-Trees with Minimum Bounding Boxes.

position and overcomes the limitation of traditional R-trees (see Fig. 7), as well as reducing storage requirements. Spatial objects are indexed with the MBB and replaced if a change occurs. The resulting indexing structures are efficient in evaluating queries regarding a particular time stamp, while search performance degrades when the query concerns trajectories that span over long time intervals [56].

The MV3R-tree [57] is instead a hybrid structure that combines a multi-version R-tree (MV) with a small 3D R-tree [58]. Multi-Version R Trees are an enhanced version of a R-tree with the aim of eliminating duplicates by sharing as many common nodes as possible between different versions of R-trees. In MV3R-tree a 3D R-tree is built on the leaf nodes of an MVR-tree. To keep the space requirement manageable, the indices of both trees share the same leaves, which leads to a rather complex insert algorithm. The combined structure opens for two possible choices in query processing: the 3D R-tree or the MVR-tree. For queries involving the timestamp, MVR-tree is preferable and vice-versa. In addition, in MVR-trees temporal changes are modeled as discrete events: this means that objects maintain the same position until an update occurs. MV3R-trees outperform other indexing structures such as the 3D R-tree and the HR-tree, but they cannot be used to represent gradual changes in position. Trajectory Bundle trees (TB-tree) [59] are also based on R-trees. This method strictly preserves trajectories, in the sense that each leaf node

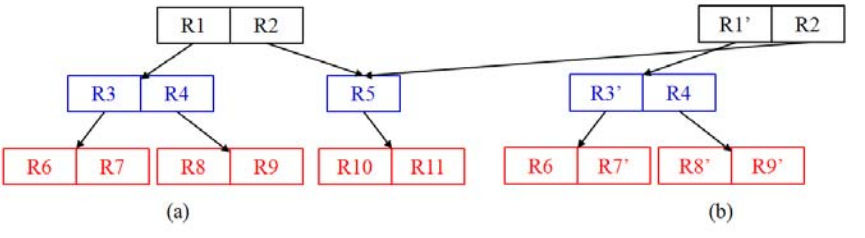


Fig. 7 HR-Tree indexing example: initial hierarchy at step 0 (a), index reconfiguration at step 1 keeping in consideration previous step (b).

only contains segments belonging to the same trajectory. The R-tree insert algorithm is modified so that leaf nodes contain trajectory segments belonging to only one moving object; the main idea is to cut the whole trajectory of a moving object into pieces and link them with a double linked list that (i) preserves trajectory evolution, and (ii) is simple in retrieving segments based on the trajectory identifier. This design aims at reducing the node accesses in retrieving a complete trajectory.

3.2 Transformation-Based Indexing

DFT is one of the most used transforms for indexing and retrieval of time sequences. The idea is to exploit the sparseness in the frequency domain to make a sort of compression of the trajectory information, keeping only the significant coefficients. A known limitation of the DFT is that it only considers spatial coordinates, discarding temporal references. Piecewise Fourier Transform of the time series has been proposed to remove this problem, but the trajectory segmentation leads to other problems: while large pieces reduce the potential of frequency representation, small pieces do not model low frequencies properly. More recently, DWT has replaced DFT in many applications of computer graphics [60] and signal processing [61], mainly thanks to its multi-resolution properties that allow analyzing the data at different levels of granularity. In [62] the DWT is applied to trajectories for dimensionality reduction in content-based search. This allows dealing with time-frequency localization, resulting in a more effective representation with respect to the DFT.

Other methods employed for dimension reduction are the Singular Value Decomposition (SVD) [63] and the KL transform, which however is computationally very demanding and therefore seldom used in real-time analysis.

4 Trajectory Matching

In order to exploit the trajectory information in a CBVR system, a final step is required, consisting in the match between users' query and indexed video data. This section deals with matching strategies that can be adopted to measure the similarity among trajectories. In developing a matching algorithm several issues should be taken into account. In fact, the extraction of object trajectories from video streams is typically imprecise due to environmental noise, illumination variations, processing errors, occlusions, and so on. The joint effect of these uncertainties typically leads to noisy trajectories that contain gaps and outliers. Moreover, even trajectories referring to similar events may present significant differences in several respects, such as initial direction and location, spatial length, temporal duration, and sampling rate, thus leading to mismatches. According to the abstraction level adopted for trajectory representation, matching techniques can be roughly divided into three categories: dynamic, statistical, and vector-based. Dynamic matching includes a set of simple yet effective comparison tools that can be applied to raw or filtered samples, and are able to deal with limited trajectory misalignments. Statistical matching is more sophisticated and requires a pre-processing to extract a set of consistent

low-level features, the similarity measure is then obtained by comparing the distribution of query and target samples in the feature space. Finally, vector matching algorithms rely on a high-level representation of the trajectories, where the feature vectors are mapped into symbols. This representation strongly simplifies the matching phase, which can be achieved through simple metrics (e.g., L_p-norm, Hausdorff or city block distances, string alignment techniques) and weighted combinations of the features.

4.1 Dynamic Matching

Methods referred to as dynamic matching are basic comparison tools, enabling the user to process sequences of different lengths. This is a key feature, since in real applications it is almost impossible to impose the same length to trajectories. The main feature of these methods is the capability of applying a local warping to the sequences, in order to achieve the best alignment. Depending on the application requirements, the stretch may be operated either in the temporal or spatial domain, thus providing time warping and spatial warping, respectively.

Concerning the temporal domain, dynamic time warping (DTW) is a distance measure used in 1-D time-series comparison [64]. Initially applied to speech signal analysis, it has been recently extended with success to different application domains including sign language recognition [65] and trajectory matching [66], because of its conceptual simplicity and versatility. Basically, the method relies on a classic distance operator (e.g., L_p-norm) and on a particular matching procedure that finds the optimal alignment between the query and the target series, allowing temporally shifted matches between samples. The matching score is calculated as the cumulative distance among samples.

The distance between two generic series $X = \{x_n\}_{i=0}^{N}$ and $Y = \{y_n\}_{j=0}^{M}$ can be measured by constructing a warping path [65], as in (5):

$$W = w_0, w_1, \ldots, w_K \qquad max\{N,M\} < K < (N+M-1) \tag{5}$$

where K is the length of the warp path and $w_k = (i, j)$ where i, j index X and Y, respectively. The warping path involves all samples in the trajectory (i.e., $w_0 = (0,0)$ and $w_K = (N,M)$); moreover, i and j have to be continuous (i.e., every index in all series has to be used) and monotonically increasing. The distance between X and Y is the optimum warp path that minimizes the overall warping distance, satisfying (6):

$$DTW(X,Y) = min\left\{\frac{1}{K}\left[\sum_{k=1}^{K} w_k\right]\right\} \tag{6}$$

where, w_k is the minimum distance between two samples indexes (one from X, one from Y) in the $k-th$ element of the warp path (see Fig. 8).

DTW presents some major drawbacks in the sensitivity to noise and outliers. Furthermore, since the global similarity score is evaluated on the basis of cumulative sample-to-sample distances, different sampling rates (scaling) and misalignments

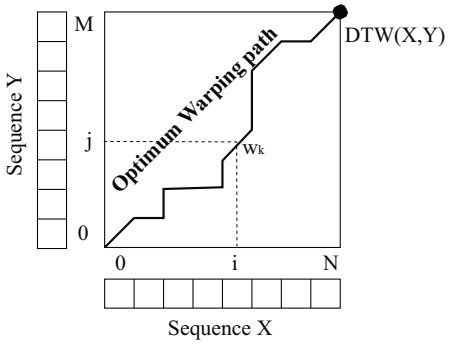


Fig. 8 DTW: optimum warping path construction.

(shifting) may lead to high distances even in the presence of very similar series. Finally, the computational complexity of the method is relatively high: $O((N+1)(M+1))$ with $N+1$, $M+1$ length of the sequences. Dynamic programming techniques are usually employed to effectively achieve the best alignment and to drastically reduce the complexity. As an example, given the two sequences X and Y, the distance $DTW(X,Y)$ is calculated as:

$$DTW(X_i,Y_j) = min \left\{ \begin{array}{l} DTW(X_i,Y_{j-1}) \\ DTW(X_{i-1},Y_j) \\ DTW(X_{i-1},Y_{j-1}) \end{array} \right\} + d(x_i,y_i) \tag{7}$$

where $d(\cdot,\cdot)$ is a distance metric that strictly depends on the employed trajectory representation, $X_i = \{x_0,x_1,\ldots,x_{i-1}\}$ and $Y_j = \{y_0,y_1,\ldots,y_{j-1}\}$. At each sample, the warping distance $DTW(X_i,Y_j)$ between X_i and Y_j indicates the cumulative sum of the local distance $d(x_i,y_j)$ and the minimum of cumulative distances among adjacent samples. In particular, $DTW(X_i,Y_j)$ is the optimum warping path between the first i samples of X and the first j of Y.

An improved approach is presented in [67], which nearly replicates the DTW matching scheme in the spatial domain with significant algorithmic enhancements. Given two sequences, the Longest Common SubSequence (LCSS) is used to optimize the alignment by finding the longest subsequence between two trajectories. This concept provides a higher flexibility allowing non consecutive samples and the insertion of gaps. This feature is fundamental, since it introduces the capability of (i) effectively processing paths of different lengths, (ii) coping with different sampling rates, and (iii) partially handling problems related to noise and outliers.

Fig. 9 sketches the difference between the alignments of the same two sequences, obtained through temporal and spatial warping, respectively. LCSS leads to a more significant alignment, since it allows excluding some samples from the matching process; on the contrary, DTW requires to match every query sample, thus causing a one-to-many association between query and target paths (Fig. 9(a)).

Similarly to DTW, LCSS relies on standard distance metrics, although employed in a different way. While in DTW the final score is evaluated by computing the

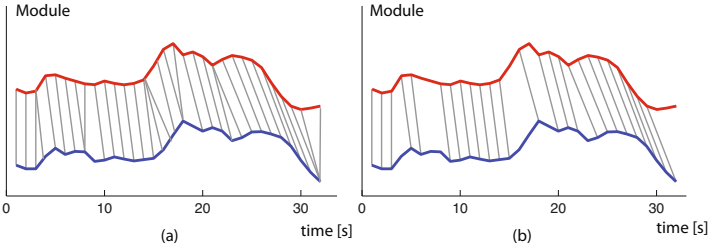


Fig. 9 Comparison between the alignments obtained through (a) DTW and (b) LCSS.

sample-to-sample distance, in LCSS the distance is used to check whether two samples are correlated or not [68]. In particular, this strategy consists in checking if the samples in the target trajectory fall within a spatio-temporal region defined in the query. If the condition is verified, the match occurs. The matching region is defined by two thresholds ε and δ, in space and time, respectively.

Since the computational burden of the alignment process is quite high, also the LCSS scheme is usually implemented with dynamic programming techniques, with a computational complexity in the order $O((N+1)(M+1))$. Given two 1D time series X and Y, the LCSS distance is evaluated by dynamically computing the matrix coefficients according the following recursion:

$$LCSS_{\varepsilon,\delta}(X_i,Y_j) = \begin{cases} 0 & \text{if } i=0 \quad \text{or } j=0 \\ 1+LCSS_{\varepsilon,\delta}(X_{i-1},Y_{j-1})) & \text{if } |x_i-y_j|<\varepsilon \text{ and } |i-j|<\delta \\ \max \begin{cases} LCSS_{\varepsilon,\delta}(X_{i-1},Y_j), \\ LCSS_{\varepsilon,\delta}(X_i,Y_{j-1}) \end{cases} & \text{otherwise} \end{cases} \tag{8}$$

Although performing generally better than DTW, LCSS still presents some limits in dealing with significant noise or outliers, since the insertion of gaps is neither penalized nor taken into account in the aligned subsequence.

4.2 Statistical Matching

In statistical matching, trajectory similarity is evaluated by analyzing the distribution of low-level features such as spatial location, local direction, or speed. In particular, these methods aim at estimating the probability density functions (*pdf*) of relevant parameters in order to build a statistical model of the target trajectory. Once the model has been defined, the similarity between query and target is calculated on the relevant distributions. A statistical inference process associates the input sequence T to the model M_n that most likely fits the query (see input/output flowchart in Fig. 10(b)). This classification requires a training phase (sketched in Fig. 10(a)), where the parameters of the machine learning algorithm (e.g., Self-Organizing

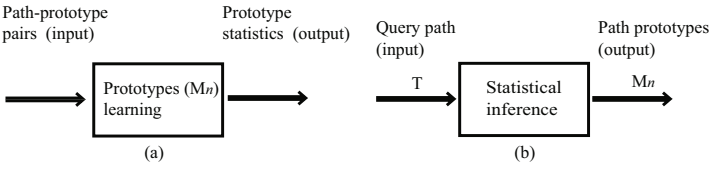


Fig. 10 Input/Output of statistic matching methods: (a) learning phase, (b) classification.

Network, Hidden Markov Models) are learned from a pre-classified set of trajectories or through unsupervised techniques.

The methods belonging to this family are proved to be particularly robust against noise and outliers. Two major aspects have to be taken into account in statistical matching: (i) the definition of target models, and (ii) the definition of the matching strategy. As to the first point, models are usually obtained by processing a significant set of paths associated to a given activity and estimating the distribution of the relevant features: spatial location, moving direction, object speed, object acceleration. The second point requires the definition of suitable metrics to evaluate the similarity between the statistics of input and model.

Among statistical methods for trajectory analysis, clustering techniques are very popular. In these techniques, similar paths are grouped together in clusters and compared against a query sample, to determine the class it belongs to, with a maximum similarity criterion. [69] proposes a strategy for trajectory distance measurement and clustering relying on Hidden Markov Models (HMM). The model of the path is build upon a mixture of HMMs and the similarity evaluation is performed by checking the statistical distribution of a given query over each model. More formally, considering two sequences X and Y, their distance is defined relying on their HMM parameterization as follows:

$$D(X,Y) = |L(X;\lambda_X) + L(Y;\lambda_Y) - L(X;\lambda_Y) - L(Y;\lambda_X)| \tag{9}$$

where λ_X and λ_Y are the models for X and Y, respectively, and the terms $L(\cdot;\cdot)$ indicates the likelihood of a path with respect to a model. It is worth noticing that when two sequences are similar, the cross terms are generally high. A major advantage of this approach is that the speed can be considered together with geometrical/spatial features of the trajectory. Moreover, the system can cope with the so-called uneven sampling instances (i.e., non-uniform trajectory sampling between consecutive points), which are typical of real-time tracking applications.

Top-down approaches introduce the concept of hierarchical clustering in statistical matching. [70] presents a hierarchical clustering strategy that first identifies global similarities and then refines the analysis of each coarse cluster. An initial wavelet decomposition is employed to tackle noise in the raw trajectory. After smoothing, a set of features is extracted concerned with the so-called trajectory resampling point set (TRPS) and trajectory directional histogram (TDH). TRPS is the result of the path resampling at regular spatial steps (i.e., it encodes specific positions); TDH is a directional histogram that considers the direction between

consecutive track samples (i.e., it encodes the statistical trend of the direction providing a rough path representation). Finally, a two step clustering is carried out: first, TDH information is exploited in a dominant-set clustering algorithm to identify coarse clusters; second, the similarity between two paths i and j is calculated with the Bhattacharyya distance as follows:

$$BD(i,j) = [1 - \sum_{b=0}^{N} \sqrt{TDH_{ib} TDH_{jb}}]^{1/2} \tag{10}$$

where TDH_{ib} and TDH_{jb} are the b^{th} elements of the directional histogram for the paths i and j, respectively. Once rough clusters have been identified, TRPS information is used to refine them, and the L_p-norm distance is used as metric.

More recently, [71] proposed an effective unsupervised clustering algorithm using mean-shift to detect coarse clusters, and a merging procedure to group adjacent blobs and eliminate outliers. Even though this method outperforms the work in [70] in the presence of noise, both methods require a resampling phase, thus not ensuring the preservation of the original temporal information. Another interesting work for trajectory clustering in video surveillance can be found in [72], where a system is proposed, able to create and update the trajectory clusters as soon as the samples are acquired by the tracker. The trajectory data are represented as concatenations of raw samples $T = \{t_i\} = \{x_i, y_i\}_{i=1}^{N}$, while each cluster C is represented by a prototype, defined as a stream of raw spatio-temporal locations in conjunction with an additional parameter (σ_i^2) that indicates the local variance of the cluster at time i:

$$C = \{c_i\} = \{x_i, y_i, \sigma_i\}_{i=1}^{M} \tag{11}$$

The metric to compare the incoming trajectory T against each detected cluster C is defined as the average normalized distance of every trajectory point (t_i) from the nearest point of the cluster, calculated within a variable-size temporal window w_i centered in i, as follows:

$$D(T,C) = \frac{1}{n} \sum_{i=1}^{N} d(t_i, C)\,; \quad d(t_i, C) = \min_j \left(\frac{dist(t_i, c_j)}{\sqrt{\sigma_j}} \right) \quad j \in w_i \tag{12}$$

where $dist(.,.)$ is the Euclidean distance.

A particular case of matching is when one wants to detect trajectories that do not comply with any model: this case is usually referred to as anomalous trajectory detection. Johnson et al. [73] propose a method for anomalous trajectory identification employing a sequence of feature vectors to represent the spatial location and the velocity of the object at each time instant. The approximation of the statistical distribution of the vectors in the feature space is achieved through a vector quantization. In particular, two concurrent neural networks are developed: initially the sequence of vectors that best represents a target trajectory is identified and then, similar tracks are clustered. According to the proposed network topology, each output node represents one of the models and it is said to *'win'* if the associated model is the nearest

to the feature vector presented as query. Leaky neurons with short-term memory capabilities are employed, in order to model also the temporal nature of the paths. The major drawback of this technique is that it cannot handle sub-trajectories. Another critical issue lies in the vector quantization phase, which provides a *pdf* approximation relying on the point distribution of prototypes. In particular, the number of the prototype vectors and their initial positioning within the feature space have to be manually defined. To cope with these problems, [74] proposes an improved method based on a completely autonomous system to detect anomalous motion. Such method extends [73], providing a learning module that ensures higher accuracy in the clustering phase and allows for an automatic setup of trajectory prototypes, i.e, the representative of the cluster. Each prototype is supposed to have a Gaussian distribution, and the anomaly detection is carried out by checking the fitness of the incoming path over the available models, according to a *Maximum-A-Posteriori* criterion. To improve the reliability of the system in detecting routes that range over wider time intervals a feedback to the neural network is introduced in [75], while Owens et al. in [76] employ a Kohonen self-organizing map [77]. The approach in [78] further improves [76] by introducing a new hierarchical network structure that allows faster learning.

4.3 Vector Matching

The general idea behind vector-based matching techniques is to extract a symbolic signature of the path in the form of a feature vector, in order to evaluate the similarity between trajectory pairs on the basis of the distance of the relevant signatures [79]. The path signature is calculated in two phases: first, the features are extracted from the raw data; second, quantization and symbolic coding are performed. Since the information is coded at the symbolic level, the vector distance can be effectively evaluated using simple metrics (e.g., Euler, Minkovsky, or Hausdorff distances). In Fig. 11 the generic flowchart of a vector-based matching is reported. The incoming path is pre-processed in order to bring its representation to a symbolic domain. The symbolic stream is then fed into the comparison routine to match the query with the signatures extracted from database entries.

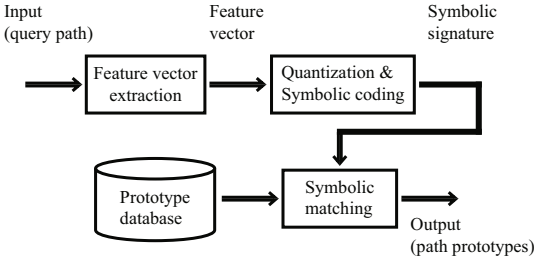


Fig. 11 Flowchart of vector-based matching methods.

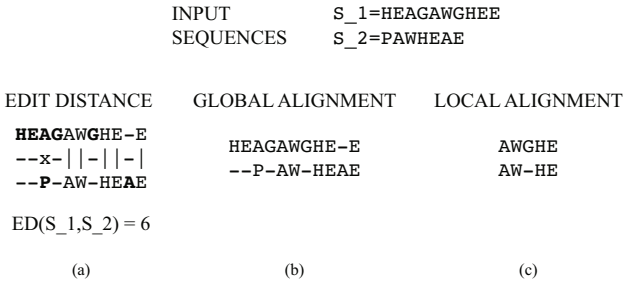


Fig. 12 Comparison between (a) edit distance, (b) global alignment and (c) local alignment.

Since the representation consists of a string of symbols, the comparison between trajectories can be casted to a string matching problem. The most popular metrics used in this context are based on the edit-distance [80], which defines the distance between two sequences as the minimum number of elementary operations required to convert one string into the other. The allowed operations are: *deletion*, *insertion* and *substitution*. Fig. 12-(a) reports an example of edit distance calculation between two textual strings, where *non-matching* characters are highlighted in bold. Referring to the alignment string, *pipe* stands for matching symbols, *cross* stands for symbol substitution, and - indicates a symbol insertion. In this example it is easy to see that the final edit distance is 6 since S_1 can be reverted to S_2 by substituting the symbol "*A*" with "*P*" and inserting 5 *gaps*. The most common way to calculate the edit distance is through a dynamic programming approach. Given two strings of symbols $X = x_0 \dots x_N$ and $Y = y_0 \dots y_M$ from a given alphabet, a matrix ED of $M+1$ rows and $N+1$ columns is initialized and filled starting from the upper left to the lower right corner, running the recursive algorithm reported in Table 2-(top). Here, d is the penalty for the gap insertion and it is set to 1 (i.e., $d = 1$), while the symbol substitution cost assumes binary values (i.e., $cost = 0, 1$), depending whether symbols match or not. Once the dynamic programming problem is solved, it returns the final edit-distance $ED(N,M)$.

Among the most interesting techniques that employ this approach, Chen et al. [81] introduce a symbolic trajectory retrieval system called movement pattern string (MPS). Raw samples are processed in order to recover information about the local direction and the distance ratio, i.e., the ratio between the current segment and the whole trajectory. These features are then quantized and each level is associated to a symbol. Accordingly, a similarity metric based on the Levenshtein distance [82] is used for trajectory alignment. Zheng et al. propose in [83] another interesting video retrieval system that compares video clips using a string-based alignment of trajectories. Although the temporal evolution of the path is a fundamental factor characterizing motion, both methods do not take into consideration temporal displacements. This problem is partially solved in [84], which proposes a complete retrieval system aiming at bridging the semantic gap between the users' query and the trajectory representation. The incoming samples are filtered and hierarchically clustered in space and time through spectral clustering. The classes are determined

Table 2 Different metrics for comparison evaluation: (top) edit-distance, (center) Global alignment, (bottom) Local alignment.

Category	***Cost* value**	**Initial matrix condition**	**Recursion**
Edit-distance [82]	Binary	$for \quad i=0 \quad to \quad N$ $\quad ED(i,0) = i \times d$ $for \quad j=0 \quad to \quad M$ $\quad ED(0,j) = j \times d$ $with \quad d=1$	$ED(i,j) = min \begin{cases} ED(i,j-1)+d \\ ED(i-1,j)+d \\ ED(i-1,j-1)+cost \end{cases}$
Global alignment [87]	Fuzzy	$for \quad i=0 \quad to \quad N$ $\quad G(i,0) = i \times d$ $for \quad j=0 \quad to \quad M$ $\quad G(0,j) = j \times d$ $with \quad d<0$	$G(i,j) = min \begin{cases} G(i,j-1)+d \\ G(i-1,j)+d \\ G(i-1,j-1)+cost \end{cases}$
Local alignment [88]	Fuzzy	$for \quad i=0 \quad to \quad N$ $\quad L(i,0) = 0$ $for \quad j=0 \quad to \quad M$ $\quad L(0,j) = 0$ $with \quad d<0$	$L(i,j) = max \begin{cases} 0 \\ L(i,j-1)+d \\ L(i-1,j)+d \\ L(i-1,j-1)+cost \end{cases}$

using the minimum cumulative square distance as a metric. Each cluster is then associated with high-level activity models automatically learned from the paths. Finally, the acquired activity models are indexed in a hierarchical tree. A more recent implementation that exploits the edit-distance, is presented in [85]. Here, the object trajectories are processed and represented by a chain of symbols indicating the direction and velocity components (sampling time is assumed unitary and constant). The symbolic mapping of the path is then achieved by quantizing each component, in order to reduce the redundancy. Since no resampling or trajectory smoothing is applied, the symbolic mapping may lead to long symbol chains where each sample is encoded as a symbol. In the same class of methods, [86] proposes a comparison strategy inspired by the alignment methods adopted in bioinformatics to match genomic sequences [87][88], also referred to as *inexact* or *approximate matching*.

The adopted metrics rely on modifications of the Levenshtein distance. As for edit-distance, they are still based on the combination of elementary operations such as deletion, insertion and substitution of symbols, but they assign arbitrary scores to each of them, in order to make more flexible the fitness function. This kind of

Table 3 Trajectory matching: a comparative analysis.

Category	Path	Description	Pros & Cons
Dynamic [64] [65] [67] [68]	Raw sample stream, syntactic-symbolic representation.	Stretching in temporal/spatial domain to determine the best alignment. The similarity is the cumulative distance between samples or longest common subsequences. Dynamic programming is used to recover the warping path and the final similarity score.	**Pros:** match series with different lengths. **Cons:** high computational complexity.
Statistic [69] [70] [71] [72] [73] [74] [75] [76] [78]	Vectors of low-level features.	A statistical model is build processing low level features. The similarity is evaluated comparing the current values with the models. Clustering is a popular approach.	**Pros:** different features are considered and modeled separately. **Cons:** large data set required for training; the *pdf* modeling heavily depends on the quality of the training.
Vector-based [44] [79] [81] [83] [84] [85] [89]	Syntactic-symbolic representation.	Extraction of low-level features, coded at symbolic level. The trajectory similarity is measured using metrics such as Euler, Minkovsky, Hausdorff distances, or modifications of the edit-distance.	**Pros:** Comparison between string of symbols. **Cons:** Pre-processing is required to map the samples into the symbolic domain. Parameters depend on the scenario.

matching algorithms provide several advantages over the implementations described in Section 4.1. In particular, they provide a confidence parameter as compared to hard (match/no-match) criteria. Referring to the formula in Table 2, the modifications concern the evaluation of substitution *costs*, expressed as real numbers in the range $[0-1]$. Since alphabet of symbols is fixed, substitution scores of each symbol pair can be encoded in a set of *substitution matrices*.

The examples in Fig. 12-(b) and 12-(c) show the results achieved when matching the same pair of genetic sequences using a global and a local alignment, respectively. The global alignment [87] optimizes the score corresponding to the overall matching of the whole sequence, while the local alignment [88] searches for the most similar subsequences. Dynamic problems that have to be solved in order to recover global or local alignments are fairly the same, except for the matrix initialization and some slight differences in the recursive algorithm [87][88]. In [88] the first row and column of the dynamic matrix are padded with zeros and the recursion is made using a *maximum* operator over the same entries as in edit-distance. In [87], the initialization follows the scheme of edit-distance and considers a negative score for gaps (i.e., $d < 0$). The recursion part is instead the same as for the edit-distance. In Table 2 a schematization is reported to underline algorithmic differences among edit-distance, global and local alignment.

Recently, a syntactic representation was introduced in [44], where the path is decomposed in high-level syntactic elements that represent significant substrings of the original trajectory. The structure of the symbols has been arranged according to a set of rules that ensure a flexible representation. Referring to Fig. 11, the flow chart of the algorithm provides a preprocessing phase to detect meaningful spatio-temporal discontinuities. Each element is quantized in direction, velocity, and differential time, and then mapped into a symbol triplet according to a predefined codebook. The matching among trajectories can be therefore expressed in terms of the highest score obtained by aligning the strings of symbols.

In Table 3 a comparison among the considered matching categories in terms of representation, strengths and weaknesses is reported.

5 Trajectory-Based Video Retrieval Applications

In this section we describe how the motion information can be exploited in CBVR, by discussing a selection of representative approaches that consider trajectory analysis for retrieval purposes. Special attention is paid to the aspect of query formulation, i.e., the instrument provided to the user to browse contents in a structured video database.

5.1 Integration of Trajectory Analysis in CBVR Tools

The existing approaches using motion information to retrieve contents emphasize the decomposition of object trajectories and adequate matching strategies to compare them [90]. Chang et al. [91] propose an online video retrieval system supporting automatic object-based indexing, and spatio-temporal queries. The system includes algorithms for automated object segmentation and tracking, and real-time video editing techniques to fulfill user queries. An approach that exploits PCA can be found in [50], where the authors use the segmentation tool to reduce the dimensionality of the trajectory data. In the wavelet framework, Sahouria and Zakhor [92] propose a trajectory-based system for video indexing, based on Haar wavelet transform coefficients. Chen and Chang [49] use a wavelet decomposition to segment each trajectory and produce an index based on velocity features. Jung et al. [93] introduce a motion model based on polynomial curve fitting, used as a key to access individual objects and an efficient indexing and retrieval tool based on object-specific features. Dagtas et al. [52] present a model that uses object motion information to characterize events and allows video retrieval through scale-invariant algorithms for spatio-temporal search. Basharat et al. [94] index and retrieve videos using the volume of the moving object instead of the appearance features. Trajectories are modeled through the Scale Invariant Feature Transform (SIFT), and are clustered to form motion segments.

All of the above approaches allow extracting a set of features from the original trajectory, which can be considered as a *signature*. Such mathematical representation has then to be matched with the users' queries. From there the importance of a suitable formulation of the query.

5.2 Query Formulation

Natural human-computer interaction is one of the paramount challenges in computer science. In content-based media retrieval, this problem arise when one has to translate a generic user request, expressed in different ways (natural language, keywords, metadata, examples), into a structured and quantitative target for the search. While browsing a video database containing thousands of sequences, corresponding to thousands of hours of contents and terabytes of data, the selection of a proper query is a critical matter. Visual queries can be formulated by specifying any combination of object shapes, textures and movements with different levels of complexity. Restricting the focus to trajectory-based systems, it is possible to summarize the query formulation into three main types:

- *Query by example (QbE).* To avoid the problem of expressing the query in a conventional way by using language, a simple alternative is to use as a query a video sample showing similar properties to what one is searching [50]. The system directly extrapolates the trajectory from the provided video sequence and matches it with database trajectories. The question to be answered is then "find videos that best match this type of motion". A major problem of QbE is the availability of a sample. Furthermore, as recalled at the beginning of the chapter, the appearance of the object motion in the scene may differ significantly from its spatio-temporal characteristics, simply due to different camera settings and point of view.
- *Query by sketch (QbS).* This method is a particular case of *QbE* [95] where the sample trajectory is input in the form of a drawing. The user might specify the size and shape of the object, its trajectory, as well as its velocity and acceleration. Systems supporting *QbS* usually smooth the sample trajectory first, to bridge the differences between the user's representation (that could be affected by irregularities) and the stored paths. One of the first systems that uses this kind of queries is *Violone* [96], even though more recent frameworks allow the user exploiting both *QbE* and *QbS* [97].
- *Semantic-based query.* The most challenging trends in trajectory-based video retrieval refer to the capability of extracting a semantic description of the object path, to generate automatic annotations to be used in database querying. Thonnat et al. [98] proposed a method to representing human activities using scenarios, which are translated into text by filling entries in a template of natural language sentences. Buxton et al. [99] use a Bayesian belief network and inference engine in highway traffic scenes, to produce high-level concepts such as lane changing or car stalling. In [100] an event-based visual surveillance system for monitoring vehicles and pedestrians is proposed, which supplies verbal descriptions of

dynamic activities in 3D scenes. Recently, a first-order complete query language for trajectory databases, allowing expressing queries directly in terms of speed and beads, has been presented in [101]. The main advantage is that the formulation is inherently invariant under speed- and bead-preserving transformations.

6 Future Perspectives and Research Challenges

In this chapter we have analyzed the current status of the research in the field of content-based video retrieval using object motion analysis. The problem has been split into three main aspects: trajectory description, trajectory-based indexing, and trajectory matching. The described methodologies and the relevant references provide a large technological toolbox, which can be used as a basis for developing powerful applications and instruments. Nevertheless, an ultimate CBVR system is still out of reach of current knowledge, and several challenging issues still remain open for researchers working in this exciting field.

As we have pointed out, one big leap that still makes the process very difficult and constrained to specific application domains is related to the dimensionality of the data. Although the use of 3D data (through stereo or multi-view capture) is more and more widespread, most of the available media, in particular at the consumer level, are still 2D. This poses several problems related to the fact that only projected motion can be detected. Even though this may change in the future, historical archives will maintain an immense quantity of 2D data for the years to come. This claims for more efficient techniques of camera motion estimation, 3D motion from 2D data, spatial reasoning, and so on.

Furthermore, on the one hand most of the presented approaches are unable to take into account at the same time all the spatio-temporal details contained in the object motion, and consider geometrical data as the main source of information, neglecting temporal references, which often play a major role in the detection of a motion model or behavior (the same spatial evolution can significantly differ if performed at different speeds). On the other hand, once all information is taken into account, it would be important to suitably manage invariance problems (scale, rotation, shift, duration), which could strongly affect the performance of a retrieval application.

Another, and probably the most important, challenge is the introduction of semantics in the analysis. There is an ever growing need of new methodologies able to increase the abstraction level in media search. Classical approaches relying on low-level descriptions have to evolve in this direction, aiming at bridging the gap between the quantitative nature of the raw data and user perception of the visual content. A fundamental contribution in this respect is given by the exploitation of the contextual information, which allows labeling a generic motion pattern with the semantics related to domain, environment, events, situations. The idea behind this is that future integrated media-based retrieval tools (using motion together with visual descriptions and context) may succeed in aligning the capabilities of current multimedia understanding algorithms with the richness and subjectivity of human interpretation of media.

References

1. Wang, Z., Lu, L., Bovik, A.C.: Video quality assessment based on structural distortion measurement. Signal Processing: Image Communication 19(2), 121–132 (2004)
2. Mansouri, A.R., Mitiche, A., El Feghali, R.: Spatio-temporal motion segmentation via level set partial differential equation. In: Fifth IEEE Southwest Symposium on Image Analysis and Interpretation, p. 243. IEEE Computer Society, Los Alamitos (2002)
3. Naftel, A., Khalid, S.: Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. Multimedia Systems 12(3), 227–238 (2006)
4. Min, J., Kasturi, R.: Activity recognition based on multiple motion trajectories. In: Int. Conf. on Pattern Recognition, vol. 4, pp. 199–202 (August 2004)
5. Bashir, F., Khokhar, A., Schonfeld, D.: Object trajectory-based activity classification and recognition using hidden Markov models. IEEE Trans. on Image Processing 16(7), 1912–1919 (2007)
6. Morris, B.T., Trivedi, M.M.: A Survey of Vision-Based Trajectory Learning and Analysis for Surveillance. IEEE Trans. on Circuits and Systems for Video Tech. 18(8), 1114–1127 (2008)
7. Oliver, N.M., Rosario, B., Pentland, A.P.: A bayesian computer vision system for modeling human interactions. IEEE Trans. on Pattern Analysis and Machine Intelligence 22(8), 831–843 (2000)
8. Parameswaran, V., Chellappa, R.: View invariance for human action recognition. Int. J. Comput. Vision 66, 83–101 (2006)
9. Ma, X., Bashir, F., Khokhar, A., Schonfeld, D.: Event analysis based on multiple interactive motion trajectories. IEEE Trans. Circuits Syst. Video Techn. 19(3), 397–406 (2009)
10. Kaneko, T., Okudaira, M.: Encoding of arbitrary curves based on the chain code representation. IEEE Trans. on Communications 33(7), 697–707 (1985)
11. Pavlidis, T.: Polygonal approximations by newton's method. IEEE Trans. on Computing 26(8), 800–807 (1977)
12. Von Stryk, O., Bulirsch, R.: Direct and indirect methods for trajectory optimization. Annals of Operations Research 37(1), 357–373 (1992)
13. Medioni, G., Yasumoto, Y.: Corner detection and curve representation using cubic B-splines. In: IEEE Int. Conf. on Robotics and Automation, vol. 3 (1986)
14. Chen, X., Schonfeld, D., Khokhar, A.: Robust null space representation and sampling for view-invariant motion trajectory analysis. In: IEEE Conf. on Computer Vision and Pattern Recognition, pp. 1–6 (June 2008)
15. Rosin, P.L.: Techniques for assessing polygonal approximations of curves. IEEE Trans. on Pattern Analisys and Machine Intelligence 19(6), 659–666 (1997), doi:10.1109/34.601253
16. Chan, W.S., Chin, F.: Approximation of polygonal curves with minimum number of line segments. LNCS, p. 378. Springer, Heidelberg (1992)
17. Sklansky, J., Gonzalez, V.: Fast polygonal approximation of digitized curves. Pattern Recognition 12(5), 327–331 (1980)
18. Kurozumi, Y., Davis, W.A.: Polygonal approximation by the minimax method. Computer Graphics and Image Processing 19(3), 248–264 (1982)
19. Wall, K., Danielsson, P.E.: A fast sequential method for polygonal approximation of digitized curves. Computer Vision Graphics Image Processing 28(3), 220–227 (1984)
20. Kumar Ray, B., Ray, K.S.: Determination of optimal polygon from digital curve using L1 norm. Pattern Recognition 26(4), 505–509 (1993)

21. Douglas, D.H., Peucker, T.K.: Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. Cartographica: The Int. J. for Geographic Information and Geovisualization 10(2), 112–122 (1973)
22. Ballard, D.H.: Strip trees: A hierarchical representation for curves. Communications of the ACM 24(5), 310–321 (1981)
23. Duda, R.O., Hart, P.E.: Pattern classification and scene analysis, New York (1973)
24. Leu, J.G., Chen, L.: Polygonal approximation of 2-D shapes through boundary merging. Pattern Recognition Letters 7(4), 231–238 (1988)
25. Ansari, N., Delp, E.J.: On detecting dominant points. Pattern Recognition 24(5), 441–451 (1991)
26. Ray, B.K., Ray, K.S.: A new split-and-merge technique for polygonal approximation of chain coded curves. Pattern Recognition Letters 16(2), 161–169 (1995)
27. Attneave, F.: Some informational aspects of visual perception. Psychological Review 61(3), 183–193 (1954)
28. Teh, C.H., Chin, R.T.: On the detection of dominant points on digital curves. IEEE Trans. on Pattern Analysis and Machine Intelligence 11(8), 859–872 (1989)
29. Held, A., Abe, K., Arcelli, C.: Towards a hierarchical contour description via dominant point detection. IEEE Trans. on Systems, Man and Cybernetics 24(6), 942–949 (1994)
30. Zhu, P., Chirlian, P.M.: On critical point detection of digital shapes. IEEE Trans. on Pattern Analysis and Machine Intelligence 17(8), 737–748 (1995)
31. Mikheev, A., Vincent, L., Faber, V., Inc, L.T., Seattle, W.A.: High-quality polygonal contour approximation based on relaxation. In: Int. Conf. on Document Analysis and Recognition, pp. 361–365 (2001)
32. Ho, S.Y., Chen, Y.C.: An efficient evolutionary algorithm for accurate polygonal approximation. Pattern Recognition 34(12), 2305–2317 (2001)
33. Yin, P.Y.: Ant colony search algorithms for optimal polygonal approximation of plane curves. Pattern Recognition 36(8), 1783–1797 (2003)
34. Yin, P.Y.: A discrete particle swarm algorithm for optimal polygonal approximation of digital curves. J. of Visual Communication and Image Representation 15(2), 241–260 (2004)
35. Yin, P.Y.: A tabu search approach to polygonal approximation of digital curves. Int. J. of Pattern Recognition and Artificial Intelligence 14(2), 243–255 (2000)
36. O'connell, K.J., Inc, M., Schaumburg, I.L.: Object-adaptive vertex-based shape coding method. IEEE Trans. on Circuits and Systems for Video Tech. 7(1), 251–255 (1997)
37. Moore, B.: Principal component analysis in linear systems: Controllability, observability, and model reduction. IEEE Trans. on Automatic Control 26(1), 17–32 (1981)
38. Jolliffe, I.T.: Principal Component Analysis. Springer, New York (2002)
39. Bashir, F., Khokhar, A., Schonfeld, D.: Segmented trajectory based indexing and retrieval of video data. In: IEEE Int. Conf. on Image Processing, vol. 2 (2003)
40. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. Molecular Biology 48(3), 443–453 (1970)
41. Gdalyahu, Y., Weinshall, D.: Flexible syntactic matching of curves and its application to automatic hierarchical classification of silhouettes. IEEE Trans. on Pattern Analysis and Machine Intelligence 21(12), 1312–1328 (1999)
42. Wu, G., Wu, Y., Jiao, L., Wang, Y.F., Chang, E.Y.: Multi-camera spatio-temporal fusion and biased sequence-data learning for security surveillance. In: ACM Int. Conf. on Multimedia, pp. 528–538. ACM, New York (2003)
43. Hsieh, J.W., Yu, S.L., Chen, Y.S.: Motion-based video retrieval by trajectory matching. IEEE Trans. on Circuits and Systems for Video Tech. 16(3), 396–409 (2006)

44. Piotto, N., Conci, N., De Natale, F.G.B.: Syntactic matching of pedestrian trajectories for ambient intelligence applications. IEEE Trans. on Multimedia 11(7) (2009)
45. Ristad, E.S., Yianilos, P.N.: Learning string-edit distance. IEEE Trans. on Pattern Analysis and Machine Intelligence 20(5), 522–532 (1998)
46. Schoenberg, I.J.: Contributions to the problem of approximation of equidistant data by analytic functions. Quarterly of Applied Mathematics 4, 45–99 (1946)
47. Loncaric, S.: A survey of shape analysis techniques. Pattern Recognition (1998)
48. Ikebe, Y., Miyamoto, S.: Shape design, representation, and restoration with splines. Picture Engineering, 75–95 (1982)
49. Chen, W., Chang, S.F.: Motion trajectory matching of video objects. In: IS&T/SPIE, San Jose, CA (2000)
50. Bashir, F.I., Khokhar, A.A., Schonfeld, D.: Real-time motion trajectory-based indexing and retrieval of video sequences. IEEE Trans. on Multimedia 9(1), 58–65 (2007)
51. Idris, F., Panchanathan, S.: Review of image and video indexing techniques. J. of Visual Communication and Image Representation 8(2), 146–166 (1997)
52. Dagtas, S., Al-Khatib, W., Ghafoor, A., Kashyap, R.L., Res, P., Manor, B.: Models for motion-based video indexing and retrieval. IEEE Trans. on Image Processing 9(1), 88–101 (2000)
53. Sellis, T., Roussopoulos, N., Faloutsos, C.: The R-tree: A dynamic index for multi-dimensional objects. The VLDB Journal, 507–518 (1987)
54. Theoderidis, Y., Vazirgiannis, M., Sellis, T.: Spatio-temporal indexing for large multimedia applications. In: IEEE Int. Conf. on Multimedia Computing and Systems, pp. 441–448 (1996)
55. Nascimento, M.A., Silva, J.R.O.: Towards Historical R-trees. In: ACM Symposium on Applied Computing, pp. 235–240. ACM, New York (1998)
56. Nascimento, M.A., Silva, J.R.O., Theodoridis, Y.: Evaluation of access structures for discretely moving points. LNCS, pp. 171–188. Springer, Heidelberg (1999)
57. Tao, Y., Papadias, D.: Mv3r-tree: a spatio-temporal access method for timestamp and interval queries. In: Int. Conf. on Very Large Data Bases, pp. 431–440. Morgan Kaufmann Publishers Inc., San Francisco (2001)
58. Becker, B., Gschwind, S., Ohler, T., Seeger, B., Widmayer, P.: An asymptotically optimal multiversion B-tree. The Int. J. on Very Large Data Bases 5(4), 264–275 (1996)
59. Pfoser, D., Jensen, C.S., Theodoridis, Y.: Novel approaches in query processing for moving object trajectories. In: Int. Conf. on Very Large Data Bases, pp. 395–406. Morgan Kaufmann Publishers Inc., San Francisco (2000)
60. Stollnitz, E.J., DeRose, T.D., Salesin, D.H.: Wavelets for computer graphics: theory and applications. Morgan Kaufmann, San Francisco (1996)
61. Akansu, A.N., Haddad, R.A.: Multiresolution signal decomposition. Academic Press, Boston (1992)
62. Chan, K.P., Fu, A.W.C.: Efficient time series matching by wavelets. In: IEEE Int. Conf. on Data Engineering, pp. 126–135. Institute of Electrical and Electronics Engineers (1999)
63. Korn, F., Jagadish, H.V., Faloutsos, C.: Efficiently supporting ad hoc queries in large datasets of time sequences. In: ACM SIGMOD Int. Conf. on Management of Data, pp. 289–300. ACM, New York (1997)
64. Berndt, D., Clifford, J.: Using dynamic time warping to find patterns in time series. In: AAAI 1994 Workshop on Knowledge Discovery and Databases, pp. 229–248 (1994)
65. Keogh, E.J., Pazzani, M.J.: Scaling up dynamic time warping for data mining application. In: ACM SIGKDD Int. Conf. on Knowledge discovery and data mining, pp. 285–289 (2000)

66. Hu, W., Tan, T., Wang, L., Maybank, S.: A survey on visual surveillance of object motion and behaviors. IEEE Trans. on Systems, Man and Cybernetics 34, 334–352 (2004)
67. Das, G., Gunopoulos, D., Mannila, H.: Finding similar time series. In: Komorowski, J., Żytkow, J.M. (eds.) PKDD 1997. LNCS, vol. 1263, pp. 88–100. Springer, Heidelberg (1997)
68. Vlachos, M., Hadjieleftheriou, M., Gunopoulos, D., Keogh, E.: Indexing multidimensional time-series with support for multiple distance measures. In: ACM SIGKDD, pp. 216–225 (2003)
69. Porikli, F.: Trajectory distance metric using hidden Markov model based representation. In: IEEE European Conf. on Computer Vision (2004)
70. Li, X., Hu, W., Hu, W.: A coarse-to-fine strategy for vehicle motion trajectory clustering. In: IEEE Int. Conf. on Pattern Recognition, vol. 1, pp. 591–594 (2006)
71. Anjum, N., Cavallaro, A.: Unsupervised fuzzy clustering for trajectory analysis. In: IEEE Int. Conf. on Image Processing, vol. 3, pp. 213–216 (2007)
72. Piciarelli, C., Foresti, G.L., Snidaro, L.: Trajectory clustering and its application for video surveillance. In: IEEE Conf. on Advanced Video and Signal Based Surveillance, pp. 40–45 (2005)
73. Johnson, N., Hogg, N.: Learning the distribution of object trajectories for event recognition. Image and Vision Computing 14, 609–615 (1996)
74. Mecocci, A., Pannozzo, M.: A completely autonomous system that learns anomalous movements in advanced video surveillance applications. In: IEEE Int. Conf. on Image Processing, vol. 2, pp. 586–589 (2005)
75. Sumpter, N., Bulpitt, A.: Learning spatio-temporal patterns for predicting object behavior. Image and Visio Computing 18, 697–704 (2000)
76. Owens, J., Hunter, A.: Application of the self-organizing map to trajectory classification. In: IEEE Int. Workshop Visual Surveillance, vol. 18, pp. 77–83 (2000)
77. Kohonen, T.: Self-organizing maps. Springer, Heidelberg (1995)
78. Hu, W.M., Xie, D., Tan, T.N.: A hierarchical self-organizing approach for learning the patterns of motion trajectories. IEEE Trans. on Neural Network 15, 135–144 (2004)
79. Imran, N., Javed, O., Shah, M.: Multi feature path modeling for video surveillance. In: Int. Conf. on Pattern Recognition, pp. 716–719 (2004)
80. Marzal, A., Vidal, E.: Computation of normalized edit distance and applications. IEEE Trans. on Pattern Analysis and Machine Intelligence 15, 926–932 (1993)
81. Chen, L., Otsu, M.T., Oria, V.: Symbolic representation and retrieval of moving object trajectories. In: ACM SIGMM Int. Workshop on Multimedia Information Retrieval, pp. 227–234 (2004)
82. Levenshtein, V.I.: Binary codes capable of correcting deletions, insertions, and reversals. Soviet Physics Doklady 10, 707–710 (1966)
83. Zheng, J.B., Feng, D.D., Zhao, R.C.: Trajectory Matching and Classification of Video Moving Objects. In: IEEE Workshop on Multimedia Signal Processing, vol. 10, pp. 1–4 (2005)
84. Hu, W., Xie, D., Fu, Z., Zeng, W., Maybank, S.: Semantic-based surveillance video retrieval. IEEE Trans. on Image Processing 16, 1168–1181 (2007)
85. Calderara, S., Cucchiara, R., Prati, A.: A Dynamic programming technique for classifying trajectories. In: Int. Conf. on Image Analysis and Processing, pp. 137–142 (2007)
86. Piotto, N., Conci, N., De Natale, F.G.B.: Syntactic matching of pedestrian trajectories for behavioral analysis. In: Proc. of 10th IEEE Workshop on Multimedia Signal Processing, pp. 877–882 (2008)

87. Needleman, S.B., Wunsch, C.D.: A general method applicable to the search for similarities in the amino acid sequence of two proteins. J. of Molecular Biology 48, 443–453 (1970)
88. Smith, T.F., Waterman, M.S.: Identification of Common Molecular Subsequences. J. of Molecular Biology 147, 195–197 (1981)
89. Vlacos, M., Kollios, G., Gunopulos, D.: Discovering similar multidimensional trajectories. In: Int. Conf. on Data Engineering, pp. 673–684 (2002)
90. Shan, M.K., Lee, S.Y.: Content-based video retrieval via motion trajectories. In: Society of Photo-Optical Instrumentation Engineers (SPIE) Conf. Series, pp. 52–61 (1998)
91. Chang, S.F., Chen, W., Meng, H.J., Sundaram, H.: A fully automated content-based video search engine supportingspatiotemporal queries. IEEE Trans. on Circuits and Systems for Video Tech. 8(5), 602–615 (1998)
92. Sahouria, E., Zakhor, A.: Motion indexing of video. In: IEEE Int. Conf. on Image Processing, vol. 2 (1997)
93. Jung, Y.K., Lee, K.W., Ho, Y.S.: Content-based event retrieval using semantic scene interpretation for automated traffic surveillance. IEEE Trans. on Intelligent Transportation Systems 2(3), 151–163 (2001)
94. Basharat, A., Zhai, Y., Shah, M.: Content based video matching using spatiotemporal volumes. Computer Vision and Image Understanding 110(3), 360–377 (2008)
95. Chang, S.F., Chen, W., Meng, H.J., Sundaram, H.: VideoQ: an automated content based video search system using visual cues. In: ACM Int. Conf. on Multimedia, pp. 313–324. ACM, New York (1997)
96. Yoshitaka, A., Hosoda, Y., Yoshimitsu, M., Hirakawa, M., Ichikawa, T.: Violone: Video retrieval by motion example. J. of Visual Languages and Computing 7(4), 423–443 (1996)
97. Aghbari, Z., Kaneko, K., Makinouchi, A.: Content-trajectory approach for searching video databases. IEEE Trans. on Multimedia 5(4), 516–531 (2003)
98. Le, T., Boucher, A., Thonnat, M.: Subtrajectory-based video indexing and retrieval. In: Cham, T.-J., Cai, J., Dorai, C., Rajan, D., Chua, T.-S., Chia, L.-T. (eds.) MMM 2007. LNCS, vol. 4351, p. 418. Springer, Heidelberg (2007)
99. Buxton, H., Gong, S.: Visual surveillance in a dynamic and uncertain world. Artificial Intelligence 78(1-2), 431–459 (1995)
100. Remagnino, P., Tan, T., Baker, K.: Multi-agent visual surveillance of dynamic scenes. Image and Vision Computing 16(8), 529–532 (1998)
101. Kuijpers, B., Othman, W.: Trajectory databases: Data models, uncertainty and complete query languages. In: Schwentick, T., Suciu, D. (eds.) ICDT 2007. LNCS, vol. 4353, pp. 224–238. Springer, Heidelberg (2006)

Trajectory Clustering for Scene Context Learning and Outlier Detection

Nadeem Anjum and Andrea Cavallaro

Abstract. We present a scene understanding strategy for video sequences based on clustering object trajectories. In this chapter, we discuss a set of relevant feature spaces for trajectory representation and we critically analyze their relative merits. Next, we examine various trajectory clustering methods that can be employed to learn activity models, based on their classification into hierarchical and partitional algorithms. In particular, we focus on parametric and non-parametric partitional algorithms and discuss the limitations of existing approaches. To overcome the limitations of state-of-the-art approaches we present a soft partitional algorithm based on non-parametric Mean-shift clustering. The proposed algorithm is validated on real datasets and compared with state-of-the-art approaches, based on objective evaluation metrics.

1 Introduction

The considerable decrease in the costs of video equipment and the dramatic increase in storage capabilities have favored the widespread use of video recording and video analytics for applications such as remote sensing, visual surveillance, home videos and sport event coverage. This generates huge volumes of visual data, which are impractical to mine or analyze effectively by a trained person or a group of experts. There is therefore the need to automatically manage these masses of visual data in order to discover and describe interesting events, enable summarization and key-frame extraction.

Nadeem Anjum
Queen Mary University of London
e-mail: nadeem.anjum@elec.qmul.ac.uk

Andrea Cavallaro
Queen Mary University of London
e-mail: andrea.cavallaro@elec.qmul.ac.uk

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 33–51.
springerlink.com

Video mining has gained considerable attention and several techniques have been presented to recognize certain events and activities, which provide means to summarize the data into a more convenient form. This also helps in the annotation that can be used for efficient searching and retrieval. However, existing approaches usually require considerable domain knowledge input prior to event analysis.

This chapter focuses on describing how to understand the semantics of a scene, with little or no contextual knowledge, by analyzing the patterns of a large number of moving objects. Starting from the observation that object motion is conditioned by the underlying scene structures and that these structures govern the expected activities in a video, we build event models by observing the motion trajectories over time. These models enable the accurate inference of the underlying scene. In particular, this chapter describes a framework to cluster trajectories for scene understanding and for abnormal trajectory detection.

A natural way to handle the challenging task of scene understanding is to first track objects and collects their trajectories over a period of time. Next, we cluster the trajectories into typical patterns and use them to build activity models. In order to compare trajectories, several spatio-temporal feature spaces are investigated and discussed. Furthermore, we discuss the notion of abnormal activities and how to detect them using the learnt activity models.

This chapter is organized as follows: Sec. 2 discusses trajectory representations for clustering. Section 3 discusses the existing trajectory clustering approaches and presents a framework for efficient clustering. Section 4 details the cluster fusion process across multiple feature spaces, whereas Section 5 explains the outlier detection techniques. Section 6 discusses the comparisons of algorithms on real datasets. Finally, in Sec. 7 we draw conclusions.

2 Trajectory Representation

Let a trajectory T^j be represented as $T^j = \{(x_i^j, y_i^j) : i = 1, \ldots, n^j\}$, where (x_i^j, y_i^j) is the estimated position of the j^{th} target at instant i on the image plane and n^j is the number of trajectory samples.

Due to non-linearity of object motions and errors generated by existing trackers, it is possible to have unequal (both in time and space) trajectories that may belong to the same class of object activity. To obtain trajectories of equal length, zero-padding and resampling are commonly used [1]. Although, these approaches are computationally simple, but due to large databases of raw trajectories and the presence of noisy observations it is very difficult to select a-priori the appropriate length of each trajectory. An important step in clustering algorithms is the choice of a suitable representation, such that spatio-temporal variability does not affect the overall clustering. This section discusses and compares existing approaches for trajectory representation.

Trajectory representation techniques can be divided into two groups, namely supervised and unsupervised. Supervised representations rely on the information

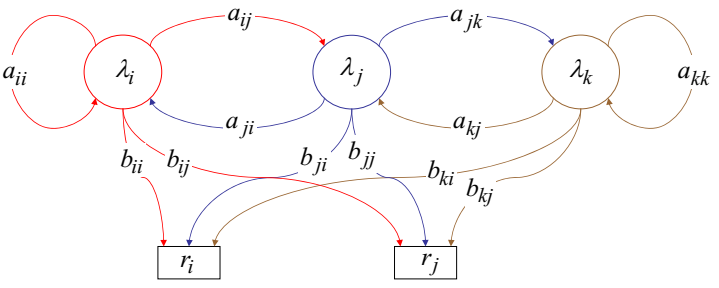


Fig. 1 HMM-based trajectory representation. Key; λ_i: i^{th} hidden state; a_{ij}: probability of transition from state i to state j; b_{ij}: probability of getting output observation j from state i and r_i: i^{th} observation.

supplied by training samples, e.g., Hidden Markov Models (HMMs) and Gaussian Mixture Models (GMMs) [2, 3, 4, 5, 6, 7].

In *HMM-based trajectory representation* each trajectory is assumed to be produced by an underlying hidden stochastic process [8]. A model $\hat{\mathscr{T}}^j$ represents the trajectory T^j as $\hat{\mathscr{T}} = (\lambda, A, B)$, where λ is a set of hidden states, A=(a_{ij}) is a transition probabilities matrix and $B = (b_{ij})$ is an observation probabilities matrix. Figure 1 shows an HMM-based trajectory representation. During the training phase, $\hat{\mathscr{T}}^j$ is learnt from T^j with appropriate initializations of state transition and prior probability matrices. Finally, each trajectory is summarized by a unique model. In order to lower the dependence on the parameter initialization, a trajectory can belong to more than one HMM, with some probability [9]. Although HMMs are robust to dynamic time warping, the structures and probability distributions are highly domain-dependent. Moreover, the parameter space increases considerably with the complexity of the events, as more hidden states are required for modeling.

GMM-based approaches are also used to associate statistical properties to each trajectory. In such approaches, a trajectory can be represented by a mixture of M Gaussian models as

$$\hat{T}^j = \sum_{m=1}^{M} w_m p(T^j | w_m, \theta_m), \tag{1}$$

where θ_m=(μ_m, Σ_m) is the Gaussian's parameter defined for the m^{th} component and w_m is the corresponding mixing weight. Figure 2 shows an example where two trajectories that started and ended at the same point are represented by two different models. Again, these approaches tend to be sensitive to the initial choice of the model parameters and the choice of number of mixing components.

An important aspect of HMM-based (with multiple states) trajectory representation is its ability to capture the temporal correlation between consecutive object detections. We assume that the state transition matrix of the HMM model will avoid abrupt state transition and hence will result in smoothed trajectory [10].

Unlike their counterpart, *unsupervised trajectory representations* do not require training samples. Examples of such representations include Principal Component

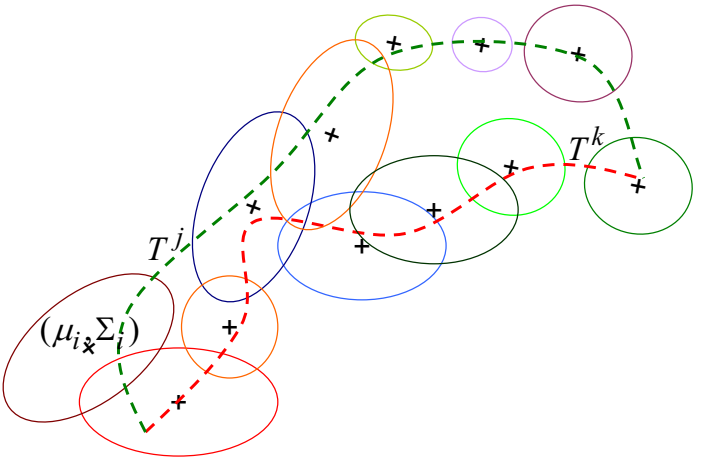


Fig. 2 GMM-based representation of a trajectory pair.

Analysis (PCA) [11, 12], Trajectory Directional Histogram (TDH) [13], average velocity, directional distance, trajectory mean, polynomial regression [14, 15, 16] and Discrete Fourier Transform (DFT) coefficients [17, 18].

Principal Components Analysis (PCA) has been used extensively, prior to clustering, to reduce the dimensionality of the data set, while extracting the most important data variations. PCA is calculated as

$$\widetilde{P}^j = P^j.T^j, \tag{2}$$

where, $\widetilde{P}^j$ is the projection of T^j in PCA space by applying transformation matrix P^j. When a trajectory is represented by number of sub-trajectories, based on temporal ordering or curvature changes, then PCA is applied on these sub-trajectories [11, 12, 19]. PCA works well for data with Gaussian distribution and requires an accurate estimation of the noise covariance matrix from the data, which is generally a difficult task. Furthermore, in its standard form, PCA representations do not contain high-order statistical information and therefore the analysis is limited to second-order statistics.

TDH is another representation to encode the statistical directional distribution of the trajectories as is calculated as $\widetilde{H}^j = \mathscr{H}(\theta^{\mathbf{j}})$, where $\mathscr{H}(.)$ is a histogram function calculated over the directional angles ($\theta^{\mathbf{j}}_{\mathbf{i}} = tan^{-1}(y^j_{i+1} - y^j_i / x^j_{i+1} - x^j_i)$).

PCA and TDH features do not encode spatial information. For this reason, two similar trajectories that are far on the image plane would be mapped on the same location. In order to incorporate spatial information, other ancillary feature representations can also be used, such as average velocity, directional distance and trajectory mean. The average velocity, $\widetilde{V}^j$, describes the rate of change of the j^{th} object position. $\widetilde{V}^j$ helps separating the trajectories of objects moving at varying pace and is defined as

$$\widetilde{V}^j = \frac{1}{n_j - 1} \sum_{i=1}^{n_j - 1} \left(x_{i+1}^j - x_i^j, y_{i+1}^j - y_i^j \right). \tag{3}$$

The directional distance, $\widetilde{D}^j$, extracts the horizontal and vertical length of a trajectory and also encodes the direction of motion (moving toward or away from the camera). $\widetilde{D}^j$ helps distinguishing longer trajectories from shorter ones and also trajectories in opposite directions and is calculated as

$$\widetilde{D}^j = \left(x_{nj}^j - x_0^j, y_{nj}^j - y_0^j \right). \tag{4}$$

The trajectory mean, $\widetilde{M}^j$, is another important feature to distinguish trajectories belonging to different regions on the image and is calculated as

$$\widetilde{M}^j = \frac{1}{n^j} \sum_{i=1}^{n_j} \left(x_i^j, y_i^j \right). \tag{5}$$

In order to model the shape of the j^{th} trajectory, irrespective of its length and sample points, polynomial regression can be used. The matrix notation for the model estimation of T^j is

$$\hat{Y}^j = (\, 1 \; X^j \; (X^j)^2 \; \ldots \; (X^j)^\rho \,) \, (\beta_0^j \beta_1^j \ldots \beta_\rho^j)^T + E, \tag{6}$$

where the first term of the R.H.S is a n^jxρ matrix with $X^j = \{x_i^j\}_{i=1}^{n^j}$, the second term is a ρx1 vector and the last term is a n^jx1 vector. The output vector is also of dimension n^jx1. The goal here is to find the optimal values of $\beta = (\beta_0^j, \beta_1^j, \ldots, \beta_\rho^j)$ for which $E = |\hat{Y}^j - Y|$ becomes minimum. The process requires an inherent trade-off between accuracy and efficiency. As the degree of the polynomial increases, the fit grows in accuracy but only up to a point. A recursive procedure is often used to find an appropriate degree.

Finally, in the frequency domain, DFT is used to decompose a trajectory into its sine and cosine components. The output of the transformation represents the trajectory in the frequency domain. In the Fourier domain, each point of a trajectory T^j represents a particular frequency and is calculated as

$$\begin{cases} \widetilde{X}^j = \frac{1}{\sqrt{n^j}} \sum_{k=0}^{n^j - 1} x_k^j e^{-i2\pi ft/n^j} \\ \widetilde{Y}^j = \frac{1}{\sqrt{n^j}} \sum_{k=0}^{n^j - 1} y_k^j e^{-i2\pi ft/n^j} \end{cases}, \tag{7}$$

where i=$\sqrt{-1}$ and $f = 0, 1, \ldots, n-1$.

Figure 3 shows an illustration of the projection of a trajectory dataset into several unsupervised feature spaces. The choice of an appropriate feature space is a critical task for optimal clustering and depends upon the application at hand [20, 14, 7, 18, 21, 22]. However, a more generic solution can be obtained by employing multiple feature spaces that can be used either simultaneously [13, 11] or independently [16].

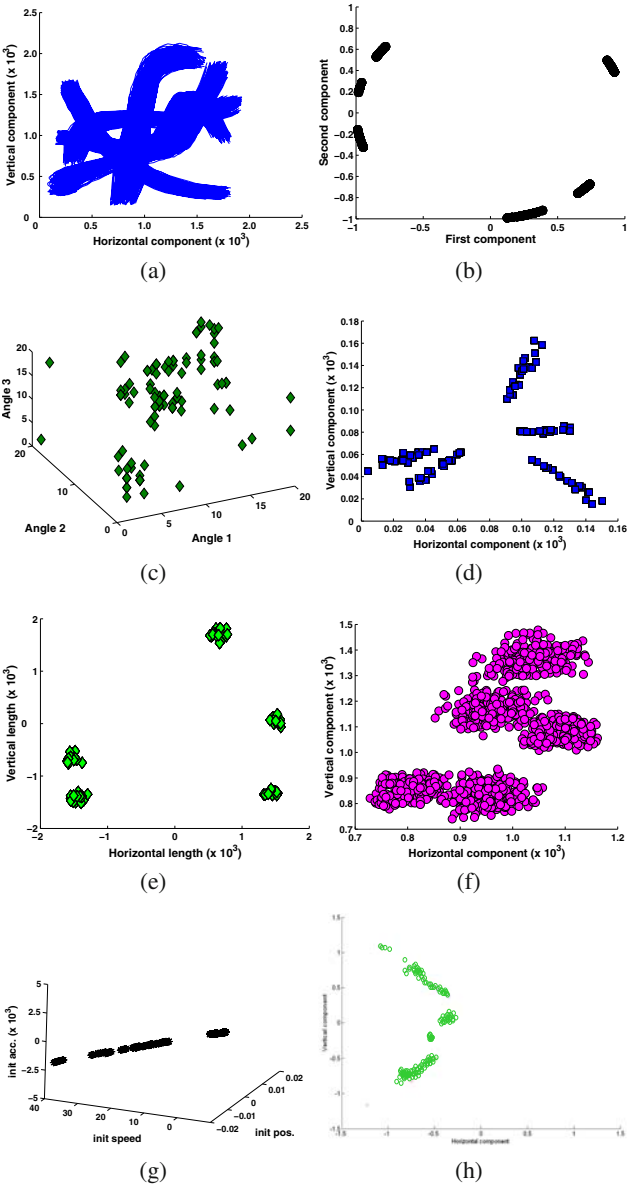


Fig. 3 Sample set of trajectories (a) and their projections on the following feature spaces: (b) principal components; (c) TDH (three dominant angles); (d) average velocity; (e) directional distance; (f) trajectory mean; (g) combination of initial position, speed and acceleration using polynomial coefficients and (h) DFT real coefficients.

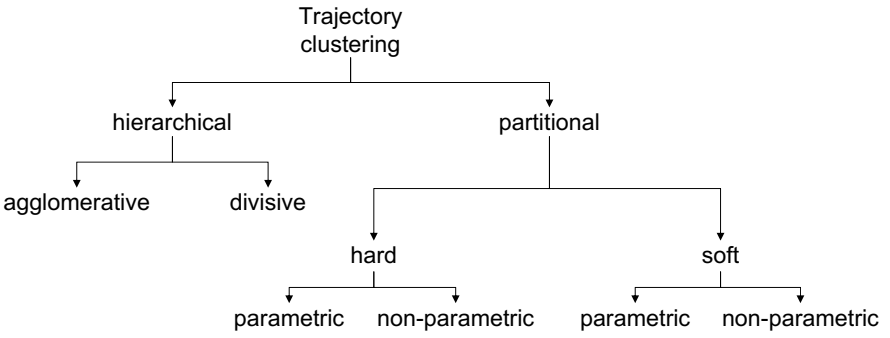


Fig. 4 Classification of trajectory clustering techniques.

3 Trajectory Clustering

After transforming the trajectories into an appropriate feature space, trajectories are grouped together based on a proximity measure (i.e., a similarity or dissimilarity measure). Trajectory clustering techniques can be classified into hierarchical and partitional [23], [24] (Fig. 4).

Hierarchical clustering provides a nested sequence of partitions [22]. These methods can further be divided into two classes, namely agglomerative and divisive. Agglomerative clustering methods start by first considering each trajectory as a separate cluster and then merge the clusters in a nested sequence [25]. On the other hand, divisive clustering starts with all trajectories in one single cluster, and then successively splits the cluster to obtain the final partition [26, 27]. The level of the tree structure depends upon the choice of threshold and is application specific. The computation of the tree structures (dendograms) are expensive and impractical with more than a few hundreds patterns [28].

Partitional clustering is more suitable for the analysis of large data collections as it generate clusters iteratively by minimizing an objective function. Partitional clustering methods can be further divided into two classes, namely hard (crisp) and soft (fuzzy). In hard clustering, each trajectory is assigned to one cluster only; in soft clustering each trajectory is assigned a degree of membership to each cluster. Furthermore, each class can be further divided into parametric and non-parametric sub-classes.

The remainder of this section focuses on these four sub-classes. To show clustering examples, we use a real highway dataset[1] from the MPEG–7, captured at 25 Hz and with resolution of 352 x 288 pixels. This dataset consists of two main clusters, primarily formed from the motion of vehicles moving either toward or away from the camera (Fig. 5(a)).

The following abbreviations are used in the remainder of the chapter: PHC for parametric-hard-clustering methods, NPHC for non-parametric-hard-clustering

[1] http://www.tele.ucl.ac.be/PROJECTS/MODEST/

methods, PSC for parametric-soft-clustering methods and NPSC for non-parametric-soft-clustering methods.

3.1 Hard Clustering Approaches

In PHC, the trajectories are clustered into pre-specified number of partitions with a cluster representative for each cluster. A widely used algorithm is the iterative K-means [29, 30], which works in two steps, namely assignment and update. In the assignment step, each trajectory T^j represented as S^j in a particular feature space is associated to the cluster (C_t^i) with the nearest mean:

$$C_t^i = \{S^j : ||S^j - \mu_t^i|| \leq ||S^j - \mu_t^k||\}, \tag{8}$$

for $k = 1, ..., n_f$, with n_f being the number of clusters, t being the iteration index and μ_t^i being the mean of the cluster C_t^i. In the update step, the mean of the cluster is re-calculated as

$$\mu_{t+1}^i = \frac{1}{|C_t^i|} \sum_{S^j \in C_t^i} S^j \tag{9}$$

The process terminates when either the change in clusters' mean is less than a threshold or the number of iterations reaches a pre-defined value.

Since K-means tends to associate each trajectory to one cluster, outlier trajectories can affect the overall shape of the clusters. To overcome this limitation, a Self-Organizing Maps (SOMs) based approach can be used [21]. Initially, each trajectory T^j can be represented with N coefficients of the Discrete Fourier Transform (DFT) (B^j). In the training phase, using these coefficients the SOM randomly initializes a weight vector W associated to the neuron outputs, which are initially set to $n_m > n_f$. B^j is assigned to an output neuron W^k for which it has minimum Euclidean distance i.e.,

$$c^* = \arg\min_k ||B^j - W_t^k||, \tag{10}$$

where c^* is the index to the output neuron and W_t^k is updated iteratively as

$$W_{t+1}^k = W_t^k + \alpha_t (B - W_t^k), \tag{11}$$

where α_t is the linearly decreasing learning rate. At the end of the training phase, the most similar cluster pairs are merged.

Figure 5(b) shows results of SOM-based approach applied on trajectories represented by 8 DFT-coefficients. The required number of clusters is set to 2. In this approach, successful clustering depends upon the correct choice of the estimated number of clusters and the rate of learning weights before the start of the process, which is not a trivial task.

Dual Hierarchical Dirichlet Processes (Dual-HDP) is an example of NPHC algorithm, which is inspired from document mining [31]. Trajectories are treated as documents and the observations of an object on a trajectory are treated as words in a

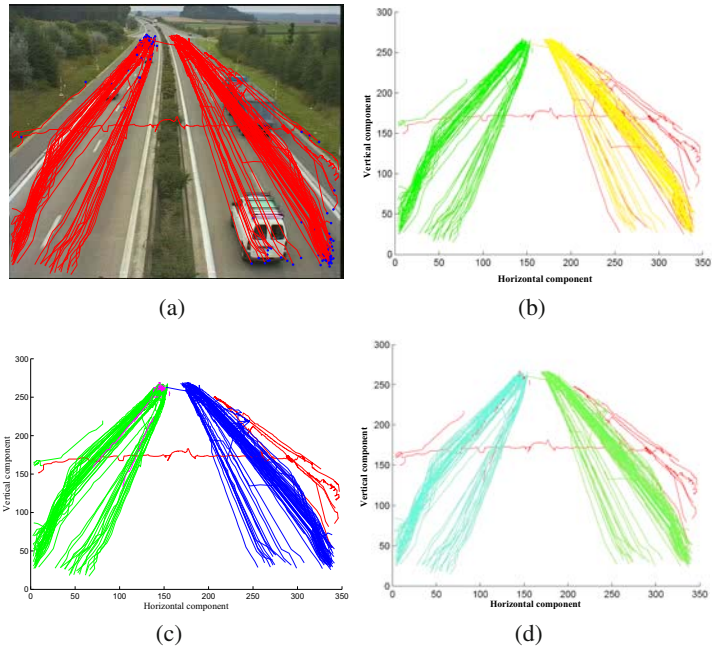


Fig. 5 Examples of trajectory clustering results: (a) input dataset; (b) SOM approach; (c) Dual-DHP approach and (d) proposed approach.

document. Trajectories are clustered in an iterative way. In each iteration the process performs clustering at two levels, namely at observation-level and at trajectory-level. The first level helps in finding the regions and the second level associates each trajectory to one region. An abnormal trajectory is defined as one that does not fall in dense regions. Figure 5(c) shows results of the Dual-HDP.

3.2 Soft Clustering Approaches

Hard partitional clustering methods (PHC and NPHC) work well when the physical boundaries of the clusters are well-defined. An advantage of soft clustering over hard clustering is that it yields more detailed information on the structure of the data as it may assign each element to multiple clusters with an associated membership value [32]. Furthermore, fuzzy clustering is less sensitive to outliers that will have a smaller membership value to a particular cluster and thus affects less on the overall cluster structure.

A popular PSC algorithm is Fuzzy K-means, a variant of the K-means algorithm, which assigns each trajectory a certain degree of belongingness to the clusters. Thus, trajectories on the edge of a cluster would have a smaller degree of membership

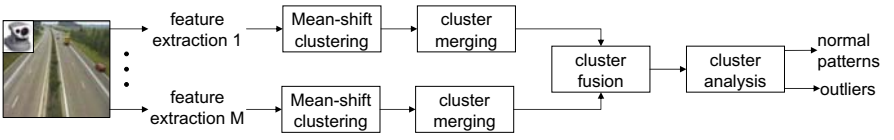


Fig. 6 Flow diagram of the proposed trajectory clustering and outlier detection framework.

than the trajectories in the center of the cluster. For each trajectory T^j we have a coefficient $u_l(T^j)$ giving the degree of belongingness to the l^{th} cluster. The sum of those coefficients for any T^j is 1. Although this algorithm minimizes the intra-cluster variance, it has the same local minimum problem as K-means, and the results depend on the initial choice of the number of clusters.

To overcome the limitations of PHC, NPHC and PSC discussed so far, we present a framework for an NPSC algorithm, where a Mean-shift-based approach operates on multiple feature spaces for clustering, without prior knowledge on the number of clusters. The flow diagram of the proposed approach is shown in Fig. 6.

Let $\mathscr{F}_l(.)$ be a feature extraction function defined as $S_l^j = \mathscr{F}_l(T^j)$, where l= $1,...,M$ and M is the total number of feature spaces. We treat each feature space independently in order to avoid normalization problems. Let us consider each feature space as the empirical probability density function (*pdf*) of the distribution of the trajectories in that particular feature space [33]. Mean-shift finds the modes of the *pdf* and then each trajectory is associated with the nearest mode to form the clusters [34]. Let $\mathbf{S}_l$=$\{S_l^j\}_{j=1}^z$ be a set of z transformed trajectories. The multivariate density estimator $\hat{F}(S_l^j)$ is defined as

$$\hat{F}(S_l^j) = \frac{1}{z_1 h_l} \sum_{i=1}^{z_1} K\left(\left| \frac{S_l^j - S_l^i}{h_l} \right| \right), \tag{12}$$

where z_1 ($\leq z$) is the number of trajectories contained in a hypersphere $\mathscr{S}_l \subseteq \mathbf{S}_l$ with radius h_l and centered at S_l^j. The choice of h_l plays an important role in Mean-shift clustering. A possible way to select this value is to employ an incremental procedure. Initially, by setting h_l to 10% of each dimension of the l^{th} feature space and iteratively increasing it to 80%. The lower bound prevents clusters from containing a single trajectory, while the upper bound avoids a cluster with all trajectories grouped together. Although a smaller h_l produces a less biased density estimator, it increases the variance. In order to find the compromise between the two quantities, the Mean Integrated Squared Error (MISE) [34] can be used

$$MISE(S_l^j) = \int \mathscr{E}(\mathscr{S}_l - \hat{F}(S_l^j))^2 dx. \tag{13}$$

The value of h_l for which $MISE(S_l^j)$ is minimum is considered to be optimal. Moreover, $K(.)$ in Eq. 12 is defined as

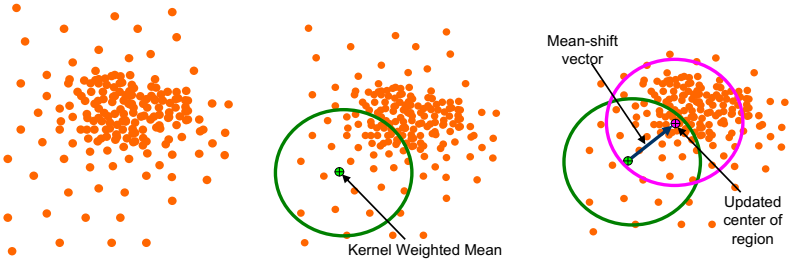


Fig. 7 An illustration of the Mean-shift process: (a) input trajectories in a feature space; (b) kernel weighted mean around a seed trajectory and (c) update of the mean in the direction of a dense region.

$$K(k) = \begin{cases} \frac{1}{2V_l}(d_l)(1-k^T k) & \text{if } k^T k < 1 \\ 0 & \text{otherwise} \end{cases}, \tag{14}$$

where $k = \left| \frac{S_l^j - S_l^i}{h_l} \right|$ and V_l represents the volume of $\mathscr{S}_l$ with d_l dimensions.

The density gradient estimate of the kernel can be written as

$$\hat{\nabla} F(S_l^j) = \nabla \hat{F}(S_l^j) = \frac{1}{z_1 h_l} \sum_{i=1}^{z_1} \nabla K \left(\left| \frac{S_l^j - S_l^i}{h_l} \right| \right). \tag{15}$$

From Eq. 12 and Eq. 15, the Mean-shift vector $M_h(S_l^j)$ is defined as

$$M_h(S_l^j) \propto \frac{\hat{\nabla} F(S_l^j)}{\hat{F}(S_l^j)}, \tag{16}$$

or

$$M_h(S_l^j) = \frac{h_l V_l}{d_l} \frac{\hat{\nabla} F(S_l^j)}{\hat{F}(S_l^j)}. \tag{17}$$

The output of the Mean-shift procedure is the set of trajectories associated to each mode. Initially, the mode seeking process starts by fixing a trajectory as a seed point. Then, after the Mean-shift process converges to the local mode, all the trajectories within the bandwidth, h_l, of the kernel, $K(.)$, are assigned to that mode [35] (Fig. 7). These trajectories are not considered for future iterations. The next seed point is selected randomly from the unprocessed trajectories. The process terminates when all trajectories are assigned to a corresponding local mode. Finally, adjacent clusters are merged if their modes are apart by less than a pre-defined threshold (Fig. 8) [36].

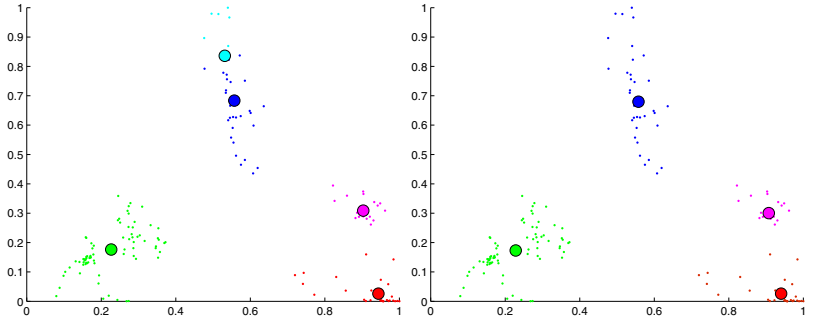


Fig. 8 Sample cluster merging results. (a) Initial trajectory clustering result (5 clusters); (b) final clustering result after cluster merging (4 clusters).

4 Cluster Fusion

The final partitioning of the trajectories is obtained by integration after analyzing the clustering results from each feature space. The integration of the clusters consists of three steps, namely the estimation of the final number of clusters, the establishment of the correspondence between clusters in different feature spaces, and the association of each trajectory to a final cluster.

Let $N = \{n_1, n_2, ..., n_m\}$ be the set containing the number of clusters for each feature space. A possible choice is to set the final number of clusters n_f as the median value of the set N. After selecting the final number of clusters, the structure of clusters are estimated as characterized by a single mode; each cluster is modeled with a univariate Gaussian with bandwidth of the kernel defining the variance of the cluster itself.

In order to find the structure of each cluster, the process starts with a feature space $F_l \in F : F = \{F_i\}_{i=1}^{m}$ such that $n_l = n_f$. The initial parameters of the final clusters $(C_i^f; i = 1, ..., n_f)$ are those defined by F_l. To refine the parameters according to the results of the other feature spaces, we find the correspondence of F_l with all the other feature spaces F_n with $F_n \in \{F_k\}_{k=1}^{m}$ and $n \neq l$.

Let $\hat{v}$ be the index of the cluster in F_n that has the maximum correspondence (maximum number of overlapping elements) with the i^{th} cluster of F_l:

$$\hat{v} = \arg\max_j \left| C_i^l \cap C_j^n \right|, \tag{18}$$

where $|.|$ is the cardinality of the set of overlapping elements, $i = 1, ..., n_f$, $j = 1, ..., n_n$, C_i^l and C_j^n represent the i^{th} and j^{th} clusters in F_l and F_n, respectively.

C_i^f is updated by taking the overlapping elements, $C_i^f = (C_i^f \cap C_{\hat{v}}^n)$. This process continues for all features spaces. This results in n_f clusters consisting of all the trajectories that are consistent across all feature spaces, and are therefore considered

to represent a reliable structure for each cluster. At this point we associate to the final clusters the trajectories ($T^{'} \subseteq \{T^j\}_{j=1}^{z}$) that are not consistent across all the feature spaces. To this end, we calculate a conditional probability of $T_o \in T^{'}$ generated from the given cluster model as

$$p(T_o|C_k^f) = \frac{1}{m}\sum_{i=1}^{m}\frac{1}{\sqrt{2\pi}\sigma_{f,k}}e^{-(\frac{\mu_{i,j}-\mu_{f,k}}{\sigma_{f,k}})^2}, \tag{19}$$

where $\mu_{i,j} = \frac{1}{|C_j^i|}\sum_{j=1}^{|C_j^i|}\bar{T}^j$ and $\mu_{f,k} = \frac{1}{|C_k^f|}\sum_{k=1}^{|C_k^f|}\bar{T}^j$ are the mean values of C_j^i and C_k^f respectively and $\sigma_{f,k}$ is the standard deviation of C_k^f. The motivation behind these parameters is that each cluster can be characterized by a single mode with the spread of the cluster represented by σ. Note that the decision is made at the cluster level and not at the feature space level, thus removing the dependence on dimensionality or normalization. T_o will be assigned to C_k^f if

$$p(T_o|C_k^f) > p(T_o|C_l^f), \tag{20}$$

where l=1, ..., n_f and $l \neq k$.

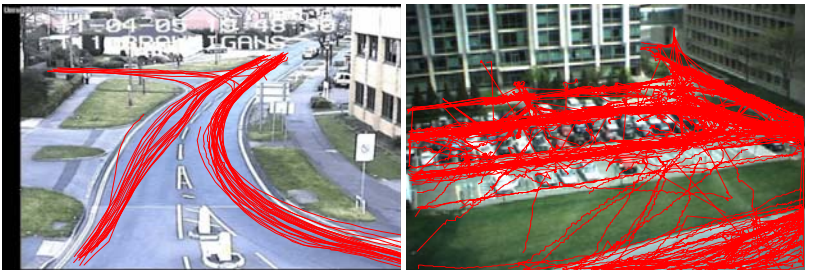
5 Outlier Detection

An outlier trajectory deviates from other trajectories as result of an abnormal event. In the anomaly detection literature, distance-based techniques are frequently used. Zhou *et al.* [37] use an Edit distance to find the common pattern. A trajectory that is far from the common pattern is considered as an anomaly. Similarly, Naftel *et al.* [21] use the Hotelling T^2 test to determine if the Mahalanobis distance of a trajectory to its nearest class center makes it an outlier. The choice of the threshold value is selected according to the given dataset. Fu *et al.* [22] use a Gaussian distribution to represent the test and template trajectories. If the difference between a test trajectory and a template trajectory is larger than one standard deviation from the mean of a template trajectory, then the test trajectory is considered as abnormal.

In this work we focus on identifying two types of outlier trajectories: (a) those existing in dense regions, but exhibiting a different behavior from the common pattern; and (b) those located in sparse regions. A trajectory $T_j^{'}$ belongs to the *first type of outliers* if $T_j^{'} \in C_k^f$ lies at least twice of the standard deviation from the center of cluster, $\mu_{f,k}$, i.e.,:

$$|\mu_{T_o} - \mu_{f,k}| > 2\sigma_{f,k}. \tag{21}$$

To detect the *second type of outliers*, we identify trajectories belonging to sparse regions by considering the size of their cluster. If a cluster has few associated trajectories and cannot be merged with a nearby cluster, then it is considered to be a set of outliers. Here the threshold value is set to the 10% of the cardinality of the cluster containing the median number of associated elements. Figure 5(d) shows an



(a) (b)

Fig. 9 Standard datasets used for performance evaluation.

example of the clustering results using the proposed approach. The results show the validity of the process in identifying the accurate number of clusters along with the detection of outliers.

6 Performance Evaluation

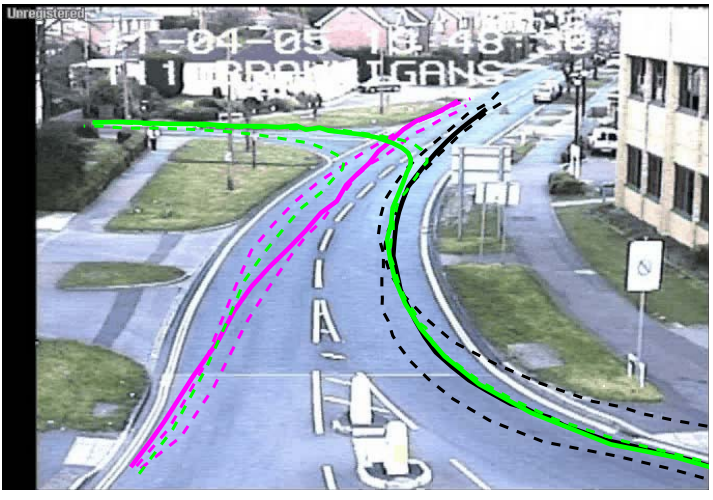
We evaluate the proposed approach based on the following independent trajectory representations: the first two components of PCA, the three dominant angles using TDH, the average velocity, the directional distance, the trajectory mean and a combination of initial position, speed and acceleration. We compare it with state-of-the-art trajectory clustering algorithms on the following real world sequences datasets: $S1$ is a traffic monitoring sequence, which consists of 47 trajectories [38], with the resolution of 720x480 pixels (25 Hz); $S2$ is a parking monitoring sequence, which consists of 535 trajectories [31] with the resolution of 360x480 pixels (25 Hz). Figure 9 shows the cumulated trajectories superimposed on a key-frame of each test sequence. For $S1$ we are interested in finding 3 clusters. For $S2$ we are interested in finding 7 clusters.

Figure 10(a) shows that the proposed approach has successfully clusters the trajectories in 3 main motion patterns. Each cluster is summarized by the representative trajectory (in bold line) and couple of boundary trajectories (in bold dash-line). The representative trajectory u^* is calculated as

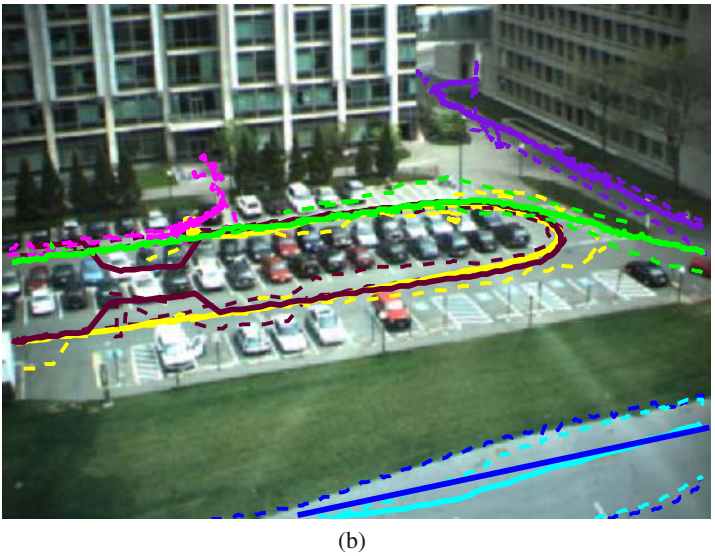
$$u^* = \arg\min_j \frac{|\mu_{T_j} - \mu_{f,k}|}{\sigma_{f,k}}, \tag{22}$$

where j is the index of all the trajectories in cluster C_k^f. Furthermore, the boundary trajectories are the top-two trajectories that are far from the center of the cluster and are calculated as

$$v^* = \arg\max_j \frac{|\mu_{T_j} - \mu_{f,k}|}{\sigma_{f,k}}, \tag{23}$$



(a)



(b)

Fig. 10 Results of proposed clustering approach on (a) $S1$ and (b) $S2$. The semantic regions are summarized by a representative trajectory (solid line) and a pair of boundary trajectories (dash line).

where j is the index of all trajectories belonging to a cluster C_k^f in two iterations. In the second iteration the already selected trajectory is not considered anymore. Figure 11(a) shows the outlier trajectories found in the sequence, which are primarily the shorter and farther trajectories compared to the rest of the trajectories in a cluster. Moreover, Figure 10(b) shows that the proposed approach has successfully

(a) (b)

Fig. 11 Outlier detected on (a) $S1$ and (b) $S2$ using the proposed framework.

Table 1 Comparison of clustering results based on Precision and Recall figures.

	$S1$						$S2$					
	SOM		Dual-HDP		Proposed		SOM		Dual-HDP		Proposed	
ID	P	R	P	R	P	R	P	R	P	R	P	R
C_1^f	0.70	1.00	0.82	1.00	0.87	1.00	0.65	0.70	0.84	0.89	0.86	0.88
C_2^f	0.87	0.65	0.79	1.00	0.84	1.00	0.73	0.71	0.86	0.90	0.86	0.95
C_3^f	0.35	0.12	0.90	0.42	1.00	0.40	0.72	0.69	0.86	0.97	0.87	0.93
C_4^f	-	-	-	-	-	-	0.77	0.68	0.89	0.94	0.92	0.91
C_5^f	-	-	-	-	-	-	0.74	0.73	0.81	0.85	0.85	0.90
C_6^f	-	-	-	-	-	-	0.72	0.74	0.79	0.84	0.91	0.85
C_7^f	-	-	-	-	-	-	0.78	0.68	0.75	0.93	0.90	0.93
Avg.	**0.64**	**0.59**	**0.84**	**0.81**	**0.90**	**0.80**	**0.73**	**0.70**	**0.83**	**0.90**	**0.88**	**0.91**

clusters the trajectories in 7 main motion patterns. Figure 11(b) shows the outlier trajectories found in the sequence, which are essentially the shorter trajectories or trajectories formed by people crossing the grass.

To objectively assess the clustering performance of the proposed approach, we use *Precision* (P_i) and *Recall* (R_i). For the i^{th} final cluster, C_i^f, P is calculated as

$$P_i = \frac{|C_i^f \cap \Gamma_i|}{|C_i^f|}, \tag{24}$$

and R_i as

$$R_i = \frac{|C_i^f \cap \Gamma_i|}{|\Gamma_i|}, \tag{25}$$

where $|.|$ is the cardinality of a cluster and Γ_i is ground-truth of the i^{th} cluster. We also compare the proposed approach with a PHC and a NPHC approaches that are based on SOM and Dual-HDP, respectively. The comparison of the three approaches is summarized in Table 1. The proposed approach has outperformed in terms of R the SOM-based approach on average by 21% for $S1$. However, the Dual-HDP approach

has better performance by 1% than the proposed framework. On the other hand, , in terms of P the proposed approach has 26% and 6% better performance than the SOM and Dual-DHP approaches, respectively. Similarly, the proposed approach has outperformed in terms of R the SOM-based and Dual-HDP approaches on average by 21% and 1%, respectively, for $S2$. Furthermore, the proposed approach has 15% and 5% better performance in terms of P than the SOM and Dual-DHP approaches respectively.

7 Conclusions

We presented a framework for scene context learning and outlier detection by clustering video object trajectories. The input trajectories are transformed into distinct feature spaces to represent the complementary characteristics of motion patterns. Several supervised and unsupervised feature spaces have been described in this chapter and their relative merits discussed. Next, Mean-shift was used to estimate clusters in each feature space and adjacent clusters in a feature space were merged to refine the initial results. A fuzzy membership of a trajectory to the final clusters was estimated, and crisp clusters were then obtained based on the maximum membership using the information from all the feature spaces. Finally, the clusters with only few associated trajectories and trajectories far from the clusters center were considered outliers. The proposed approach was validated on standard datasets and compared with state-of-the-art approaches.

References

1. Morris, B.T., Trivedi, M.M.: A survey of vision-based trajectory learning and analysis for surveillance. IEEE Trans. on Circuits and Systems for Video Technology 18(8), 1114–1127 (2008)
2. Brand, M., Oliver, N., Pentland, A.: Coupled hidden markov models for complex action recognition. In: Proc. of IEEE Int'l. Conf. on Computer Vision and Pattern Recognition (CVPR), San Juan, Puerto Rico (June 1997)
3. Porikli, F.: Trajectory pattern detection by HMM parameter space features and eigenvector clustering. In: Proc. of 8^{th} European Conf. on Computer Vision (ECCV), Prague, Czech Republic (May 2004)
4. Wilson, A.D., Bobick, A.F.: Recognition and interpretation of parametric gesture. In: Proc. of IEEE 6^{th} Intl. Conf. on Computer Vision, Bombay, India (January 1998)
5. Oliver, N., Rosario, B., Pentland, A.: A bayesian computer vision system for modeling human interactions. IEEE Trans. on Pattern Analysis and Machine Intelligence 22(8), 831–843 (2000)
6. Chudova, D., Gaffney, S., Smyth, P.: Probabilistic models for joint clustering and time warping of multi-dimensional curves. In: Proc. of 19^{th} Conf. on Uncertainty in Artificial Intelligence, Acapulco, Mexico (August 2003)
7. Gaffney, S., Smyth, P.: Trajectory clustering with mixtures of regression models. In: Proc. of Intl. Conf. on Knowledge Discovery and Data Mining, San Diego, CA, USA (August 1999)

8. Ma, X., Schonfeld, D., Khokhar, A.: Video event classification and image segmentation based on non-causal multi-dimensional hidden markov models. IEEE Trans. on Image Processing 18(6), 1304–1313 (2009)
9. Alon, J., Sclaroff, S., Kollios, G., Pavlovic, V.: Discovering clusters in motion time-series data. In: Proc. of IEEE Conf. on Computer Vision and Pattern Recognition (CVPR), Madison, Wisconsin (June 2003)
10. Song, G., Martynovich, P.: A study of hmm-based bandwidth extension of speech signals. Elsevier Journal of Signal Processing 89(10), 2036–2044 (2009)
11. Bashir, F., Khokhar, A., Schonfeld, D.: Real-time motion trajectory-based indexing and retrieval of video sequences. IEEE Trans. on Multimedia 9(1), 58–65 (2007)
12. Bashir, F., Khokhar, A., Schonfeld, D.: Segmented trajectory based indexing and retrieval of video data. In: Proc. of Intl. Conf. on Image Processing (ICIP), Barcelona, Catalonia, Spain (September 2003)
13. Li, X., Hu, W., Hu, W.: A coarse-to-fine strategy for vehicle motion trajectory clustering. In: Proc. of Intl. Conf. on Pattern Recognition (ICPR), Hong Kong, China (August 2006)
14. Antonini, G., Thiran, J.: Counting pedestrians in video sequences using trajectory clustering. IEEE Trans. on Circuit and Systems for Video Technology 16(8), 1008–1020 (2006)
15. Hu, W., Xie, D., Tan, T., Maybank, S.: Learning activity patterns using fuzzy self-organizing neural network. IEEE Trans. on Systems, Man and Cybernetics, Part B 34(3), 334–352 (2004)
16. Anjum, N., Cavallaro, A.: Multi-feature object trajectory clustering for video analysis. IEEE Trans. on Circuits and Systems for Video Technology 18(11), 1555–1564 (2008)
17. Sumpter, N., Bulpitt, A.J.: Learning spatio-temporal patterns for predicting object behaviour. In: Proc. of British Conf. on Machine Vision, Southampton, UK (September 1998)
18. Khalid, S., Naftel, A.: Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. In: Proc. of third ACM Intl. workshop on Video Surveillance and Sensor Networks, Singapore (September 2005)
19. Bashir, F., Khokhar, A., Schonfeld, D.: Object trajectory-based activity classification and recognition using hidden markov models. IEEE Trans. on Image Processing 16(7), 1912–1919 (2007)
20. Amasyali, F., Albayrak, S.: Fuzzy c-means clustering on medical diagnostic systems. In: Proc. of Intl. Twelfth Turkish Symp. on Artificial Intelligence and Neural Networks (TAINN 2003), Canakkale, Turkey (July 2003)
21. Naftel, A., Khalid, S.: Classifying spatiotemporal object trajectories using unsupervised learning in the coefficient feature space. Trans. on Multimedia Systems 12(3), 227–238 (2006)
22. Fu, Z., Hu, W., Tan, T.: Similarity based vehicle trajectory clustering and anomaly detection. In: Proc. of IEEE Intl. Conf. on Image Processing, ICIP, Genova, Italy (September 2005)
23. Frigui, H., Krishnapuram, R.: A robust competitive clustering algorithm with applications in computer vision. IEEE Trans. on Pattern Analysis and Machine Intelligence 21(5), 450–465 (1999)
24. Wilson, H., Boots, B., Millward, A.: A comparison of hierarchical and partitional clustering techniques for multispectral image classification. In: Proc. of IEEE Intl. Geoscience and Remote Sensing Symp (IGARSS), Toronto, Canada (June 2002)
25. Buzan, D., Sclaroff, S., Kollios, G.: Extraction and clustering of motion trajectories in video. In: Proc. of IEEE Intl. Conf. on Pattern Recognition (ICPR), Cambridge, UK (August 2004)

26. Li, M., Wu, C., Han, Z., Yue, Y.: A hierarchical clustering method for attribute discretization in rough set theory. In: Proc. of Intl. Conf. on Machine Learning and Cybernetics, Shanghai, China (August 2004)
27. Biliotti, D., Antonini, G., Thiran, J.: Multi-layer hierarchical clustering of pedestrian trajectories for automatic counting of people in video sequences. In: Proc. of IEEE Workshop on Motion and Video Computing, Colorado, USA (January 2005)
28. Jain, A.K., Murty, M.N., Flynn, P.: Data clustering: A review. ACM Computing Surveys 31(3), 264–323 (1999)
29. Melo, J., Naftel, A., Bernardino, A., Victor, J.: Retrieval of vehicle trajectories and estimation of lane geometry using non-stationary traffic surveillance cameras. In: Proc. of Advanced Concepts for Intelligent Vision Systems (ACIVS), Brussels, Belgium (August 2004)
30. Seber, G.A.F.: Multivariate Observations. Wiley, New York (1984)
31. Wang, X., Ma, K.T., Ng, G.W., Grimson, W.E.L.: Trajectory analysis and semantic region modeling using a nonparametric bayesian model. In: Proc. of IEEE Intl. Conf. on Computer Vision and Pattern Recognition, Anchorage, Alaska, USA (June 2008)
32. Kwok, T., Smith, R., Lozano, S., Taniar, D.: Parallel fuzzy c-means clustering for large data sets. In: Proc. of 8th Intl. Euro-Par Conf. on Parallel Processing, Paderborn, Germany (August 2002)
33. Comaniciu, D., Meer, P.: Mean shift: a robust approach toward feature space analysis. IEEE Trans. on Pattern Analysis and Machine Intelligence 24(5), 603 (2002)
34. Comaniciu, D., Meer, P.: Distribution free decomposition of multivariate data. IEEE Trans. on Pattern Analanlysis and Applications 2(1), 22–30 (1999)
35. Comaniciu, D., Ramesh, V., Meer, P.: The variable bandwidth mean shift and data-driven scale selection. In: Proc. of IEEE Int'l. Conf. on Computer Vision (ICCV), Vancouver, Canada (July 2001)
36. Wang, H., Suter, D.: False-peaks-avoiding mean shift method for unsupervised peak-valley sliding image segmentation. In: Proc. of Intl. Conf. on Digital Image Computing: Techniques and Applications (DICTA), Sydney, Australia (December 2003)
37. Zhou, Y., Yan, S., Huang, T.: Detecting anomaly in videos from trajectory similarity analysis. In: Proc. of Intl. Conf. on Multimedia and Expo., Beijing, China (July 2007)
38. Kasturi, R.: Performance evaluation protocol for face, person and vehicle detection & tracking in video analysis and content extraction, vace-ii (January 2006)

Motion Trajectory-Based Video Retrieval, Classification, and Summarization

Xiang Ma, Xu Chen, Ashfaq Khokhar, and Dan Schonfeld

Abstract. This chapter provides an overview of various methods for motion trajectory-based video content modeling, retrieval and classification. The techniques discussed form the foundation for content-based video indexing and retrieval (CBVIR) systems. We focus on view-invariant representations of single and multiple motion trajectories based on null-space invariants that allows for video retrieval and classification from unknown and moving camera views. We introduce methods based on matrix and tensor decomposition for efficient storage and retrieval of single and multiple motion trajectories, respectively. We subsequently explore the use of one- and multi-dimensional hidden Markov models for video classification and recognition based on single and multiple motion trajectories. We summarize the basic concepts and present computer simulation results to demonstrate the fundamental notions introduced throughout the chapter. We finally discuss several open problems in the field of motion trajectory analysis and future trends in content-based video modeling, retrieval and classification.

1 Video Content Modeling

1.1 Video Structure

A *video* consists of multiple scenes. Each *scene* is composed of contiguous shots that share a semantic theme. A *shot* is formed by successive frames acquired by a continuous recording from a single camera. The shot is regarded as the fundamental building block of the video sequence. Given the significance of shots in videos, extraction of shots, or shot boundary detection, is a key step in video analysis.

1.2 Video Segmentation

We use the term *video segmentation* to refer to the problem of *shot boundary detection*. The aim of video segmentation, or shot boundary detection, is to identify

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 53–82.
springerlink.com

the frames in the video sequence in which a transition between shots has occurred. The simplest example of transition between video shots is known as a cut. A *cut* represents a sharp transition between shots that occur when the camera recording is suddenly terminated and later restarted. Most shot boundary detection methods are designed for identification of cuts and rely on correlation between adjacent frames [1][2][3]. However, other forms of gradual transitions between video shots are often introduced in the video sequence by use of special effects (e.g. fades, dissolves, etc.). Numerous techniques have been developed to address different potential special effects introduced through video editing. Lelescu and Schonfeld [4] have developed a unified method for video segmentation that addresses both abrupt and gradual shot transitions by relying on sequential hypothesis testing.

1.3 Video Tracking

A key feature in video analysis is provided by object motion [5]. Motion can be captured by extraction of the object's movement within shots using video tracking algorithms. Numerous methods have been developed over the past few decades for tracking of single and multiple objects [6][7][8]. Among the most popular video tracking techniques are block-based methods [9][10] and tracking based on particle filters [11]. Qu et al. [12] have developed an efficient, distributed framework for multiple object tracking based on particle filtering. Video tracking algorithms can be used to characterize the object's shape, orientation, and position in each frame.

1.4 Motion Trajectories

A collection of the object's coordinates over time is referred to as a *motion trajectory*. The coordinates selected for representation of motion trajectories often represent the centroid, or other salient features of the object. The importance of motion trajectories as a critical feature for activity modeling has been established in video analysis applications [13][14][15][16]. Video sequences consisting of multiple moving objects lead to the representation of multiple simultaneous motion trajectories.

An object motion trajectory in a video shot of length L is usually represented by an L-tuple formed by a collection of the x- and y-axes of the object's centroid in a two-dimensional coordinate system at each time instant; i.e.

$$r^L = \{x[t], y[t]\}, t = 1, ..., L. \tag{1}$$

In some cases, motion trajectories are represented by a three-dimensional coordinate system for applications in which the depth of the object from the camera is important.

Typically, for a video clip of length L with M moving objects, the multiple motion trajectories are represented as a set $\mathscr{S}$ of M motion trajectories given by

$$\mathscr{S} = \{r_1^L, r_2^L, r_3^L, ..., r_M^L\}. \tag{2}$$

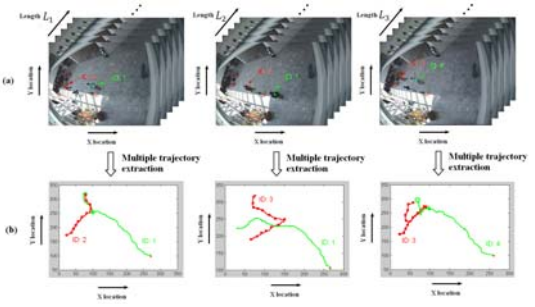


Fig. 1 Multiple motion trajectories: (a) snapshots extracted from three video sequences; and (b) the corresponding multiple motion trajectories.

Figure 1 illustrates the representation of multiple motion trajectories. Snapshots extracted from three video clips are shown in Fig. 1(a) and the corresponding multiple motion trajectories are depicted in Fig. 1(b).

2 View-Invariant Representation

View-Invariant representation is a very important and sometimes difficult aspect of an intelligent system. The problem with this is that the system can only recognize a predefined set of behaviors [17] [18] [19]. In this section, the state-of-the-art approaches in motion trajectory-based video content modelling and representation are represented. These include: curvature scale space (CSS) and centroid distance functions (CDF)-based representations. The null space invariant (NSI) representations for video classification and retrieval due to camera motions are further discussed. Moreover, the tensor null space invariant (TNSI) representation for high dimensional data is presented in the last part of this section.

2.1 Curvature Scale-Space (CSS) and Centroid Distance Function (CDF)-Based Representation

As described in [20], scale-space is a multi-resolution technique used to represent data of arbitrary dimension without any knowledge of noise level and preferred scale (smoothness). The curvature $\kappa[k]$ for the trajectory represented as in (1) can be expressed as:

$$\kappa[k] = \frac{x'[k]y''[k] - y'[k]x''[k]}{\{x'[k]^2 + y'[k]^2\}^{3/2}} \tag{3}$$

The curvature of a trajectory has several desirable computational and perceptual characteristics. One such property is that it is invariant under planar rotation and translation of the curve. Curvature is computed from dot and cross products of parametric derivatives and these are purely local quantities, hence independent of

rotations and translations. The dot and cross products are based only on the lengths, and angles between, vectors. Hence, these are also independent of rigid transformations of the coordinate system. Given a trajectory as in Eq.(1), the evolved version of the trajectory in terms of scale-space is defined by:

$$r_\sigma[k] = X[k,\sigma], Y[k,\sigma] \ , \tag{4}$$

where

$$X[k;\sigma] = x[k] \otimes g[k;\sigma], \tag{5}$$
$$Y[k;\sigma] = y[k] \otimes g[k;\sigma] \tag{6}$$

with $g[k;\sigma]$ being the symmetric Gaussian kernel used for smoothing. At each level of scale, governed by the increasing standard deviation of Gaussian kernel, curvature of the evolved trajectory is computed. Then the function implicitly defined by

$$\kappa[k;\sigma] = 0 \tag{7}$$

is the curvature scale space image of the trajectory. It is defined as a binary image with a value of 1 assigned to the points of curvature zero crossing at each scale level. Note that each of the arch-shaped contours in the CSS image corresponds to a convexity or concavity on the original trajectory with the size of the arch being proportional to the size of the corresponding feature. The CSS image is a very robust representation under the presence of noise in trajectory data due to small camera motions and minor jitters in tracking. Noise amplifies only the small peaks, with no effect on the location or scale of the feature contour maxima. As evident from the figure, the major peaks of the CSS image remain quite preserved after significant affine transformation.

The centroid distance function is another invariant representation of the raw shape data used in the affine invariant image retrieval applications. The centroid distance function is expressed by the distance of each point in trajectory from the centroid of the trajectory:

$$c[k] = \sqrt{[x[k]-x_c]^2 + [y[k]-y_c]^2}, k = 0,1,\ldots,N-1 \tag{8}$$

where $x_c = \frac{1}{N}\sum_{t=o}^{N-1} x[k]$,$y_c = \frac{1}{N}\sum_{t=o}^{N-1} y[k]$. Let us denote by uniform affine transformation the set of affine transforms including translation, rotation and uniform scaling. This excludes the shear transformation in general affine transformation. Let us denote the centroid distance function of a trajectory before affine transformation as $C[k]$ and after uniform affine transformation as $C^{'}[k]$. Then it can be easily proved that under uniform affine transformation, the following relation holds between the centroid distance functions computed from original and affined version of the trajectory:

$$C'[k] = \alpha C[k],$$

$$\alpha = 1, for \begin{pmatrix} u[k] \\ v[k] \end{pmatrix} = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \begin{pmatrix} x[k] \\ y[k] \end{pmatrix} + \begin{pmatrix} \beta \\ \gamma \end{pmatrix}$$

$$\alpha = \alpha_s, for \begin{pmatrix} u[k] \\ v[k] \end{pmatrix} = \alpha_s \begin{pmatrix} x[k] \\ y[k] \end{pmatrix}. \tag{9}$$

Figure 2 displays one of the trajectories from ASL dataset along with two of its rotated versions, and its representation in terms of the two feature spaces explored in this presentation. As seen from this figure, the CDF-based representation is absolutely invariant to rotational deformations. The CSS-based representation, on the other hand, results in the curvature zero crossings being consistent but the CSS maxima tend to shift around. In the case of CSS representation, as outlined previously, the trajectory is represented by the locations of CSS maxima in terms of their temporal ordering and the scale of concavity. In this context, the trajectory data is represented by a time-ordered sequence of two-dimensional feature vectors containing CSS maxima. Results on a range of dataset sizes from the ASL [21] dataset are reported, in terms of the ROC curves [22] in Figure 3. The ROC curves are two-dimensional depiction of classifier performance.

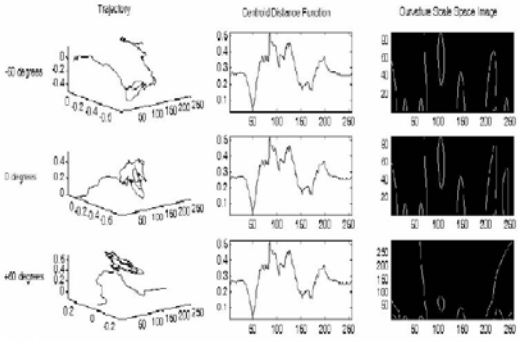


Fig. 2 Trajectory of the hand motion for signing the word 'Alive' in Australian Sign Language with its two rotated versions, and their corresponding representations using centroid distance functions (CDFs) and curvature scale-space (CSS) images.

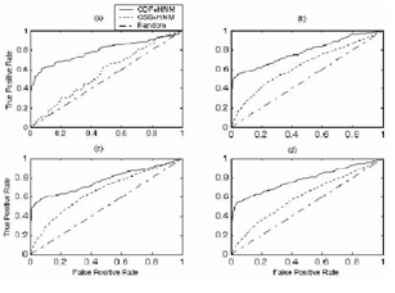


Fig. 3 ROC curves for the rotated trajectories posed for classification.

2.2 Null-Space Invariance (NSI) Representation

Null Space Invariant (NSI) of a trajectories matrix (each row in the matrix corresponds to the positions of a single object over time) is introduced as a new and powerful affine invariant space to be used for trajectory representation. This invariant, which is a linear subspace of a particular vector space, is the most natural invariant and is definitely more general and more robust than the familiar numerical invariants. It does not need any assumptions and after invariant calculations it conserves all the information of original raw data.

Let $Q_i = (x_i, y_i)$ be a single 2-D point, $i = 0, 1, \ldots, n-1$, among n ordered nonlinear points in R^2, representing a trajectory. Consider the following arrangement of the n 2-D points in a $3 \times n$ matrix M:

$$M = \begin{pmatrix} x_0 & x_1 & \ldots & x_{n-1} \\ y_0 & y_1 & \ldots & y_{n-1} \\ 1 & 1 & \ldots & 1 \end{pmatrix} \tag{10}$$

(n-3)-dimensional linear subspace H^{n-3} can be associated to a 2-D trajectory whose features set is $Q_0, Q_1, \ldots, Q_{n-1}$:

$$H^{n-3} = \{q = (q_0, q_1, \ldots, q_{n-1})^T, i.e. Mq = (0,0,0)^T\} \tag{11}$$

Since at least one determinant of 3×3 minor of M is not zero because of nonlinear feature points, H^{n-3} has a dimension of n-3. The attractive property of the linear subspace is that it does not change when it undergoes any of the affine transformations. The new invariant of the trajectory matrix M is used to represent each trajectory. Moreover,

$$H^{n-3} \subset R^{n-1} = \{q = (q_0, q_1, \ldots, q_{n-1})^T \in R^{n-1}, and \sum_{i=0}^{n-1} q_i = 0\} \tag{12}$$

which produces (n-3)-planes in (n-1)-space, $Gr_R(n-3, n-1)$. $Gr_R(n-3, n-1)$ is a well understood manifold of dimension 2n-6, which is the number of invariants associated to the matrix M. H^{n-3} is spanned by the vectors $v_i = (q_0^i, q_1^i, \ldots, q_{n-1}^i)^T, i = 3, 4, \ldots, n-1$, where

$$q_0^i = -det \begin{pmatrix} x_1 & x_2 & x_i \\ y_1 & y_2 & y_i \\ 1 & 1 & 1 \end{pmatrix} / det \begin{pmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{pmatrix} \tag{13}$$

$$q_1^i = det \begin{pmatrix} x_0 & x_2 & x_i \\ y_0 & y_2 & y_i \\ 1 & 1 & 1 \end{pmatrix} / det \begin{pmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{pmatrix} \tag{14}$$

$$q_2^i = -det \begin{pmatrix} x_0 & x_1 & x_i \\ y_0 & y_1 & y_i \\ 1 & 1 & 1 \end{pmatrix} / det \begin{pmatrix} x_0 & x_1 & x_2 \\ y_0 & y_1 & y_2 \\ 1 & 1 & 1 \end{pmatrix} \tag{15}$$

$$q_i^i = -1 and q_i^j = 0 for j = 3,4,\ldots,i-1,i+1,\ldots,n-1 \tag{16}$$

Now each trajectory is represented now with a matrix $NSI_{n\times(n-3)}$, which is composed of v_i columns, from the trajectory matrix $M_{3\times n}$. A method for dimensionality-reduction and classification based on PCNSA [23] is chosen for the null space operator. For noisy data, as described in [24] [25], based on the perturbed null operator, it is desirable to know the ratio of the input error and the output error where the input error is referred to the error of the trajectory matrix and the output error is referred to the error of the null operator. Therefore, the ratio of the output error and input error is:

$$\tau = \frac{E\|Q-\tilde{Q}\|_F^2}{E\|Z\|_F^2} = \frac{1}{N}[(N-3)(r_{01}^2+r_{02}^2+r_{12}^2)+\sum_{i=3}^{N-1}(r_{0i}^2+r_{1i}^2+r_{2i}^2)] \ . \tag{17}$$

It can be seen that the ratio only relies on the trajectory itself while independent of the noise. Defining the power of the output signal as $\|Q\|_F^2$, SNR can be computed by $\Delta_{SNR} = \frac{\|Q\|_F^2}{E\|Q-\tilde{Q}\|_F^2}$ as:

$$\Delta_{SNR} = \{A\sum_{i=3}^{N-1} y_i^2 + B\sum_{i=3}^{N-1} x_i^2 + C\sum_{i=3}^{N-1} x_i y_i + D\sum_{i=3}^{N-1} x_i + E\sum_{i=3}^{N-1} y_i + F\}/\{2\delta^2[(N-3)\sum_{j,k=0}^{2} r_{jk}^2$$
$$+ \sum_{i=3}^{N-1}(r_{0i}^2+r_{1i}^2+r_{2i}^2)]\} \ , \tag{18}$$

where

$$A = \sum_{j,k=0}^{2}(x_j-x_k)^2, B = \sum_{j,k=0}^{2}(y_j-y_k)^2 \ , C = -2\sum_{j,k=0}^{2}(x_j-x_k)(y_j-y_k) \ , \tag{19}$$

$$D = 2\sum_{j,k=0}^{2}(y_j-y_k)(x_j y_k - x_k y_j) \ , E = 2\sum_{j,k=0}^{2}(x_j-x_k)(x_j y_k - x_k y_j) \ , \tag{20}$$

$$F = (N-3)[\sum_{j,k=0}^{2}(x_j y_k - x_k y_j)^2 + (x_1 y_2 - x_2 y_1 +$$
$$x_0 y_1 - x_1 y_0 - x_0 y_2 + x_2 y_0)^2] \ , \tag{21}$$

where $j \neq k$. It can be seen that the critical points of SNR are trajectory-dependent. Expanding arbitrary trajectories in x and y directions in Maclaurin series, we obtain:

$$x = f(t) = f(0) + f^{'}(0)t + \frac{f^{''}(0)}{2!}t^2 + \ldots + \frac{f^{(n)}(0)}{n!}t^n \ . \tag{22}$$

$$y = g(t) = g(0) + g^{'}(0)t + \frac{g^{''}(0)}{2!}t^2 + \ldots + \frac{g^{(n)}(0)}{n!}t^n \ . \tag{23}$$

With regard to SNR, we have the following property:

Property 1: With uniform sampling $t_k = kT$,

$$\lim_{N\to\infty} \Delta_{SNR} = \frac{A(g^{(n)}(0))^2 + B(f^{(n)}(0))^2}{6\delta^2[(g^{(n)}(0))^2 + (f^{(n)}(0))^2]} + \frac{Cf^{(n)}(0)g^{(n)}(0)}{6\delta^2[(g^{(n)}(0))^2 + (f^{(n)}(0))^2]} \quad (24)$$

where A, B and C are defined in the expression of SNR.

Property 2: $\lambda = O(N)$ should be chosen for Poisson sampling to guarantee the convergence of τ, where N is the total number of samples.

Property 3: τ converges in mean sense given $\lambda = O(N)$. Specifically, for $\lambda = \frac{N}{T}$,

$$\lim_{N\to\infty} E(\tau) = 3\sum_{k=1}^{2n} \frac{(a_k + b_k)T^k}{k+1} , \quad (25)$$

where k is the index for Taylor series and for a_k and b_k, if k is odd,

$$a_k = \sum_{i=1}^{\frac{k-1}{2}} \frac{2g^{(i)}(0)g^{(k-i)}(0)}{i!(k-i)!} , \quad (26)$$

$$b_k = \sum_{i=1}^{\frac{k-1}{2}} \frac{2f^{(i)}(0)f^{(k-i)}(0)}{i!(k-i)!} , \quad (27)$$

if k is even,

$$a_k = \sum_{i=1}^{\frac{k-2}{2}} \frac{2g^{(i)}(0)g^{(k-i)}(0)}{i!(k-i)!} + \frac{(g^{(\frac{k}{2})}(0))^2}{(\frac{k}{2}!)^2} , \quad (28)$$

$$b_k = \sum_{i=1}^{\frac{k-2}{2}} \frac{2f^{(i)}(0)f^{(k-i)}(0)}{i!(k-i)!} + \frac{(f^{(\frac{k}{2})}(0))^2}{(\frac{k}{2}!)^2} . \quad (29)$$

Property 4: If the trajectories are sampled with $\lambda = O(N)$, the variance of the error ratio converges to zero, namely,

$$\lim_{N\to\infty} Var(\tau) = 0 . \quad (30)$$

Remark: Property 3 can be proved by showing that

$$\lim_{N\to+\infty} \frac{1}{N}\sum_{j=3}^{N-1} E(t_j^k) = \frac{1}{k+1} . \quad (31)$$

In this framework, the density λ corresponds to the average number of samples per unit-length; i.e. $\lambda = \frac{N}{T}$.

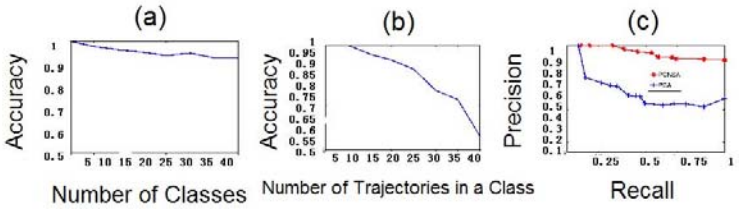


Fig. 4 (a) Accuracy values for the classification problem on increasing number of classes, (b) Accuracy values for the classification problem on increasing number of trajectories in a class. (c) The comparison with Precision-Recall Curve for retrieval with 20 classes for perfect trajectories.

Since in real life trajectories in a class may have different lengths, the length is normalized by taking the Fourier Transform and choosing the biggest n=32 coefficients and then taking the Inverse Fourier Transform so that all the trajectories are of size 32 before invariant matrix calculations. For all the simulations $\delta_1 = 10^7$, $\delta_2 = 10^{-4}$ as thresholds and $L = 32$ in PCNSA. Figure.4(a) depicts accuracy of the proposed classification system versus number of classes. There are $K = 20$ trajectories in each class word. Simulation results show that this system preserves its efficiency even for higher number of different classes. Figure.4(b) depicts accuracy values versus increase in the number of trajectories within a class. There are $C = 20$ classes in the system. Figure.4(c) shows Precision vs. Recall curves for indexing and retrieval problem by using 40 classes, each class having 20 trajectories. For retrieval problems, the distance of the query trajectory to any other trajectory is computed by using PCNSA on NSI as $D(X_i, Y) = \|W_{NSA,i}(X_i - Y)\|$, where Y is the query trajectory. This distance is then used to find α nearest trajectories, where α is a user specified parameter. There are two curves in Figure 4(c), one is with using PCA on NSI directly, where PCA is basically used for dimension reduction. As it can be

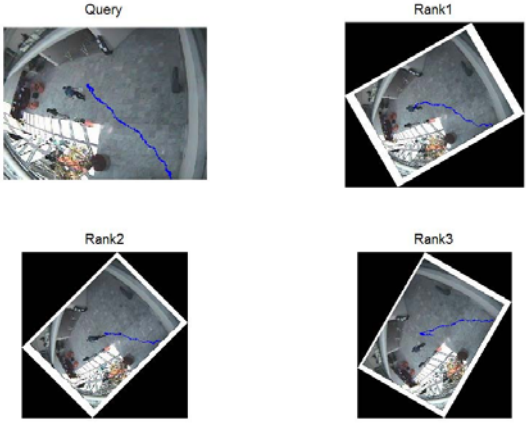


Fig. 5 Visual illustration for retrieval results with 20 classes with motion trajectories.

seen from Figure 4(c) that the result of using PCNSA on NSI is much superior to the one using PCA on NSI directly. The visual illustration of the query and the three most similar retrieval are shown in Fig. 5.

2.3 Tensor Null-Space (TNSI) Invariance Representation

Among affine view-invariance systems, majority of them represent affine view-invariance in a single dimension, thus limiting the system to only single object motion based queries and single dimension affine view-invariance. In many applications, it is not only the individual movement of an object that is of interest, but also the motion patterns that emerge while considering synchronized or overlapped movements of multiple objects. In this subsection, a novel fundamental mathematical framework is proposed for tensor null space invariants and is further used for the important application of view-invariant classification and retrieval of motion events involving multiple motion trajectories. Let us denote the tensor $A \in R^{I_1 \times I_2 \ldots I_{N-1} \times I_N}$ as the multi-dimensional data. Elements of A are denoted as $a_{i_1 i_2 \ldots i_N}$. A generalization of the product of two matrices is the product of a tensor and a matrix. The *mode-n product* of a tensor $A \in R^{I_1 \times I_2 \ldots \times I_n \ldots I_{N-1} \times I_N}$ by a matrix $U \in R^{I_n \times J_n}$, denoted by $A \times_n U$, is a tensor $B \in R^{I_1 \times \ldots I_{n-1} \times J_n \times I_{n+1} \ldots \times I_N}$ whose entries are:

$$(A \times_n U)_{i_1 \ldots i_{n-1} j_n i_{n+1} \ldots i_N} = \sum_{i_n} a_{i_1 \ldots i_{n-1} i_n i_{n+1} \ldots i_N} u_{i_n j_n} \tag{32}$$

The mode-n product $B = A \times_n U$ can be computed via the matrix multiplication $B_{(n)} = UA_{(n)}$, followed by a re-tensorization to undo the mode-n flattening. As described in the equations (10)-(12), applying the affine transformation T on the matrix $M_1 = TM$, the null spaces of M_1 and M are identical. Similarly, applying the affine transformation T_m, T_n on the mth, nth unfolding of the multi-dimensional data M, respectively, if the resulting tensor null space Q is invariant in both dimensions, the it is referred to as *mode-m,n invariant*. Let us derive the mathematical formulation of the mode-1,2,3 invariant tensor Q for three dimensional data $M \in R^{I_1 \times I_2 \times I_3}$. To be rotation invariant, we have:

$$M_{(1)} \times Q_{(3)} = 0 \ , M_{(2)} \times Q_{(2)} = 0 \ , M_{(3)} \times Q_{(1)} = 0 \ , \tag{33}$$

where $M_{(1)}, M_{(2)}, M_{(3)}$ are the unfolding of the three order tensor M into matrices with the dimension $I_2 I_3 \times I_1$, $I_1 \times I_3 I_2$ and $I_1 \times I_2 I_3$, respectively, and $Q_{(3)}, Q_{(2)}, Q_{(1)}$ are the corresponding unfoldings of the tensor Q. The condition for invariance of translation for the mode-1,2,3 invariant tensor Q:

$$\sum Q_{(1)} = 0, \sum Q_{(2)} = 0, \sum Q_{(3)} = 0 \tag{34}$$

Combining the conditions for rotational and translational invariance, we can solve the TNSI Q subject to the mode-1,2,3 affine view-invariant. If the order of the

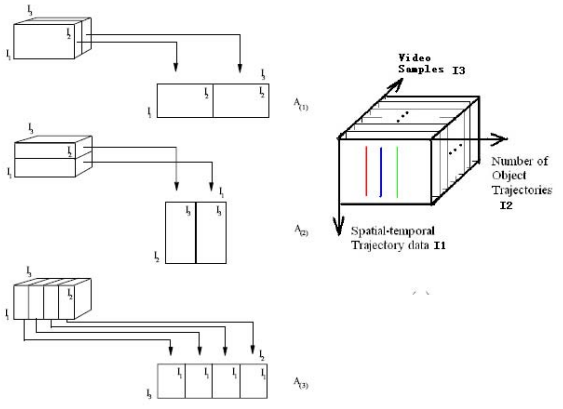


Fig. 6 Flattening a 3^{rd} order motion event tensor for multiple trajectories representation.

tensor M is 2, it is easy to show that the condition boils down to the Stiller's one dimensional null space invariants [6]. It is also easy to extend the result to the case of the Nth order tensor with mode-$I_a, \ldots, I_k$ affine view-invariant.

We align each trajectory as two rows in a matrix according to x and y coordinates, and the number of rows of a matrix is set to be twice the number of the objects in the motion event under analysis. $M = (M_{i,j})_{i=1,2,\ldots,2J;j=1,2,\ldots,P}$,, where P denotes the temporal length of normalized trajectories, J represents the number of trajectories within one motion event. Finally, multiple trajectory matrices are aligned in the direction orthogonal to the plane spanned by them, and form a three dimensional matrix, or tensor. We refer to it as *Motion Event Tensor* T. $T = (T_{i,j,k})_{i=1,2,\ldots,2J;j=1,2,\ldots,P;k=1,2,\ldots,K}$, where K is the number of motion event samples (trajectory video clips).

3 Video Indexing and Retrieval

In this section, the concepts, problems and state-of-the-art approaches for content-based video indexing and retrieval (CBVIR) are discussed. We first investigate one of the key problems in video retrieval-video summarization, where we summarize the state-of-the-art approaches for both shot-boundary detection and key frame extractions. Then we focus on spatial-temporal motion trajectory analysis, and present both single and multiple motion trajectory-based CBVIR techniques, especially, (i) geometrical multiple-trajectory indexing and retrieval (GMIR) algorithm, (ii) unfolded multiple-trajectory indexing and retrieval (UMIR) algorithm, and (iii) concentrated multiple-trajectory indexing and retrieval (CMIR) algorithm. The above algorithms not only remarkably reduce the dimensionality of the indexing space but also enable the realization of fast retrieval systems.

3.1 Motion-Trajectory Representation

An object trajectory-based system for video indexing is proposed in [26], in which the normalized x- and y-projections of trajectory are separately processed by wavelet transform using Haar wavelets. Chen et al. [27] segment each trajectory into subtrajectories using fine-scale wavelet coefficients at high levels of decomposition. A feature vector is then extracted from each subtrajectory and Euclidean distances between each subtrajectory in the query trajectory and all the indexed subtrajectories are computed to generate a list of similar trajectories in the database. Bashir et al. [19][28] proposed a Principle Component Analysis (PCA)-based approach to object motion trajectory indexing and retrieval, which has been shown to provide a very effective method for indexing and retrieval of single object motion trajectory.

3.1.1 Multiple Motion-Trajectory Representation

For a video database consisting of many video clips, each video clip has an unique corresponding set of multiple motion trajectories, which characterizing the dynamics and motion pattern of multiple objects within that particular video clip. Thus by indexing and retrieving a set of multiple object trajectories, we are able to index and retrieve its corresponding video clip in the database.

Each set of multiple object trajectories $\mathscr{S} = \{r_1^L, r_2^L, r_3^L, ..., r_M^L\}$, extracted from a particular video clip of length L with M objects, is modelled as a *Multiple Trajectory Matrix* $\mathscr{M}$ of size $2L$ by M.

We firstly smooth out each of the noisy trajectories by applying wavelet transform using Daubechies wavelet DB4, and taking coarse coefficients corresponding to the low subband in a three level decomposition. After that, $x-$ and $y-$ location information of each object trajectory $r_i^L (i = 1, 2, ..., M)$ is concatenated into one column vector, we refer to as *Single Trajectory Vector* $\mathscr{V}_i$.

$$\mathscr{V}_i = (x_i[1], x_i[2], ..., x_i[L], y_i[1], y_i[2], ..., y_i[L])^T \tag{35}$$

The single trajectory vectors are then aligned as columns of a matrix, where the number of columns is set to be the number of objects in the particular set of multiple object trajectories. We name this matrix as *Multiple Trajectory Matrix* $\mathscr{M}$.

$$\mathscr{M} = [\mathscr{V}_1 | \mathscr{V}_2 | \mathscr{V}_3 | ... | \mathscr{V}_M] \tag{36}$$

Here, each multiple trajectory matrix represents the motion dynamics of a group of objects within one particular video clip, please note that the order of single trajectory vectors placed in the multiple trajectory matrix is very important, for simplicity, we focused exclusively on the simplest case, where orders of multiple trajectories in both the dataset and the query are known. However, the tensor-space representation can be easily extended to the case where correspondence of multiple trajectories between the dataset and the query is unknown. For more details, please refer to [29].

In a motion video database with many video clips, multiple trajectories are firstly extracted for each video, then multiple trajectory matrix is constructed. For compact representation, multiple trajectory matrices of the same number of columns (video clips with the same number of acting objects), are grouped together. Then each group of video clips are resampled to a median size for further processing. The median size is the median of lengths from all video clips within the same group. Let there be N sets of M trajectories extracted from N video clips, and the original lengths of each set of M trajectories be L_1, L_2,...,L_N. Suppose the desired median length after sampling is L'. For each set of M trajectories, we resample the whole set of trajectories to the median length L'. For each trajectory within the same set, we first use 2D fourier transform to get coefficients of each trajectory in frequency domain; then we retain the first p biggest coefficients while ignoring the rest; finally we perform L'-point inverse 2D fourier transform to get the re-sampled trajectory with desired length L'. Figure 7 (b) depicts three sets of 2-trajectories S_1, S_2 and S_3, extracted from three video clips of lengths L_1,L_2 and L_3, respectively; while Fig. 7 (c) shows the resampled multiple trajectory sets with the same median size L'. The re-sampling process is a necessary step to transform different sets of trajectories with varying lengths to the same format, such that they can be assembled into a compact form (e.g. matrix or tensor form) for further efficient analysis.

After resampling, within each group, multiple trajectory matrices are of the same size, then they are aligned in the direction orthogonal to the plane spanned by them, and form a three-dimensional matrix, or tensor [30]. We refer to it as *Multiple Trajectory Tensor* $\mathscr{T}$

$$\mathscr{T} = (T_{i,j,k})_{i=1,2,...,2L';j=1,2,...,M;k=1,2,...,K.} \tag{37}$$

where L and M are the same as previous defined, K is the depth of the tensor, referring to total number of video clips indexed so far.

Figure 7 depicts an example of constructing a multiple trajectory tensor by using three set of multiple trajectories. The reason to align multiple trajectory matrices and form a three-dimensional tensor is, to assemble as much as possible date into a compact form and extract intrinsic and common characteristics of data for efficient analysis. By assembling multiple trajectory into three-dimensional tensors, the multiple trajectory data spans a three-dimensional tensor-space.

Please note that this approach is built on trajectory representation by instantaneous x- and y- coordinates of object centroid at each frame. While processing trajectories, it is assumed that the frame of reference for all of the trajectories is held fixed during the generation of trajectories. This is consistent with the fixed-camera scenario. For the moving camera case, such as PTZ cameras or airborne surveillance, an additional step of trajectory registration will be needed to generate trajectories which are all registered to a common frame of reference. The proposed algorithms can then be used for indexing and retrieval of the registered trajectories from multiple objects for the moving camera scenario.

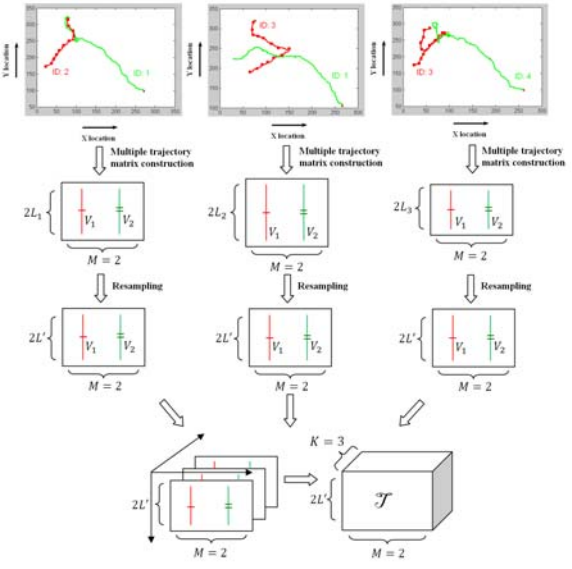


Fig. 7 Tensor-Space Representation of Multiple-Object Trajectories: (a) Three sets of multiple trajectories $\mathscr{S}_1$,$\mathscr{S}_2$,$\mathscr{S}_3$, extracted from three video clips displayed in Fig. 1. (b) Three corresponding *Multiple Trajectory Matrices* $\mathscr{M}_1$, $\mathscr{M}_2$, $\mathscr{M}_3$. (c) Three *Multiple Trajectory Matrices* $\mathscr{M}'_1$, $\mathscr{M}'_2$, $\mathscr{M}'_3$ after resampling. (d) *Multiple Trajectory Tensor* $\mathscr{T}$ constructed from $\mathscr{M}'_1$, $\mathscr{M}'_2$, $\mathscr{M}'_3$.

3.1.2 Global and Segmented Multiple Motion-Trajectory Representation

Two types of multiple trajectory tensors based on two types of multiple trajectory data are introduced.

Global multiple trajectory tensor. The "Global Multiple Trajectory Tensor" is constructed by using sets of multiple trajectories extracted from video clips, as shown in Fig. 1. Here, "Global" refers to the multiple trajectories with global lengths.

Segmented multiple trajectory tensor. The "Segmented Multiple Trajectory Tensor" is constructed by using sets of multiple subtrajectories. We jointly segment multiple trajectories with global lengths into atomic "units" which are called *multiple subtrajectories*. Those multiple subtrajectories represent certain joint motion patterns of multiple objects within a certain time interval. The segmentation points that define the joint segmentation of multiple trajectories depend on the segmentation technique and in general shall correspond to changes in joint motion patterns of multiple objects, such as changes in velocity (1st order derivative) and/or acceleration (2nd order derivative). The proposed segmentation technique ensures that only joint change of motions of multiple objects would be chosen as segmentation point, based on spatial curvatures of multiple trajectories. The spatial curvature of a 2-D trajectory is given by:

$$\Theta[t] = \frac{x'[t]y''[t] - y'[t]x''[t]}{[(x'[t])^2 + (y'[t])^2]^{3/2}}. \tag{38}$$

The value of curvature at any point is a measure of inflection point, an indication of concavity or convexity in the trajectory. For multiple trajectories, curvature data is calculated for each trajectory. For example, for $M(\geq 1)$ trajectories, there would be M curvatures. We segment the multiple trajectories by applying a moving window scheme and a hypothesis testing method on their spatial curvatures. Let X and Y be two non-overlapping windows of size n, where X contains the first n M-dimensional curvature vector samples, and Y contains the next n samples. Each M-dimensional curvature vector consists of curvatures from M trajectories. Let Z be the window of size $2n$ formed by concatenating window X and window Y. Then we perform a likelihood ratio hypothesis test to determine if the two windows X and Y have data drawn from the same distribution. If curvatures of multiple trajectories in X and Y are from different distributions, then there would be a change of concavity or convexity of multiple trajectories.

Specifically, we have two hypothesis:

$$\begin{cases} H_0 : f_x(X;\theta_x) = f_y(Y;\theta_y) = f_z(Z;\theta_z). \\ H_1 : f_x(X;\theta_x) \neq f_y(Y;\theta_x). \end{cases}$$

Assume that curvature data in each window forms an i.i.d random variable and the data is M-dimensional jointly Gaussian. We first compute the maximum likelihood estimator of mean and variance for each hypothesis,

$$L_0 = \left[\frac{1}{(2\pi)^{\frac{M}{2}}|\Sigma_3|^{\frac{1}{2}}}\right]^{2n} \exp\{-\frac{1}{2}\sum_{i=s}^{s+2n}(\underline{k_i} - \underline{\mu_3})^T \Sigma_3^{-1}(\underline{k_i} - \underline{\mu_3})\}$$

$$L_1 = \left[\frac{1}{(2\pi)^M|\Sigma_1|^{\frac{1}{2}}|\Sigma_2|^{\frac{1}{2}}}\right]^{n} \exp\{-\frac{1}{2}[\sum_{i=s}^{s+n}(\underline{k_i} - \underline{\mu_1})^T \Sigma_1^{-1}(\underline{k_i} - \underline{\mu_1}) + \sum_{i=s+n+1}^{s+2n}(\underline{k_i} - \underline{\mu_2})^T \Sigma_2^{-1}(\underline{k_i} - \underline{\mu_2})]\}. \tag{39}$$

then the likelihood ratio is defined as,

$$\lambda_L = \frac{L_0}{L_1}. \tag{40}$$

We finally calculate the distance d between the distributions of X and Y. We define the distance function d between X and Y as

$$d(s) = -log(\lambda_L) = -nlog(\frac{|\Sigma_1|^{1/2}|\Sigma_2|^{1/2}}{|\Sigma_3|})$$

$$+\frac{1}{2}\Big[\sum_{i=s}^{s+2n}(\underline{k_i}-\underline{\mu_3})^T\Sigma_3^{-1}(\underline{k_i}-\underline{\mu_3})-\sum_{i=s}^{s+n}(\underline{k_i}-\underline{\mu_1})^T\Sigma_1^{-1}(\underline{k_i}-\underline{\mu_1})$$

$$-\sum_{i=s+n+1}^{s+2n}(\underline{k_i}-\underline{\mu_2})^T\Sigma_2^{-1}(\underline{k_i}-\underline{\mu_2})\Big]. \quad (41)$$

where $\underline{\mu_i}$, Σ_i $(i=1,2,3)$ are mean column vectors and variance matrices of M-dimensional Gaussian distributions, which represent distributions of data in windows X,Y and Z, respectively. s is the start point of samples in window X, where $s=1,2,...,L-2n$. $\underline{k_i}$ is curvature column vector which consists of curvature samples from M trajectories at time instant i, $\underline{k_i}=[\Theta_1[i],...,\Theta_M[i]]^T$. The distance d is large if the data in window X and Y have different distributions. The windows are moved by m $(<n)$ samples and the process is repeated in the same manner. A 1-D vector of distance function is then formed. The segmentation positions are chosen at the locations along the trajectory where the likelihood ratio d attains a local maximum. To select the local maxima, the 1-D vector of likelihood ratio d is partitioned into segments and the global maximum is selected within each partition to represent the segmentation location. The threshold α within each segment is chosen such that only the global maximum of the likelihood ratio d is selected to represent the segmentation location within each partition. Figure 8 displays the segmentation results of a set of multiple trajectories. Please note that the segmentation positions are chosen when both curvatures of the two trajectories have a sudden change, e.g. at segmentation points 2 and 4, which ensures that in the proposed segmentation only joint change of motions of multiple objects would be chosen as segmentation point.

3.2 Multiple-Trajectory Indexing and Retrieval

Three multiple-object trajectory indexing and retrieval algorithms, that mainly differ in terms of representation techniques, are presented: (i) geometrical multiple-trajectory indexing and retrieval (GMIR) algorithm, (ii) unfolded multiple-trajectory

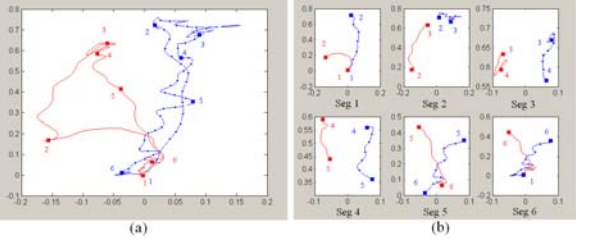


Fig. 8 Example of segmentation of a set of global multiple trajectories (2 trajectories) into segmented multiple subtrajectories. (a) a set of 2 trajectories with global length. (b) 6 sets of segmented multiple subtrajectories, segmented from (a). In both figures, solid squares with numbers indicate segmentation positions, the horizontal axis is x- location, the vertical axis is y- location within the scene.

indexing and retrieval (UMIR) algorithm, and (iii) concentrated multiple-trajectory indexing and retrieval (CMIR) algorithm.

3.2.1 Geometrical Multiple-Trajectory Indexing and Retrieval (GMIR) Algorithm

In the following we outline the Indexing and Retrieval processes of the GMIR algorithm.

Indexing

1. Generate geometric bases G_1, G_2 and G_3 from multiple trajectory tensor $\mathscr{T}$. The geometric bases represent the principle axes of variation across each coordinate axis in a 3D coordinate system. Specifically, G_1 spans the space of spatial-temporal multiple trajectory data, G_2 spans the space of object trajectory cardinality, G_3 spans the space of sets of multiple trajectories. They can be obtained by using various tensor analysis techniques, such as[31].
2. Project each multiple trajectory matrix onto the first two bases G_1 and G_2 and transform the multiple trajectory data into low-dimensional subspace defined by those two bases:
$$M_{Coeff} = G_1^T \times M_{Multi-Traj} \times G_2. \tag{42}$$
3. Use the coefficient matrices obtained in step 2 as indices of their corresponding multiple-trajectory matrices in the database.

Retrieval

1. Project multiple trajectories in the input query on the 2 bases G_1 and G_2, and get the GMIR coefficient matrix:
$$M_{QueryCoeff} = G_1^T \times M_{TrajQuery} \times G_2. \tag{43}$$
2. Calculate the Euclidean distance norm D_{GMIR} between the GMIR coefficients of query multiple-trajectory and the GMIR coefficients matrices stored in the database, and return the ones that have the distances within a threshold.
$$D_{GMIR} = ||(M_{Coeff} - M_{QueryCoeff})||^2. \tag{44}$$

where in eqns.(6)-(8), G_i is the i^{th} geometric base, G_i^T is transpose of G_i, M_{Coeff} and $M_{QueryCoeff}$ are the coefficient matrices of multiple object trajectory in multiple trajectory tensor and query multiple-trajectory, $M_{Multi-Traj}$ and $M_{TrajQuery}$ are multiple trajectory matrices in multiple trajectory tensor and query multiple object trajectory, respectively.

3.2.2 Unfolded Multiple-Trajectory Indexing and Retrieval (UMIR) Algorithm

The procedure of generating data-dependent bases of a multi-dimensional matrix or tensor using unfolded multiple-trajectory indexing and retrieval (UMIR) algorithm

can be viewed as a recursive "unfold" process of tensor. By viewing slices of tensor, which are matrices, as vectors, the tensor can be viewed as a matrix, then first use SVD on the matrix, and then use SVD on the slice-matrices, as shown below:

$$\mathscr{T} = \sigma_1 \times_1 T_{1\ldots(N-1)} \times_2 U_N. \tag{45}$$

$$T_{1\ldots(N-1)} = \sigma_2 \times_1 T_{1\ldots(N-2)} \times_2 U_{N-1}. \tag{46}$$

$$\ldots$$

$$T_{12} = \sigma_{N-1} \times_1 U_1 \times_2 U_2. \tag{47}$$

Where in the above eqns., $\times_n$ refers to the mode-n product [32]. The indexing and retrieval processes of the UMIR algorithm are summarized below:

Indexing

1. Generate unfolded bases U_1, U_2 and U_3 from multi-dimensional data, by recursive "unfolding" of tensor $\mathscr{T}$:

$$\mathscr{T} = \sigma_1 \times_1 T_{12} \times_2 U_3. \tag{48}$$

$$T_{12} = \sigma_2 \times_1 U_1 \times_2 U_2. \tag{49}$$

2. Project each multiple trajectory matrix onto the 2 bases U_1 and U_2 and transform the multiple trajectory data into low-dimensional subspace spanned by those two bases:

$$M_{Coeff} = U_1^T \times M_{Multi-Traj} \times U_3. \tag{50}$$

3. Use the coefficient matrices obtained in step 2 as indices of their corresponding multiple-trajectory matrices in the database.

Retrieval

1. Project query multiple trajectories on the 2 bases U_1 and U_3, and get UMIR coefficient matrix:

$$M_{QueryCoeff} = U_1^T \times M_{TrajQuery} \times U_3. \tag{51}$$

2. Calculate the Euclidean distance norm D_{UMIR} between the UMIR coefficients of query multiple-trajectory and those of each multiple-trajectory input indexed in the database, and return the ones that have the distances within a threshold.

$$D_{UMIR} = ||(M_{Coeff} - M_{QueryCoeff})||^2. \tag{52}$$

Where in eqns.(9)-(16), $T_{1\ldots N}$ is a matrix indexed by 1 to N, and U_i are unfolded bases. U_i^T is transpose of U_i $(i = 1,2,3)$. M_{Coeff1}, M_{Coeff2}, $M_{QueryCoeff1}$, $M_{QueryCoeff2}$ are coefficient matrices; $M_{Multi-Traj}$, $M_{TrajQuery}$ are the multiple-trajectory matrix and query multi-trajectory matrix.

3.2.3 Concentrated Multiple-Trajectory Indexing and Retrieval (CMIR) Algorithm

The Indexing and Retrieval processes used in the CMIR algorithm are outlined below:

Indexing

1. Generate concentrated bases C_1, C_2 and C_3 from multiple trajectory tensor $\mathscr{T}$ by minimizing the following sum-of-squares loss function:

$$\min_{C_1,C_2,C_3} \sum_{i,j,k} ||\mathscr{T}_{ijk} - \sum_{r}^{R} c_{ir}^1 c_{jr}^2 c_{kr}^3||^2. \tag{53}$$

   Where c_{ir}^1,c_{jr}^2, c_{kr}^3 are i, j and k-th column vectors of C_1, C_2 and C_3, respectively. The data-dependent bases can be obtained by solving the above equation. One solution to extract those bases is called Parallel Factors Decomposition (PARAFAC) [33].
2. Project each multiple trajectory matrix onto the 2 bases C_1 and C_2 and transform the multiple trajectory data into low-dimensional subspace spanned by those two bases:

$$M_{Coeff} = C_1^T \times M_{Multi-Traj} \times C_2. \tag{54}$$

3. Use the coefficient matrices obtained in step 2 as indices of their corresponding multiple-trajectory matrices in the database.

Retrieval

1. Project query multiple trajectories on the 2 bases C_1 and C_2, and get CMIR coefficient matrix:

$$M_{QueryCoeff} = C_1^T \times M_{TrajQuery} \times C_2. \tag{55}$$

2. Calculate the Euclidean distance norm D_{UMIR} between the UMIR coefficients of query multiple-trajectory and those of each multiple-trajectory input indexed in the database, and return the ones that have the distances within a threshold.

$$D_{CMIR} = ||(M_{Coeff} - M_{QueryCoeff})||^2. \tag{56}$$

Where in eqns.(14)-(17) C_i^T is transpose of $C_i i = 1,2,3$, $M_{Multi-Traj}$ and $M_{TrajQuery}$ are trajectory matrices of multiple-trajectory in the database and query multiple-trajectory, M_{Coeff} and $M_{QueryCoeff}$ are their corresponding coefficient matrices.

The experiment shown in Fig. 9 demonstrates the retrieval results on two trajectories in CAVIAR [34] dataset. The query is represented in Fig. 9(a). The most-similar and second-most-similar retrieval results are illustrated in Figs. 9(b) and 9(c), respectively. In this case, the query depicts a video clip "two people who meet, fight and chase each other". The most-similar retrieved result show in Fig. 9(b) is a video

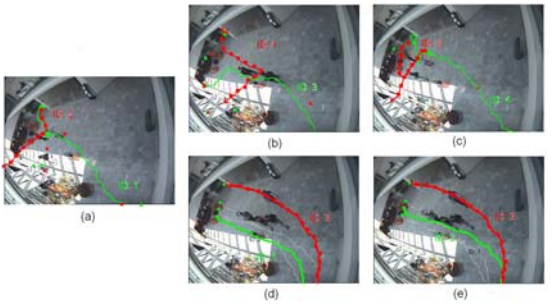


Fig. 9 Retrieval results for *CAVIAR* dataset (INRIA) using the proposed CMIR algorithm for multiple trajectory representation (2 trajectories): (a) the query; (b) the most-similar retrieval; (c) the second-most-similar retrieval; (d) the most-dissimilar retrieval; (e) the second-most-dissimilar retrieval.

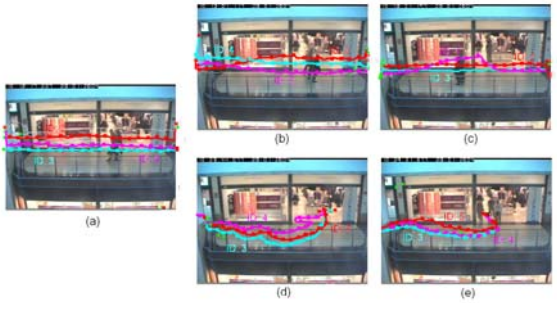


Fig. 10 Retrieval results for *CAVIAR* dataset (Shopping Center in Portugal) using the proposed CMIR algorithm for multiple trajectory representation (3 trajectories): (a) the query; (b) the most-similar retrieval; (c) the second-most-similar retrieval; (d) the most-dissimilar retrieval; (e) the second-most-dissimilar retrieval.

clip in which "two people meet, walk together and split apart"; whereas the second-most similar retrieved result portrayed in Fig. 9(c) is a video clip in which "two people meet, fight and run away". We can see the retrieved multiple trajectories are visually quite similar with the query. Figure 10 shows another retrieval result on *CAVIAR* data for three-trajectory case. Trajectory data are selected from the clips of shopping center in Portugal (2nd subset in CAVIAR). In this scenario, the query depicts a video sequence "3 persons walking in the corridor". The most-similar retrieved result show in Fig. 10(b) is a video clip in which "Another 3 persons walking in the corridor"; whereas the second-most similar retrieved result portrayed in Fig. 10(c) is a video clip in which "3 people walking together along the corridor". On contrast, the most-dissimilar retrieval and the second-most-dissimilar retrieval are depicted in Figs. 10 (d) and (e), respectively. Both of retrieved dissimilar results visually vary a lot from the query.

4 Video Classification and Recognition

Video classification differs from video indexing and retrieval, since in video classification, all videos are put into categories, and each video is assigned a meaningful label. While in video indexing and retrieval, the aim is to accurately retrieve videos that match a user's query. Many automatic video classification algorithms have been proposed, most of them can be categorized into four groups: text-based approaches, audio-based approaches, visual-based approaches, and combination of text, audio and visual features. Many standard classifiers, such as Gaussian Mixture Models (GMM), Bayesian, support vector machines (SVM), neural networks and hidden Markov Models (HMMs) have been applied in video classification and recognition. For more details, we refer to [35]. We propose a novel distributed multi-dimensional hidden Markov Model (DHMM) for modelling of interacting trajectories involving multiple objects. The proposed model is capable of conveying not only dynamics of each trajectory, but also interactions information between multiple trajectories, while requiring no further semantic analysis.

4.1 Hidden Markov Model

Hidden Markov model (HMM) is very powerful tool to model temporal dynamics of processes, and has been successfully applied to many applications such as speech recognition [36], gesture recognition [37], musical score following [38]. Bashir et al. [39] presented a novel classification algorithm of object motion trajectory based on 1D HMM. They segmented single trajectory into atomic segments called subtrajectories based on curvature of trajectory, then the subtrajectories are represented by their principal component analysis (PCA) coefficients. Temporal relationships of subtrajectories are represented by fitting a 1D HMM. However, all the above applications rely on a one-dimensional HMM structure. Simple combinations of 1D HMMs can not be used to characterize multiple trajectories, since 1D models fail to convey interaction information of multiple interacting objects. The major challenge here is to develop a new model that will semantically reserve and convey the "interaction" information.

4.2 Multi-dimensional Distributed Hidden Markov Model

A novel distributed multi-dimensional hidden Markov Model (DHMM) for modelling of interacting trajectories involving multiple objects is proposed. In this model, each object-trajectory is modelled as a separate Hidden Markov process; while "interactions" between objects are modelled as dependencies of state variables of one process on states of the others. The intuition of this work is that, HMM is very powerful tool to model temporal dynamics of each process (trajectory); each process (trajectory) has its own dynamics, while it may be influenced by or influence others. In the proposed model, "influence" or "interaction" among processes (trajectories) are modelled as dependencies of state variables among processes

(trajectories). This model is capable of conveying not only dynamics of each trajectory, but also interactions information between multiple trajectories, while requiring no semantic analysis.

A solution for non-causal, multi-dimensional HMMs is proposed by distributing the non-causal model into multiple distributed causal HMMs. Then the simultaneous solution of multiple distributed HMMs is approximated on a sequential processor by an alternate updating scheme. Subsequently the training and classification algorithms presented in [45] are extended to a general causal model. A new Expectation-Maximization (EM) algorithm for estimation of the new model is derived, where a novel General Forward-Backward (GFB) algorithm is proposed for recursive estimation of the model parameters. A new conditional independent subset-state sequence structure decomposition of state sequences is proposed for the 2D Viterbi algorithm. The new model can be applied to many problems in pattern analysis and classification. For simplicity, the presentation in this paper will focus primarily on a special case of the proposed model in two-dimensions, which we referred to as distributed 2D hidden Markov Models (2D DHMMs).

Suppose there are $M \in \mathbb{N}$ interacting objects in a scene. Recall that in the proposed model, each object-trajectory is modelled as a Hidden Markov process of time; while "interactions" between object trajectories are modelled as dependencies of state variables of one process on those of the others. We constrain the probabilistic dependencies of state in one process (trajectory) at time t, on its own state at time t-1, as well as on the states of other processes (trajectories) that "interact" or influence on it at time t and t-1, i.e.

$$Pr(s(m,t)|s(l,t),s(n,1:t-1)) = Pr(s(m,t)|s(n,t-1),s(l,t)) \tag{57}$$

where $m,n,l \in \{1,...,M\}$ are indexes of processes (trajectories), $l \neq m$. The above constrain of state dependencies makes the desired model non-causal, since each process (trajectory) can influence others, there is no guarantee that the influence should be directional or causal. Figure 4(a) shows an example of the proposed non-causal 2D HMM, which is used to model two interacting trajectories. Each node S(i; t) in the figure represents one state at specific time t for trajectory i, where $t = \{1,2,...,T\}$, i = {1,2}; each node O(i,t) represents observations corresponding to S(i,t), and each arrow indicates transition of states (the reverse direction of it indicates dependency of states). The first row of states is the state sequence for trajectory 1, and the second row corresponds to trajectory 2. As can be seen, each state in one HMM chain (trajectory) will depend on its past state, the past state of the other HMM chain (trajectory), and the concurrent state of the other HMM chain (trajectory).

The above model is capable of modelling multiple processes and their interactions, but it is intractable since it is non-causal. A novel and effective solution is proposed, where the model is "decomposed" into M causal 2D hidden Markov models with multiple dependencies of states, such that each HMM can be executed in parallel in a distributed framework. In each of the distributed causal HMM, state

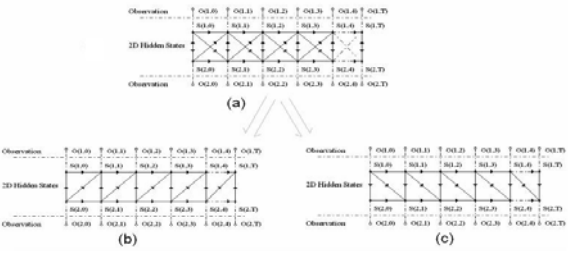


Fig. 11 Distributed 2D Hidden Markov Models: (a) Non-causal 2D Hidden Markov Model. (b) Distributed 2D Hidden Markov Model 1. (c) Distributed 2D Hidden Markov Model 2.

transitions (or state dependencies) must follow the same causality rule. For example, we distributed the non-causal 2D HMM in Fig. 11(a) to two causal 2D HMMs, shown in Figs. 11(b) and 11(c), respectively. In Fig. 11(b), state transitions follow the same rule, so do state transitions in Fig. 11(c). The above rules ensure the homogeneous structure of each distributed HMM, which further enable us to develop relatively tractable training and classification algorithms.

When trajectory number M=3, the proposed non-causal 2D hidden Markov model is depicted as in Fig. 12(a), the same distributing scheme is used to get 3 distributed 2d hidden Markov models, as shown in Fig. 12(b), 12(c), and 12(d), respectively. Using the same distributing scheme, any non-causal 2D hidden Markov model that characterizing $M(>3)$ trajectories can be distributed to M distributed causal 2D hidden Markov models.

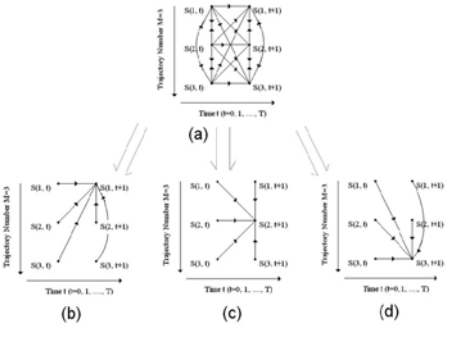


Fig. 12 Distributed 2D Hidden Markov Models with application to 3 trajectories: (a) Non-causal 2D Hidden Markov Model that treat 3 object trajectory as one system(only 2 adjacent time slots of the system states are shown). (b) Distributed 2D Hidden Markov Model 1 for Object Trajectory 1. (c) Distributed 2D Hidden Markov Model 2 for Object Trajectory 2. (d) Distributed 2D Hidden Markov Model 3 for Object Trajectory 3. (Please note in Figures (b) (c) and (d), only state transitions to one state point is shown, other state points follow the same rules, respectively).

4.2.1 DHMM Training and Classification

Define the observed feature vector set O = {o(m,t), m = 1,2,...,M; t = 1,2,...,T} and corresponding hidden state set S = {s(m,t), m = 1,2,...,M; t = 1,2,...,T }, and assume each state will take N possible values. The model parameters are defined as a set $\Theta = \{\Pi, \mathbf{A}, \mathbf{B}\}$, where Π is the set of initial probabilities of states $\Pi = \{\pi(m,n)\}$; $\mathbf{A}$ is the set of state transition probabilities $A = \{a_{i,j,k,l}(m)\}$, and $a_{i,j,k,l}(m) = Pr(s(m,t) = l|s(m',t) = k, s(m,t-1) = i, s(m',t-1) = j)$, and $\mathbf{B}$ is the set of probability density functions (PDFs) of the observed feature vectors given corresponding states, assume $\mathbf{B}$ is a set of Gaussian distribution with means $\mu_{m,n}$ and variances $\Sigma_{m,n}$, where $m, m' = 1,...,M; m \neq m'; n, i, j, k, l = 1,...,N; t = 1,...,T$. Due to space limit, the case $M = 2$ is discussed. For more details, we refer to [40] [41] [42] [43].

Expectation-maximization (EM) algorithm. A new Expectation-Maximization (EM) algorithm suitable for estimation of parameters of the M Distributed 2D Hidden Markov Models in M trajectory system is proposed. The proposed algorithm is analogous to the classical EM algorithm for 1D HMM [44].

Define $F^{(p)}_{m,n,k,l}(i,j)$ as the probability of state corresponding to observation $o(i-1,j)$ is state m, state corresponding to observation $o(i-1,j-1)$ is state n, state corresponding to observation $o(i,j-1)$ is state k and state corresponding to observation $o(i,j)$ is state l, given the observations and model parameters,

$$F^{(p)}_{m,n,k,l}(i,j) = P\Big(m = s(i-1,j), n = s(i-1,j-1), k = s(i,j-1), l = s(i,j)|O,\Theta^{(p)}\Big), \tag{58}$$

and define $G^{(p)}_m(i,j)$ as the probability of the state corresponding to observation $o(i,j)$ is state m, then

$$G^{(p)}_m(i,j) = P(s(i,j) = m|O,\Theta^{(p)}). \tag{59}$$

We can get the iterative updating formulas of parameters of the proposed model,

$$\pi^{(p+1)}_m = P(G^{(p)}_m(1,1)|O,\Theta^{(p)}). \tag{60}$$

$$a^{(p+1)}_{m,n,k,l} = \frac{\sum_i^I \sum_j^J F^{(p)}_{m,n,k,l}(i,j)}{\sum_{l=1}^M \sum_i^I \sum_j^J F^{(p)}_{m,n,k,l}(i,j)}. \tag{61}$$

$$\mu^{(p+1)}_m = \frac{\sum_i^I \sum_j^J G^{(p)}_m(i,j) o(i,j)}{\sum_i^I \sum_j^J G^{(p)}_m(i,j)}. \tag{62}$$

$$\Sigma^{(p+1)}_m = \frac{\sum_i^I \sum_j^J G^{(p)}_m(i,j)(o(i,j) - \mu^{(p+1)}_m)(o(i,j) - \mu^{(p+1)}_m)^T}{\sum_i^I \sum_j^J G^{(p)}_m(i,j)}. \tag{63}$$

In eqns. (1)-(6), p is the iteration step number. $F^{(p)}_{m,n,k,l}(i,j)$, $G^{(p)}_{m}(i,j)$ are unknown in the above formulas, a General Forward-Backward (GFB) algorithm will be introduced to estimate those parameters.

General forward-backward (GFB) algorithm. Forward-Backward algorithm was firstly proposed by Baum et al. [44] for 1D Hidden Markov Model and later modified by Li et al. [45]. Here, the Forward-Backward algorithm in [44][45] is generalized so that it can be applied to the proposed model, the proposed algorithm is called General Forward-Backward (GFB) algorithm. Assume the probability of all-state sequence S can be decomposed as products of probabilities of conditional-independent subset-state sequences $U_0, U_1, ...$, i.e.,

$$P(S) = P(U_0)P(U_1/U_0)...P(U_i/U_{i-1})... \tag{64}$$

where $U_0, U_1, ..., U_i ...$are subsets of all-state sequence in the HMM system, we call them *subset-state sequences*. Define the observation sequence corresponding to each subset-state sequence U_i as O_i.

Define the forward probability $\alpha_{U_u}(u), u = 1, 2, ...$ as the probability of observing the observation sequence $O_v (v \leq u)$ corresponding to subset-state sequence $U_v (v \leq u)$ and having state sequence for u-th product component in the decomposing formula as U_u, given model parameters Θ, i.e. $\alpha_{U_u}(u) = P\{S(u) = U_u, O_v, v \leq u|\Theta\}$, and the backward probability $\beta_{U_u}(u), u = 1, 2, ...$ as the probability of observing the observation sequence O_v $(v > u)$ corresponding to subset-state sequence $U_v (v > u)$, given state sequence for u-th product component as U_u and model parameters Θ, i.e. $\beta_{U_u}(u) = P(O_v, v > u|S(u) = U_u, \Theta)$.

The recursive updating formula of forward and backward probabilities can be obtained as

$$\alpha_{U_u}(u) = [\sum_{u-1} \alpha_{U_{u-1}}(u-1)P\{U_u|U_{u-1}, \Theta\}]P\{O_u|U_u, \Theta\}. \tag{65}$$

$$\beta_{U_u}(u) = \sum_{u+1} P(U_{u+1}|U_u, \Theta)P(O_{u+1}|U_{u+1}, \Theta)\beta_{U_{u+1}}(u+1). \tag{66}$$

Then, the estimation formulas of $F_{m,n,k,l}(i,j)$, $G_m(i,j)$ are :

$$G_m(i,j) = \frac{\alpha_{U_u}(u)\beta_{U_u}(u)}{\sum_{u:U_u(i,j)=m} \alpha_{U_u}(u)\beta_{U_u}(u)}. \tag{67}$$

$$F_{m,n,k,l}(i,j) = \frac{\alpha_{U_{u-1}}(u-1)P(U_u|U_{u-1}, \Theta)P(O_u|U_u, \Theta)\beta_{U_u}(u)}{\sum_u \sum_{u-1}[\alpha_{U_{u-1}}(u-1)P(U_u|U_{u-1}, \Theta)P(O_u|U_u, \Theta)\beta_{U_u}(u)]}. \tag{68}$$

Viterbi algorithm. For classification, a two-dimensional Viterbi algorithm [46] is employed to search for the best combination of states with maximum a posteriori probability and map each block to a class. This process is equivalent to search for the state of each block using an extension of the variable-state Viterbi algorithm presented in [45]. If we search for all the combinations of states, suppose the number

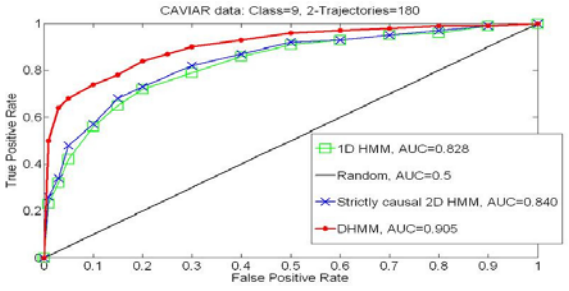


Fig. 13 ROC curve of DHMM, Strictly Causal 2D HMM [45] and 1D HMM for CAVIAR data

of states in each subset-state sequence U_u is $w(u)$, then the number of possible sequences of states at every position will be $M^{w(u)}$, which is computationally infeasible. To reduce the computational complexity, only N sequences of states with highest likelihoods out of the $M^{w(u)}$ possible states are used.

For simplicity, the DHMM model-based multiple trajectory classification algorithm is tested on the M=2 trajectory cases. The classification performance of proposed distributed 2D HMM-based classifier, causal 2D HMM-based classifier and traditional 1D HMM-based classifier are computed on subset of the CAVIAR [34] dataset. The classification results are reported in terms of ROC curve [22]. As can be seen, the proposed model outperforms the existing ones.

5 Summary

This chapter presented recent advances in motion trajectory-based video content modeling, retrieval and classification. The developments in the chapter focused on view-invariant representations for video retrieval and classification methods that are robust to unknown camera views and dynamic camera motion. Moreover, special attention was devoted in the chapter to the emerging focus on multiple motion trajectory analysis. In particular, we introduced techniques for video retrieval and classification based on multiple simultaneous motion trajectories.

Despite the success of recent developments in content-based video retrieval and classification, several fundamental open problems remain open and must be addressed prior to the widespread use of video retrieval and classification in commercial systems. We shall conclude with a brief discussion of some of the open problems and future trends in content-based video analysis.

5.1 Open Problems and Future Trends

1. **Query-Database Matching:** One of the main bottlenecks in content-based video analysis pertains to the correspondence between objects in the query and database. For instance, retrieval and classification of multiple motion trajectories

leads to different performance characteristics depending on the pairing of objects in the query and database. Current approaches to this problem require permutation of the object pairing and results in exponential computational complexity. Practical methods that address the matching of objects in the query and database must provide scalable solutions for large video databases [29][47].

2. **Partial Query-Database Representation:** A critical limitation of existing video retrieval and classification methods is the need for the representation of entire objects. For example, identical motion trajectory queries and database elements may be represented very differently due to various factors: differences in the instant of video capture initiation/termination, object occlusion, frame rate, etc. Development of schemes that are flexible and robust to partial information must be devised to accommodate the diversity of data captured in real-world video applications [25][48].
3. **Dynamic Databases:** Representation of large video databases that can effectively be used for video retrieval and classification is only useful if it can efficiently handle insertions and deletions of entries in the database. Specifically, practical video database systems cannot represent and index the entire video database from scratch every time a video object is removed or added to the archive. This requirement demands the development of efficient methods for dynamic updating and downdating techniques for the representation of motion trajectories [25][48].
4. **Feature Fusion:** Motion trajectories provide an important spatio-temporal feature for video analysis. However, integration of motion trajectories with other video features can be used to extract much more information about the spatio-temporal structure of video queries. Moreover, future multimedia systems will require a unified representation of a flexible feature space used for the content-based representation of text, audio, and visual data.
5. **Semantic Gap:** Visual feature-based techniques at low-level of abstraction have been explored in the literature. Current research efforts aim to extend content-based video analysis to high-level description of visual content. A critical need arises to bridge low-level features and high-level semantics by providing a unified representation between pixels and predicates for intelligent video systems.

References

1. Yuan, J., Wang, H., Zheng, W., Li, J., Lin, F., Zhang, B.: A Formal Study of Shot Boundary Detection. IEEE Transactions on Circuit and Systems for Video Technology, 168–186 (2007)
2. Hanjalic, A.: Shot-boundary detection: unraveled or resolved? IEEE Transactions on Circuit and Systems for Video Technoligy 12(2), 90–105 (2002)
3. Lienhart, R.: Reliable Transition Detection in Videos: A Survey and Practitioner's Guide. International Journal of Image and Graphics 1, 469–486 (2001)
4. Lelescu, D., Schonfeld, D.: Statistical sequential analysis for real-time scene hange detection on compressed multimedia bitstream. IEEE Transactions on Multimedia 5, 106–117 (2003)

5. Johansson, G.: Visual Perception of Biological Motion and a Model for its Analysis. Perception and Psychophysics 14(2), 201–211 (1973)
6. Yilmaz, A., Javed, O., Shah, M.: Object tracking: A survey. ACM Comput. Surv. 38(4), 45 (2006)
7. Zhao, T., Nevatia, R.: Tracking multiple humans in crowded environment. In: Proc. IEEE Int. Conf. Compt. Vision and Pattern Recognit., vol. 2, pp. 406–413 (2004)
8. Chang, C., Ansari, R., Khokhar, A.: Multiple Object Tracking with Kernal Particle Filter. In: Proc. IEEE Int. Conf. Compt. Vision and Pattern Recognit., vol. 1, pp. 566–573 (2005)
9. Schonfeld, D., Lelescu, D.: VORTEX: Video retrieval and tracking from compressed multimedia databases-multiple object tracking from MPEG-2 bitstream. Journal of Visual Communications and Image Representation, Special Issue on Multimedia Database Management 11, 154–182 (2000) (invited paper)
10. Hariharakrishnan, K., Schonfeld, D.: Fast object tracking using adaptive block matching. IEEE Transactions on Multimedia 7(5), 853–859 (2005)
11. Isard, M., Blake, A.: Condensation-Conditional Density Propagation for Visual Tracking. International Journal of Computer Vision 29(1) (1998)
12. Qu, W., Schonfeld, D., Mohamed, M.: Real-time distributed multi-object tracking using multiple interactive trackers and a magnetic-inertia potential model. IEEE Transactions on Multimedia 9, 511–519 (2007)
13. Chang, S.F., Chen, W., Meng, H.J., Sundaram, H., Zhong, D.: A Fully Automated Content-Based Video Search Engine Supporting Spatiotemporal Queries. IEEE Trans. Circuits Syst. Video Technol. 8(5), 602–615 (1998)
14. AbouGhazaleh, N., Gamal, Y.E.: Compressed Video Indexing Based on Object's Motion. In: Int. Conf. Visual Communication and Image Processing, VCIP 2000, Perth, Australia, pp. 986–993 (2000)
15. Katz, B., Lin, J., Stauffer, C., Grimson, E.: Answering questions about moving objects in surveillance videos. In: Proceedings of 2003 AAAI Spring Symp. New Directions in Question Answering, Palo Alto, CA (2003)
16. Rea, N., Dahyot, R., Kokaram, A.: Semantic Event Detection in Sports Through Motion Understanding. In: Enser, P.G.B., Kompatsiaris, Y., O'Connor, N.E., Smeaton, A., Smeulders, A.W.M. (eds.) CIVR 2004. LNCS, vol. 3115, pp. 21–23. Springer, Heidelberg (2004)
17. Bashir, F.I., Khokhar, A.A., Schonfeld, D.: A Hybrid System for Affine- Invariant Trajectory Retrieval. In: Proceedings of the 6th ACM SIGMM International Workshop on Multimedia Information Retrieval, New York (2004)
18. Bashir, F.I., Khokhar, A.A., Schonfeld, D.: Object trajectory-based activity classification and recognition using hidden Markov models. IEEE Transactions on Image Processing 16, 1912–1919 (2007)
19. Bashir, F.I., Khokhar, A.A., Schonfeld, D.: Real-time motion trajectory-based indexing and retrieval of video sequences. IEEE Transactions on Multimedia 9, 58–65 (2007)
20. Bashir, F.I., Khokhar, A.A., Schonfeld, D.: Real-time affine-invariant motion trajectory-based retrieval and classification of video sequences from arbitrary camera view. ACM Multimedia Systems Journal, Special Issue on Machine Learning Approaches to Multimedia Information Retrieval 12, 45–54 (2006)
21. The University of California at Irvine Knowledge Discovery in Databases (KDD) archive, http://kdd.ics.uci.edu
22. Fawcett, T.: Roc graphs: Notes and practical considerations for researchers, Technical Report, HP Labs, HPL-2003-4 (2004)

23. Vaswani, N., Chellappa, R.: Principal Components Null Space Analysis for Image and Video Classification. IEEE Trans. Image Processing (July 2006)
24. Chen, X., Schonfeld, D., Khokhar, A.: Robust null space representation and sampling for view invariant motion trajectory analysis. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
25. Chen, X., Schonfeld, D., Khokhar, A.: Localized Null Space Representation for Dynamic Updating and Downdating in Image and Video Databases. In: IEEE International Conference on Image Processing, ICIP (2009)
26. Sahouria, E., Zakhor, A.: A Trajectory Based Video Indexing System For Street Surveillance. In: IEEE Int. Conf. on Image Processing (ICIP), pp. 24–28 (1999)
27. Chen, W., Chang, S.F.: Motion Trajectory Matching of Video Objects. In: IS&T/ SPIE, pp. 544–553 (2000)
28. Bashir, F.I., Khokhar, A.A., Schonfeld, D.: Segmented trajectory based indexing and retrieval of video data. In: Proc. IEEE Int. Conf. Image Processing, pp. 623–626 (2003)
29. Ma, X., Bashir, F., Knokhar, A., Schonfeld, D.: Event Analysis Based on Multiple Interactive Motion Trajectories. IEEE Trans. on Circuits and Syst. for Video Technology 19(3) (2009)
30. Ma, X., Bashir, F., Knokhar, A., Schonfeld, D.: Tensor-based Multiple Object Trajectory Indexing and Retrieval. In: Proc. IEEE Int. Conf. on Multimedia and Expo. (ICME), Toronto, Canada, pp. 341–344 (2006)
31. Lathauwer, L., Moor, B.D., Vandewalle, J.: A multilinear singular value decomposition. SIAM Journal on Matrix Analysis and Applicat. (SIMAX) 21(4), 1253–1278 (2000)
32. Lathauwer, L.D., Moor, B.D.: From Matrix to Tensor: Multilinear Algebra and Signal Processing. In: Proc. 4th IMA Int. Conf. Mathmatics in Signal Process., pp. 1–15 (1996)
33. Harshman, R.A.: Foundations of the PARAFAC procedure: Model and Conditions for an "explanatory" multi-mode factor analysis. UCLA Working Papers in Phonetics, pp.1-84 (1970)
34. The Context Aware Vision using Image-based Active Recognition (CAVIAR) dataset, http://homepages.inf.ed.ac.uk/rbf/CAVIAR/
35. Brezeale, D., Cook, D.J.: Automatic Video Classification: A Survey of the Literature. IEEE Transactions on Systems, Man and Cybernetics, Part C: Applications and Reviews 38(3) (2008)
36. Rabiner, L.R.: A tutorial on hidden markov models and selected applications in speech recognition. Proceedings of the IEEE 77, 257–286 (1989)
37. Starner, T., Pentland, A.: Real-Time American Sign Language Recognition From Video Using Hidden Markov Models, Technical Report, MIT Media Lab, Perceptual Computing Group, vol. 375 (1995)
38. Raphael, C.: Automatic Segmentation of Acoustic Musical Signals Using Hidden Markov Models. IEEE Transactions on Pattern Analysis and Machine Intelligence 21(4), 360–370 (1999)
39. Bashir, F.I., Khokhar, A.A., Schonfeld, D.: HMM based motion recognitionsystem using segmented pca. In: IEEE International Conference on Image Processing (ICIP 2005), vol. 3, pp. 1288–1291 (2005)
40. Ma, X., Schonfeld, D., Khokhar, A.: Distributed multidimensional hidden Markov Model: theory and application in multiple-object trajectory classification and recognition. In: SPIE International Conference on Multimedia Content Access: Algorithms and Systems, San Jose, California (2008)
41. Ma, X., Schonfeld, D., Khokhar, A.: Image segmentation and classification based on a 2D distributed hidden Markov model. In: SPIE International Conference on Visual Communications and Image Processing (VCIP 2008), San Jose, California (2008)

42. Ma, X., Schonfeld, D., Khokhar, A.: Distributed Multi-dimensional Hidden Markov Models for Image and Trajectory-Based Video Classification. In: IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2008), Las Vegas, Nevada (2008)
43. Ma, X., Schonfeld, D., Khokhar, A.: Video Event Classification and Image Segmentation Based on Non-Causal Multi-Dimensional Hidden Markov Models. IEEE Transactions on Image Processing, T-IP (May 2009) (to appear)
44. Baum, L.E., Petrie, T., Soules, G., Weiss, N.: A maximization technique occuring in the statistical analysis of probabilistic functions of markov chains. Ann. Math. Stat. 1, 164–171 (1970)
45. Li, J., Najmi, A., Gray, R.M.: Image classification by a two-dimensional hidden markov model. IEEE Trans. on Signal Processing 48, 517–533 (2000)
46. Schonfeld, D., Bouaynaya, N.: A new method for multidimensional optimization and its application in image and video processing. IEEE Signal Processing Letters 13, 485–488 (2006)
47. Ma, X., Khokhar, A., Schonfeld, D.: Robust video mining based on local similarity alignment of motion trajectories. In: IEEE Conference on Image Processing (ICIP 2009), Cairo, Egypt (2009)
48. Ma, X., Schonfeld, D., Khokhar, A.: Dynamic updating and downdating matrix SVD and tensor HOSVD for adaptive indexing and retrieval of motion trajectories. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2009), Taipei, Taiwan (2009)

Part II
High-Dimensional Video Representation

Three Dimensional Information Extraction and Applications to Video Analysis

Arturo Donate and Xiuwen Liu

Abstract. This chapter explores the idea of extracting three dimensional features from a video, and using such features to aid various video analysis and mining tasks. The use of 3D information in video analysis is scarce in the literature due to the inherent difficulties of such a system. When the only input to the system is a video stream with no previous knowledge of the scene or camera (a typical scenario in video analysis), computing an accurate 3D representation becomes a difficult task; however, several recently proposed methods can be applied to solving the problem efficiently, including simultaneous localization and mapping, structure from motion, and 3D reconstruction. These methods are surveyed and presented in the context of video analysis and demonstrated using videos from TRECVID 2005; their limitations are also discussed. Once an accurate 3D representation of a video is obtained, it can be used to increase the performance and accuracy of existing systems for various video analysis and mining tasks. Advantages of utilizing 3D representation are illustrated using several of these tasks, including shot boundary detection, object recognition, content-based video retrieval, as well as human activity recognition. The chapter concludes with a discussion on limitations of existing 3D methods and future research directions.

1 Introduction

With all the current and ever-increasing uses for video databases, the need for fast and accurate mining systems grows every day. On account of this, video analysis is quickly becoming a prominent field of active research. This field can be broken

Arturo Donate
Department of Computer Science, Florida State University, Tallahassee, FL 32312
e-mail: donate@cs.fsu.edu

Xiuwen Liu
Department of Computer Science, Florida State University, Tallahassee, FL 32312
e-mail: liux@cs.fsu.edu

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 85–114.
springerlink.com

down into five main areas: multimodal analysis, video representation, summarization, browsing, and retrieval [43]. The area of multimodal analysis studies aspects such as automatic shot boundary detection and key frame extraction. Video representation studies different ways of representing a video. Video summarization studies the problem of summarizing the entire content of a video and generating metadata. Video browsing handles the problem of browsing sections of the video without viewing it in its entirety. Finally, video retrieval deals with the problem of retrieving videos (either an entire video, or particular sections of one) based on some user-defined content-based query.

Video analysis tasks typically employ the use of image descriptors and other statistical measurements. Some of the more popular descriptors are Harris features [16], SIFT [22], and SURF [3] descriptors. Such descriptors rely on image gradients and statistics to detect salient regions. Although such features have been proven to be successful in various tasks in the literature, they are still unable to model the inherent three dimensional structures of many real world objects. As such, these two dimensional descriptors will always be limited in their explicative power.

Ultimately, the aim of this study is to extract three dimensional structure of the environment observed in a video, and use this information to solve various video analysis and mining tasks. The motivation for such an approach stems from the recent advances made in various areas of computer vision in recent years. There has been much work done in the areas of simultaneous localization and mapping (SLAM) using a single camera, as well as structure from motion and 3D reconstruction from a single view. This chapter will present several approaches to extract 3D information from a scene using only the video stream as input; such methods can be adapted to video analysis since a video is essentially a view of a scene obtained from a single monocular camera. We exclude special effect frames, where 3D information is not well defined.

In a real-world scenario, a given video clip can be arbitrarily large. For this reason, the methods chosen here perform accurate 3D reconstructions with real time performance, typically 30 frames per second on typical hardware. In order to illustrate the idea that such methods are excellent candidates for various video analysis and mining tasks, several videos from the TRECVID 2005 database [33] were reconstructed in 3D. The following sections contain sample video frames as well as reconstruction results in order to illustrate their performance.

This chapter is organized as follows: the next section explores several different approaches for obtaining 3D information using only the image frames obtained from a monocular video; such methods employ the use of structure from motion, simultaneous localization and mapping, as well as 3D reconstruction. The following section discusses several ways in which this information can be utilized to perform several video analysis and mining tasks including accurate and intrinsic shot boundary detection, search and retrieval of videos in a database, object detection, as well as recognition and analysis of human activities in a video. The final section presents a summary of the current state of research on this particular topic, discusses several problems and limitations, and presents several promising directions for future research.

2 Extracting 3D Cues from Videos

The use of 3D information has not been widely used in video analysis due to the inherent difficulties of extracting accurate depth information of a scene from a single monocular view. This section provides a review of recent works dealing with the problem of estimating 3D geometry of a scene from a single camera. The works presented in this section employ the use of various techniques including structure from motion, simultaneous localization and mapping, and real time 3D reconstruction.

Structure from motion (SFM) methods estimate the 3D structure of a scene using observations of objects and the motion of a camera. They can be very useful in extracting the structure of a scene, particularly when the scene is static and the only motion present is that of the camera. Recent work has extended SFM approaches to incorporate dynamically moving objects into the 3D estimation. Such advancements may prove extremely valuable to the video analysis community, since it would be an ideal candidate for reconstructing an arbitrary video in 3D.

Simultaneous localization and mapping (SLAM) is the process of determining both the location of the system in a given world, as well as mapping the scene observed by this system. This technique is often used in robotics to allow the robot to model its surroundings. There are two basic categories of SLAM algorithms, EKF-SLAM and FastSLAM [7]. Typically, EKF-SLAM approaches keep track of a small subset of high quality features over time. Predicted motions and locations are calculated using the Extended Kalman Filter [7, 13, 35] according to some predefined state transition functions. FastSLAM, on the other hand, approaches the problem by estimating the robot position then estimating landmarks based on the robot pose estimate [27]. Estimations are calculated using a particle filter, and each landmark is measured independently of all the others (unlike EKF-SLAM, which often models the covariance between landmarks). In doing so, FastSLAM methods are able to track a much larger set of points over time. Both of these SLAM methods can function with various forms of input, including cameras as well as laser scanners. Typically, stereo cameras are used for vision-based SLAM [7, 42]. In recent years, however, much research has been geared towards single view SLAM, where the only input to the system is a video stream from a monocular camera. Much like typical SLAM algorithms, these recent methods must accurately calculate the 3D location of the observed features. Since the only input to such systems is a video stream obtained from a monocular camera, any standard video may be used as input to the SLAM algorithm, in an attempt to calculate 3D locations of salient features in the scene.

Methods based on 3D reconstruction typically use a more geometric approach to estimating depth of features. Some simulate the availability of stereo image pairs by using video frames that are at a time t apart from each other. By doing so, they are able to employ the use of essential and fundamental matrices in order to calculate 3D reconstructions using stereo reconstruction methods such as the 5-point and 8-point algorithms [13, 17, 20, 29, 30]. Typically, such methods achieve high accuracy and are computationally efficient.

2.1 Structure from Motion

Structure from motion is an area of computer vision dealing with the problem of extracting an estimate of the structure of each object in the scene based on observations of object and camera motions. It is useful in estimating the three dimensional structure of a scene, particularly when viewed from a monocular camera moving under arbitrary motion.

Recent work by Tola *et al.* [40] shows that structure from motion approaches can be extended to handle independently moving objects in a scene, thus solving the multi-body structure from motion problem for the single camera case. In other words, the problem is to estimate a 3D reconstruction of the scene containing a camera under arbitrary motion as well as independently moving objects. The authors make several assumptions in order to solve the problem. First, they assume there are only two types of motion present in the scene: camera motion and the independently moving objects. Each motion is assumed to be slow and constant in one direction. Second, the scene does not contain any abrupt changes in lighting over time, and each object in the scene is rigid. Finally, the intrinsic parameters of the camera are assumed to be known beforehand.

The first step of the algorithm is to solve the correspondence problem over a large number of frames. This is done by using the Lucas-Kanade tracking algorithm implemented into the OpenCV library [5], using the trajectory of points to solve correspondences. For each independently moving object, a fundamental matrix F_i can be computed that satisfies the epipolar constraint, which states that given two views of a scene and a set of corresponding points $\{x, x'\}$ in each of these views, these points are related to each other via the fundamental matrix F as:

$$x'^T F x = 0, \tag{1}$$

where x is a set of points in one view, and x' is the corresponding set of points in the other view.

Using a RANSAC-based fundamental matrix estimation, the F matrix estimated by random sampling will correspond to the points of the most prominent motion in the scene (typically background points due to camera motion), leaving all other trajectories as outliers. If the same process is repeated on the outlying trajectories, the next F matrix computed should correspond to the next most dominant motion; it can be repeated to estimate a fundamental matrix for all independently moving objects in the scene.

In order to simplify the calculation, trajectories are divided into one of four categories. The first are *complete* trajectories, which are trajectories visible in all the frames of the video. The second are *incomplete right* trajectories which begin somewhere within the video and continue to the end. The third are *incomplete left* trajectories, which are present in the beginning of the video, but end before the video ends. The fourth and final are *incomplete* trajectories, which begin and end within the video, and are not visible in the first or last frames. The first three categories of trajectories are used. This segmentation of individual trajectories has four steps:

1. Perform RANSAC using first and last frame in the video to extract first F matrix.
2. Perform RANSAC on outliers of previous step to find independently moving objects (IMO).
3. Perform RANSAC on the IMO points of step 2 using the first and next-to-last frame to find complete-left trajectories.
4. Perform RANSAC on the IMO points of step 2 using the second and last frame to find complete-right trajectories.
5. Repeat steps 3 and 4 on the rest of the frames.

The *incomplete* trajectories, along with points labeled as outliers after performing RANSAC n number of times, are discarded from the rest of the algorithm.

Given a fundamental matrix F_i, it is possible to decompose it into rotation and translation matrices since the intrinsic parameters of the camera are assumed to be known. With these matrices, the points can be triangulated to find the 3D location in the world coordinate frame using the triangulation method described by Hartley and Zisserman [17]. The overall reconstruction is then refined by minimizing the re-projection error of the 3D points using global bundle adjustment, parameterized by the equation:

$$\sum_{i=1}^{m}\sum_{j=1}^{n} d(x_i^j, P^j, X_i), \tag{2}$$

where x_i^j represents the i^{th} image point seen with camera j, P^j is the set of all camera matrices, and X_i is the set of all 3D points calculated from all the cameras. Minimizing equation 2 with respect to P^j and X_i is the final step in the 3D structure estimation.

The importance of this work stems from the fact that this is one of the few available methods that can reconstruct a scene observed by a single camera, while also reconstructing the objects moving independently in the scene. This is crucial to estimating 3D structure for video analysis and mining, since most other approaches assume a static scene and are unable to reconstruct the moving objects.

2.2 Simultaneous Localization and Mapping

One of the most popular single camera SLAM frameworks is MonoSLAM by Davison *et al.* [7]. The method presents an efficient approach capable of localizing a single monocular camera and simultaneously estimating the relative 3D structure of the environment seen by the camera, all with real time performance.

The world is modeled as a probabilistic 3D map and the system stores the current state of the camera, interesting features observed, as well as the uncertainty of the estimates. Features may be added or removed from the map according to their uncertainty. The map is initialized at startup and is continuously being updated via an Extended Kalman Filter (EKF) [7, 13, 35].

The Kalman filter [7, 13, 35] is used to determine the state of a system based on noisy measurements. For example, it can be used to compute the position and velocity of an object given only information about the object's position at previous points

in time, along with an uncertainty factor for each measurement. In the Kalman filter, state transition models are built from linear equations. The Extended Kalman Filter is an extension to the traditional Kalman filter where state transition models to be built from non-linear equations by linearization of these non-linear functions [35].

The map is made up of a state vector $\hat{x}$ and covariance matrix P. The state vector is an estimate of the camera and visual features mapped in the world, composed of the camera vector $\hat{x}_v$ and all the feature vectors $\hat{y}_i$. The covariance matrix P is a square matrix that can be divided into individual sub-matrix elements, allowing the probability distribution to be approximated by a single multivariate Gaussian distribution. The state vector and covariance matrix are defined as:

$$\hat{x} = \begin{pmatrix} \hat{x}_v \\ \hat{y}_1 \\ \hat{y}_2 \\ \vdots \end{pmatrix}, \quad P = \begin{bmatrix} P_{xx} & P_{xy_1} & P_{xy_2} & \cdots \\ P_{y_1x} & P_{y_1y_1} & P_{y_1y_2} & \cdots \\ P_{y_2x} & P_{y_2y_1} & P_{y_2y_2} & \cdots \\ \vdots & \vdots & \vdots & \ddots \end{bmatrix}. \tag{3}$$

The camera and feature vectors describe the main scene elements. The elements of the camera vector $\hat{x}_v$ essentially describe the extrinsic parameters of the camera, while the elements of the feature vectors $\hat{y}_i$ describe the 3D location of each feature in the world coordinate frame. These vectors are defined as:

$$\hat{x}_v = \begin{pmatrix} r^{WC} \\ q^{WC} \\ v^W \\ \omega^W \end{pmatrix}, \quad y_i = \begin{pmatrix} x_i \\ y_i \\ z_i \\ \theta_i \\ \phi_i \\ \rho_i \end{pmatrix} \tag{4}$$

respectively. For the camera vector, the term r^{WC} stores the current 3D position of the camera in world coordinates, q^{WC} is an orientation quaternion describing the orientation of the camera in terms of azimuth and elevation, v^W is the velocity of the camera, and ω^W is the angular velocity. For the feature vector, the first three terms (x_i, y_i, z_i) define the 3D location of the camera's optical center at the time the feature was first observed, (θ_i, ϕ_i) is the azimuth and elevation for the ray $\mathbf{m}(\theta_i, \phi_i)$ from the camera to the observed feature, and ρ_i is the inverse depth $(\frac{1}{d_i})$ of the feature along this ray. These parameters keep track of a 3D point located at:

$$\begin{pmatrix} x_i \\ y_i \\ z_i \end{pmatrix} + \frac{1}{\rho_i} \mathbf{m}(\theta_i, \phi_i). \tag{5}$$

The geometry of this scene can be seen in Figure 1.

The primary function of the map is to provide localization of the camera, not a dense mapping of the environment. Because of this, a sparse set of landmarks are used for localization. Keeping the number of landmarks small helps achieve real time performance. In order for this probabilistic map to function properly, the

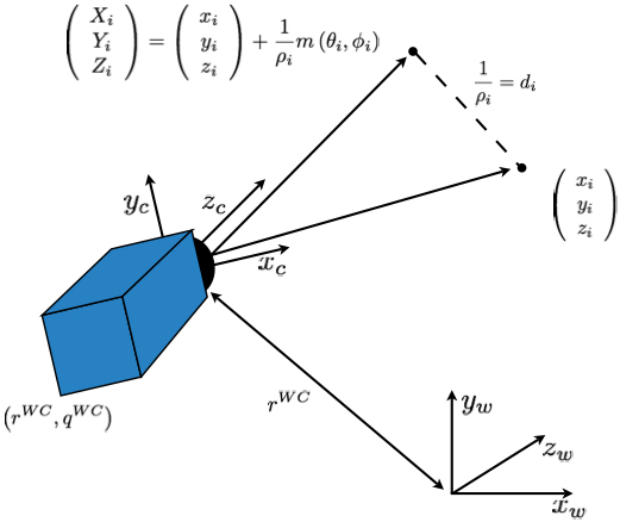


Fig. 1 Basic geometry of the scene, for details see [28].

assumption is made that the map is rigid and all landmarks are stationary. The only motion in the scene is caused by the arbitrary camera motion.

Landmarks are initially found in the image by searching for salient image regions using the Shi and Tomasi operator [36]. A window of 11×11 pixels is used to extract an image patch at the given location. The patch is assumed to be planar and the surface normal is initially set to be equal to the optical axis of the camera (to be updated later). This image patch is projected as an image and saved as a template for matching. When a new view of this feature occurs during the execution of MonoSLAM, the new visible image is compared to this template. These templates are created from the first time a landmark is observed, and are never updated.

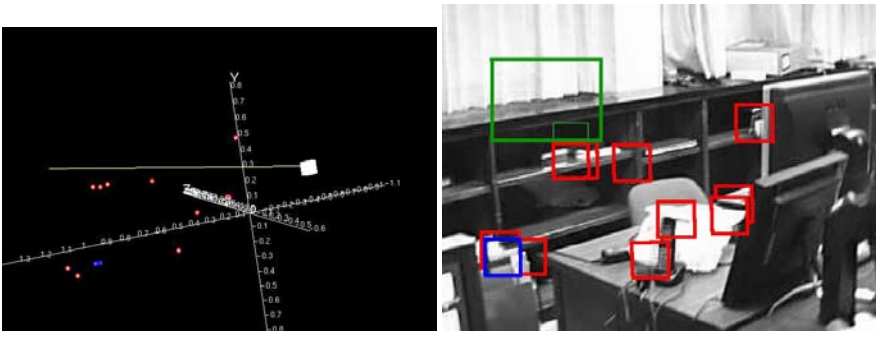


Fig. 2 Left image shows the camera viewpoint. Right image shows a 3D representation of the probabilistic map.

When a feature is detected, it is immediately inserted into the map with a large possible depth range of $[1, \text{inf}]$, coded as a Gaussian [28]. As points are re-observed over time, the Extended Kalman Filter re-estimates the depth of each feature until it converges to a more accurate depth value. For this to occur, there must be enough parallax observed by the camera. Otherwise, the system assumes the features are at infinity. Such features contribute only to estimating the camera position.

In order to model the movement of the camera, the authors employ a "constant velocity, constant angular velocity model" [7]. This model does not necessarily assume the camera moves at a constant velocity, but instead expects that accelerations occur with a Gaussian profile. In other words, there are no sudden accelerations. At a given point in time, an acceleration a^W and angular acceleration α^R cause a change in the velocity and angular velocity of the camera, described as:

$$n = \begin{pmatrix} V^W \\ \Omega^R \end{pmatrix} = \begin{pmatrix} a^W \, \Delta t \\ \alpha^R \, \Delta t \end{pmatrix} \tag{6}$$

which in turn is used to update the camera state vector in the following manner:

$$f_v = \begin{pmatrix} r^{WC}_{new} \\ q^{WC}_{new} \\ v^W_{new} \\ \omega^W_{new} \end{pmatrix} = \begin{pmatrix} r^{WC} + (v^W + V^W)\Delta t) \\ q^{WC} \times q((\omega^W + \Omega^W)\Delta t) \\ v^W + V^W \\ \omega^W + \Omega^W \end{pmatrix} \tag{7}$$

This updated camera vector is accompanied by the uncertainty measurement of the camera after the motion (required by the EKF), which is expressed as

$$Q_v = \frac{\partial f_v}{\partial n} P_n \frac{\partial f_v^T}{\partial n}, \tag{8}$$

where P_n is the covariance of the noise vector n described in Equation 6. The rate of growth in this uncertainty model depends on the covariance matrix P_n. When P_n is small, the change in velocities is small and the system expects a smooth camera motion. Likewise, when P_n is large, the uncertainty also grows large in order to cope with rapid camera motions.

In order to provide good localization while achieving real time performance, the system continuously measures 12 features at a time. If the number of currently visible features falls below 12 at any point in time, a new feature is added to the map. If a feature is expected to be seen but fails to be found, the system continues trying to look for it until the ratio of times found and times not found falls below a certain threshold. When this happens, the feature is removed from the map entirely. This usually happens when a feature is on a moving object, occluded, or drastically changes in appearance (possibly caused by sudden changes in lighting).

Figure 2 shows the algorithm executing at a point in time. The left image illustrates the current frame of the video, while the right image illustrates the visual 3D representation of the probabilistic map. Here, red features are correctly measured,

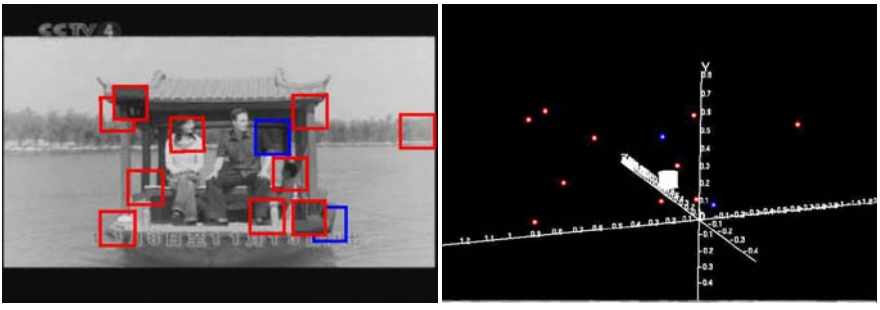


Fig. 3 Left image shows a sample image being tracked using the MonoSLAM algorithm, while the right image shows the corresponding 3D location of the features. The reconstruction was performed on videos from the TRECVID 2005 database [33].

blue features have failed measurements, and green features are currently being initialized into the system.

This MonoSLAM algorithm has several benefits that make it an excellent candidate for extracting 3D information from a video. The only input to the system is a video stream from a monocular camera, and it is able to extract 3D points in the scene in real time from the input video. Unfortunately, however, the runtime of the algorithm is $O(N^2)$ where N is the number of points in the probabilistic map. In order to achieve real time performance, the authors place an upper bound of $N = 100$ and perform the SLAM cycle with a sparse set of features. Since the primary goal is to provide accurate localization, a sparse set of points works nicely to solve this problem. In order to map the environment with a dense cloud of points, however, the value of N would have to be much larger than 100, causing a larger degradation in performance.

By sacrificing the real time performance of the system and increasing the number of features tracked, it is possible to extract a relatively dense set of 3D points from the observed scene. Figure 3 illustrates the performance of estimating 3D locations of features on one of the videos from the TRECVID 2005 database [33]. There are some limitations to the process, however. First, this system is unable to extract 3D points from dynamically moving objects. Tracked features in such objects will fail to be recognized and be removed from the map. Second, the camera must have some motion in order to achieve 3D point location accuracy. If the camera is static, the system will assume distances from the camera to all the feature points are infinity, and hence all lie on a single plane.

2.3 Real Time 3D Reconstruction

In the work by Mouragon *et al.* [29, 30], the authors propose a method to perform real time 3D reconstruction of a scene viewed by a single monocular camera. The method finds features in video frames and matches them across frames to exploit

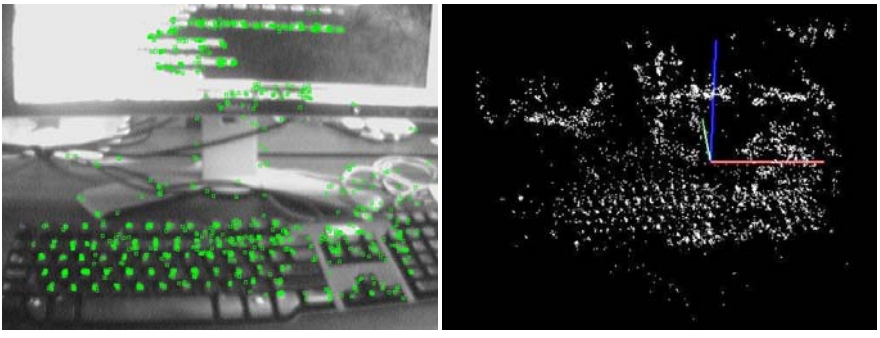


Fig. 4 Left image shows several feature points being tracked over time. Right image shows these feature points in a 3D space.

parallax and estimate 3D structure. By tracking features over time, the authors use separate video frames to simulate a stereo view of a scene.

The first step of the method is to find salient image features in the frames of the video. For this, the Harris corner detector [16] is employed. For a given image point in the first frame I_1, the corresponding point is searched inside a region of interest in the second frame I_2. Zero-mean normalized cross correlation (ZNCC) is used as a distance measure to find the matching point inside the region of interest. Pairs with high ZNCC scores are selected as corresponding points between the pair of video frames I_1, I_2. The left image of Figure 4 shows a sample video frame with the detected Harris features.

In order to achieve good reconstruction results, the two frames used must be far enough away from each other so that enough parallax is visible, but not too far so that they still have a large number of points in common. To achieve this, not all frames are used for 3D triangulation, only selected key frames. When the system is first initialized, the first frame is always considered a key frame and is considered I_1. The second key frame I_2 is selected so that there are as many image frames as possible between I_1 and I_2, but the two key frames have at least M points in common. In [29, 30], the authors use a value of $M = 400$.

A third key frame I_3 is selected in a similar manner as the first two key frames. The distance between I_2 and I_3 should be as large as possible with I_2 and I_3 having at least M points in common, and I_1 and I_3 having at least M' points in common. The authors use a value of $M' = 300$ in the experiments presented in [29, 30]. The coordinate system of I_1 is set as the world coordinate frame, and the relative poses between all the views are calculated using the 5-point algorithm along with RANSAC. The 3D points are then triangulated using I_1 and I_3.

After initialization, estimating the camera pose C_i at time i is done by first selecting a set of points p visible from the last video frame, as well as the last key frame. The projection of these points from previous camera poses (C_{i-1}, C_{i-2}, ...) is known, as well as their 3D coordinates. Using these 3D coordinates, the authors employ the use of Grunert's pose estimation algorithm to find camera pose [15].

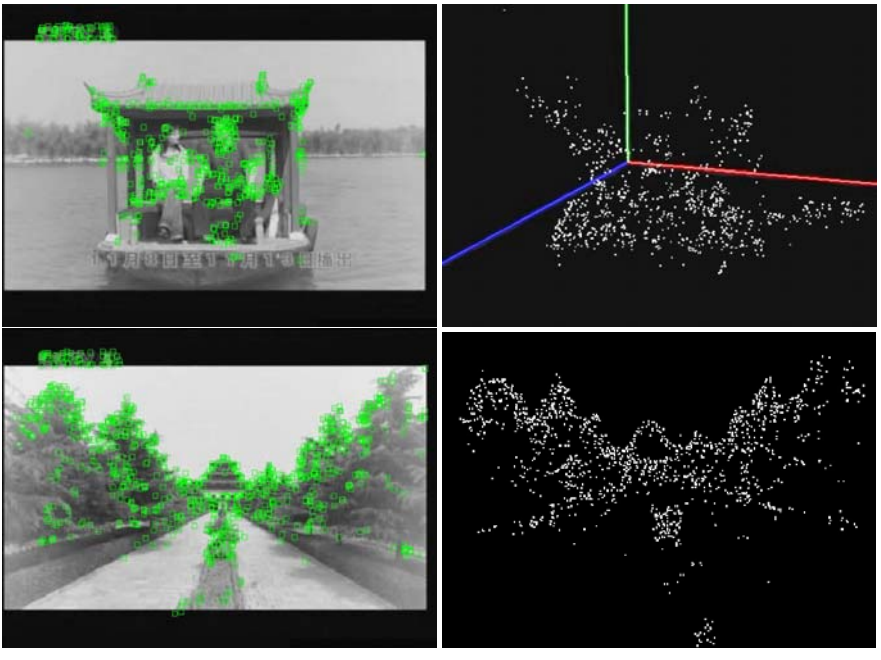


Fig. 5 3D reconstruction of a video using the TREVID 2005 database [33]. The left column shows several feature points being tracked over time. The right column shows these feature points in a 3D space.

Figure 4 shows a sample image frame from a monocular camera with detected Harris features, and the corresponding 3D reconstruction.

As previously mentioned, only a small set of key frames are used for 3D estimation. For a new key frame to be added by the system, one of two conditions must be met. If either the number of matched points between the current image frame and the last key frame drops below the previously mentioned threshold M, or if the uncertainty of the camera pose becomes too large. If either of these two conditions is true, the system selects the next incoming frame as a key frame.

After a new key frame is added to the system, a bundle adjustment optimization takes place. This optimization is a Levenberg-Marquard minimization of a cost function f^i at time step i defined as

$$f^i(C^i, P^i), \tag{9}$$

where C^i contains the extrinsic parameters of the current camera, and P^i is the set of 3D points visible from the camera at pose C^i. It is essentially a minimization of the re-projection error of the points. The authors here recognize that bundle adjustment calculations become increasingly expensive as the number of key frames grows. To solve this problem and retain real-time performance, the authors perform local

bundle adjustment, where the process only optimizes the extrinsic parameters of the last n cameras, taking into account the projection of the points in the last N frames.

The cost function to be minimized by local bundle adjustment can now be represented as

$$f^i(C^i, P^i) = \sum_{C_i \in \{C_i - N + 1;\, C_i\}} \sum_{p_j \in P^j} d^2\left(p_{ij}, K_i p_j\right), \tag{10}$$

where the right hand side of the equation is the sum of the squared Euclidean distances between the estimated projection of a point from the i^{th} camera ($K_i p_j$), and its observed location in the image (p_j). It is important to note that K_i is the projection matrix of the i^{th} camera C_i, and is composed of the extrinsic parameters (calculated from the camera pose) as well as the intrinsic parameters which are assumed to be known by the system.

In this local bundle adjustment scheme, N and n are the two main parameters affecting optimization performance. Larger values take more cameras and more key frames into account, resulting in better data accuracy at the cost of calculation time. Smaller values provide quicker performance, but may not have a very high degree of accuracy. At system startup, when the number of key frames is below some threshold N_f, the system performs global bundle adjustment; when $i > N_f$, the values for n and N are set and local bundle adjustment is performed. As long as the value of N_f is relatively low (approximately 20), there should be little to no loss of performance.

Out of all the possible ways discussed in this chapter for estimating 3D structure of a video, this one appears to be by far the most effective. The method is not only able to extract very accurate estimations for camera pose and 3D scene structure, but it can do so in real time. These two factors are crucial for use in a video analysis scenario. This system offers accurate reconstruction with minor errors, and is able to compute them in an efficient manner. Figure 5 illustrates the reconstruction results obtained using sample videos from the TRECVID 2005 database [33]; the frame in the first row contained almost zero camera motion, making an accurate reconstruction very difficult. Still, however, the method was able to estimate the depth in the scene quite accurately. The frame in the second row corresponds to a point in time where the camera was moving around in the scene. As illustrated, the reconstruction results were very successful in this particular case.

2.4 Other Methods for 3D Estimation

In [20], Klein and Murray present a framework for performing real time augmented reality using a single monocular camera. This method borrows many aspects from the method previously discussed by Mouragon *et al.* [29, 30], but makes some improvements upon it. First, the method splits the tasks of tracking and mapping into separate threads, taking advantage of the fact that most modern computers contain multiple CPU cores. Not all frames are used for mapping, only a small subset of frames. These key frames do not need to be processed in real time, as long as the computation is finished before the next key frame is retrieved.

The method also deals with rapid camera motions by using a 4-level Gaussian pyramid of the image, and performing a coarse-to-fine tracking procedure. When a new key frame is first processed, the system initially searches for a small number of the coarsest-scale features (typically, 50 features are used) according to the previous camera pose estimate, and these are used to calculate the updated camera pose. Then, a large number of fine-level features are searched for in the image (in their experiments, the authors used 1000 features) according to the updated camera pose, and a final pose estimate is calculated.

During tracking, the method keeps a fraction of the number of successfully tracked features over the total number of features. If this fraction drops below a threshold T_{min}, the tracking is considered poor and no new key frames are inserted until the fraction goes above the threshold again. This is so because poor tracking results indicate there are poor tracking conditions currently in the frame (caused by occlusion, motion blur, etc.). While this fraction is above T_{min}, several conditions must be met in order for the system to add a new key frame. First, the previous key frame must be at least 20 frames apart. This is so that the stereo calculations have a wide baseline to work with, and can achieve more accurate reconstructions. Also, the camera must be a minimum distance away from the closest feature in the map. This prevents a problem that may occur of stationary features corrupting the map.

During initialization, the mapping uses stereo to build the initial map. Since the input video stream comes from a single monocular camera, the system requires some user intervention at startup. The user grabs two frames so that the camera is translated between them, creating a stereo image pair. Next, 1000 points are tracked at the coarsest pyramid level and the 5 point algorithm along with RANSAC is used to estimate the essential matrix. The essential matrix is related to the previously mentioned fundamental matrix as described by the following equation

$$E = K^T F K, \tag{11}$$

where F is the fundamental matrix, and K is the intrinsic camera matrix. The equation defining the essential matrix is

$$\hat{x}'^T E \hat{x} = 0, \tag{12}$$

where $\hat{x} \leftrightarrow \hat{x}'$ are corresponding points in stereo image pairs, and the $\hat{x}$ notation denotes the set of image points x after being converted to normalized image coordinates [17]. Correlation between frames is performed using zero-mean sum of squared differences, and are calculated on pixels along the epipolar line of the second image. At this step, the corresponding points in the images can be triangulated to determine the 3D location of the points in the world coordinate frame. Once the points have been estimated, the dominant plane in the image is estimated using RANSAC in order to find the best surface for projecting augmented reality objects.

The pose estimate and 3D point locations of the system are optimized using local bundle adjustment to minimize the re-projection error of the points. This process is done much in the same way as it was done by Mouragon *et al.* [29, 30], as previously described in Equations 9 and 10.

Another improvement presented by Klein and Murray over the real time 3D reconstruction algorithm of Mouragon *et al.* [29, 30] has to do with the incorporation of data association refinement. In the case that the local bundle adjustment optimization converges, and the system does not need to add new key frames, then the thread responsible for mapping can use its free CPU cycles to improve the map. This is done by re-measuring features in previous key frames. Given a particular feature already in the map, the original measurement was based on two image frames (used to simulate a stereo pair). During this step, if the feature is visible in other key frames, it is re-measured and re-inserted into the map. This refinement step is given very low priority in the system and only occurs if the mapping thread is idle (i.e., optimization has converged, and no new key frames are being inserted into the system). As soon as a new key frame is inserted, this process is interrupted in order to handle the new frame and perform the necessary calculations.

3 Video Analysis Using 3D Information

As mentioned earlier in this chapter, the number of published methods in the area of video analysis which incorporate the use of 3D information is somewhat limited. This is likely due to the difficulty of extracting accurate 3D measurements from a video stream with no prior knowledge. Although still a relatively new approach, several works exist which employ methods similar to the ones described so far. These works attempt to solve problems such as shot boundary detection, object recognition, content-based video retrieval, as well as human activity recognition. These methods successfully show that 3D cues provide enough information to solve various video analysis and mining tasks.

Shot boundary detection refers to the development of algorithms for detecting boundaries between shots (i.e., the point in time when one shot ends and another begins). The literature is filled with methods for performing shot boundary detection using 2D features and statistical approaches, including various surveys and comparisons of different methods [2, 4, 9, 14, 21, 44]. For example, Abd-Almageed [1] proposed a method for shot boundary detection that works by first extracting multivariate feature vectors. These vectors are then placed in a matrix which is later decomposed via singular value decomposition. The claim is that tracking the rank of the singular vectors using a sliding window offers an accurate measurement of shot boundaries. Such an approach, however, does not take into consideration the inherent three-dimensional structure of the observed scene, and thus may fail in certain scenarios.

Object recognition is one of the classic problems in computer vision, and deals with searching for instances of objects in images and video. Queries may pertain to a specific object, or a particular class of objects. One such work is done by Visser *et al.* [41]. In [41], the authors first detect segmented blobs from image frames using the Kalman filter. Each blob is then classified under one of three categories (person,

automobile, other) using eigenvector decomposition. Finally, several temporal filters are applied in order to perform sequential classification over all the frames of the video. The drawback of methods such as this one lies in the fact that they do not take into consideration 3D shape or curvature. Most objects in real life are three dimensional; therefore to truly generate a list of descriptive features for an object, one must take the third dimension into account.

Content-based video retrieval is another popular problem. The idea is to classify and query a database of video clips based not on meta tags, but rather on the actual content within each video. One such example of a content-based video retrieval system that provides excellent performance is the "Video Google" framework developed by Sivic and Zisserman [38, 37, 39]. The main idea is to create a dictionary of visual words used to index video clips. When the user inputs to the system an image of an object to look for, the system extracts the visual words from the input image and searches for videos containing the same visual words (i.e., videos which contain the same object). These visual words are built by first extracting salient features in the image using one of two feature detectors. The first was proposed by Mikolajczyk and Schmid, and is constructed by elliptical shape adaptation around a Harris interest point [26]. The second feature detector, proposed by Matas *et al.* [25], detects stable regions of an image after applying several intensity thresholds [25]. Each detected area is then described using a SIFT descriptor [22]. The first detector has a bias to be centered at image corners, while the second tends to be centered around high contrast blobs. Neither, however, makes use of 3D information. Recognizing the importance of incorporating 3D measurements into their content-based video retrieval system, Sivic and Zisserman challenge the readers to extend their framework to incorporate three dimensional information into their feature descriptors, and provide the readers with possible research directions for doing so [39].

Activity recognition deals with the problem of tracking and recognizing the activities of individual agents in videos. Typically, this area is applied to humans in an attempt to track and classify human actions. One such dealing with this problem is the work done by Kellokumpu *et al.* [19]. The authors use texture descriptors to describe human movements. Originally proposed by Ojala *et al.* [32], the local binary patterns descriptor represents a texture using binary patterns around a neighborhood. Image textures can thus be represented as histograms of these binary patterns. Kellokumpu *et al.* use these descriptors to describe the movements of humans in a spatio-temporal manner. These features are then modeled using Hidden Markov Models. The authors show that the performance of their proposed method exceeds that of several other methods which attempt to solve the same task. Their approach is not without drawbacks, however. Since the descriptors used here rely on texture measurements, they are sometimes unable to cope with dynamic backgrounds. Texture measurements, although very successful in the applications, do not encompass enough information about a subject to be the sole descriptor used. Incorporation of 3D measurements into this framework may aid in removing some of the limitations of the method.

3.1 Shot-Boundary Detection

Most videos can be decomposed according to a hierarchical structure. This structure begins at the highest level with the actual video. The video can then be decomposed into different scenes in a way such that each scene contains semantically related content. According to Xiong *et al.* [43], scenes typically convey some high level concepts and are divided using semantic boundaries. Each scene can then be broken down into video shots, and each shot is composed of image frames. Consider an artificial example to describe all these concepts. Assuming the video is an arbitrary theatrical movie, an example scene may be a section of the movie where there are two characters having a conversation at a coffee shop. This scene can be broken down into groups, one of which may be a discussion between two characters. Assuming that the camera switches back and forth between the two characters according to whomever is taking at the current point in time, an example shot is when the camera is concentrated on one speaker. As soon as the camera cuts away from the first speaker to show the second speaker, this is considered a new shot. Shots can be viewed as small independent video clips which may or may not have semantic meaning.

As mentioned earlier in this chapter, the MonoSLAM framework keeps track of a small subset of high quality features to perform localization and mapping. Such a framework provides an excellent tool for automatic shot boundary detection. The idea is to create a system that tracks several high quality features across the video. Since the system can cope with camera rotations about 3D objects in the scene, it is much more than just a simple tracking algorithm. The underlying details of the algorithm are essentially identical to MonoSLAM, only a few minor changes have been incorporated. Figure 6 shows a sample frame being tracked. Here, the red squares denote successfully tracked features, yellow squares are features not currently being measured, and blue squares denote features for which the tracking has failed.

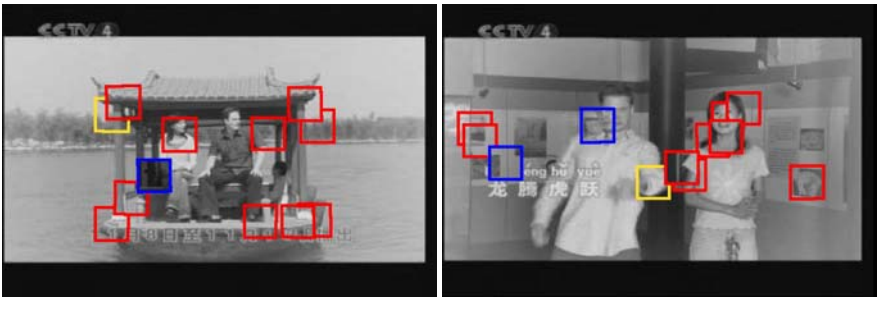


Fig. 6 Sample frames using MonoSLAM to track features in videos from the TRECVID 2005 [33] database. Red squares denote successfully tracked regions, yellow squares are regions about to be initialized, and blue squares denote regions for which the tracking has failed.

Fig. 7 Four frames making up a diagonal wipe transition from the lower right of the frame to the upper left. Video was obtained from the TRECVID 2005 database [33].

The algorithm first begins by finding 16 salient regions in a given image frame, as opposed to the eight regions that traditional MonoSLAM uses. These regions are tracked over time and updated via the EKF. As long as tracking does not fail for a given region, it remains in the probabilistic map. If tracking for a given region fails for 15 consecutive frames, that region is then removed from the map and a new region is detected to take its place after n frames. If at any point, the system fails to successfully track all 16 regions simultaneously, such an event is considered to be a shot boundary. The idea behind the approach is that at a given point in time, if at least some regions of the image have not changed, a shot boundary must not be occurring. Here, the SLAM loop operates as normal (blue regions here correspond to areas which fail tracking due to the motion of the independent objects in the scene).

The algorithm is constantly trying to keep track of 16 non-overlapping regions in the video, each with dimensions 21×21 pixels. These numbers are rather large considering MonoSLAM falls under the EKF-SLAM category, and tracking such a large number of features causes a decrease in the runtime performance of the algorithm. The reason for such a large number is based on the fact that MonoSLAM expects a static scene, and as such is unable to handle moving objects. Since most videos will contain independently moving objects in the scene, by having a large number of features, the hope is that at least several of those features will find static regions of the scene (such as background objects, buildings, etc). Additionally, by

Fig. 8 Six frames illustrating the tracking performance on dynamic scenes. Video was obtained from the TRECVID 2005 database [33].

increasing the size of each region to 21×21 pixels, the tracking quality of each independent region increases significantly and reduces false negatives.

There are several limitations to such a system. For instance, if the system is unable to detect any features belonging to the static background, such features will eventually fail to be tracked and a false shot boundary may be detected. Additionally, if the majority of the features are located on independently moving objects, and the system is unable to locate better features on static objects, this may also lead to a false boundary detection. Lastly, at times the contents of a window will have little to no change across shot boundaries. In such cases, this system may not be able to detect the present boundaries.

This system was tested on sample videos from the TRECVID 2005 [33] database for shot boundary detection. The system was successful in detecting complex shot boundaries characterized by different video transitions. These transitions, such as

wipes and fades, pose a serious challenge because at a given point in time, both shots may be at least partially present on the screen. As seen in Figure 7, this method can easily cope with such transitions. This figure illustrates a diagonal wipe transition beginning from the lower right corner to the upper left corner of the video frame to the upper left. In the first image of the sequence (top left), the algorithm is tracking several high quality features in the current frame. The second image (top right) shows the right third of the video screen has already changed to the next shot, and all the features in that section of the video frame have failed tracking (denoted by the blue squares). By the third frame of the sequence (bottom left), the transition has wiped all but the upper left corner of the video frame. At this time, most of the features are failing the tracking step, except for those in the upper left corner of the frame. In the last frame of the sequence (bottom right), the system detects that all current features have failed to be tracked, and thus the system considers the current point in time to be a shot boundary. The tracking then re-initializes in order to detect the next shot boundary.

One of the limitations of using MonoSLAM for this framework is its inability to track moving objects. However, the effects of the limitation may be reduced by increasing the size of each feature tracked, as well as the total number of features tracked over time. In doing so, the system will be able to find and keep static features on the scene, while quickly removing dynamic features from the map. Figure 8 illustrates such an example. The six video frames were obtained from one of the videos from the TRECVID 2005 [33] database. In this scene, we see a woman talking to the camera, and a man moving around the woman while moving his arms around vigorously. Observe that the system manages to find features in both static and dynamic objects. The tracking quality of the static background features is strong enough to maintain good overall tracking performance, even though the frames contain several dynamic features that are constantly being added and removed from the map. As before, red features are successfully being tracked, yellow features are new features just inserted, and blue features are currently unable to be tracked (soon to be removed). By the second to last frame of the sequence (bottom left of Figure 8), a wipe transition from left to right has begun. By the last frame (bottom right), all features have failed tracking and a shot boundary has been detected.

3.2 Object Recognition

Recently, Castle *et al.* [6] published an extension to MonoSLAM which builds up on the previous work done by Davison *et al.* [7] and Montiel *et al.* [28]. In their work, the focus is developing SLAM algorithms for wearable cameras. According to the authors, there are two preconditions needed for wearable SLAM. First, the camera must be able to establish its location in an initially unknown environment. This problem was addressed already by Davison *et al.* [7] and Montiel *et al.* [28] as previously discussed. The second precondition is that some landmarks must have some sort of higher level meaning to the system. This is achieved by integrating a recognition and tracking functionality into MonoSLAM.

It is important to note that the method proposed here does not change any part of the existing MonoSLAM framework, but instead incorporates the functionality of being able to insert into the probabilistic map objects which have been recognized from a previously built database of known objects. In order to simplify the computation of this problem, the system uses the same feature points for localization and for recognition. Features are first detected using SIFT [22], similarly to the work of Sivic and Zisserman [39], described earlier in this chapter. Features detected using SIFT are invariant to image scale and rotation, and are robust to changes in lighting, noise, and small changes in viewpoint [6, 22].

In [6], Castle *et al.* build a database of objects to recognize. The objects are restricted to be planar. The database contains an image of the object, the SIFT features for the given object, as well as the location of each of the features in the image. When a new object is observed in the scene, the SIFT features of the object are compared with the SIFT features of the database entries. The matching algorithm is based on the original work by Lowe [22], and it computes the Euclidean distance between each feature descriptor, as well as the distance between the 2 nearest neighbors. If the number of matched features is greater than some threshold T_{min} (in [6], the authors use a value of eight), the object is labeled as a possible candidate for recognition.

It may be the case that some SIFT features may be incorrectly matched. To resolve this problem, RANSAC is employed. Let x_i be the points in the image of the observed candidate object, and x_0 be the points in the image of the database object. Since the objects in the database are known to be planar, the candidate and database object points are related by a homography

$$x_i = H_i x_0, \tag{13}$$

and RANSAC can be used to estimate the homography H_i. If RANSAC is able to find a large enough consensus, the candidate object has been recognized successfully. The incorrectly matched SIFT features should be considered as outliers by the RANSAC.

The rest of the algorithm works in the exact same manner as the original MonoSLAM proposed by Davison *et al.* [7] with the inclusion of inverse depth parameterization as proposed by Montiel *et al.* [28]. In order to incorporate the recognized objects into the probabilistic map without having to add any extra functionality for handling SIFT feature vectors, these SIFT features are not directly inserted into the map. Instead, each object in the database keeps track of the location of three points on its boundary. These three points are inserted into the probabilistic map, and MonoSLAM continues as normal. The number three comes from the fact that three is the smallest number of points required to define a plane in three dimensions.

The drawback of the recognition process is computational cost. MonoSLAM is bounded by a $O(N^2)$ complexity[7], where N is the number of features in the probabilistic map, and is capable of running at 30Hz on a desktop computer with standard hardware for a small value of N. The recognition process, however, can be computationally expensive to calculate all the SIFT features and estimate homographies via

RANSAC. Because of this limitation, the recognition process is executed only once every 30 frames, while the SLAM cycle is executed for every frame. This brings the performance of the overall algorithm from 30Hz without recognition, down to 1Hz with recognition performed once every 30 frames.

The main benefit provided by the method is that of greater accuracy for some measurements, as well as incorporating recognition into the single camera SLAM framework. This brings the monocular SLAM algorithms one step closer to being able to track moving objects in a scene. Such functionality would be crucial to the scope of research mentioned in this chapter. Being able to track the position of a moving object from a single video, while concurrently estimating the 3D structure of the scene, would allow video analysis and mining methods to segment objects based on their movements and positions in a 3D world. Such information could be used to create descriptive index terms for recognized objects, and would be beneficial for various tasks.

This current method, however, suffers from many of the drawbacks previously mentioned for MonoSLAM, but also from a performance perspective. Losing the real time performance of the original MonoSLAM framework is a crucial drawback, particularly since the 1Hz performance of the proposed method is achieved with a sparse set of landmarks. Once the sparseness is removed and the system uses a dense set of landmarks (for estimating scene 3D structure), the performance of the algorithm may degrade further. Depending on the application and the length of the input video, a large performance penalty such as this one may be highly undesirable.

3.3 Video Retrieval

Ewerth *et al.* [10] use depth information in order to index videos for retrieval. They claim to be the first authors to use depth features describing the spatial arrangement of three-dimensional objects in a video scene under a content-based video retrieval framework. Although the input to the system is a video from a monocular camera, the authors exploit the motion parallax visible under camera motion to use methods normally applied to stereo vision problems.

The first step in the method is to detect salient features in the initial image frame. These features are detected based on the eigenvalues of the covariance matrix of intensity derivatives. Each feature is then tracked over time using the Lucas-Kanade optical flow algorithm [23] implemented into Intel's OpenCV library [5].

In order for stereo reconstruction algorithms to estimate the three dimensional location of a point from two different viewpoints, that point must first be found in both images. In order to build the correspondence, the authors exploit the epipolar constraint. Given two cameras C_1 and C_2 (corresponding to two views of an object from different angles) observing a point X in the world, the point maps to each camera's image plane as x_1 and x_2, respectively. The line of projection between the camera C_1 and the 3D point X (which passes through the projected point x_1), projects as a line in the image plane of the second camera, C_2. This relation between C_1 and C_2 is represented by the fundamental matrix F [17]. Figure 9 illustrates this

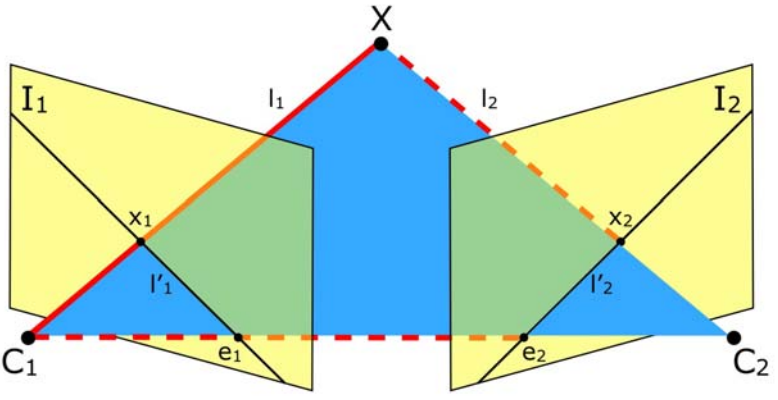


Fig. 9 Geometry of a scene observed by two cameras.

representation. The epipolar constraint is defined in terms of this fundamental matrix as previously described in Equation 1.

The fundamental matrix F can be estimated using the normalized 8 point algorithm [13, 17]. Typically, the algorithm employs the use of singular value decomposition to decompose a matrix containing feature points on both images (provided by the image planes of C_1 and C_2 as previously described). The authors note that this is error prone due to the fact that the method expects a moving camera observing a static scene. If there is any additional motion in the scene, the reconstruction results suffer. To overcome this limitation, the authors use RANSAC [12] to eliminate outlier trajectories.

In order to complete the scene reconstruction, the camera matrices must be computed for each frame. To do so, the authors first calculate the epipole e' of the second image from the equation $F^T e' = 0$. After solving for e', the camera matrices can be computed as

$$P = [I|0] \quad and \quad P' = \left[\left[e' \right]_x F | e' \right], \tag{14}$$

where e' is the epipole vector consisting of (e'_1, e'_2, e'_3) and $[e']_x$ is a 3×3 matrix composed from the epipole vector. After these camera matrices have been calculated, the points can be triangulated to find the corresponding 3D point for all image point pairs between the two images [17].

The 3D reconstruction begins calculations when the first two image frames are available. Afterwards, for every new frame that is obtained, a new camera matrix P must be computed. In order to improve these results, subsequent image frames may be added in order to minimize 3D reconstruction errors. In order to optimize the reconstruction, the authors use bundle adjustment to minimize the re-projection error of the observed m points in n views (image frames). This re-projection error is parameterized by Equation 2. Minimizing this equation with respect to the camera matrices P^j and the set of all 3D points X_i is the final step in the 3D structure estimation.

Before a proper depth map can be generated, the camera must be calibrated in order to calculate proper depth values. The authors incorporate the automatic camera calibration technique proposed by Pollefeys *et al.* [34]. The distance of a point in the 3D world coordinate frame can now be described as the Euclidean distance between the point and the camera's center of projection, $d(X_i, C^j)$. Although this center of projection C^j is initially unknown, it is easily calculated from the equation $P^j C^j = 0$ using singular value decomposition [10, 17].

The depth information computed is sparse and covers a very small percentage of the pixels visible in the image with arbitrary coordinates. In order to build complete depth maps, the authors propagate the depth value of each pixel to its neighbors. For a given pixel, the depth value assigned to it is that of the closest feature (i.e., the closest point with an actual calculated value), as long as the two distances are no more than some threshold t_{dist} apart. Note that it may be the case where certain regions of the image are not assigned a depth value.

These generated depth maps are calculated for each video and used as indexing terms for the task of video retrieval. In order to generate such index terms, first the middle frame (in regards to time) is selected as the representative frame for the video. The depth map generated for this frame is first divided into fixed size blocks and two features are used to describe each block. The first is the normalized mean depth value d' within the block, and the second is the percentage of pixels with depth values. These two features are used to describe each block of the frame, and are inserted into a feature vector which will be used as the index term.

The final component from the framework is a way of measuring the distance between two videos, calculated from their respective feature vectors v_1, v_2. Let B be the set of all blocks in a frame, and let $B_1 \subseteq B$ and $B_2 \subseteq B$ be two subsets of B for whom the values of the second features (percentage of features with depth information) is above some set threshold for vectors v_1 and v_2 respectively. The authors define a distance measurement between two such vectors as:

$$d(v_1, v_2) = \sum_{i \in B_1 \cap B_2} \left| d_1'^i - d_2'^i \right| + \lambda_1 \cdot (|B_1 \backslash B_2| + |B_2 \backslash B_1|) + \lambda_2 \cdot |B \backslash (B_1 \cup B_2)|. \quad (15)$$

The authors evaluate the proposed method using 136 videos from the TRECVID 2005 [33] data set. Although in the case where there is little or no camera motion the 3D reconstruction should yield poor results, the depth maps generated were still quite successful in achieving good retrieval results. Using videos similar to news broadcasts (static camera, one or multiple speakers in the video), the authors claim an average retrieval precision between 63% and 79% for the top 50 ranked videos resulting from each query.

This is the first video retrieval system to incorporate the notion of 3D and depth into content-based retrieval. The precision measurements calculated from the conducted experiments show that such information can be very useful in describing the contents of a video. The method is limited, however, by the sparse 3D reconstruction generated. A much dense reconstruction, although more computationally expensive, should yield much better results since it may provide more detailed depth maps as well as information about each object. As it stands, the method provides very little

information about each object in the scene aside from the depth, relative to other objects.

3.4 Activity Recognition

The area of activity recognition deals with the detection and understanding of activities performed by autonomous entities in a video. Typically, this area of research concentrates on human activities.

In [24], Luo and Hwang propose a video content analysis framework which infers 3D parameters of objects in videos. The authors use a coarse-to-fine approach to segment videos containing sports video sequences. In the first (coarse) stage, the system performs shot segmentation on the video, then uses Hidden Markov Models to classify shots containing appropriate human body shapes. These body shapes are then analyzed in the second (fine) stage.

In the second stage, the idea is to first segment the video objects from the shot, then fit a 3D model to each one. Luo and Huang aim to detect video shots where a tennis player is serving the ball. In order to identify such video shots, the authors propose the use of Gaussian Mixture Models. Training the algorithm on test datasets provides the parameters necessary for the Gaussian mixtures. Maximum likelihood is then used to calculate the probability of each object in a shot to the GM of the training data. If the maximum likelihood ratio of one of the objects exceeds a predefined threshold T_{min}, the object is declared to be in a serving pose.

The next step of the algorithm is to infer the 3D parameters of the human from the sequence of 2D image frames. This is done in a two stage process which fits a generic 3D model of a human onto the the segmented human poses extracted in the previous step. First, the frames containing the extracted human poses are divided into two groups. Cluster analysis is used on the first half of the group. Hu's moments [18] along with K-Means clustering [8] are used to extract the mean clusters of the images. The image with the smallest distance to the mean image is selected, and the 3D model is initially fitted onto this image using manual interaction with the system (the user is asked to manually fit the model as close as possible to the image using the provided GUI).

The second step of the 3D model fitting is to refine the manually adjusted 3D model. First, all extracted image frames are compared against the cluster mean of the previous stage in order to find the frame that is closest to this mean. Nelder-Mead's algorithm [31] is then used to fit the 3D model into the shape of the segmented human in this frame, using the parameters of the manually-adjusted 3D model as the initial parameters to the fitting algorithm. This fitting process is then repeated for every frame in the sequence.

The proposed method attempts to perform activity recognition by fitting a 3D model of a human to extracted video frames containing a specific activity. Although the authors do not actually extract 3D features from the video frames autonomously, the method still provides excellent recognition results by employing the use of a 3D model. The reason lies in the fact that the model used allows for the different

articulations of the human body. When applied to an activity (such as tennis, as illustrated in their work [24]) containing large ranges of motion for the human body, the results illustrate the benefit of incorporating the use of 3D information into activity recognition tasks.

According to the study performed by Fenton *et al.* [11] in 2007, the use of 3D motion analysis is also very helpful to athletes. In this study, a group of athletes were chosen at random and presented either a two dimensional or a three dimensional analysis of their performance. The goal of this study was to determine if the use of 3D data is more beneficial than 2D data in assessing athletic performance. The study concluded that the 3D data was more successful in assessing performance according to the survey, although athletes complained that 2D data was easier to comprehend.

4 Current State of Research

The use of 3D information in video analysis has become increasingly popular in recent years. As shown earlier in this chapter, if an accurate 3D representation of a video clip is obtained, such information is extremely useful in solving various video analysis tasks.

Currently, several approaches which use a video stream as input may be used to estimate the 3D structure of a scene. These approaches use techniques such as structure from motion, simultaneous localization and mapping, as well as 3D reconstruction. Although these methods may be used to estimate 3D structure, none of them are without drawbacks. This section provides a discussion of such drawbacks and limitations, as well as possible ways of overcoming such drawbacks which would make promising future research directions.

4.1 Problems and Limitations

Although the methods presented in this chapter show great promise for solving various tasks, each is not without its own set of drawbacks and limitations. Most of the problems stem from the fact that these algorithms may not have been designed with video analysis in mind. Therefore, they have some inherent limitations which make it difficult for them to function flawlessly for this sort of application.

The work by Tola [40] mentioned in section 2.1 presents a structure from motion framework capable of extracting the motion of multiple independently moving objects, as well as camera motion, all from a set of images obtained using a single monocular camera. The importance of this work is its ability to not only calculate accurate 3D estimations of points in a scene, but it is also able to handle arbitrary camera motions along with the motions of independently moving objects. The drawback of this method, however, stems from the fact that there are several strict assumptions in regards to the motion of the objects. If these strict assumptions are not met, the system may not be able to perform an accurate reconstruction of the observed scene.

As previously discussed in section 2.2, MonoSLAM is a recent algorithm that proposes to solve the SLAM problem using only a single monocular camera as input, and requires no previous knowledge of the scene. Since it is based on an EKF-SLAM framework, it keeps track of high quality features and uses them to accurately localize the camera in a 3D environment. Although the method performs an excellent job of estimating the depth of features, its primary goal is to perform camera (or robot) localization. As a result, when applied to a video sequence, it generates a very sparse representation of the scene. Of course, as shown in the work done by Ewerth *et al.* [10] in section 3.3, sparse 3D representations can still be very useful, even if it is to segment objects based on depth alone.

Another limitation of MonoSLAM is its inability to track moving objects. The seriousness of such a limitation is dependent on the content of the video being analyzed, but is a major limitation regardless. Depending on the scene being viewed, if there are any moving objects, then features on such objects will fail to be recognized (since they will not be located where the system predicts it should be), and thus will be eventually removed from the probabilistic map. At times, such features may also introduce errors into the camera location estimation, which in turn may result in erroneous 3D point estimations for the other features.

The work by Mouragon *et al.* [29, 30] discussed in section 2.3 proposes to solve the problem of scene reconstruction by using certain video frames as key frames and treating them as stereo image pairs. The 3D point locations are then calculated via triangulation, and later refined using bundle adjustment. Unlike MonoSLAM, the method computes a dense reconstruction of the scene while still maintaining efficient computational time. Unfortunately, however, it is also unable to extract 3D locations of moving objects in a scene. Since each point is estimated independently of the others, any points that are located on a moving object are simply discarded once they fail to be tracked.

4.2 Promising Research Directions

As illustrated in this chapter, extracting 3D cues from video streams provides an inherently powerful representation of a scene. Such a representation has already been shown to be useful for solving certain video analysis problems. Still, there remains much work to be done related to this topic. Many of these tasks present new and exciting topics for future research directions.

In the area of 3D estimation from a single view, the biggest limitation to date is the inability to estimate 3D points on moving objects. There has been some recent work in solving this problem for the robotics community using Bayesian probability to track the location of moving objects in a scene [42]. Such work, however, does not use a video camera as input, but rather a 3D scanner which automatically outputs a set of 3D points. As is, such an approach cannot be applied to video analysis, but may provide a good starting point to develop methods for tracking and reconstructing objects in a video.

Another potential direction of research is to study the best way of incorporating the available 3D information into video analysis and mining tasks. Possible topics include fitting 3D models to the available data, or estimating the shape of objects present in the scene for use in certain analysis tasks. For example, given a set of 3D points from a video, how key words can be extracted from the scene in a way such that they help to improve the performance of a content-based video retrieval system.

5 Conclusion

The amount of available video data has seen a rapid increase in recent years, partly due to the growth of personal, industrial, and online databases such as YouTube (http://www.youtube.com). With the increase in activity and data availability comes an increase in necessity of improved methods for video analysis and mining. This chapter discusses the usefulness of incorporating 3D measurements into such tasks. The idea is to estimate a 3D reconstruction from a video clip, and use the available information to solve various tasks such as shot boundary detection, object recognition, content-based video retrieval, as well as activity recognition.

Methods that make use of 3D cues to solve such problems are scarce in the literature due to the inherent difficulty of extracting an accurate 3D representation of a scene in a video clip. This chapter presents several approaches which may be used for such a task, each with its own strengths as well as weaknesses. Structure from motion methods may be used to solve the problem, although they may be computationally expensive and do place certain limitations on videos. SLAM-based methods are also quite useful for solving such a task. Some methods which track a small subset of high quality features have been shown to provide very good results in automatic shot boundary detection due to their excellent three-dimensional tracking abilities. Others, which estimate a dense 3D point cloud of the observed surface, may be used to solve more traditional video problems such as content-based video retrieval and object recognition. These methods have successfully shown the feasibility of such approaches, and provide a new and exciting direction for potential research.

Much work still remains to be done on this particular topic, however. Although current structure from motion, SLAM, and 3D reconstruction methods have shown that 3D cues can be very powerful, they still lack the reliability and robustness required for a video analysis and mining application. Even with these limitations, however, these works show us that 3D estimation of a video stream is not only possible, but also very useful. The next few years should bring very promising advances in this field, which will translate to much higher performance in the analysis, search, and mining of videos.

Acknowledgements. This research was supported in part by NSF grant DMS-0713012. The authors like to thank Gary Gramajo and Brandon Grant for their help with experiments and insightful discussions.

References

1. Abd-Almageed, W.: Online, simultaneous shot boundary detection and key frame extraction for sports videos using rank tracing. In: International Conference on Image Processing, pp. 3200–3203 (2008)
2. Ahanger, G., Little, T.D.C.: A survey of technologies for parsing and indexing digital video. Journal of Visual Communication and Image Representation 7, 28–43 (1996)
3. Bay, H., Tuytelaars, T., Van Gool, L.J.: Surf: Speeded up robust features. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3951, pp. 404–417. Springer, Heidelberg (2006)
4. Boreczky, J.S., Rowe, L.A.: Comparison of video shot boundary detection techniques. JEI 5(2), 122–128 (1996)
5. Bradski, G.: The opencv library. Dr. Dobb's Journal of Software Tools, 120–126 (November 2000)
6. Castle, R.O., Gawley, D.J., Klein, G., Murray, D.W.: Towards simultaneous recognition, localization and mapping for hand-held and wearable cameras. In: Proc. International Conference on Robotics and Automation, Rome, Italy, April 10-14, pp. 4102–4107 (2007)
7. Davison, A.J., Reid, I.D., Molton, N.D., Stasse, O.: MonoSLAM: Real-time single camera SLAM. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(6), 1052–1067 (2007)
8. Duda, R.O., Hart, P.E., Stork, D.G.: Pattern Classification, 2nd edn. Wiley-Interscience, Hoboken (2000)
9. El Qawasmeh, E., Al Badarneh, A.: A survey of digital video shot boundary detection algorithms. Applied Informatics, 497–502 (2002)
10. Ewerth, R., Schwalb, M., Freisleben, B.: Using depth features to retrieve monocular video shots. In: International Conference on Image and Video Retrieval, New York, NY, USA, pp. 210–217 (2007)
11. Fenton, G., Churchill, S., Castle, P.: How useful do athletes find 2d video analysis compared to 3d motion analysis? - a preliminary study (2007), http://eprints.worc.ac.uk/238/
12. Fischler, M.A., Bolles, R.C.: Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. Communications of the ACM 24(6), 381–395 (1981)
13. Forsyth, D.A., Ponce, J.: Computer Vision: A Modern Approach. Prentice Hall, Englewood Cliffs (August 2002)
14. Gargi, U., Kasturi, R., Strayer, S.H.: Performance characterization of video-shot-change detection methods. CirSysVideo 10(1) (2000)
15. Haralick, R., Lee, C.n., Ottenberg, K., Nolle, M.: Analysis and solutions of the three point perspective pose estimation problem. International Journal of Computer Vision, 592–598 (1991)
16. Harris, C., Stephens, M.: A combined corner and edge detection. In: Proceedings of The Fourth Alvey Vision Conference, pp. 147–151 (1988)
17. Hartley, R., Zisserman, A.: Multiple View Geometry. Cambridge Press, New York (2003)
18. Hu, M.K.: Visual pattern recognition by moment invariants. IRE Transactions on Information Theory IT-8, 179–187 (1962)
19. Kellokumpu, V., Zhao, G., Pietikäinen, M.: Human activity recognition using a dynamic texture based method. In: British Machine Vision Conference (2008)
20. Klein, G., Murray, D.W.: Parallel tracking and mapping for small ar workspaces. In: International Symposium on Mixed Augmented Reality (2007)

21. Koprinska, I., Carrato, S.: Temporal video segmentation: A survey. Signal Processing: Image Communication 16(5), 477–500 (2001)
22. Lowe, D.G.: Object recognition from local scale-invariant features. In: International Conference on Computer Vision, pp. 1150–1157 (1999)
23. Lucas, B.D., Kanade, T.: An iterative image registration technique with an application to stereo vision. In: Proceedings of the 7th International Joint Conference on Artificial Intelligence, pp. 674–679 (April 1981)
24. Luo, Y., Hwang, J.-N.: A comprehensive coarse-to-fine sports video analysis framework to infer 3d parameters of video objects with application to tennis video sequences. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (2005)
25. Matas, J., Chum, O., Urban, M., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: British Machine Vision Conference, pp. 384–393 (2002)
26. Mikolajczyk, K., Schmid, C.: Scale and affine invariant interest point detectors. International Journal of Computer Vision 60(1), 63–86 (2004)
27. Montemerlo, M., Thrun, S., Koller, D., Wegbreit, B.: Fastslam: A factored solution to the simultaneous localization and mapping problem. In: Proceedings of the AAAI National Conference on Artificial Intelligence, pp. 593–598 (2002)
28. Montiel, J.M.M., Civera, J., Davison, A.: Unified inverse depth parametrization for monocular slam. In: Proceedings of Robotics: Science and Systems (August 2006)
29. Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., Sayd, P.: Real time localization and 3d reconstruction. In: Proceedings of the 2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Washington, DC, USA, pp. 363–370. IEEE Computer Society, Los Alamitos (2006)
30. Mouragnon, E., Lhuillier, M., Dhome, M., Dekeyser, F., Sayd, P.: Generic and real-time structure from motion using local bundle adjustment. Image and Vision Computing (2008)
31. Nelder, J.A., Mead, R.: A simplex method for function minimization. The Computer Journal 7(4), 308–313 (1965)
32. Ojala, T., Pietikäinen, M., Mäenpää, T.: Multiresolution gray-scale and rotation invariant texture classification with local binary patterns. Pattern Analysis and Machine Intelligence 24(7), 971–987 (2002)
33. Over, P., Ianeva, T., Kraaij, W., Smeaton, A.F.: Trecvid 2005 - an overview. In: TREC Video Retrieval Evaluation Online Proceedings (2006)
34. Pollefeys, M., Van Gool, L., Vergauwen, M., Verbiest, F., Cornelis, K., Tops, J., Koch, R.: Visual modeling with a hand-held camera. International Journal of Computer Vision 59(3), 207–232 (2004)
35. Ribeiro, M.I.: Kalman and extended kalman filters: Concept, derivation and properties (February 2004)
36. Shi, J., Tomasi, C.: Good features to track. In: International Conference on Computer Vision and Pattern Recognition, pp. 593–600. Springer, Heidelberg (1994)
37. Sivic, J.: Efficient Visual Search of Images and Videos. PhD thesis, University of Oxford (2006)
38. Sivic, J., Zisserman, A.: Video google: Efficient visual search of videos. In: Ponce, J., Hebert, M., Schmid, C., Zisserman, A. (eds.) Toward Category-Level Object Recognition. LNCS, vol. 4170, pp. 127–144. Springer, Heidelberg (2006)
39. Sivic, J., Zisserman, A.: Efficient visual search for objects in videos. Proceedings of the IEEE 96(4), 548–566 (2008)
40. Tola, E., Knorr, S., Imre, E., Alatan, A.A., Sikora, T.: Structure from motion in dynamic scenes with multiple motions. In: 2nd Workshop on Immersive Communication and Broadcast Systems (ICoB 2005), Berlin, Germany (October 2005)

41. Visser, R., Sebe, N., Bakker, E.: Object recognition for video retrieval. In: International Conference on Image and Video Retrieval, pp. 262–270 (2002)
42. Wang, C.-C., Thorpe, C., Hebert, M., Thrun, S., Durrant-Whyte, H.: Simultaneous localization, mapping and moving object tracking. The International Journal of Robotics Research 26(6) (June 2007)
43. Xiong, Z., Radharkishnan, R., Divakaran, A., Rui, Y., Huang, T.S.: A Unified Framework for Video Summarization, Browsing and Retrieval. Elsevier, Amsterdam (2006)
44. Yuan, J., Wang, H., Xiao, L., Zheng, W., Li, J., Lin, F., Zhang, B.: A formal study of shot boundary detection. IEEE Transaction on Circuit and Systems For Video Technology 17(2), 168–186 (2007)

Statistical Analysis on Manifolds and Its Applications to Video Analysis*

Pavan Turaga, Ashok Veeraraghavan, Anuj Srivastava, and Rama Chellappa

Abstract. The analysis and interpretation of video data is an important component of modern vision applications such as biometrics, surveillance, motion-synthesis and web-based user interfaces. A common requirement among these very different applications is the ability to learn statistical models of appearance and motion from a collection of videos, and then use them for recognizing actions or persons in a new video. These applications in video analysis require statistical inference methods to be devised on non-Euclidean spaces or more formally on manifolds. This chapter outlines a broad survey of applications in video analysis that involve manifolds. We develop the required mathematical tools needed to perform statistical inference on manifolds and show their effectiveness in real video-understanding applications.

1 Introduction

Applications in computer vision often involve the study of geometric scenes and their interplay with physical phenomena such as illumination and motion. When these scenes are imaged using cameras, the observed appearances obey certain mathematical constraints that are induced by the underlying physical

Pavan Turaga and Rama Chellappa
Institute for Advanced Computer Studies, University of Maryland,
College Park, MD 20742
e-mail: {pturaga,rama}@umiacs.umd.edu

Ashok Veeraraghavan
Mitsubishi Electric Research Labs, Cambridge MA 02139
e-mail: veerarag@merl.com

Anuj Srivastava
Department of Statistics, Florida State University, Tallahasse FL
e-mail: anuj@stat.fsu.edu

* This work was partially supported by the Office of Naval Research under the Grant n00014-09-10664.

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 115–144.
springerlink.com

constraints. Examples include the observation that images of a convex object under all possible illumination conditions lie on the so called 'illumination-cone' [17]. Images taken under a stereo-pair are constrained by the epipolar geometry of the cameras [22]. Similarly, the 3D pose of the human head is parameterized by three angles – hence, under constant illumination and expression, the observed face of a human under different viewing directions lies on a three-dimensional manifold. In a particular application, if the physical and mathematical constraints are well-understood, such as in epipolar geometry and illumination modeling, then one can design accurate modeling and inference techniques derived from this understanding.

In several applications of video analysis such as gait-based human ID, activity recognition, shape-based dynamics modeling, and video-based face recognition, some of the constraints that arise have a special form. These special constraints can often be expressed in the form of an equation with some smoothness criterion. Such constraints can be formally defined as manifolds. Before we give precise definitions of what is meant by a manifold, let us first consider some simple problems that illustrate why special attention is needed to study them. To enable this discussion, we shall for the time-being assume that a 'manifold' is defined as a set of points in $\mathbb{R}^n$ that satisfy an equation $f(x) = 0$ (with appropriate conditions on $f()$ that shall be spelt out in a later section). For example, the set of points that satisfy the equation $f(x) = x^T x - 1 = 0$ is the unit hyper-spherical manifold in $\mathbb{R}^n$.

Now one might ask, what is special about these constraints that require new mathematical tools from differential geometry and topology ? Can we not use the classical Euclidean methods and multi-variate statistics, with perhaps some loss of accuracy ? To answer these questions, we will consider a very simple engineering problem. Suppose, a highway construction engineer is laying out a road between two cities which are far apart. Given two cities on the earth, the engineer wants to know a) what is the length of the road required (so as to estimate the amount of building material that needs to be ordered), b) where should a rest-area that is mid-way between the two cities be placed.

Given two points x_1 and x_2 on the earth, he/she would like to compute the shortest distance between them. If the curvature of the earth were not taken into account, and all he/she knew was that the points are in $\mathbb{R}^3$, he/she might choose to use the standard Euclidean norm $\|x_1 - x_2\|$. Unfortunately, this would lead to the engineer underestimating the distance between the two cities. But equipped with the additional knowledge that the earth is well-approximated as a sphere in $\mathbb{R}^3$, we can interpret the Euclidean norm as the 'chordal-length' between these points. The knowledge of this geometry of the constraint set also shows that the Euclidean distance is not *intrinsic* i.e. if we sample points along the shortest straight line path, the samples do not lie on the sphere. This distance is thus meaningless for the engineer since it would require him to lay a tunnel underneath the surface instead of a road on the surface of the earth.

Similarly, given two cities/points that lie on the unit-hypersphere as before, we wish to compute a mean-point where a rest-area may be constructed. Once again, if we did not know the nature of the constraint set, we might use the arithmetic-mean as the mean-point. Now given the extra information that these points need to lie on a hypersphere, it is obvious that the arithmetic mean is not intrinsic either since it does not lie on the hypersphere. The arithmetic mean of these points would lie *under* the surface of the earth rendering it physically meaningless. A much more complicated situation arises when we want to place say 3 rest areas in the midst of 10 cities with some optimality criterion such as reducing the overall length of road to be laid. This requires solving an optimization problem with manifold-valued constraints.

Even though these are fairly simple applications, they illustrate the need to understand the underlying constraints to obtain geometrically meaningful distances and statistics. Naturally, in the presence of such constraints, classical Euclidean geometry fails to provide meaningful solutions. This motivates the need to study such non-Euclidean spaces via methods from differential geometry.

Organization: We begin the chapter by motivating the study of manifold analysis for video processing applications. We then provide an introduction to manifold theory and describe relevant manifolds and provide an introduction to differential geometry on these manifolds. In Section 4, we present methods to perform statistical inference on these manifolds. In Section 5, we present several applications of the presented theory to problems in video understanding.

2 Motivation for Studying Manifolds in Video Analysis

Let us first consider some real applications in video understanding that require appreciating the geometry of some non-Euclidean manifolds. Once again, we shall for the time-being assume that a 'manifold' is defined as a set of points in $\mathbb{R}^n$ that satisfy an equation $f(x) = 0$ (with appropriate conditions on $f()$ that shall be spelt out in the next section).

The problem of video understanding can be studied from three widely differing perspectives a) The feature space, b) The model space and c) The transformation space. Even though the specific nature of these spaces can be quite different, a large class of these spaces can be described mathematically as manifolds. Traditionally, 'manifold-learning' methods have been at the forefront of these applications where an analytical characterization of these spaces cannot be found. In the past few years, computer vision researchers have made significant advancement in the analytical and geometric understanding of these varied spaces. This marks an important development in computer vision by moving away from data-driven approaches to geometry-driven approaches for characterizing videos. We provide specific examples of various analytical manifolds found in different applications of computer vision below.

1. **Feature Spaces:** Video understanding typically begins with the extraction of some specific features from the videos. Examples of these features include background subtracted images, shapes, intensity features, motion vectors etc. These features extracted from the videos might satisfy certain geometric and photometric constraints. The feature space deals with understanding and characterizing the geometry of features that can be extracted from videos. The study of this space then enables appropriate modeling methodologies to be designed. Consider the example of the shape feature. Shapes in images are commonly described by a set of landmarks on the object being imaged. After appropriate translation, scale and rotation normalization it can be shown that shapes reside on a complex spherical manifold. Further, by factoring out all possible affine transformations, it can be shown that shapes reside on a Grassmann Manifold.
2. **Model Spaces:** After features are extracted from each frame of the video, the next step in video analysis, is to describe a sequence of such features using appropriate spatio-temporal models. One specific example of this is modeling the feature sequence as realizations of dynamical systems. Examples include dynamic textures, human joint angle trajectories and silhouette sequences. One popular dynamical model for such time-series data is the autoregressive and moving average (ARMA) model. The space spanned by the columns of the observability matrix of the ARMA model can be identified as a point on the Grassmann manifold. Time-varying and switching linear dynamical systems can then be interpreted as paths on the Grassmann manifold.
3. **Transformation Spaces:** Finally, the transformation space encompasses all possible manifestations of the same semantic activity. The study of this space is important to achieve invariance to factors such as view-changes and execution-rate changes. In this chapter we consider the specific instance of execution-rate variations in human activities, which is modeled as temporal warps of feature trajectories. The space of these warps is the space of positive and monotonically increasing functions mapping the unit-interval to the unit-interval. The derivatives of warping functions can be interpreted as probability density functions. The square-root form of pdfs can then be described as a sphere in the space of functions. Variability in sampling closed planar curves gives rise to variations in observed feature points on shapes. This variability can also be modeled as a sphere in the space of functions (also known as a Hilbert sphere).

As these examples illustrate, manifolds arise quite naturally in several vision-based applications.

2.1 Manifold Theory in Vision

There has been an increasing awareness of the need to perform statistical inferences on non-Euclidean domains for a variety of reasons. There has been a significant amount of work in this area in several disciplines. Here, we will

review some of these works. This treatment by no means should be considered exhaustive. The use of certain groups, e.g. Euclidean groups, have been fundamental in physics since Einstein and perhaps earlier. The Euclidean motion group plays a fundamental role in rigid body dynamics and uncertainties in modeling dynamic systems have been characterized using probability measures on this group. Another community that has combined the strengths of geometry and statistics is stochastic control [11, 10] where system variables and controls are constrained to be on certain non-Euclidean manifolds.

To the best of our knowledge, the first major effort in using geometry and statistics in pattern recognition was introduced by Ulf Grenander in the early 70s [19, 18]. Grenander created the field of pattern theory which had the following important components: (i) represent the systems of interest using algebraic structures that favor rule-based compositions, (ii) capture variability in these systems using probabilistic super-structures, and (iii) develop efficient algorithms for inferences using geometries of underlying spaces. Over the last three decades, this philosophy has been implemented in a number of contexts with explicit involvement of statistics on non-Euclidean manifolds. We list a few here: The work on analyzing anatomical variability using non-invasive imaging (such as MRI, PET, etc) involved probabilistic structures on high-dimensional deformation groups – this area has recently been labeled as *computational anatomy* [20, 31]. An algebraic pattern theoretic approach has not been exclusive to medical imaging only. It has also been used in addressing computer vision and image analysis problems. For example, in the problem of recognizing objects in images, the variability due to viewing angle of the camera is very important. [21] deals with the problem of estimating the pose as an element of $SO(3)$ and that of bounding the estimating error using statistical bounds. [40] studies the problem of using Markov Chain Monte Carlo methods for performing estimation on some matrix Lie groups e.g. $SO(n)$, and their quotient spaces, e.g. a Grassmann manifold, while [41] studies the problem of subspace tracking (in signal processing) as a problem of nonlinear filtering on a complex Grassmann manifold. While these papers involve statistical inferences on manifolds, there is a strong literature on more general optimization problems. For example, a major work in the area of optimization algorithms on Grassmann and Stiefel manifolds was presented by Edelman et al. [16, 1].

Another prominent area that employed statistical models and inferences on non-Euclidean manifolds is shape analysis. Starting with a trend-setting paper by Kendall [26, 28], there has been a remarkable literature on representing and analyzing shapes of objects, in images or otherwise, using a "landmark-based" approach. In terms of statistical analysis, this is perhaps the most mature area involving manifolds as domains [15, 36]. In more recent years, there has been an extension of Kendall's shape theory to infinite-dimensional representations of shapes of curves and surfaces [27, 39, 32].

The area of statistics and inference on manifolds has seen a large growth in recent years. Many of the ideas have been formally introduced and advanced through the efforts of many researchers. One of the landmark works in

establishing mean estimation and central limit theorems for manifold-valued variables is Bhattacharya and Patrangenaru [5, 4]. Another important piece of work comes from Pennec [33] who has applied these notions for detection and classification of anatomical structures in medical images. Recent applications in computer vision have included study of Kendall's shape spaces for human gait analysis [48], and Hilbert sphere modeling of time warp functions for human activities in [49]. Other applications include classification over Grassmann manifolds for shape and activity analysis [3, 45], and face recognition [30]. A recently developed formulation of using the covariance of features in image-patches has found several applications such as texture classification [46], and pedestrian detection [47]. Mean-shift clustering was extended to general Riemannian manifolds in [42].

3 Introduction to Manifolds

We shall first start with the topological definition of a manifold in terms of charts and atlases. Using them, we will show that $\mathbb{R}^n$ is indeed a differentiable manifold. Then, we state a theorem that defines sub-manifold of a manifold as a solution of an equation. This shall be specialized to the case of manifolds that are actually sub-manifolds of $\mathbb{R}^n$, arising as solutions of an equation in $\mathbb{R}^n$ with some conditions. Furthermore, we will establish the notions of tangent vectors and tangent spaces on non-Euclidean manifolds. This will then allow the use of classical statistical methods on the tangent planes via the exponential map and its inverse. We shall provide specific examples to illustrate these notions.

3.1 General Background from Differential Geometry

We start by considering the definition of a general differentiable manifold. The material provided here is brief and by no means comprehensive. We refer the interested readers to two excellent books [9, 38] for a more detailed introduction to differential geometry and manifold analysis. A topological space M is called a **differentiable manifold** if, amongst other properties, it is *locally Euclidean*. This means that for each $p \in M$, there exists an open neighborhood U of p and a mapping $\phi : U \to \mathbb{R}^n$ such that $\phi(U)$ is open in $\mathbb{R}^n$ and $\phi : U \to \phi(U)$ is a diffeomorphism. The pair (U, ϕ) is called a *coordinate chart* for the points that fall in U; for any point $y \in U$, one can view the Euclidean coordinates $\phi(y) = (\phi_1(y), \phi_2(y), \ldots, \phi_n(y))$ as the coordinates of y. The dimension of the manifold M is n. This is a way of flattening the manifold locally. Using ϕ and ϕ^{-1}, one can move between the sets U and $\phi(U)$ and perform calculations in the more convenient Euclidean space. If there exists multiple such charts, then they are compatible, i.e. their compositions are smooth. We look at the some simple manifolds as examples.

Example 1. **($\mathbb{R}^n$ is a manifold)**

1. The Euclidean space $\mathbb{R}^n$ is an n-dimensional differentiable manifold which can be covered by the single chart $(\mathbb{R}^n, \phi)$, $\phi(x) = x$.
2. Any open subset of a differentiable manifold is itself a differentiable manifold. A well known example of this idea comes from linear algebra. Let $M(n)$ be the set of all $n \times n$ matrices; $M(n)$ can be identified with the set $\mathbb{R}^{n \times n}$ and is, therefore, a differentiable manifold. Define the subset $GL(n)$ as the set of non-singular matrices, i.e. $GL(n) = \{A \in M(n) | \det(A) \neq 0\}$, where $\det(\cdot)$ denotes the determinant of a matrix. Since $GL(n)$ is an open subset of $M(n)$, it is also a differentiable manifold.

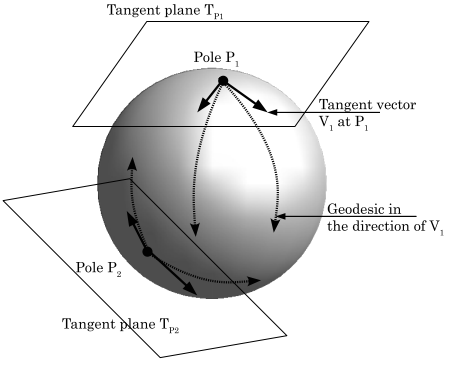


Fig. 1 Figure illustrating the notions of tangent spaces, tangent vectors, and geodesics

In order to perform differential calculus, i.e. to compute gradients, directional derivatives, critical points, etc., of functions on manifolds, one needs to understand the tangent structure of those manifolds. Although there are several ways to define tangent spaces, one intuitive approach is to consider differentiable curves on the manifold passing through the point of interest, and to study the velocity vectors of these curves at that point. To help visualize these ideas, we illustrate the notions of tangent planes, geodesics in figure 1. More formally, let M be an n-dimensional manifold and, for a point $p \in M$, consider a differentiable curve $\gamma : (-\epsilon, \epsilon) \to M$ such that $\gamma(0) = p$. The velocity $\dot{\gamma}(0)$ denotes the velocity of γ at p. This vector has the same dimension as the manifold M itself and is an example of a **tangent vector** to M at p. The set of all such tangent vectors is called the **tangent space** to M at p. Even though the manifold M maybe nonlinear, the tangent space $T_p(M)$ is always linear and one can impose probability models on it using more traditional approaches.

Example 2. 1. In case of the Euclidean space $\mathbb{R}^n$, the tangent space $T_p(\mathbb{R}^n) = \mathbb{R}^n$ for all $p \in \mathbb{R}^n$.

2. For $GL(n)$, the space of non-singular matrices and for an $A \in GL(n)$, let $\gamma(t)$ be a path in $GL(n)$ passing through $A \in GL(n)$ at $t = 0$. The velocity vector at p, $\dot{\gamma}(0)$, is an element of $M(n)$, the set of all $n \times n$ matrices.

Next we introduce the notion of a differential which is important in defining the submanifolds of interest to us. Several of the spaces we will study can be viewed as submanifolds of larger manifolds such as $\mathbb{R}^n$ and $GL(n)$. The differential of a smooth mapping $f : M \to N$ at $p \in M$, denoted by df_p, is a linear map $df_p : T_p(M) \to T_{f(p)}(N)$ specified as follows. Let $g : N \to \mathbb{R}$ be a smooth function. Then, for any $v \in T_p(M)$, define $(df_p(v))(g) = v(f \circ g)(p)$. A point $p \in M$ is said to be a **regular point** if df_p is onto, and its image under f is said to be a **regular value**.

Theorem 1. *Suppose M and N are manifolds of dimensions m and n respectively, and let $f : M \to N$ be a smooth map, with a regular value $y \in N$. Then $f^{-1}(y)$ is a submanifold of M of dimension $m - n$.*

This theorem states that the pullback sets of certain types of points under smooth maps have the submanifold structure. Important examples of such pullback sets include spheres in Euclidean spaces.

Example 3. 1. **Unit Sphere**: Using this theorem, let us check if $\mathbb{S}^n$ is indeed a submanifold of $\mathbb{R}^{n+1}$. Let $f : \mathbb{R}^{n+1} \to \mathbb{R}$ be a map given by $f(p) = \sum_{i=1}^{n+1} p_i^2$, where $p = (p_1, \ldots, p_{n+1})$. The differential of f is given $df_p(u) = 2\langle p, u \rangle$, which is clearly onto for all $p \in f^{-1}(1)$. Thus, 1 is a regular value of f and the set $f^{-1}(1)$ given by $\mathbb{S}^n$ is an n-dimensional submanifold of $\mathbb{R}^{n+1}$. Also, the tangent space $T_p(\mathbb{S}^n)$ is just the orthogonal complement of $p \in \mathbb{R}^{n+1}$.

2. **Orthogonal Matrices**: We now consider the set $O(n)$ of orthogonal matrices, which is a subset of the manifold $GL(n)$. We define $O(n)$ to be the set of all $n \times n$ invertible matrices O that satisfy $OO^T = I$. Define $S(n)$ to be the set of $n \times n$ symmetric matrices, and then define $f : GL(n) \to S(n)$ by $f(O) = OO^T$. It can easily be shown that I is a regular value of f and, hence, $f^{-1}(I) = O(n)$ is a submanifold of $GL(n)$. Note that $O(n)$ is not connected, but has two components: those orthogonal matrices with determinant $+1$, and those with determinant -1. The set of orthogonal matrices with determinant 1 is called the **special orthogonal group**, and denoted by $SO(n)$. The dimension of $O(n)$ can be determined by the above theorem; it is $n^2 - n(n+1)/2 = n(n-1)/2$. One can show that $T_O O(n) = \{OX | X \text{ is an } n \times n \text{ skew-symmetric matrix}\}$.

We now consider the task of measuring distances on a manifold. This is accomplished using a Riemannian metric defined as follows.

Definition 1. A **Riemannian metric** on a differentiable manifold M is a map $\langle \cdot, \cdot \rangle$ that smoothly associates to each point $p \in M$ a symmetric, bilinear, positive definite form on the tangent space $T_p(M)$.

A differentiable manifold with a Riemannian metric on it is called a **Riemannian manifold**.

Example 4. 1. $\mathbb{R}^n$ is a Riemannian manifold with the Riemannian metric $\langle v_1, v_2 \rangle = v_1^T v_2$, the standard Euclidean product.
2. We have earlier examined the manifold $O(n)$ and stated that its tangent space is: $T_O O(n) = \{OX : X \text{ is skew-symmetric}\}$. Define the inner product for any $Y, Z \in T_O O(n)$ by $\langle Y, Z \rangle = trace(YZ^T)$, where *trace* denotes the sum of diagonal elements. With this metric $O(n)$ becomes a Riemannian manifold.
3. Similarly, for the unit sphere $\mathbb{S}^n$ and a point $p \in \mathbb{S}^n$, the Euclidean inner product on the tangent vectors make $\mathbb{S}^n$ a Riemannian manifold. That is, for any $v_1, v_2 \in T_p(\mathbb{S}^n)$, we used the Riemannian metric $\langle v_1, v_2 \rangle = v_1^T v_2$.

Using the Riemannian structure, it becomes possible to define lengths of paths on a manifold. Let $\alpha : [0,1] \mapsto M$ be a parameterized path on a Riemannian manifold M that is differentiable everywhere on $[0,1]$. Then $\frac{d\alpha}{dt}$, the velocity vector at t, is an element of the tangent space $T_{\alpha(t)}(M)$, with length given by $\sqrt{\langle \frac{d\alpha}{dt}, \frac{d\alpha}{dt} \rangle}$. The length of the path α is then given by:

$$L[\alpha] = \int_0^1 \sqrt{\left(\left\langle \frac{d\alpha(t)}{dt}, \frac{d\alpha(t)}{dt} \right\rangle \right)} dt \ . \tag{1}$$

For any two points $p, q \in M$, one can define the distance between them as the infimum of the lengths of all smooth paths on M which start at p and end at q:

$$d(p,q) = \inf_{\{\alpha:[0,1]\mapsto M | \alpha(0)=p, \alpha(1)=q\}} L[\alpha] \ . \tag{2}$$

A path $\hat{\alpha}$ which achieves the above minimum, if it exists, is a **geodesic** between p and q on M.

Example 5. 1. Geodesics on a unit sphere $\mathbb{S}^n$ are great circles [9]. The distance minimizing geodesic between two points p and q is the shorter of the two arcs of a great circle joining them between them. As a parameterized curve, this geodesic is given by:

$$\alpha(t) = \frac{1}{\sin(\theta)} [\sin(\theta - t)p + \sin(t)q] \tag{3}$$

where $\theta = \cos^{-1}(\langle p, q \rangle)$.
2. To define geodesics on $SO(n)$, we introduce the notion of matrix exponential. For a matrix $A \in M(n)$, define its matrix exponential $\exp(A)$ by:

$$\exp(A) = I + \frac{A}{1!} + \frac{A^2}{2!} + \frac{A^3}{3!} + \ldots \tag{4}$$

Using the matrix exponential, one can define geodesics on $SO(n)$ (with respect to the Riemannian metric defined earlier) as follows: for any $O \in SO(n)$ and any skew-symmetric matrix X, $\alpha(t) \equiv O \exp(tX)$, is the unique geodesic in $SO(n)$ passing through O with velocity OX at $t = 0$ [9].

An important tool in studying statistics on a manifold is an exponential map. If M is a Riemannian manifold and $p \in M$, the **exponential map** $\exp_p : T_p(M) \to M$, is defined by $\exp_p(v) = \alpha_v(1)$ where α_v is a constant speed geodesic whose velocity vector at p is v. For $\mathbb{R}^n$, under the Euclidean metric, since geodesics are given by straight lines, the exponential map is a simple addition: $\exp_p(v) = p + v$, for $p, v \in \mathbb{R}^n$. The exponential map on a sphere, $\exp : T_p(\mathbb{S}^n) \mapsto \mathbb{S}^n$, is given by $\exp_p(v) = \cos(\|v\|)p + \sin(\|v\|)\frac{v}{\|v\|}$. In case of $SO(n)$, the exponential is given by $\exp_O(X) = O\exp(X)$, where the exponential on the right side is defined in Eqn. 4. We illustrate the notions of the exponential map in figure 2.

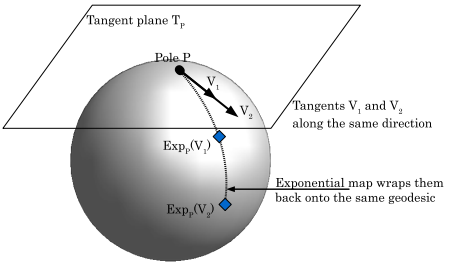


Fig. 2 Figure illustrating the notion of exponential maps and inverse exponential maps.

3.2 Special Manifolds of Interest

We are interested in quotient spaces of the special orthogonal group $SO(n)$ studied earlier. We start by introducing the notion of a quotient space of a group. A group G is a set having an associative binary operation, denoted by $\cdot$, such that: (i) there is an identity element e in G, and (ii) for each element, there exists a unique inverse. Let H be a subgroup of G. For any element $g \in G$, define a left **coset** of H in G by $gH = \{g \cdot h \big| h \in H\}$. In general, the cosets are not subgroups and the only coset that is a subgroup of G is H itself (eH). For different elements g_1 and g_2, the cosets g_1H and g_2H will either be identical or disjoint. They will be identical when $g_2^{-1}g_1$ is an element of H; otherwise they will be disjoint. This is similar to an equivalence relation that partitions a set into disjoint equivalence classes. In fact, one can define an equivalence relation using membership of these cosets: we define $g_1 \sim g_2$ if $g_1 \in g_2H$, i.e. $g_1 = g_2h$ for some $h \in H$. In the notation of equivalence classes, we have $[g] = gH$. The quotient space $G/\sim$, also denoted by G/H to emphasize the role of H in defining $\sim$, is the set of all left cosets of H in G. The quotient space G/H is also called the space G *modulo* H, or the space that results when H *is removed from* G.

Now we consider three specific manifolds that are important in our analysis of features in videos.

1. **Stiefel Manifold:** Let the set of all $n \times d$ orthogonal matrices be $\mathcal{S}_{n,d}$,

$$\mathcal{S}_{n,d} = \{U \in \mathbb{R}^{n \times d} | U^T U = I_d\} \quad \subset GL(n,d). \tag{5}$$

$\mathcal{S}_{n,d}$ is called a **Stiefel** manifold, and each element of $\mathcal{S}_{n,d}$ provides an orthonormal basis for a d-dimensional subspace of $\mathbb{R}^n$. $\mathcal{S}_{n,d}$ can also be viewed as a quotient space of $SO(n)$ as follows. First, consider $SO(n-d)$ as a subgroup of $SO(n)$ using the embedding: $\phi_1 : SO(n-d) \mapsto SO(n)$, defined by

$$\phi_1(V) = \begin{bmatrix} I_d & 0 \\ 0 & V \end{bmatrix} \quad \in \quad SO(n) \ . \tag{6}$$

With this embedding, we can generate left cosets of $SO(n)$: for an $O \in SO(n)$, a coset is given by $O\phi_1(S(n-d))$. This defines an equivalence relation $\sim$ in $SO(n)$ according to: for $Q_1,\ Q_2 \in SO(n)$,

$$Q_1 \sim Q_2, \quad \text{if and only if} \quad Q_1 = Q_2\phi_1(V), \quad \text{for some} \quad V \in SO(n-d) \ .$$

In other words, $Q_1 \sim Q_2$ if and only if their first d columns are identical, irrespective of the remaining columns. Therefore, $\mathcal{S}_{n,d}$ can be viewed as the quotient space

$$\mathcal{S}_{n,d} = SO(n)/\sim \ \text{ or } \ SO(n)/\phi_1(SO(n-d)) \ \text{ or simply } \ SO(n)/SO(n-d) \ .$$

2. **Grassmann Manifold:** If one is interested only in the subspace spanned by the columns of U, and not in a particular basis, then the required space is reduced further. Let $SO(d) \times SO(n-d)$ be a subset of $SO(n)$ using the embedding $\phi_2 : (SO(d) \times SO(n-d)) \mapsto SO(n)$:

$$\phi_2(V_1, V_2) = \begin{bmatrix} V_1 & 0 \\ 0 & V_2 \end{bmatrix} \quad \in \quad SO(n) \ , \quad V_1 \in SO(d), \quad V_2 \in SO(n-d). \tag{7}$$

As for $\mathcal{S}_{n,d}$, define an equivalence relation as a coset of $SO(n)$ generated by the subgroup $\phi_2(SO(d) \times SO(n-d))$ and let $\mathcal{G}_{n,d}$ be the quotient space $SO(n)/\phi_2(SO(d) \times SO(n-d))$, or simply $SO(n)/(SO(d) \times SO(n-d))$. We will use the square-brackets to denote elements of $\mathcal{G}_{n,d}$:

$$[U] = \{UO | U \in \mathcal{S}_{n,d},\ O \in SO(d)\} \ .$$

3. **Kendall's Shape Manifold:** Kendall [25] provided a mathematical theory for the description of landmark based shapes. Bookstein [8] and later Dryden and Mardia [14] have furthered the understanding of such landmark based shape descriptions. Kendall's representation of shape describes the shape configuration of n landmark points in an d-dimensional space as a $n \times d$ matrix containing the coordinates of the landmarks. Pre-shape is the geometric information that remains when location and scale effects are filtered out. Let the configuration of a set of n landmark points be given by a n-dimensional complex vector containing the positions of landmarks.

Let us denote this configuration as X. The centered pre-shape is obtained by subtracting the mean from the configuration and then scaling to norm one. The centered pre-shape is given by

$$Z_c = \frac{CX}{\parallel CX \parallel}, \quad where \quad C = I_n - \frac{1}{n} 1_n 1_n^T, \tag{8}$$

where I_n is a $n \times n$ identity matrix and 1_n is a n-dimensional vector of ones. The pre-shape vector that is extracted by the method described above lies on a spherical manifold. Let us denote this pre-shape space as $\mathcal{P}_{n,d}$. The shape space is now the quotient of the preshape space obtained by removing all rotations of the shape i.e. $\mathcal{P}_{n,d}/SO(d)$.

Example 6 **(Kendall Shape Metrics).** Several distance metrics have been defined in [14] to measure distances between shapes using the Kendall's shape representation. Here, we shall describe some of them. Consider two complex configurations X and Y with corresponding preshapes α and β. The full Procrustes distance between the configurations X and Y is defined as the Euclidean distance between the full Procrustes fit of α and β. Full Procrustes fit is chosen so as to minimize

$$d(Y, X) = \parallel \beta - \alpha s e^{j\theta} - (a + jb) 1_n \parallel, \tag{9}$$

where s is a scale, θ is the rotation and $(a + jb)$ is the translation. The Full Procrustes distance is the minimum Full Procrustes fit i.e.,

$$d_{Full}(Y, X) = \inf_{s,\theta,a,b} d(Y, X). \tag{10}$$

We note that the preshapes are actually obtained after filtering out effects of translation and scale. Hence, the translation value that minimizes the full Procrustes fit is given by $(a + jb) = 0$, while the scale $s = 1$. The rotation angle θ that minimizes the Full Procrustes fit is given by $\theta = arg(|\alpha^* \beta|)$. The partial Procrustes distance between configurations X and Y is obtained by matching their respective preshapes α and β as closely as possible over rotations, but not scale. So,

$$d_{Partial}(X, Y) = \inf_{\Gamma \epsilon SO(d)} \parallel \beta - \alpha \Gamma \parallel . \tag{11}$$

It is interesting to note that the optimal rotation θ is the same whether we compute the full Procrustes distance or the partial Procrustes distance. The Procrustes distance $\rho(X, Y)$ is the closest great circle distance between α and β on the preshape sphere. The minimization is done over all rotations. Thus ρ is the smallest angle between complex vectors α and β over rotations of α and β. The three distance measures defined above are all trigonometrically related as

$$d_{Full}(X, Y) = \sin \rho(X, Y), \quad d_{Partial}(X, Y) = 2 \sin(\frac{\rho(X, Y)}{2}). \tag{12}$$

3.2.1 Tangent Structure of the Special Manifolds

If M/H is a quotient space of M under the action of a group $H \subset M$ (assuming H acts on M), then, for any point $p \in M$, a vector $v \in T_p(M)$ is also tangent to M/H as long as it is perpendicular to the tangent space $T_p(pH)$. Here, $T_p(pH)$ is considered as a subspace of $T_p(M)$. We will use this idea to find tangent spaces on $\mathcal{S}_{n,d}$, $\mathcal{G}_{n,d}$, and Kendall's shape space, using the tangent structure of $SO(n)$.

1. **Tangent Structure of $\mathcal{S}_{n,d}$**: Since $\mathcal{S}_{n,d} = SO(n)/\phi_1(SO(n-d))$, set $M = SO(n)$ and $H = \phi_1(SO(n-d))$, with ϕ_1 as defined in Eqn. 6. Let $J \in \mathbb{R}^{n\times d}$ be a tall-skinny matrix, made up of the first d columns of I_n; J acts as the "identity" element in $\mathcal{S}_{n,d}$. A vector in $T_{I_n}(SO(n))$, that is perpendicular to $T_{\phi_1(I_{n-d})}(I_n SO(n-d))$, when multiplied on right by J results in a tangent to $\mathcal{S}_{n,d}$ at J. This gives:

$$T_J(\mathcal{S}_{n,d}) = \{ \begin{bmatrix} C \\ -B^T \end{bmatrix} | C = -C^T, C \in \mathbb{R}^{d\times d}, B \in \mathbb{R}^{d\times(n-d)} \} \ . \tag{13}$$

   For any other point $U \in \mathcal{S}_{n,d}$, let $Q \in SO(n)$ be a matrix that rotates the columns of U to align with the columns of J, i.e. let $U = Q^T J$. Note that the choice of Q is not unique. It follows that the tangent space at U is given by: $T_U(\mathcal{S}_{n,d}) = \{Q^T G | G \in T_J(\mathcal{S}_{n,d})\}$.
2. **Tangent Structure of $\mathcal{G}_{n,d}$**: In this case, set $M = SO(n)$ and $H = \phi_2(SO(d) \times SO(n-d))$, with ϕ_2 as given in Eqn. 7. Using the same argument made before, the vectors tangent to $SO(n)$ and perpendicular to the space $(T_{I_d}(SO(d)) \times T_{I_{n-d}}(SO(n-d)))$, will also be tangent to $\mathcal{G}_{n,d}$ after multiplication on right by J. Thus, the tangent space at $[J] \in \mathcal{G}_{n,d}$ is given by:

$$T_{[J]}(\mathcal{G}_{n,d}) = \{ \begin{bmatrix} 0 \\ -B^T \end{bmatrix} \ | \ B \in \mathbb{R}^{d\times(n-d)} \} \tag{14}$$

   For any other point $[U] \in \mathcal{G}_{n,d}$, let $Q \in SO(n)$ be a matrix such that $U = Q^T J$. Then, the tangent space at $[U]$ is given by $T_U(\mathcal{G}_{n,d}) = \{Q^T G | G \in T_J(\mathcal{G}_{n,d})\}$.
3. **Tangent Structure of Kendall's Shape Space**: The pre-shape formed by n points lie on a $n-1$ dimensional complex hypersphere of unit radius. The Procrustes tangent coordinates of a preshape α are given by

$$v(\alpha, \mu) = \alpha\alpha^*\mu - \mu|\alpha^*\mu|^2. \tag{15}$$

   where μ is the Procrustes mean shape of the data.

So far we have introduced several manifolds of interest – namely $\mathbb{S}^n$, $\mathcal{S}_{n,d}$ and $\mathcal{G}_{n,d}$ – and have defined their geometries, including their tangent spaces, Riemannian metrics, geodesics and exponential maps. Now we consider the task of studying statistics on these manifolds.

4 Statistical Inference on Manifolds

What are the challenges in performing a statistical analysis if the underlying state space is non-Euclidean? Take the case of the simplest statistic, the sample mean, for a sample set $(x_1, x_2, \ldots, x_n)$ on $\mathbb{R}^n$:

$$\bar{x}_k = \frac{1}{k}\sum_{i=1}^{k} x_i, \qquad x_i \in \mathbb{R}^n \ . \tag{16}$$

Since $\bar{x}_k$ is a widely used and studied statistic, one already knows the pros and cons of using $\bar{x}_k$ as an estimate of the population mean. For example, we know that $\bar{x}_k$ is an unbiased and efficient estimator, but it is susceptible to the outliers. Now what if the underlying space is not $\mathbb{R}^n$ but a non-Euclidean manifold instead? To answer this question we consider an n-dimensional Riemannian manifold M. Let $d(p, q)$ denote the length of the shortest geodesic between arbitrary points $p, q \in M$. To facilitate a general discussion, we will assume that there exists an embedding $\varepsilon : M \to V$ where V is an m-dimensional Hilbert space ($n \leq m$). We have chosen V to be a vector space so that we can perform a statistical analysis in V using standard techniques from multivariate calculus. The distance between any two elements $p, q \in M$ is the geodesic distance $d(p, q)$ when the geodesic is restricted to be in M and it is $\|\varepsilon(p) - \varepsilon(q)\|$, with the norm of V, when the geodesic is allowed to be in V. The latter distance, of course, depends on the choice of the embedding ε. We start the analysis by assuming that we are given a probability density function f on M. This function, by definition, satisfies the properties that $f : M \to \mathbb{R}_{\geq 0}$ and $\int_M f(p)dp = 1$, where dp denotes the reference measure on M with respect to which the density f is defined. We can extend f to the larger set V by simply setting:

$$\tilde{f}(x) = \begin{cases} f(p) & \text{if } \ x = \varepsilon(p),\ p \in M \\ 0 & \text{if } \ x \notin \varepsilon(M) \end{cases} \ . \tag{17}$$

Naturally, $\tilde{f}$ is a probability density function on V. There are two possibilities for computing statistics on M – intrinsic and extrinsic. We describe them next.

4.1 Intrinsic Statistics

The first question that we consider is: What is a suitable notion of mean on the Riemannian manifold M? A popular method for defining a mean on a manifold was proposed by Karcher [24] who used the centroid of a density as its mean.

Definition 2 (Karcher Mean [24]). The Karcher mean μ_{int} of a probability density function f on M is defined as local minimizer of the cost function: $\rho : M \to \mathbb{R}_{\geq 0}$, where

$$\rho(p) = \int_M d(p,q)^2 f(q)\ dq\ . \tag{18}$$

dq denotes the reference measure used in defining the probability density f on M. The value of function ρ at the Karcher mean is called the **Karcher variance**. How does the definition of Karcher mean adapt to the sample set, i.e. a finite set of points drawn from an underlying probability distribution? Let $q_1, q_2, \ldots, q_k$ be independent random samples from the density f. Then, the sample Karcher mean of these points is defined to be the local minimizer of the function:

$$\rho_k(p) = \frac{1}{k}\sum_{i=1}^{k} d(p,q_i)^2\ . \tag{19}$$

An iterative algorithm for computing the sample Karcher mean is as follows. Let μ_0 be an initial estimate of the Karcher mean. Set $j = 0$.

1. For each $i = 1, \ldots, k$, compute the tangent vector v_i such that the geodesic from μ_j, in the direction v_i, reaches q_i at time one, i.e. $\psi_1(\mu_j, v_i) = q_i$ or $v_i = \exp_{\mu_j}^{-1}(q_i)$.
2. Compute the average direction $\bar{v} = \frac{1}{k}\sum_{i=1}^{k} v_i$.
3. If $\|\bar{v}\|$ is small, then stop. Else, update μ_j in the update direction using

$$\mu_{j+1} = \psi_\epsilon(\mu_j, \bar{v}),$$

   where $\epsilon > 0$ is small step size, typically 0.5. $\psi_t(p,v)$ denotes the geodesic path starting from p in the direction v parameterized by time t. In other words, $\mu_{j+1} = \exp_{\mu_j}(\epsilon\bar{v})$.
4. Set $j = j + 1$ and return to Step 1.

It can be shown that this algorithm converges to a local minimum of the cost function given in Eqn. 19 which by definition is μ_{int}. Depending upon the initial value μ_0 and the step size ϵ, it converges to the nearest local minimum.

We exploit the fact that the tangent spaces of M are vector spaces and can provide a domain for defining covariances. We can transfer the probability density f from M to a tangent space $T_p(M)$, using the inverse exponential map, and then use the standard definition of central moments in that vector space. For any point $p \in M$, let $p \to v \equiv \exp_\mu^{-1}(p)$ denote the inverse exponential map at μ from M to $T_\mu(M)$. The point μ maps to the origin $\mathbf{0} \in T_\mu(M)$ under this map. Now, we can define the Karcher covariance matrix as:

$$K_{int} = \int_{T_\mu(M)} vv^T f_v(v)dv, \quad v = \exp_\mu^{-1}(q)\ ,$$

where f_v is the induced probability density on the tangent space. For a finite sample set, the sample Karcher variance is given by

$$\hat{K}_{int} = \frac{1}{k-1}\sum_{i=1}^{k} v_i v_i^T, \quad \text{where} \quad v_i = \exp_\mu^{-1}(q_i)\ . \tag{20}$$

4.2 Extrinsic Statistics

The other possibility for performing statistics is to use the vector space structure of V to simplify calculations. In this case one transfers the probability measure to V, computes the pertinent statistical quantities in V and projects the final results back to M. Let $\Pi : V \rightarrow M$ be a projection map defined in such a way that

$$\Pi(v) = argmin_{p \in M} \|v - \varepsilon(p)\|^2 \ . \tag{21}$$

The existence and the uniqueness of Π, of course, depend on the nature of M, p and ε. Now, the extrinsic mean of a density f on M is defined as follows.

Definition 3 (Extrinsic Mean). The extrinsic mean of density f on M, specified with respect to an embedding ε of M in a larger vector space V, is given by

$$\mu_{ext} = \Pi(\nu),$$

where:

- Π is the projection defined in Eqn. 21,
- $\nu = \int_V v\tilde{f}(v)dv$ is the standard mean of $\tilde{f}$ in V, and
- $\tilde{f}$ is the unique extension of f from M to V (given by Eqn. 17).

Once the embedding ε has been chosen, and a mechanism for projection Π has been established, the rest of the process is quite straightforward. It requires computing the mean of $\tilde{f}$ in V and projecting it down to M. In case M is a Euclidean space, the projection is simply the identity operation and the extrinsic mean coincides with the classical mean. Additionally, in this case, if the Euclidean metric is chosen as the Riemannian metric, then the intrinsic mean also coincides with the classical mean.

What about the covariance analysis in an extrinsic framework? An extrinsic covariance can be defined similar to the extrinsic mean. Let $\pi : V \rightarrow T_\nu(M)$ be any linear map. Since it is a linear map, it can be written as a $n \times m$ matrix A so that $\pi(v) = Av$. Define the covariance

$$K_v = \int_V (v - \nu)(v - \nu)^t \tilde{f}(v)dv \ ,$$

in the vector space V and project it using:

$$K_{ext} = AK_vA^T \ . \tag{22}$$

The advantages and disadvantages of an extrinsic mean, with respect to the Karcher mean, are straightforward. The main advantage is its computational simplicity. Once an embedding ε is chosen, the rest of the analysis is quite standard and typically very fast. In contrast, computation of the Karcher mean requires repeated computations of the exponential and inverse exponential maps. The disadvantage is that the result $\Pi(\nu)$ depends on the

choice of embedding ε which is quite arbitrary. Different embeddings will result in different solutions, and the projection Π itself may not be unique.

Example 7 **(Extrinsic Mean of Subspaces).** As discussed in section 3.2, the Grassmann manifold can be viewed as a quotient space of the set of full-rank $n \times d$ orthonormal matrices. We can also associate to each d-dimensional subspace an $n \times n$ idempotent projection matrix P of rank d (not to be confused with the projection operation Π), such that $P = YY^T$, where Y is a point on the $\mathcal{S}_{n,d}$ whose columns span the subspace. The space of $n \times n$ projectors of rank d, denoted by $\mathbb{P}_{n,d}$ can be embedded into the set of all $n \times n$ matrices – $\mathbb{R}^{n \times n}$ – which is a vector space. The projection Π from $\mathbb{R}^{n \times n}$ to $\mathbb{P}_{n,d}$ is given by

$$\Pi(M) = UU^T, \text{where } M = USV^T \text{ is the d-rank SVD of } M. \tag{23}$$

Using this embedding, we can define an extrinsic distance metric on the Grassmann manifold using the distance metric inherited from $\mathbb{R}^{n \times n}$.

$$d^2(P_1, P_2) = tr(P_1 - P_2)^T (P_1 - P_2) \tag{24}$$

Given a set of sample points on the Grassmann manifold represented uniquely by projectors $\{P_1, P_2, \ldots, P_N\}$, we can compute the extrinsic mean by first computing the mean of the P_i's and then projecting the solution to the manifold by means of equation (23). i.e.

$$\mu_{ext} = \Pi(P_{avg}), \text{ where } P_{avg} = \frac{1}{N} \sum_{i=1}^{N} P_i \tag{25}$$

4.3 Learning Distributions from Data

In addition to sample statistics such as the mean and covariance, it is possible to define parametric probability distribution functions on manifolds. The intrinsic distributions are defined on the manifolds of interest directly without embedding them into a vector space. Examples of such distributions include the Langevin distribution for spherical data. Another intrinsic way of defining probability distributions is to project parametric distributions onto the manifold of interest. In addition to intrinsic methods such as these, we can estimate extrinsic distributions as well.

Example 8 **(Intrinsic Density Estimation).** Suppose, we have n sample points, given by $q_1, q_2, ...q_n$ from a manifold $\mathcal{M}$. Then, we first compute their Karcher mean $\bar{q}$ as discussed before. The next step is to define and compute a sample covariance for the observed q_i's. The key idea here is to use the fact that the tangent space $T_{\bar{q}}(q)$ is a vector space. For a d-dimensional manifold, the tangent space at a point is also d dimensional. Using a finite-dimensional approximation, say $V \subset T_{\bar{q}}(q)$, we can use the classical multivariate calculus for this purpose. The resulting sample covariance matrix is given by:

$$\bar{\Sigma} = \frac{1}{n-1}\sum_{i=1}^{n} v_i v_i^T$$

where each v_i is a d-dimensional sample of the function $exp_{\bar{q}}^{-1} q_i$. Note that by definition, the mean of v_is should be zero. In cases where the number of samples n is smaller than d, one can apply an additional dimension-reduction tool to work on a smaller space. For instance, we can use the singular value decomposition (SVD) of the sample covariance matrix $\bar{\Sigma}$ and retain only the top m significant singular values and the corresponding singular vectors. In such cases, the covariance matrix is indirectly stored using $\lambda_1, \lambda_2, ...\lambda_m$ singular values and their corresponding singular vectors $u_1, u_2, ...u_m$.

The exponential map: $\exp_{\bar{q}} : T_{\bar{q}}(q) \rightarrow \mathcal{M}$ maps this covariance back to $\mathcal{M}$. Specifically, this approach is widely used to define wrapped-Gaussian densities on a given manifold. In general, one can define arbitrary pdfs on the tangent plane such as mixtures of Gaussians, Laplace etc and project it back to the manifold via the exponential map. This allows us to experiment with and choose an appropriate pdf that works well for a given problem domain.

Example 9 **(Extrinsic Densities using Kernels).** Here we discuss density estimation over the Grassmann manifold using extrinsic methods proposed by [12]. Given two orthonormal bases Y_1 and Y_2 we define the distance between the subspaces as the smallest squared Euclidean distance between their corresponding equivalence classes on the Stiefel manifold. Hence,

$$d^2([Y_1],[Y_2]) = \min_{R \in SO(d)} tr(Y_1 - Y_2 R)^T (Y_1 - Y_2 R) \tag{26}$$

This distance is called the Procrustes distance [12]. This minimization can be solved in closed form. It is possible to relax the constraint that $R \in SO(d)$ to $R \in GL(d)$. In this case, the minimum is attained at $R = A$ and the distance is given by $d^2(Y_1, Y_2) = tr(I_k - A^T A)$, where $A = Y_1^T Y_2$. We refer the reader to [12] for derivations and other cases. Using this interpretation, we can define extrinsic statistics on the Grassmann manifold. Here, we discuss a non-parametric method for estimation of pdfs. Given several samples from a pdf, represented by orthonormal basis $(Y_1, Y_2, \ldots, Y_n)$, the density can be estimated using extrinsic methods and the Procrustes metric [12] as

$$\hat{f}(Y; M) = \frac{1}{n} C(M) \sum_{i=1}^{n} K[M^{-1/2}(I_k - Y_i^T Y Y^T Y_i) M^{-1/2}] \tag{27}$$

where $K(T)$ is the kernel function, M is a $k \times k$ positive definite matrix which plays the role of the kernel width or a smoothing parameter. $C(M)$ is a normalizing factor chosen so that the estimated density integrates to unity.

5 Applications and Experiments

In this section, we present several examples where an understanding of the manifold that the data lies on can provide a principled means of solving the problem. The examples we discuss include 1. human gait analysis, 2. activity analysis via state-space modeling and 3. modeling execution-rate variations in human activities.

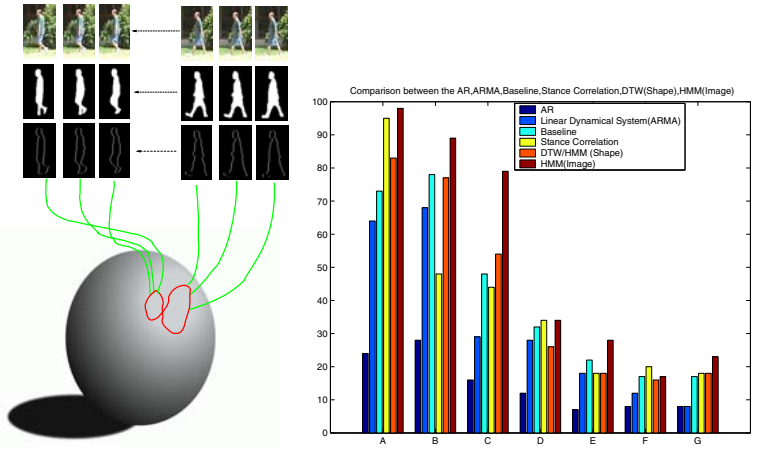
5.1 Feature Space Manifold: Kendall's Shape Sphere for Human Gait Analysis

Shape analysis plays a very important role in object recognition, matching and registration. There has been substantial work in shape representation and on defining a feature vector which captures the essential attributes of the shape. A description of shape must be invariant to translation, scale and rotation. The Kendall's shape space is a natural feature to use in such cases. Given a binary image consisting of the silhouette of a person, we extract the shape from this binary image. The procedure for obtaining shapes from the video sequence is graphically illustrated in Figure 3(a). Note that each frame of the video sequence maps to a point on the spherical shape manifold.

Consider a situation where there are two shape sequences and we wish to compare how similar these two shape sequences are. One may want to use non-parametric sequence matching such as Dynamic-Time warping or a parametric approach such as state-space modeling. In either case, we need to take into account the geometry of the shape-manifold for matching. Consider dynamic time warping, which has been successfully used by the speech recognition [34] community for performing non-linear time normalization. Pre-shape, as we have already discussed lies on a spherical manifold. In our experiments, we use the Procrustes shape distance described in section 3.2 during the DTW distance computations. For state-space modeling such as autoregressive (AR) or ARMA, we use the tangent structure of the manifold. We project a given sequence to the tangent plane constructed at the mean-point. The AR and ARMA model parameters are then estimated on the tangent-planes. The tangent structure for Kendall's shape manifold was discussed in 3.2.1. Once the model parameters are estimated, computing similarity between two sequences can be performed by computing the distance between the model parameters. We refer the reader to [48] for details of model fitting and computing similarity between the model-parameters. Next, we present some experiments that demonstrate the utility of these methods.

5.1.1 Gait Recognition Experiment on the USF Gait Database

The USF database [35] consists of 71 people in the Gallery. Various covariates such as camera position, shoe type, surface and time were varied in



(a) (b)

Fig. 3 (a)Graphical illustration of the sequence of shapes obtained during a walking cycle, (b)Bar Diagram comparing the identification rate of various algorithms.

a controlled manner to design a set of challenge experiments[1] [35]. On the USF database we conducted experiments on recognition performance using these methods- Stance Correlation, DTW on shape space, Stance based AR (a slight modification of the AR model [48]) and the ARMA model. Gait recognition experiments were designed for challenge experiments A-G. These experiments featured and tested the recognition performance against various covariates like the camera angle, shoe type, surface change etc. Refer to [35] for a detailed description of the various experiments and the covariates in these experiments. Figure 3(b) shows a comparison of the identification rate (rank 1) of the various shape and kinematics based algorithms. It is clearly seen that shape-based algorithms perform better than purely kinematics-based algorithms.

5.2 Model Space Manifold: Grassmann Manifold for Human Activity Analysis

Modeling of human activities is an important problem in video-understanding. Applications of activity recognition include activity-based indexing, biometrics, motion synthesis, and anomaly detection. Human activity analysis typically proceeds in a hierarchical fashion. At lower-levels, some features pertaining to motion of the human are extracted from video sequences such as optical flow or background subtracted masks. Then, a model is imposed on the feature evolution such as Hidden Markov Models (HMMs) or Linear Dynamic Systems (LDS). Given training data, the goal is to estimate the

[1] Challenge Experiments:Probes A-G in increasing order of difficulty.

model parameters. Here we study ARMA models and show that the study of these models can be formulated as a study of the geometry of the Grassmann manifold. A wide variety of time series data such as dynamic textures, human joint angle trajectories, shape sequences, video based face recognition etc are frequently modeled as ARMA models [37, 6, 48, 2]. The ARMA model equations are given by

$$f(t) = Cz(t) + w(t) \quad w(t) \sim N(0, R) \tag{28}$$

$$z(t+1) = Az(t) + v(t) \quad v(t) \sim N(0, Q) \tag{29}$$

where, z is the hidden state vector, A the transition matrix and C the measurement matrix. f represents the observed features while w and v are noise components modeled as normal with 0 mean and covariance R and Q respectively.

The model parameters (A, C) learned as above do not lie on a Euclidean space. The transition matrix A is constrained to be stable with eigenvalues inside the unit circle. The observation matrix C is constrained to be an orthonormal matrix. Now, starting from an initial condition $z(0)$, it can be shown that the *expected* observation sequence is given by

$$E\begin{bmatrix} f(0) \\ f(1) \\ f(2) \\ . \\ . \end{bmatrix} = \begin{bmatrix} C \\ CA \\ CA^2 \\ . \\ . \end{bmatrix} z(0) = O_\infty(M)z(0) \tag{30}$$

Thus, the expected observation sequence generated by a time-invariant model (A, C) lies in the column space of the extended *observability* matrix given by $O_\infty = [C^T, (CA)^T, (CA^2)^T, ...]^T$. However, motivated by the fact that human actions are of a finite-duration in time and not infinitely extending in time, we can simplify the study of the model by considering only an n-length expected observation sequence instead of the infinite sequence as above. Let the size of the temporal window be n. Thus, the n-length expected observation sequence generated by the model (A, C) lies in the column space of the *finite* observability matrix given by

$$O_n^T = \left[C^T, (CA)^T, (CA^2)^T, \dots (CA^{n-1})^T\right] \tag{31}$$

We can thus identify a dynamical model by a point on the Grassmann manifold, corresponding to the subspace spanned by the columns of the observability matrix. Since, the geometry of the Grassmann manifold is known we can use its geometry as discussed in sections 3.2 and 3.2.1 to define distances, exponential maps, and statistics (section 4) for video classification.

5.2.1 INRIA iXMAS Activity Recogntion Experiment

We performed a recognition experiment on the publicly available INRIA dataset [50]. The dataset consists of 10 actors performing 11 actions, each

Table 1 Comparison of view invariant recognition of activities in the INRIA dataset using a) Best DimRed [50] on $16 \times 16 \times 16$ features, b) Best Dim. Red. [50] on $64 \times 64 \times 64$ features, c) Nearest Neighbor using Subspace angles ($16 \times 16 \times 16$ features) d) Nearest Neighbor using Procrustes distance ($16 \times 16 \times 16$ features), e) Maximum likelihood using wrapped Gaussian($16 \times 16 \times 16$ features) f) Maximum likelihood using Parzen windows on the Grassmann manifold ($16 \times 16 \times 16$ features)

Activity	**Dim. Red.** [50] 16^3 volume	**Best Dim. Red.** [50] 64^3 volume	**Subspace Angles** 16^3 volume	**Procrustes Metric** 16^3 volume	**Wrapped Normal** 16^3 volume	**Extrinsic Kernel** 16^3 volume
Check Watch	76.67	86.66	93.33	90	100	100
Cross Arms	100	100	100	96.67	96.67	100
Scratch Head	80	93.33	76.67	90	100	96.67
Sit Down	96.67	93.33	93.33	93.33	90	93.33
Get Up	93.33	93.33	86.67	80	96.67	96.67
Turn Around	96.67	96.67	100	100	96.67	100
Walk	100	100	100	100	100	100
Wave Hand	73.33	80	93.33	90	90	100
Punch	83.33	96.66	93.33	83.33	100	100
Kick	90	96.66	100	100	93.33	100
Pick Up	86.67	90	96.67	96.67	93.33	100
Average	88.78	93.33	93.93	92.72	96.06	98.78

action executed 3 times at varying rates while freely changing orientation. We used the view-invariant representation and features as proposed in [50]. Specifically, we used the $16 \times 16 \times 16$ circular FFT features proposed by [50]. Each activity was modeled as a linear dynamical system. Testing was performed using a round-robin experiment where activity models were learnt using 9 actors and tested on 1 actor. In table 1, we show the recognition results obtained using four methods. The first column shows the results obtained using dimensionality reduction approaches of [50] on $16 \times 16 \times 16$ features. [50] reports recognition results using a variety of dimensionality reduction techniques (PCA, LDA, Mahalanobis) and here we choose the row-wise best performance from their experiments (denoted 'Best Dim. Red.') which were obtained using $64 \times 64 \times 64$ circular FFT features. The third column presents results using the method of using subspace angles based distance between dynamical models [13]. This is closely related to the geodesic on the Grassmann manifold for finite observability matrices. Column 4 shows the nearest-neighbor classifier performance using Procrustes metric on the Grassmann manifold ($16 \times 16 \times 16$ features). We see that the manifold Procrustes distance performs as well as subspace angles. But, statistical modeling of class conditional densities for each activity using parametric and non-parametric methods, leads to a significant improvement in recognition performance. In addition to activity analysis and ARMA modeling, we refer the reader to [45] for more example applications of statistical modeling on the Grassmann manifold in computer vision applications.

5.2.2 Activity Based Summarization

The ARMA model described above in conjunction with statistical models on the Grassmann manifold can be used to summarize long videos. Towards this

purpose, we describe long videos as outputs of time-varying ARMA models given by

$$f(t) = C(t)z(t) + w(t) \quad w(t) \sim N(0, R(t)) \tag{32}$$
$$z(t+1) = A(t)z(t) + v(t) \quad v(t) \sim N(0, Q(t)) \tag{33}$$

Note that here the model parameters (A, C, Q, R) are allowed to vary with time. Further, we assume that the model parameters change slowly with time so that they can be approximated as locally constant. Thus, parameter estimation is done in short-temporal windows (say of length 20 frames). This gives rise to a sequence of model parameters $M_t = (A_t, C_t)$. Each element in the sequence can be considered to be a point on the Grassmann manifold arising due to the time-varying observability matrix.

$$O_n(M_t) = \left[C_t; C_t A_t; \ldots; C_t A_t^{n-1} \right] \tag{34}$$

Thus, the time-varying model can be viewed as a sequence of subspaces S_t, where each subspace is spanned by the columns of the observability matrix at the corresponding time instant. Thus, the sequence of subspaces can be seen as a trajectory on the Grassmann manifold. To compactly represent the subspace variations, we parametrize the trajectory using a switching model akin to the HMM on the Grassmann manifold. This representation can be used to provide a visual summarization of long videos [43]. The clusters of the HMM represent the distinct actions in the video e.g. spins, leaps, glides for the case of skating. The transition structure between the clusters represents how the overall activity in the video proceeds. In this experiment we show the results of summarizing a long video containing a complex activity – the game of Blackjack. For this, we used the dataset reported in [51]. A few sample frames from the dataset are shown in figure 4. The game of Blackjack consists of a few elements such as dealing cards, waiting for bids, shuffling the cards etc. We try to estimate a Grassmann switching model for the entire video of Blackjack. The Grassmann switching model would then represent a 'summary' of the game, where the clusters of the model represent various elements of the game and the switching structure represents how the game progresses. This video consists of about 1700 frames. We extracted the motion-histogram features as proposed in [51] for each frame of the video. The time-varying model parameters are estimated in sliding windows of size 10. The dimension of the state vector is chosen to be $d = 5$. To estimate the Grassmann switching model for the game of Blackjack, we manually set the number of clusters to 5. In figure 5(a), we show an embedding of the video obtained from the model parameters using Laplacian eigenmaps. Each point corresponds to a time-invariant model parameter (A, C) pair or equivalently a point on the Grassmann manifold. Each cluster was found to correspond to a distinct element of the game as shown. The switching structure between the clusters is encoded in the transition matrix and is shown in figure 5(b).

Fig. 4 A few sample frames from the Blackjack dataset of [51].

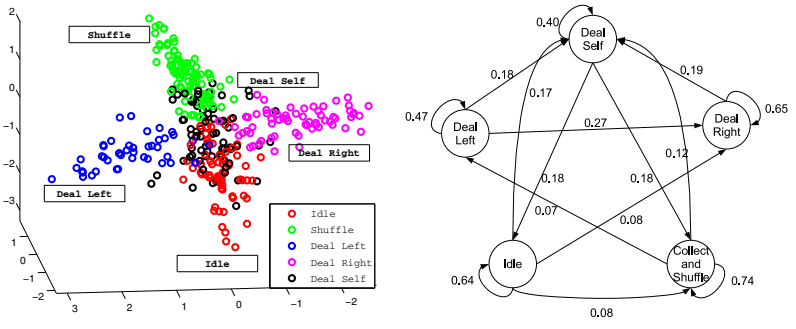


Fig. 5 (a) An embedding of the entire Blackjack video sequence. Figure best viewed in color. (b) Estimated structure of the game of Blackjack. (For the sake of clarity arcs with low weights have not been shown)

Similar ARMA models were also used in [44] for summarizing a long skating video sequence.

5.3 Transformation Space Manifold: Hilbert Sphere for Modeling Execution-Rate Variations in Activities

In activity recognition, different instances of the same activity may consist of varying relative speeds at which the actions are executed, in addition to other intra- and inter- person variabilities. Most existing algorithms for activity recognition are not very robust to intra- and inter-personal changes of the same activity, and are sensitive to warping of the temporal axis due to variations in speed profile. Results on gait-based person identification shown in [7] indicate that it is very important to take into account the temporal variations in the person's gait. In [49], it was shown that accounting for execution rate enhances recognition performance for action recognition. Typical approaches for accounting for variations in execution rate are either directly based on the dynamic time warping (DTW) algorithm [34] or some variation of this algorithm [49].

For now, let us assume that for each frame of the video, an appropriate feature has been extracted and that the video data has now been converted

into a feature sequence given by $f^1, f^2, ...$, for frames $1, 2, ...$ respectively. We will use $\mathcal{F}$ to denote the feature space associated with the chosen feature. Let γ be a diffeomorphism (A diffeomorphism is a smooth, invertible function with a smooth inverse) from $[0, 1]$ to itself with $\gamma(0) = 0$ and $\gamma(1) = 1$. Also, let $\mathbf{\Gamma}$ be the set of all such functions. We will use elements of $\mathbf{\Gamma}$ to denote time warping functions. Our model for an activity consists of an average activity sequence given by $a : [0, 1] \to \mathcal{F}$, a parameterized trajectory on the feature space. Any time-warped realization of this activity is then obtained using:

$$r(t) = a(\gamma(t)), \quad \gamma \in \mathbf{\Gamma} \ . \tag{35}$$

Equation (35) actually defines an action of $\mathbf{\Gamma}$ on $\mathcal{F}^{[0,1]}$, the space of all continuous activities. In our model, the variability associated with γ in each class will be modeled using a distribution P_γ on $\mathbf{\Gamma}$. For the convenience of analysis and computation, we prefer to work with $\psi = +\sqrt{\dot{\gamma}}$ instead of γ directly. There is a bijection between γ and ψ and the probability models on ψ directly relate to equivalent models on γ. Thus, we will introduce probability distributions P_ψ on the set of all ψs, for each activity class.

The parameters of this model are $a(t)$, the nominal activity trajectory, and P_ψ, the probability distribution on square-root representations of time warping functions. In general, the nominal activity trajectory $a(t)$ can also be chosen to be random. Here, we restrict our analysis to cases where the nominal activity trajectory $a(t)$ is deterministic but unknown. We will consider parametric forms of densities for P_ψ and reduce the problem of learning P_ψ to one of learning the parameters of the distribution P_ψ.

Let the space of all square-root density forms be given by

$$\mathbf{\Psi} = \{\psi : [0, 1] \to \mathbb{R} | \psi \geq 0, \int_0^1 \psi^2(t)dt = 1\} \ . \tag{36}$$

This is the positive orthant of a unit hypersphere in the Hilbert space of all square-integrable functions on $[0, 1]$. Let $T_\psi(\mathbf{\Psi})$ be the tangent space to $\mathbf{\Psi}$ at any given point ψ. Then, for any v_1 and v_2 in $T_\psi(\mathbf{\Psi})$, the Fisher-Rao metric is given by

$$\langle v_1, v_2 \rangle = \int_0^1 v_1(t)v_2(t)dt. \tag{37}$$

Since $\mathbf{\Psi}$ is a sphere, its geometry is well known and we can directly use known expressions for geodesics, exponential maps, and inverse exponential maps on $\mathbf{\Psi}$ as discussed in sections 3.2 and 3.2.1. Consequently, the algorithms for computing sample statistics, defining probability density functions, and generating inferences also become straightforward.

5.3.1 Common Activities Dataset

We used the UMD common activities dataset [49], a dataset of common activities to perform preliminary experiments to validate our model. The dataset

consists of 10 activities and 10 different instances of each activity. We partition the dataset into 10 disjoint sets each containing 1 instance of every activity. In order to test the recognition performance for each set, we first learn the model parameters from the remaining nine sets and then perform recognition for the test sequences. We repeat the process for each of the 10 sets. Thus we ensure that there is no overlap between the training set and the test sequences. Figure 6 shows the 10×100 similarity matrix for using the function space algorithm with the uniform distribution on the space of temporal warps. Each column corresponds to a different test sequence while each row corresponds to a different activity. The strongly block diagonal nature of the similarity matrix indicates that the recognition algorithm performs well. In fact, on this database we obtained 100% recognition using both our algorithms.

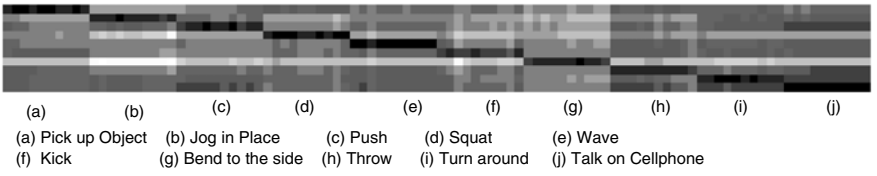


Fig. 6 10 X 100 Similarity matrix of 100 sequences and 10 different activities using the function space algorithm.

5.3.2 USF Gait Database

Since the model for learning the function space of time-warpings is not explicitly dependent on the choice of features, one could potentially use the same model to learn individual specific function spaces in order to perform activity-based person identification. The only difference would be that we would choose a feature that is person-specific (e.g., silhouette). The nominal activity trajectory would be individual specific in this case. Various external conditions (such as surface, shoe) induce systematic time-warping variations within the gait signatures of each individual. The function space of temporal warpings for each individual amounts to learning the class of person specific warping functions. By learning the function space of these variations we are able to account for the effects of such external conditions.

In order to compare the performance of our algorithm with the current state of the art algorithms, we also performed a gait-based person identification experiment on the publicly available USF gait database [35]. The USF database consists of 71 people in the Gallery. Various covariates like camera position, shoe type, surface and time were varied in a controlled manner to design a set of challenge experiments [35]. We performed a round-robin recognition experiment in which one of the challenge sets was used as test while the other seven were used as training examples. The process was repeated for each of the seven challenge sets on which results have been reported.

Table 2 Comparison of Identification rates on the USF dataset. Note that the experimental results reported in this table contain varying amounts of training data. While columns 2-6 (Baseline - pHMM) used only the gallery sequences for training, the results reported in columns 7-10 (P_{LW} - $P_{GaussIm}$) used all the probes except the test probe during training.

Probe	Baseline	DTW Shape	HMM Shape	HMM Image	pHMM [29]	P_{LW}	P_{Unif}	P_{Gauss}	$P_{GaussIm}$
Avg.	42	42	41	50	65	51.5	59	59	64
A	79	81	80	96	85	68	70	78	82
B	66	74	72	86	89	51	68	68	78
C	56	52	56	74	72	51	81	82	76
D	29	29	22	32	57	53	40	50	48
E	24	20	20	28	66	46	64	51	54
F	30	19	20	17	46	50	37	42	56
G	10	19	19	21	41	42	53	40	55

Table 2 shows the identification rates of our algorithm with a uniform distribution on the space of warps (P_{Unif}), our algorithm with a wrapped Gaussian distribution on the tangent space of warps with shape as a feature and with binary image feature (P_{Gauss} and $P_{GaussIm}$). For comparison the table also shows the baseline algorithm [35], simple DTW on shape features [48] and the image-based HMM [23] algorithm on the USF dataset for the 7 probes A-G. Since most of these other algorithms could not account for the systematic variations in time-warping for each class the recognition experiment they performed was not round robin but rather used only one sample per class for learning. Therefore, to ensure a fair comparison, we also implemented a round-robin experiment using the linear warping (P_LW).

The average performance of our algorithms P_{Unif} and P_{Gauss} are better than all the other algorithms that use the same feature, (DTW/HMM (Shape) [48] and Linear warping P_{LW}) and is also better than the baseline [35] and HMM [23] algorithms that use the image as a feature. The improvement in performance while using binary image as a feature is shown in the last column ($P_{GaussIm}$). The experimental results presented here clearly show that using multiple training samples per class and learning the distribution of their time warps makes significant improvement to gait recognition results. While most algorithms based on learning from a single sample led to overfitting and therefore performed much better when the gallery was similar to the probe (Probe A-C), they also performed very poorly when the gallery and the probes were significantly different. But, since our algorithm has good generalization ability the performance of our algorithm did not suffer from overfitting and therefore did not drop as much when moving from probes A-C to Probes D-G.

6 Conclusions

In this chapter we provided a brief overview of the usefulness and effectiveness of statistical analysis on manifolds to specific applications in video analysis. Typical video analysis is usually composed of three stages of processing - feature extraction, building models and accounting for transformation invariance. We highlight three different applications of manifold analysis, one for each of the three stages in a typical video analysis framework. We describe Kendall shape manifold for shape feature representation. We show the applicability of the Grassmann manifold for understanding dynamical models. Finally, we show the space of time-warp transformations as a spherical manifold of functions. In all applications, we show experiments that illustrate the superior performance of algorithms that exploit the geometric properties of the underlying manifold.

References

1. Absil, P.A., Mahony, R., Sepulchre, R.: Optimization Algorithms on Matrix Manifolds. Princeton University Press, Princeton (2008)
2. Aggarwal, G., Roy-Chowdhury, A., Chellappa, R.: A system identification approach for video-based face recognition. In: International Conference on Pattern Recognition (ICPR) (2004)
3. Begelfor, E., Werman, M.: Affine invariance revisited. In: IEEE International Conference on Computer Vision and Pattern Recognition, pp. 2087–2094 (2006)
4. Bhattacharya, R., Patrangenaru, V.: Nonparametric estimation of location and dispersion on Riemannian manifolds. Journal for Statistical Planning and Inference 108, 23–36 (2002)
5. Bhattacharya, R., Patrangenaru, V.: Large sample theory of intrinsic and extrinsic sample means on manifolds- I. The Annals of Statistics 31(1), 1–29 (2003)
6. Bissacco, A., Chiuso, A., Ma, Y., Soatto, S.: Recognition of human gaits. In: IEEE International Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 52–57 (2001)
7. Bobick, A., Tanawongsuwan: Performance analysis of time-distance gait parameters under different speeds. In: Kittler, J., Nixon, M.S. (eds.) AVBPA 2003. LNCS, vol. 2688. Springer, Heidelberg (2003)
8. Bookstein, F.: Size and shape spaces for landmark data in two dimensions. Statistical Science 1, 181–242 (1986)
9. Boothby, W.M.: An introduction to differentiable manifolds and Riemannian geometry. Academic Press Inc., London (1975)
10. Brockett, R.: Notes on Stochastic Processes on Manifolds. In: Systems and Control in the Twenty-First Century: Progress in Systems and Control, vol. 22. Birkhäuser, Basel (1997)
11. Brockett, R.W.: System theory on group manifolds and coset spaces. SIAM Journal on Control 10(2), 265–284 (1972)
12. Chikuse, Y.: Statistics on special manifolds. Lecture Notes in Statistics. Springer, Heidelberg (2003)

13. Cock, K.D., Moor, B.D.: Subspace angles and distances between ARMA models. In: Proceedings of the Intl. Symposium of Mathematical Theory of Networks and Systems, MTNS (2000)
14. Dryden, I., Mardia, K.: Statistical shape analysis. John Wiley and Sons, Chichester (1998)
15. Dryden, I.L., Mardia, K.V.: Statistical Shape Analysis. John Wiley & Son, Chichester (1998)
16. Edelman, A., Arias, T., Smith, S.T.: The geometry of algorithms with orthogonality constraints. SIAM Journal of Matrix Analysis and Applications 20(2), 303–353 (1998)
17. Georghiades, A.S., Kriegman, D.J., Belhumeur, P.N.: Illumination cones for recognition under variable lighting: Faces. In: IEEE International Conference on Computer Vision and Pattern Recognition, pp. 52–59 (1998)
18. Grenander, U.: Probabilities on Algebraic Structures. Wiley, Chichester (1963)
19. Grenander, U.: General Pattern Theory. Oxford University Press, Oxford (1993)
20. Grenander, U., Miller, M.I.: Computational anatomy: An emerging discipline. Quarterly of Applied Mathematics LVI(4), 617–694 (1998)
21. Grenander, U., Miller, M.I., Srivastava, A.: Hilbert-Schmidt lower bounds for estimators on matrix Lie groups for ATR. IEEE Transactions on Pattern Analysis and Machine Intelligence 20(8), 790–802 (1998)
22. Hartley, R.I., Zisserman, A.: Multiple View Geometry in Computer Vision, 2nd edn. (2004)
23. Kale, A., Sundaresan, A., Rajagopalan, A., Cuntoor, N., Roy Cowdhury, A., Krueger, V., Chellappa, R.: Identification of humans using gait. IEEE Transactions on Image Processing 13(9), 1163–1173 (2004)
24. Karcher, H.: Riemannian center of mass and mollifier smoothing. Communications on Pure and Applied Mathematics 30, 509–541 (1977)
25. Kendall, D.: Shape manifolds, procrustean metrics and complex projective spaces. Bulletin of London Mathematical society 16, 81–121 (1984)
26. Kendall, D.G.: Shape manifolds, procrustean metrics and complex projective spaces. Bulletin of London Mathematical Society 16, 81–121 (1984)
27. Klassen, E., Srivastava, A., Mio, W., Joshi, S.: Analysis of planar shapes using geodesic paths on shape spaces. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(3), 372–383 (2004)
28. Le, H.L., Kendall, D.G.: The Riemannian structure of Euclidean shape spaces: a novel environment for statistics. Annals of Statistics 21(3), 1225–1271 (1993)
29. Liu, Z., Sarkar, S.: Improved gait recognition by gait dynamics normalization. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(6), 863–876 (2006)
30. Lui, Y.M., Beveridge, J.R.: Grassmann registration manifolds for face recognition. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part II. LNCS, vol. 5303, pp. 44–57. Springer, Heidelberg (2008)
31. Miller, M.I., Younes, L.: Group actions, homeomorphisms, and matching: A general framework. International Journal on Computer Vision 41(1/2), 61–84 (2001)
32. Mio, W., Srivastava, A., Joshi, S.: On shape of plane elastic curves. International Journal on Computer Vision 73(3), 307–324 (2007)
33. Pennec, X., Ayache, N.: Uniform distribution, distance and expectation problems for geometric features processing. Journal of Mathematical Imaging and Vision 9(1), 49–67 (1998)

34. Rabiner, L., Juang, B.: Fundamentals of speech recognition. Prentice-Hall, Englewood Cliffs (1993)
35. Sarkar, S., Phillips, P., Liu, Z., Vega, I., Grother, P., Bowyer, K.: The humanid gait challenge problem: data sets, performance, and analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence, 162–177 (2005)
36. Small, C.G.: The Statistical Theory of Shape. Springer, Heidelberg (1996)
37. Soatto, S., Doretto, G., Wu, Y.N.: Dynamic textures. In: IEEE International Conference on Computer Vision, vol. 2, pp. 439–446 (2001)
38. Spivak, M.: A Comprehensive Introduction to Differential Geometry, vol. 1. Publish or Perish, Inc., Houston (1970)
39. Srivastava, A., Joshi, S., Mio, W., Liu, X.: Statistical shape analysis: Clustering, learning and testing. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(4), 590–602 (2005)
40. Srivastava, A., Klassen, E.: Monte Carlo extrinsic estimators for manifold-valued parameters. IEEE Trans. on Signal Processing 50(2), 299–308 (2001)
41. Srivastava, A., Klassen, E.: Bayesian, geometric subspace tracking. Journal for Advances in Applied Probability 36(1), 43–56 (2004)
42. Subbarao, R., Meer, P.: Nonlinear mean shift over riemannian manifolds. International Journal on Computer Vision 84(1), 1–20 (2009)
43. Turaga, P., Chellappa, R.: Locally time-invariant models of human activities using trajectories on the grassmannian. In: IEEE International Conference on Computer Vision and Pattern Recognition, pp. 2435–2441 (2009)
44. Turaga, P., Veeraraghavan, A., Chellappa, R.: Unsupervised view and rate invariant clustering of video sequences. Computer Vision and Image Understanding 113(3), 353–371 (2009)
45. Turaga, P.K., Veeraraghavan, A., Chellappa, R.: Statistical analysis on Stiefel and Grassmann manifolds with applications in computer vision. In: IEEE International Conference on Computer Vision and Pattern Recognition (2008)
46. Tuzel, O., Porikli, F., Meer, P.: Region covariance: A fast descriptor for detection and classification. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 589–600. Springer, Heidelberg (2006)
47. Tuzel, O., Porikli, F., Meer, P.: Pedestrian detection via classification on Riemannian manifolds. IEEE Transactions on Pattern Analysis and Machine Intelligence 30(10), 1713–1727 (2008)
48. Veeraraghavan, A., Roy-Chowdhury, A., Chellappa, R.: Matching shape sequences in video with an application to human movement analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence 27(12), 1896–1909 (2005)
49. Veeraraghavan, A., Srivastava, A., Roy Chowdhury, A.K., Chellappa, R.: Rate-invariant recognition of humans and their activities. IEEE Transactions on Image Processing 18(6), 1326–1339 (2009)
50. Weinland, D., Ronfard, R., Boyer, E.: Free viewpoint action recognition using motion history volumes. Computer Vision and Image Understanding 104(2), 249–257 (2006)
51. Zhong, H., Shi, J., Visontai, M.: Detecting unusual activity in video. In: IEEE International Conference on Computer Vision and Pattern Recognition, pp. 819–826 (2004)

Part III
Semantic Video Analysis

Semantic Video Content Analysis

Massimiliano Albanese, Pavan Turaga, Rama Chellappa,
Andrea Pugliese, and V.S. Subrahmanian

Abstract. In recent years, there has been significant interest in the area of automatically recognizing activities occurring in a camera's field of view and detecting abnormalities. The practical applications of such a system could include airport tarmac monitoring, or monitoring of activities in secure installations, to name a few. The difficulty of the problem is compounded by several factors: detection of primitive actions in spite of changes in illumination, occlusions and noise; complex multi-agent interaction; mapping of higher-level activities to lower-level primitive actions; variations in which the same semantic activity can be performed. In this chapter, we develop a theory of semantic activity analysis that addresses each of these issues in an integrated manner. Specifically, we discuss ontological representations of knowledge of a domain, integration of domain knowledge and statistical models for achieving semantic mappings, definition of logical languages to describe activities, and design of frameworks which integrate all the above aspects in a coherent way, thus laying the foundations of effective Semantic Video Content Analysis systems.

1 Introduction

Interaction amongst humans and with objects forms the basis of almost all human activities. Semantics refers to the meaning associated with a particular interaction. While the set of all possible interactions can be quite large, many activities are constrained in nature. These constraints are induced in part by the surrounding environment under consideration, such as an airport, and in part by the underlying motive of the interaction, such as a hand-shake. The visual analysis of activities performed

Massimiliano Albanese, Pavan Turaga, Rama Chellappa, V.S. Subrahmanian
Institute for Advanced Computer Studies,University of Maryland, College Park, MD 20742
e-mail: albanese,pturaga,rama,vs@umiacs.umd.edu

Andrea Pugliese
DEIS, University of Calabria, Rende, Italy
e-mail: apugliese@deis.unical.it

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 147–176.
springerlink.com

by a group of humans in such settings is of great importance in applications such as automated security and surveillance systems. In real life, one usually has some prior knowledge of the semantic structures of activities that occur in a given setting. Taking into account this knowledge, we need to understand how to represent the constraints induced both by the surrounding environment and by the motive of the particular activity. The representation and analysis of these constraints and the mapping to semantically meaningful concepts form the subject of this chapter.

There has been significant interest in the area of automatically recognizing activities occurring in a camera's field of view and detecting abnormalities. Several factors contribute to the complexity of the problem: a) unambiguous detection of low-level primitive actions in spite of changes in illumination, occlusions and noise; b) high-level representation of activities in settings which are usually characterized by complex multi-agent interaction; c) mapping higher-level concepts to lower-level primitive actions; d) understanding the variations in which the same semantic activity can be performed and achieving robustness to such variations during recognition.

In this chapter, we develop a theory of semantic activity analysis that addresses each of these issues in an integrated manner. Specifically, Section 2 summarizes the major challenges and issues in the field of semantic activity analysis, and proposes a general framework which integrates all these aspects in a coherent way. Section 3 discusses ontological representations of domain knowledge, while Section 4 proposes two different solutions – based on Finite State Automata and Petri Nets respectively – for integrating domain knowledge and statistical models, and achieving semantic mappings. Section 5 explores logical languages for describing activities. Finally, experimental evaluations of the proposed techniques are reported in Section 6 and concluding remarks are given in Section 7.

2 Human Interactions: Challenges and Issues

Designing a system for semantic analysis of human actions from video requires one to adopt a principled approach starting from defining an activity. This requires specifying the domain of interest and the types of activities of interest. Once we enumerate and define what we mean by activities, we then need to represent them by appropriate computational models. This forms the basis for any automatic approach to recognizing human activities. However, video data brings with it unique challenges that require specific attention to enable deploying semantic models for real applications. Many of the key challenges are a consequence of the nature of the imaging process. Cameras generate mappings from the three-dimensional world to a two-dimensional image plane. This necessarily introduces several ambiguities which make reasoning difficult and low-level processing prone to errors. Therefore, video analysis requires methods that are not only semantically rich, but also robust to errors in the low-level processing. Furthermore, achieving all of these goals in a real-time setting requires designing computationally efficient algorithms.

Another major issue in automatic activity detection is the semantic gap between the types of primitive actions that can be automatically recognized using state of

the art image processing algorithms (e.g., a person walking on a street, the presence of a package-like object in the scene) and the type of complex interactions that automatic surveillance systems are intended to detect. Therefore, expressive and flexible models are needed to describe complex interactions in terms of primitive actions. The challenge here is to build such models in a way that is general enough to be applied to different domains and robust to noise and variations from standard scenarios. Thus, video mining requires addressing the following core issues:

1. **Representation:** Defining the meaning of an activity.
2. **Model:** Defining computational models for an activity.
3. **Robustness:** Designing algorithms robust to ambiguities in the imaging process.
4. **Efficiency:** Designing efficient recognition algorithms.

In this chapter, we discuss each of these problems and provide the reader with a deeper understanding of the underlying challenges. We demonstrate the effectiveness of the proposed design philosophy using experiments on real video sequences.

2.1 Unified Framework for Activity Semantics

Although a huge amount of work has been done on many of the individual issues discussed in the previous section, considerably less effort has been put towards the definition of a framework where all the aspects of Semantic Video Content Analysis are integrated in a coherent and effective way. Broadly speaking, there are three main classes of problems – from a user's perspective – that such an integrated system should address:

- **Evidence.** Given a video v, a set of activity definitions $\mathscr{A}$, a time interval (t_s, t_e), and a probability threshold p_t, find all the minimal subsequences of v containing occurrences X of activities in $\mathscr{A}$ such that X occurs within the interval (t_s, t_e) with probability $p \geq p_t$.
- **Identification.** Given a video v, a set of activity definitions $\mathscr{A}$, a time interval (t_s, t_e), find the activity which occurs in v within the interval (t_s, t_e) with maximal probability among the activities in $\mathscr{A}$.
- **Online Identification.** Given a real-time video feed $v = \langle f_1, f_2, \ldots, f_t \rangle$, where f_t is the current frame, and a set of activity definitions $\mathscr{A}$, find the probability that each activity in $\mathscr{A}$ is unfolding at current time t.

In this chapter, we put particular emphasis on discussing how all the components of a Semantic Video Content Analysis system can be integrated and propose a framework and design methodology that takes into account these requirements and the issues outlined in the previous section. We show in real experiments the effectiveness of the proposed design philosophy.

Figure 1 shows the general architecture of the proposed framework. The front-end of the system consists of the usual physical elements, such as cameras and video storage units. At the core, there is an ontological repository of activity and domain definitions, possibly provided by an expert. The Ontology consists of the 'vocabulary' and the 'grammar' of human activities. To bridge the gap between the

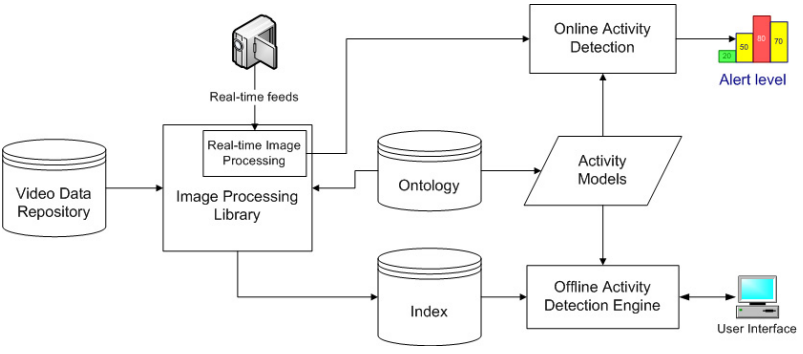


Fig. 1 General architecture of a Semantic Video Content Analysis system

low-level features at the front-end and the activity-definitions at the core, we require image and video analysis methods that can provide detection and recognition of the 'vocabulary'. Computational algorithms for activity recognition draw upon these, and identify activities of interest defined according to the 'grammar'.

3 Knowledge Representation

With the proliferation of visual surveillance systems in a wide variety of domains such as banks, airports, and convenience stores, it is necessary to create a general representation framework for modeling activities. Examples of such knowledge-bases, or more formally, ontologies have been in existence in other fields of AI such as the semantic web, image and video annotation and computational genomics. Designing ontologies for human activities has only recently gained interest in the computer vision community. The advantages of such a centralized representation are easily seen. It standardizes activity definitions, allows for easy portability to specific deployments, enables interoperability of different systems and allows easy replication and comparison of system performance. Since video-mining approaches rely on a domain expert to provide activity semantics, it is useful to create a standardized ontology from which to draw upon. Recent efforts have focused at creating ontologies for activities in specific scenarios. Examples include analysis of social interactions in nursing homes [6], classification of meeting videos [13], and activity detection in a bank monitoring setting [10]. As a result of the Video Event Ontology Challenge Workshop held in 2003 [12], ontologies have been defined for six video surveillance domains: 1) Perimeter and Internal Security, 2) Railroad Crossing Surveillance, 3) Visual Bank Monitoring, 4) Visual Metro Monitoring, 5) Store Security, 6) Airport-Tarmac Security. The workshop led to the development of two formal languages – Video Event Representation Language (VERL) [15], which provides an ontological representation of complex events in terms of simpler sub-events, and Video Event Markup Language (VEML), which is used to annotate VERL events in videos.

In most practical deployments, activity definitions are constructed in an empirical or ad-hoc manner. Though empirical constructs are fast to design and even work

```
PROCESS(cruise-parking-lot(vehicle v, parking-lot lot),
    Sequence(enter(v, lot), Repeat(AND(move-in-circuit(v), inside(v, lot))),
        exit(v, lot)))
```

Fig. 2 Cruise Parking Lot ontology

very well in most cases, they are limited in their utility to the specific deployment for which they have been designed. A principled approach must then be adopted. An ontology for human activities describes entities, environment, interaction among them and the sequence of events that is semantically identified with an activity. It specifies how an activity can be composed using lower-level primitive events and identifying the role played by each entity in the sequence of events. Building activity ontologies for a domain may be broadly classified into the following steps:

Entities: Define animate and inanimate entities relevant to an activity. The list could include humans, vehicles and environment (e.g. scene location, buildings).

Identity: Specify the properties – such as physical attributes and appearance – that uniquely identify each entity.

Spatio-temporal Attributes: Specify spatio-temporal attributes representing the state of an entity. Examples include definitions of 'near', 'far', 'before', 'after'.

Activity Primitives: Describe primitive events involving each entity individually and in relation to others.

Description of Activities: Define the relationship between entities and the spatio-temporal sequence of change and interaction among the entities.

Human activities are characterized by complex spatio-temporal interactions. Consider the activity of a vehicle cruising in a parking lot whose definition is given in Figure 2. This definition illustrates the importance of encoding spatio-temporal constraints. Without regard to such constraints, we might say that it is composed of the primitives: 'enter', 'move-in-circuit' and 'exit'. But, a normal car-parking activity can also be described using the same set of primitives. The difference that sets apart the two activities lies in the temporal span of the 'move-in-circuit' primitive.

3.1 Designing a Good Ontology

Thomas Gruber [11] made one of the earliest systematic attempts to propose guidelines for the design of ontologies intended for knowledge sharing and interoperability. Five important criteria for ontology design – clarity, coherence, extendibility, minimal encoding bias and minimal ontological commitment – were proposed.

Clarity: An ontology should convey the meaning of all conceptualizations unambiguously.

Coherence: An ontology should be coherent and allow for meaningful inferences to be drawn that are consistent with definitions and axioms.

Extendibility: The design of the ontology should allow future extensions without the need for revising definitions.

```
SINGLE-THREAD(suspicious-load(vehicle v, person p, ent obj, facility fac),
    AND(zone(loading-area), near(loading-area, facility), portable(obj),
        Sequence(approach(v, fac), AND(stop(v), near (v, fac), NOT(inside(v, loading-area))),
                AND(approach(p, v), carry(p, obj)), AND(stop(p), near (p, v)),
                cause(p, open(portal-of(v))), enter (obj, v),
                cause(p, close(portal-of(v))), leave(v, facility))))
```

Fig. 3 Suspicious Load in Perimeter Security

Minimal Encoding Bias: An ontology should have symbol-independent conceptualizations.

Minimal Ontological Commitment: An ontology should make as few assumptions as possible about the domain being modeled.

Additional issues that are relevant from a computer vision perspective are:

View Invariance: An ontology should not be tuned to one particular camera view, but should generalize across views.

Invariance to Rate of Activity Execution: The same activity performed by different agents may have different temporal spans. The ontology should not make any assumptions about the activity execution rate.

Invariance to Sensor Characteristics: An ontology should be insensitive to variations of sensor characteristics such as resolution, frame-rate, etc.

We now show how to resolve ambiguities according to the design principles outlined above. Let us consider the 'suspicious load' example from the Perimeter Security ontology given in Figure 3. Though this example maintains clarity, further inspection shows that minimal ontological commitment is not preserved. According to the definition the vehicle's portal has to be opened in order to load the object. This does not encompass other possible scenarios. For example, the suspicious load can be placed onto the trailer of a truck which is open from the top, hence not using any portal, or it can even be an explosive that is placed under the body of the vehicle. Moreover, it is not necessary for the vehicle to stop. For instance, somebody inside the vehicle could grab a bag from a suspicious pedestrian through the window. Hence, minimal ontology should only include the object being on the vehicle's exterior, and then being transferred to the vehicle's interior.

As this example illustrates, Gruber's criteria provide a principled method to design and refine ontologies to make them generalizable to new deployments. We refer the reader to [1] for several more detailed examples illustrating the use of Gruber's criteria in designing good ontologies for video surveillance.

4 Computational Models: Integrating Structure and Statistics

Defining semantic equivalence between two sets of interactions amounts to defining equivalence classes on the space of all interactions. The space of interactions can be quite large and doing this individually is not a feasible solution. In addition to a priori knowledge, one may also have access to a limited training set, which in most

cases is not exhaustive. Thus, we can leverage available domain knowledge to design activity models and define semantic equivalence using statistical approaches. The fusion of these disparate approaches – statistical, structural and knowledge based – has not yet been fully realized and has gained attention only in recent years. The two approaches described in this section fall in this category – we exploit domain knowledge to create rich and expressive models for activities and augment them with probabilistic extensions.

For this discussion, we present two computational models – Finite State Automata (FSA) and Petri Nets (PN). At this stage, the choice of the model is driven mostly by the complexity of the activities that need to be represented. If activities of interest are mostly sequential in nature, with only one or two agents involved in the activity, then a FSA may suffice. On the other hand, if activities of interest involve multiple agents performing different actions in parallel and occasionally interacting with each other, then Petri Nets may be the model of choice, since it allows modeling more complex behavior such as parallelism and synchrony. Once we choose a model for a particular domain, it is fundamental to equip this model with the ability to handle (a) 'noise' in the labeling of video data, (b) inaccuracies in vision algorithms and (c) variations from a hard-coded activity model. We illustrate how this can be achieved using FSA and PN as examples. However, it is important to note that the same design principles can be extended to other activity models as well.

In the following, we will often refer to the output of the image processing library with the term 'observations', or 'observation table', implicitly assuming that video data was processed offline and the output stored in a database, which might have been eventually indexed (see Section 4.1.2).

4.1 Probabilistic Automata

There is a large body of work in the AI community on plan and activity recognition, a large portion of which relies on Hidden Markov Models (HMMs) and their variants. Luhr at al. [16] use Hierarchical HMMs to learn the probabilistic nature of simple sequences of activities. Duong at al. [9] introduce the Switching Hidden Semi-Markov Model (S-HSMM), a two-layered extension of the Hidden Semi-Markov Model (HSMM). The bottom layer represents atomic activities using HSMMs, while the top layer represents a sequence of high-level activities, defined in terms of atomic activities. [17] uses non-stationary HSMMs to model the dependency of transition probabilities on the duration an agent has spent in a given state. Dynamic Bayesian networks have also been used for tracking and recognizing multi-agent activities [14]. The CFG-based representation of human activities and interactions [20] enables to formally define complex activities based on simple actions. The problem of recognizing multiple interleaved activities has been studied in [5], where the authors propose a symbolic plan recognition approach, relying on a hierarchical plan-based representation.

Fig. 4 Frames from the Bank dataset depicting (a) a staged robbery; (b) regular bank operations.

Example 1. Consider the two sequences of frames in Figures 4 depicting a staged robbery and regular bank operations respectively. Suppose that the following five activities may occur in a bank setting, and interleaving of activities is possible.

- (a_1) Regular customer-employee interaction, similar to the example in Figure 4b.
- (a_2) Outsider enters the safe.
- (a_3) A bank robbery attempt – the suspected assailant does not make a getaway.
- (a_4) A successful bank robbery, similar to the example in Figure 4a.
- (a_5) An employee accessing the safe on behalf of a customer.

We are interested in monitoring and detecting all of these activities concurrently.

In this section, we begin with the stochastic automaton activity model for video described in [3], and extend it in order to capture multiple activities in a single labeled stochastic automaton. We then define an index to index large numbers of observations from interleaved activities and efficiently retrieve completed instances of these activities. The stochastic activity model and the associated index presented in this section enable activity recognition not only from video data labeling, but also from any type of time-stamped data. In order to make this point clear, we will present examples where this approach has been applied to log data.

Definition 1 (Stochastic activity). A *stochastic activity* is a labeled graph (V, E, δ) where: V is a finite set of action symbols; E is a subset of $(V \times V)$; $\exists v \in V$ s.t. $\nexists v' \in V$ s.t. $(v', v) \in E$, i.e., there exists at least one *start node* in V; $\exists v \in V$ s.t. $\nexists v' \in V$ s.t. $(v, v') \in E$, i.e., there exists at least one *end node* in V; $\delta : E \rightarrow [0, 1]$ is a function that associates a probability distribution with the outgoing edges of each node, i.e., $\forall v \in V \sum_{\{v' \in V | (v,v') \in E\}} \delta(v, v') = 1$.

Example 2. Figure 5 shows the stochastic activity associated with ordering products from an online store. A user first accesses the product catalog (*catalog*) and either inspects the details of an item (*itemDetails*) or continues with a previously saved cart

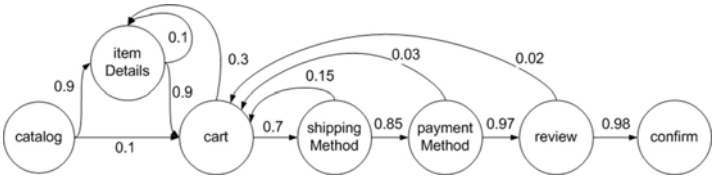


Fig. 5 Online purchase stochastic activity

Table 1 Example of a web log

id	ts	action	id	ts	action	id	ts	action	id	ts	action
1	1	catalog	6	5	itemDetails	11	8	itemDetails	16	11	paymentMethod
2	2	catalog	7	5	shippingMethod	12	9	cart	17	11	review
3	3	itemDetails	8	6	cart	13	9	review	18	12	confirm
4	3	cart	9	7	shippingMethod	14	10	confirm	19	13	paymentMethod
5	4	itemDetails	10	7	paymentMethod	15	11	shippingMethod	20	13	review

(*cart*). The checkout process requires the user to select a shipping method (*shipping-Method*), choose a payment method (*paymentMethod*), review the order (*review*) and confirm it (*confirm*). At each stage during checkout, the user can cancel and return to the cart and from there on to one of the items. The probabilities labeling the edges have the following intuitive meaning: once the *catalog* action has been taken, there is a .9 probability that the user will check details of an item (*itemDetails*) and a .1 probability that she will continue with a previously saved cart (*cart*).

Definition 2 (Activity instance). Let (V,E,δ) be a stochastic activity. An *instance* of (V,E,δ) is a sequence $\langle v_1,\ldots,v_n\rangle$ with $v_i \in V$ such that: (i) v_1 is a start node and v_n is an end node in (V,E,δ); (ii) $\forall i \in [1,n-1]$, $(v_i,v_{i+1}) \in E$. Thus, an activity instance is a path from a start node to an end node in (V,E,δ). The *probability* of the instance is $prob(\langle v_1,\ldots,v_n\rangle) = \Pi_{i\in[1,n-1]}\delta(v_i,v_{i+1})$.

Intuitively, an activity instance is a path from a start node to an end node – the probability of this activity instance is the product of the edge probabilities on the path. We assume that each node in an activity is an observable event. We also assume that the probability of taking an action at any time only depends on the previous event. Thus the probability labeling an edge (v_1,v_2) can be seen as the conditional probability of observing action v_2 given that action v_1 has been observed. These assumptions allow to easily learn the probabilities labeling the edges from training data and to compute the probability of an instance as a product of probabilities.

4.1.1 The Evidence and Identification Problems

We assume that all observations are stored in a single relational database table O. Each row (or tuple) o in the table denotes a single action, $o.action$, which is observed at a given time, $o.ts$.

Definition 3 (Activity occurrence). Let (V,E,δ) be a stochastic activity and let O be an observation table. An occurrence of (V,E,δ) in O with probability p is a

set $\{o_1,\ldots,o_n\} \subseteq O$ s.t. $o_1.ts \leq o_2.ts \ldots \leq o_n.ts$ and $\langle o_1.action,\ldots,o_n.action\rangle$ is an instance of (V,E,δ) with $prob(\langle o_1.action,\ldots,o_n.action\rangle) \geq p$. The *span* of the occurrence is the time interval $span(\{o_1,\ldots,o_n\}) = [o_1.ts, o_n.ts]$.

Intuitively, an activity occurrence is a sequence $\{o_1,\ldots,o_n\}$ of observations in O corresponding to the nodes of an activity instance and the probability of this occurrence is the probability of the instance.

Example 3. The activity in Figure 5 occurs in the server log of Table 1. In fact the sequence of observations with identifiers 1, 4, 7, 10, 13, and 14, corresponds to the instance $\{catalog, cart, shippingMethod, paymentMethod, review, confirm\}$.

Proposition 1. *Given an observation table O and a stochastic activity A, the problem of finding all occurrences of A in O takes exponential time, w.r.t.* $|O|$.

The reason for this behavior is interleaving of activities, which leads to an exponential number of identifiable occurrences of A in O. As an example, the set of observations: $\{catalog, cart, shippingMethod, paymentMethod, review, confirm, confirm\}$ leads to two occurrences of the activity in Figure 5, one for each of the *confirm* actions. Therefore, it is not feasible in practice to try to find all occurrences. Instead, we impose restrictions on what constitutes a valid occurrence in order to greatly reduce the number of possible occurrences. We propose two constraints applicable in most real-world scenarios. In addition, due to the size of the search space, it is important to have data structures that enable very fast searches for activity occurrences. We describe the Multi-Activity Graph Index Creation (MAGIC) [4] that allows to solve the *Evidence* and *Identification* problems efficiently.

Our first restriction requires that if the span of an occurrence O_2 is contained **within** the span of an occurrence O_1 we will discard O_1 from the set of results. This is called the *minimal span restriction* (MS for short).

Definition 4 (MS restriction). Let $O_1 = \{o_1,\ldots,o_k\} \subseteq O$ and $O_2 = \{o'_1,\ldots,o'_l\} \subseteq O$ be two occurrences of the same activity. We say that the span of O_1 is less than or equal to the span of O_2 – $span(O_1) \leq span(O_2)$ – iff $o_1.ts \geq o'_1.ts$ and $o_k.ts \leq o'_l.ts$. Under the *MS restriction*, we only consider occurrences that are minimal w.r.t. span.

The MS restriction may still allow exponentially many occurrences of an activity, since multiple occurrences may have the same span. The *earliest action* (EA for short) restriction requires that when looking for the next action in an activity occurrence, we always choose the first possible successor. For instance, consider the activity definition $v_1 \xrightarrow{1} v_2 \xrightarrow{1} v_3$ and the observation sequence $\{v_1^1, v_2^2, v_1^3, v_2^4, v_3^5\}$ where v_j^i denotes the fact that action v_j was observed at time i. There are two occurrences starting with v_1^1, namely $\{v_1^1, v_2^2, v_3^5\}$ and $\{v_1^1, v_2^4, v_3^5\}$. Under the EA restriction we only consider $\{v_1^1, v_2^2, v_3^5\}$, since v_2^2 is the first possible successor to v_1^1. This restriction makes the search space linear in the size of the observation sequence.

Definition 5 (EA restriction). An activity occurrence $\{o_1,\ldots,o_n\} \subseteq O$ is said to have the *earliest action* property if $\forall i \in [2,n], \nexists w_i \in O$ s.t. $w_i.ts < o_i.ts$ and $\{o_1,\ldots,o_{i-1}, w_i, o_{i+1}, \ldots, o_n\}$ is an occurrence of the same activity.

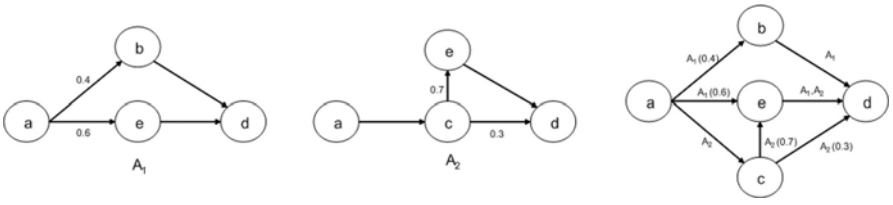


Fig. 6 Stochastic activities and multi-activity graph

It is now straightforward to generalize the *Evidence* and *Identification* problems stated in Section 2.1. In particular, in the Evidence problem we are now interested in recognizing all the *possibly restricted* occurrences of activities in $\mathscr{A}$, from any sequence of time-stamped observations, including video data as a particular case.

4.1.2 Multi-activity Graph Index Creation

In order to concurrently monitor multiple activities, we first merge all activity definitions from $\mathscr{A} = \{A_1, \ldots, A_k\}$ into a single graph.

Definition 6 (Multi-activity graph). Let $\mathscr{A}=\{A_1, \ldots, A_k\}$ be a set of stochastic activities, where $A_i = (V_i, E_i, \delta_i)$. A *Multi-Activity Graph* is a triple $G = (I_{\mathscr{A}}, V_G, \delta_G)$ where: (i) $I_{\mathscr{A}} = \{id(A_1), \ldots, id(A_k)\}$ is a set of identifiers for activities in $\mathscr{A}$; (ii) $V_G = \cup_{i\in[1,k]} V_i$ is a set of action symbols; (iii) $\delta_G : V_G \times V_G \times I_{\mathscr{A}} \rightarrow [0,1]$ is a function that associates a triple $(v, v', id(A_i))$ with $\delta_i((v, v'))$, if $(v, v') \in E_i$ and 0 otherwise.

A multi-activity graph can be graphically represented by labeling nodes with action symbols and edges with id's of activities containing them, along with the corresponding probabilities. The multi-activity graph for a set $\mathscr{A}$ of activities can be computed in time polynomial in the size of $\mathscr{A}$. Figure 6 shows two stochastic activities, A_1 and A_2, and the corresponding multi-activity graph. If $A = (V, E, \delta)$ is an activity and $v \in V$, we use $A.p_{max}(v)$ to denote the maximum product of probabilities on any path in A between v and an end node. The *multi-activity graph index* allows us to efficiently monitor activity occurrences new observations occur.

Definition 7 (Multi-activity graph index). Let $\mathscr{A} = \{A_1, \ldots, A_k\}$ be a set of stochastic activities, where $A_i = (V_i, E_i, \delta_i)$, and let $G = (I_{\mathscr{A}}, V_G, \delta_G)$ be the multi-activity graph built over $\mathscr{A}$. A *Multi-Activity Graph Index* is a 6-tuple $I_G = (G, start_G, end_G, max_G, tables_G, completed_G)$, where:

- $start_G : V_G \rightarrow 2^{I_{\mathscr{A}}}$ is a function that associates each node $v \in V_G$ with the set of activity-id's for which v is a start node;
- $end_G : V_G \rightarrow 2^{I_{\mathscr{A}}}$ is a function that associates each node $v \in V_G$ with the set of activity id's for which v is an end node;
- $\max_G : V_G \times I_{\mathscr{A}} \rightarrow [0,1]$ is a function that associates a pair $(v, id(A_i))$ with $A_i.p_{max}(v)$, if $v \in V_i$ and 0 otherwise;
- For each $v \in V_G$, $tables_G(v)$ is a set of tuples of the form $(current^{\uparrow}, actID, t_0$ is a timestamp, $probability$, $previous^{\uparrow}$, $next^{\uparrow})$, where $current^{\uparrow}$ is a pointer to an

Algorithm 1. *MAGIC-insert*(t_{new}, I_G, p_t)

Require: New observation to be inserted t_{new}, multi-activity graph index I_G, probability threshold p_t
Ensure: Updated multi-activity graph index I_G

```
1: for all tuple t ∈ tables_G(t_new.action) do {Look at start nodes}
2:     if t.activityID ∈ start_G(t_new.action) and t.next = ⊥ then
3:         t.current↑ ← t↑_new
4:     end if
5: end for
6: for all activity id for which no tuple was updated in the previous loop do
7:     add (t↑_new, id, t_new.ts, 1, ⊥, ⊥) to tables_G(t_new.action)
8: end for
9: for all action symbol v ∈ V_G s.t. ∃id ∈ I_A, δ_G(v, t_new.action, id) ≠ 0 do {Look at intermediate nodes}
10:    for all tuple t ∈ tables_G(v) with t.next = ⊥ and maximal t_0 w.r.t. tuples in tables_G(v) with the same actID do
11:        a ← t_new.action, id ← t.activityID
12:        if δ_G(v, a, id) ≠ 0 and t.probability · δ_G(v, a, id) · max_G(a, id) ≥ p_t then
13:            t' ← (t↑_new, id, t.t_0, t.probability · δ_G(v, a, id), t↑, ⊥)
14:            add t' to tables_G(a)
15:            t.next ← t'↑
16:            if id ∈ end_G(a) then {Look at end nodes}
17:                add t'↑ to completed_G(id)
18:            end if
19:        end if
20:    end for
21: end for
```

observation, $actID \in I_{\mathscr{A}}$, t_0, $probability \in \mathbb{R}^+$, $previous^{\uparrow}$ and $next^{\uparrow}$ are pointers to tuples in $tables_G$;

- $completed_G : I_{\mathscr{A}} \rightarrow 2^{\mathscr{P}}$, where $\mathscr{P}$ is the set of tuples in $tables_G$, is a function that associates an activity identifier $id(A)$ with a set of references to tuples in $tables_G$ corresponding to a completed instance of activity A.

Note that $G, start_G, end_G, max_G$ can be computed a-priori. All the tables that are part of the index will be initially empty. As new observations are added, the index tables will be updated accordingly. Therefore, MAGIC can track *partially-completed* activity occurrences.

4.1.3 MAGIC Insertion Algorithm

This section describes an algorithm to update the MAGIC index under the MS and EA restrictions when new observations occur (Algorithm 1). We refer the reader to [4] for a detailed example. Lines 1–8 handle the case when the observation contains an action that is the start node of an activity in $\mathscr{A}$. Tuples in the table associated with the action are updated to minimize span, unless they already have a successor. In this case a new tuple is added, indicating the start of a new concurrent occurrence. Lines 9–15 look at the tables associated with the predecessors of $t_{new}.action$ in the multi-activity graph and check whether the new observation can be linked to existing partial occurrences. For each activity in $\mathscr{A}$, the tuple with the most recent t_0 is linked, to minimize the span. Moreover, in order to impose the EA restriction, the algorithm requires that each tuple has at most one successor. We also enforce the probability threshold by detecting whether the partial occurrence can still have a probability above the threshold on completion. If all these conditions are met, a new tuple t' is added to the table associated with $t_{new}.action$. Finally, lines 16–18 check

whether $t_{new}.action$ is an end node for some activity; in this case, a pointer to t' is added to $completed_G$ signaling that an occurrence has been completed.

Proposition 2. *Algorithm* MAGIC-insert *runs in time* $\mathcal{O}(|\mathscr{A}| \cdot max_{(V,E,\delta)\in\mathscr{A}}(|V|) \cdot |O|)$*, where O is the set of observations indexed so far.*

The *MAGIC-evidence* algorithm finds all minimal sets of observations that validate the occurrence of activities in $\mathscr{A}$ with a probability exceeding a given threshold. The *MAGIC-id* algorithm, which solves the *Identification* problem, identifies the activities in $\mathscr{A}$ that have maximum probability of occurring in the required time span. For reasons of space, we omit a detailed description of the two algorithms and refer the reader to [4] for further details.

4.2 Probabilistic Petri Nets

Petri Nets (PNs) [21] were defined as a mathematical tool for describing relations between conditions and events, and are particularly useful to model and visualize behaviors such as sequencing, concurrency, synchronization and resource sharing. David *et al.* [8] and Murata [19] provide comprehensive surveys on Petri Nets. Several forms of PNs such as Colored PNs, Continuous PNs, Stochastic timed PNs, and Fuzzy PNs have been proposed. Colored PNs associate a *color* to each token, hence are useful to model complex systems where each token can potentially have a distinct identity. In Continuous PNs, the markings of places are *real numbers*, hence they can be used to model situations where the underlying physical processes are continuous in nature. Stochastic timed PNs [18] associate, to each transition, a probability distribution representing the delay between the enabling and firing of the transition. Fuzzy PNs [7] are used to represent fuzzy rules between propositions.

In real-life situations, vision systems have to deal with ambiguities and inaccuracies in the lower-level detection and tracking systems. Moreover, activities may not unfold exactly as described by the model that represents them. The aforementioned types of PNs are not well suited to deal with these situations. The probabilistic PN model [2] described in this section is better suited to express uncertainty in the state of a token or associate a probability to a particular unfolding of the Petri Net.

Definition 8 (Constrained Probabilistic Petri Net). A *Constrained Probabilistic Petri Net* PPN is a 5-tuple $\{P,T,\rightarrow,F,\delta\}$, where

- P and T are finite disjoint sets of places and transitions respectively, i.e., $P \cap T = \emptyset$.
- $\rightarrow$ is the flow relation between places and transitions, i.e., $\rightarrow \subseteq (P \times T) \cup (T \times P)$.
- The preset ${}^{\cdot}x$ of a node $x \in P \cup T$ is the set $\{y | y \rightarrow x\}$.
- The postset $x^{\cdot}$ of a node $x \in P \cup T$ is the set $\{y | x \rightarrow y\}$.
- F is a function defined over the set of places P, that associates each place x with a local probability distribution $f_x(t)$ over $\{t | x \rightarrow t\}$. For each place $x \in P$, we denote by $p^*(x)$ the maximum product of probabilities over all paths from x to a terminal node. We will often abuse notation and use $F(x,t)$ to denote $f_x(t)$.

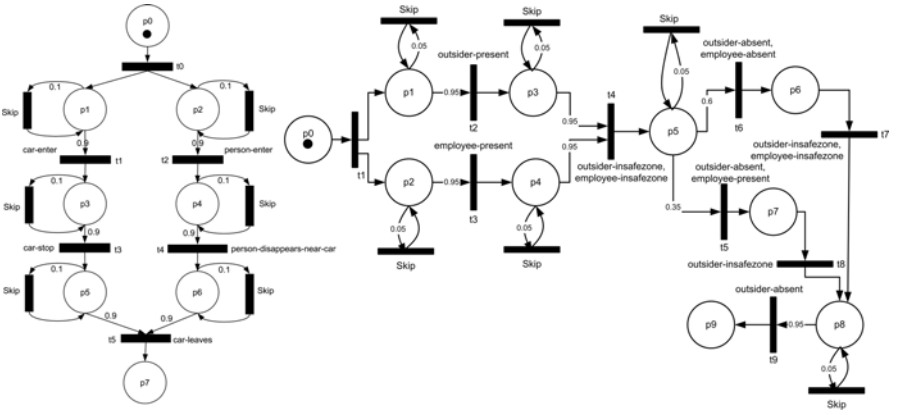
- $\delta : T \rightarrow 2^{\mathscr{A}}$ associates a set of action symbols with every transition in the Petri Net. Transitions will only fire when all the action symbols in the constraint are also encountered in the video sequence labeling.
- $\exists x \in P$ s.t. $x^{\cdot} = \emptyset$, i.e., there exists at least one terminal node in the Petri Net.
- $\exists x \in P$ s.t. ${}^{\cdot}x = \emptyset$, i.e., there exists at least one start node in the Petri Net.

4.2.1 Tokens and Firing of Transitions

Petri Net dynamics are represented via 'markings'. For a PPN $(P, T, \rightarrow, F, \delta)$, a *marking* is a function $\mu : P \rightarrow \mathbb{N}$ that assigns a number of *tokens* to each place in P. In a PPN modeling an activity, a marking μ represents the current state of completion of that activity. The execution of a PN is controlled by its current marking. A transition is *enabled* if and only if all its input places (the preset) have a token. When a transition is enabled, it may *fire*. When a transition fires, all enabling tokens are removed and a token is placed in each of the output places of the transition (the postset). We use μ_0 to denote the *initial marking* of a PPN. A *terminal marking* is reached when one of the terminal nodes contains at least one token. In the simplest case, all the enabled transitions may fire. However, to model more complex scenarios we can impose other conditions to be satisfied before an enabled transition can fire. This set of conditions is represented by δ.

Example 4. Consider the car pickup activity modeled by the PPN in Figure 7.a, with places labeled $p_1, \ldots, p_7$ and transitions $t_0, \ldots, t_5$, p_0 being the start node and p_7 the terminal node. In the initial marking depicted in the figure all places except p_0 have 0 tokens. Transition t_0 is unconstrained and it is always fired, adding a token in both p_1 and p_2. When a car enters the scene, t_1 fires and a token is placed in p_3 and we have a new marking μ_1 such that $\mu_1(p_3) = 1$ and $\mu_1(p) = 0$ for any $p \neq p_3$. The transition t_3 is enabled in this state, but it cannot fire until the condition associated with it is satisfied – i.e., when the car stops. When this occurs, and a person enters the parking lot and then disappears near the car, the Petri Net evolves to a state where there is a token in each of the enabling places of transition t_5. Once the car leaves, t_5 fires and both the tokens are removed and a token placed in the final place p_7. This example illustrates sequencing, concurrency and synchronization.

In the above discussion, we have not yet discussed the probabilities labeling the place-transition edges. Note that the postsets of $p_1, \ldots p_6$ contain multiple transitions. The *skip* transitions are used to *explain away* deviations from the base activity pattern – each such deviation is penalized by a low probability. The probabilities assigned to skip transitions control how tolerant the model is to deviations from the base activity pattern. All tokens are initially assigned a probability of 1. Probabilities are accumulated by multiplying the token and transition probabilities on the edges. When two or more tokens are removed from the net and replaced by a single token then the probability for the new token is set to be the product of the probabilities of the removed tokens. We will use the final probability of a token in a terminal node as the probability that the activity is satisfied.



(a) Car-pickup (b) Bank attack

Fig. 7 Examples of Probabilistic Petri Nets

4.2.2 Activity Recognition

We now define the concepts of *PPN trace* and *activity satisfaction* and provide a method for computing the probability with which the PPN is satisfied.

Definition 9 (PPN trace). A trace for a PPN $(P,T,\rightarrow,F,\delta)$ with initial marking μ_0 is a sequence of transitions $\langle t_1,\ldots,t_k\rangle \subseteq T$ such that:

(i) t_1 is enabled in marking μ_0.

(ii) Firing $t_1,\ldots,t_k$ in that order reaches a terminal marking.

For each $i \in [1,k]$, let $p_i = \Pi_{\{x\in P|x\rightarrow t_i\}} F(x,t_i)$. The trace $\langle t_1,\ldots,t_k\rangle$ has probability $p = \Pi_{i\in[1,k]} p_i$[1]. Let p_{max} be the maximum probability of any trace for $(P,T,\rightarrow,F,\delta)$. Then $\langle t_1,\ldots,t_k\rangle$ has *relative probability* $\frac{p}{p_{max}}$. Intuitively, the relative probability measures how close that trace is to the *ideal* execution of the PPN, i.e., the execution that leads to the maximum possible absolute probability.

Example 5. Consider the PPN in Figure 7b. $\langle t_1,t_2,t_3,t_4,t_6,t_7,t_9\rangle$ is a trace with probability .464 and a relative probability 1.

Definition 10 (Activity satisfaction). Let $\mathscr{P} = (P,T,\rightarrow,F,\delta)$ be a PPN, let v be a video sequence and let ℓ be the labeling of v. We say ℓ satisfies $\mathscr{P}$ with *relative probability* $p \in [0,1]$ iff there exists a trace $\langle t_1,\ldots,t_k\rangle$ for $\mathscr{P}$ such that:

(i) There exist frames $f_1 \leq \ldots \leq f_k$ such that $\forall i \in [1,k]$, $\delta(t_i) \subseteq \ell(f_i)$ AND

(ii) The relative probability of $\langle t_1,\ldots,t_k\rangle$ is equal to p.

We will refer to a video sequence v satisfying an activity definition, as an equivalent way of saying the labeling of v satisfies the activity. If ℓ is the labeling of v and $v' \subseteq v$ is a subsequence of v, we will also use ℓ to refer to the restriction of ℓ to v'.

[1] We are assuming that the initial probability assigned to tokens in the start nodes is 1.

4.2.3 The Evidence and Identification Problems

In this section we present an algorithm for the Evidence Problem stated in Section 2.1, PPN-evidence (Algorithm 2), which simulates the constrained PPN forward. The algorithm uses an ad-hoc data structure to store tuples of the form $\langle \mu, p, f_s \rangle$, where μ is a valid marking, p is the probability of the partial trace that leads to μ and f_s is the frame when the first transition in that trace was fired.

The algorithm starts by adding $\langle \mu_0, 1, 0 \rangle$ to the store S (line 1). This means we start with the initial marking μ_0 and probability equal to 1. The value 0 for the start frame means the first transition has not yet fired. Whenever the first transition fires and the start frame is 0, the current frame is chosen as a start for this candidate subsequence (lines 8–9). The algorithm iterates through the video frames, and at each iteration analyzes the current candidates in S. Any transitions t enabled in the current marking that have the conditions $\delta(t)$ satisfied are fired. The algorithm generates a new marking, and with it a new candidate partial subsequence (line 7). If the new probability p' is still above the threshold (line 15) and can remain above the threshold on the best path to a terminal marking (line 16), the new state is added to the store S. This first pruning does away with any states that will result in low probabilities. Note that we have not yet removed the old state from S, since we also need to fire any enabled skip transition, in order to explore the space of solutions. This is done on line 26. At this point (line 27) we also prune any states in S that have no possibility of remaining above the threshold as we reach a terminal marking.

The following two theorems state correctness and complexity results for the PPN-evidence algorithm. We refer the reader to [2] for proofs of both theorems.

Theorem 1 (PPN-evidence correctness). *Let $\mathscr{P} = (P, T, \rightarrow, F, \delta)$ be a PPN with initial configuration μ_0, let v be a video sequence and ℓ its labeling, and let $p_t \in [0,1]$ be a probability threshold. Then for any subsequence $v' \subseteq v$ that satisfies $\mathscr{P}$ with probability greater than or equal to p_t, one of the following holds:*

(i) v' is returned by PPN-evidence OR

(ii) $\exists v'' \subseteq v$ that is returned by PPN-evidence such that $v'' \subseteq v'$.

Theorem 2 (PPN-evidence complexity). *Let $\mathscr{P} = (P, T, \rightarrow, F, \delta)$ be a PPN with initial configuration μ_0, such that the number of tokens in the network at any marking is bounded by a constant k. Let v be a video sequence and ℓ its labeling. Then PPN-evidence runs in time $\mathscr{O}(|v| \cdot |T| \cdot |P|^k)$.*

We now briefly describe an algorithm for the Identification Problem. Listing of the algorithm, detailed description and correctness theorem are omitted for reasons of space. We refer the reader to [4] for further details.

Let $\{\mathscr{P}_1, \ldots, \mathscr{P}_n\}$ be a given set of activity definitions. If we assume that there is only one activity $\mathscr{P}_l$ which satisfies the video sequence v with maximal probability, a simple binary search iterative method – naivePPN-ident – can employ PPN-evidence to find the answer: naivePPN-ident runs PPN-evidence for all activities in $\{\mathscr{P}_1, \ldots, \mathscr{P}_n\}$ and for different thresholds until a delimiter threshold is found (i.e., one for which only one activity $\mathscr{P}_l$ has a non-empty result from PPN-evidence).

Algorithm 2. PPN-evidence($\mathscr{P}$, μ_0, ℓ, p_t)

Require: $\mathscr{P} = (P, T, \rightarrow, F, \delta)$, initial configuration μ_0, video sequence labeling ℓ, probability threshold p_t
Ensure: Set of minimal subsequences that satisfy $\mathscr{P}$ with relative probability above p_t

$S \leftarrow \{\langle \mu_0, 1, 0\rangle\}$
$R \leftarrow \emptyset$
$max_p \leftarrow$ the maximum probability for any trace of $\mathscr{P}$
for all f frame in video sequence **do**
  for all $\langle \mu, p, f_s\rangle \in S$ **do**
    for all $t \in T$ enabled in μ s.t. $\delta(t) \subseteq \ell(f)$ **do**
      $\mu' \leftarrow$ fire transition t for marking μ
      if $f_s = 0$ and $\delta(t) \neq \emptyset$ **then**
        $f' \leftarrow f$
      else
        $f' \leftarrow f_s$
      end if
      $p^* \leftarrow \Pi_{\{x \in P | x \rightarrow t\}} F(x,t)$
      $p' \leftarrow p \cdot p^*$
      if $p' \geq p_t \cdot max_p$ **then**
        if $\exists x \in P$ s.t. $\mu(x) < \mu'(x) \wedge p' \cdot p^*(x) \geq p_t \cdot max_p$ **then**
          $S \leftarrow S \cup \{\langle \mu', p', f'\rangle\}$
        end if
      end if
      if μ' is a terminal configuration **then**
        $S \leftarrow S - \{\langle \mu', p', f'\rangle\}$
        $R \leftarrow R \cup \{[f_s, f]\}$
      end if
    end for
    for all skip transition $t \in T$ enabled **do**
      Fire skip transition if no other transitions fired and update $\langle \mu, p, f_s\rangle$
      Prune $\langle \mu, p, f_s\rangle \in S$ s.t. $\forall x \in P$ s.t. $\mu(x) > 0$, $p \cdot p^*(x) < p_t \cdot max_p$
    end for
  end for
end for
Eliminate non-minimal subsequences in R
return R

A more efficient solution is PPN-ident which uses a similar storage structure as PPN-evidence (tuples of the form $\langle \mu, p, \mathscr{P}\rangle$, where $\mathscr{P}$ is the activity definition to which marking μ applies). The algorithm maintains a global maximum relative probability max_p with which the video satisfies any of the activities in $\{\mathscr{P}_1, \ldots, \mathscr{P}_n\}$ and a list R of the activity definitions that have been satisfied with probability max_p. max_p is updated any time a better relative probability is found.

Theorem 3 (PPN-ident complexity). *Let* $\{(P_i, T_i, \rightarrow_i, F_i, \delta_i)\}_{i \in [1,m]}$ *be a set of PPNs, let v be a video sequence. PPN-ident runs in time* $\mathscr{O}(m \cdot |v| \cdot |T| \cdot |P|^k)$.

5 Language Based Paradigm: PADL

The task of modeling complex activities using the models discussed in the previous section may be challenging when the models need to be created manually by users. Although a lot of work has been done on learning model parameters from training data, learning the whole model from training data is still an open issue. Therefore, we also explore the definition of logical languages to define complex activities in an expressive yet formal fashion. Thus, we introduce PADL (Probabilistic Activity Detection Language), an extensible logical language including a set of boolean and

probabilistic predicate symbols which map to primitive actions that can be recognized through image processing algorithms. Along with PADL, we propose a suite of offline and real-time algorithms that solve the Evidence and Online Identification problems stated in Section 2.1.

5.1 Syntax of PADL

PADL is a *logical language* which has a set of constant, function and variable symbols, a set of boolean predicate symbols, and, unlike ordinary logic, a set of probabilistic predicate symbols. PADL builds on top of a library of image processing algorithms. Each algorithm implements either a boolean or a probabilistic predicate. PADL can be extended by extending the underlying image processing library.

Definition 11 (atom). If p is a predicate symbol with k arguments, and $t_1,\ldots,t_k$ are either variables or constants of the right types, then $p(t_1,\ldots,t_k)$ is an *atom*. $p(t_1,\ldots,t_k)$ is an *uninstantiated atom* if at least one of the t_i's is a variable.

Note that the definition of an atom does not distinguish between boolean and probabilistic predicates. We will address the issue in Section 5.2 when discussing probabilistic activity satisfaction. We now define the set of well-formed formulas in PADL. The degree of a formula is the number of alternating $\forall$ and $\exists$ quantifiers.

Definition 12 (activity formula). An activity formula is defined as follows:

1. Every atom is an activity formula of degree 0.
2. If F is a formula of degree 0 and x is a variable, then $(\forall x)F$ and $(\exists x)F$ are activity formulas of degree 1.
3. If F, G are activity formulas of degree m,n respectively, then the conjunction $(F \wedge_{\otimes} G)$ and the disjunction $(F \vee G)$ are activity formulas of degree $\max(m,n)$. By $(F \wedge_{\otimes} G)$ we denote that, if F and/or G are probabilistic, the probability of their conjunction can be obtained using the t-norm $\otimes$[2]. We will assume that the default t-norm is independence ($x \otimes y = xy$), unless otherwise specified.
4. If F is an activity formula of degree k and x is a variable that is not within the scope of any quantifier in F, then:
   a. If F has the form $(\forall\ldots)G$ then $(\forall x)F$ is an activity formula of degree k.
   b. If F has the form $(\exists\ldots)G$ then $(\forall x)F$ is an activity formula of degree $k+1$.
   c. If F has the form $(\forall\ldots)G$ then $(\exists x)F$ is an activity formula of degree $k+1$.
   d. If F has the form $(\exists\ldots)G$ then $(\exists x)F$ is an activity formula of degree k.

A well known result in logic [22] states that, as the degree of an activity formula becomes higher, the problem of finding occurrences of the activity becomes more complex. Moreover, if two formulas have the same form, but a different leading quantifier, $(\exists)$ and $(\forall)$ respectively, then the former is easier to solve than the latter.

[2] We omit the notations on disjunctions, as they can be expressed by conjunction and negation.

5.2 Probabilistic Activity Satisfaction

As mentioned earlier, PADL assumes the existence of a set of image processing algorithms By applying these algorithms to each frame in a video, we obtain a *labeling* of the video.

Definition 13 (Frame labeling). Suppose $\mathscr{O}$ is a universe of distinct reference image objects (e.g. a database of mugshots or a set of car images). A *labeling* is a pair (ℓ, pl) where ℓ is a set of ground boolean atoms and pl is a function which assigns values in $[0,1]$ to each probabilistic ground atom, such that for all objects o of interest in a given frame $\sum_{o' \in O} \mathsf{pl}(\mathsf{eq}(o,o')) = 1$[3].

We will abuse notation and use (ℓ, pl) to denote the labeling of the entire video sequence or any subsequence of it. Once we obtain the video labeling (ℓ, pl), we need to compute the probability that it satisfies an activity formula. Intuitively, a formula is satisfied if we can substitute objects from the labeling for the variables in the formula. A *substitution* θ is any set $\theta = \{X_1 = v_1, \ldots, X_n = v_n\}$ where the X_i's are variables and the v_i's are constants. Given an activity formula F, we use $F\theta$ to denote the replacement of all occurrences of X_i in F by v_i. If θ_1, θ_2 are substitutions, $(\theta_1 \cup \theta_2)$ is solvable if and only if the substitutions are consistent.

Given a labeling (ℓ, pl) of a video and an activity formula F, we now define the concept of a substitution set $\delta(F)$ for F. Intuitively, $\delta(F)$ contains all pairs (θ, p) where θ is a substitution that binds existentially quantified variables in F to objects or frame numbers and p is the probability that F is satisfied by the labeling when applying substitution θ. These informal notions are formalized below.

Definition 14 (Substitution sets for activity formulas). Let F be an activity formula, v a video sequence, and (ℓ, pl) a labeling of v. The *substitution set* for F, denoted $\delta(F)$, is defined as follows:

(i) If F is a boolean atom, then $\delta(F) = \{(\theta, 1) \mid F\theta \in \ell(v)\}$.
(ii) If F is a probabilistic atom, then $\delta(F) = \{(\theta, \mathsf{pl}(F\theta)) \mid (F\theta, \mathsf{pl}(F\theta)) \in \mathsf{pl}(v)\}$.
(iii) If $F = (\exists\, x)G$, then $\delta(F) = \delta(G)$.
(iv) If $F = (\forall\, x)G$, $\delta(F) = \delta(\bigwedge_{o \in \mathscr{O}_x} G([x/o]))$, where $\mathscr{O}_x$ is the set of possible values for x.
(v) If $F = G \wedge_{\otimes} H$, then $\delta(F) = \{(\theta_1 \cup \theta_2, v_1 \otimes v_2) \mid (\theta_1, v_1) \in \delta(F) \wedge (\theta_2, v_2) \in \delta(G) \wedge (\theta_1 \cup \theta_2)$ is solvable $\}$.
(vi) If $F = \neg G$, let Θ_G contain all substitutions for variables in G. Then $\delta(F) = \{(\theta, p) \mid (\theta, 1-p) \in \delta(G)\} \cup \{(\theta, 1) \mid \nexists (\sigma, v) \in \delta(G)$ s.t. $\sigma \cup \theta$ is solvable $\}$
(vii) If $F = G \vee H$, then $\delta(F) = \delta(G) \cup \delta(H)$.

Example 6. Consider the frame sequence in Figure. 8.a. We assume that the set of reference objects consists of *John Doe*, *Joe Doe* – the two protagonists – and the object *Bag*. The boolean labeling ℓ and the probabilistic labeling pl might

[3] For each object o in the video, $\mathsf{pl}(\mathsf{eq}(o,o'))$ is the probability that o is the same as some $o' \in \mathscr{O}$.

Fig. 8 Frames depicting (a) a package transfer from the ITEA CANDELA dataset; (b) tarmac operations from the TSA dataset.

be: $\ell = \{\text{in}(p_1, f_1), \text{haspkg}(p_1, pkg_1, f_1), \text{in}(p_1, f_2), \text{in}(p_2, f_2), \text{haspkg}(p_1, pkg_1, f_2), \text{in}(p'_1, f_3), \ldots\}$ $\mathsf{pl} = \{(\mathsf{eq}(p_1, p'_1), 0.91), (\mathsf{eq}(p_2, p'_2), 0.95), (\mathsf{eq}(p_1, p_2), 0.07), (\mathsf{eq}(p_1, \textit{John Doe}), 0.94), \ldots\}$

Now consider the activity formula $F = \text{in}(P_1, F_1) \wedge \text{in}(P_2, F_2) \wedge F_1 < F_2 \wedge eq(P_1, P_2)$, which requires the same person appear in two different frames F_1 and F_2. The substitution set for F is $\delta(F) = \{(\{P_1 \rightarrow p_1, P_2 \rightarrow p_1, F_1 \rightarrow f_1, F_2 \rightarrow f_2\}, 1), (\{P_1 \rightarrow p_1, P_2 \rightarrow p'_1, F_1 \rightarrow f_2, F_2 \rightarrow f_3\}, 0.91), \ldots\}$.

We now formally define what constitutes an answer to the *Evidence* problem stated in Section 2.1, we then present the OffPad algorithm that solves the Evidence problem, and the OnPad algorithm that solves the Online Identification problem.

Definition 15 (answer). Suppose (ℓ, pl) is the labeling of video v, F is an activity formula, and p is a real number in the $[0,1]$ interval. A sub-sequence sv of v is said to *minimally satisfy F with probability p* iff

i There exists a pair $(\theta, q) \in \delta(F)$ such that $sv = [start(F), end(F)]$ and $q \geq p$[4].

ii There is no strict sub-sequence sv' of sv which satisfies the above condition.

5.3 The OffPad Algorithm

The OffPad algorithm (Algorithm 3) uses the *answer_af* method (Algorithm 4) to compute the substitution set $\delta(F)$ for the activity formula F, whereas the main body of the algorithm uses the substitutions returned by *answer_af* to compute the minimal contiguous sub-sequences of the video that satisfy the given activity definition.

[4] We assume that the user has explicitly denoted some variables in F as *frame* variables, and marked two frame variables as a start variable $start(F)$ and an end variable $end(F)$ respectively.

A brief outline of the algorithm reads as follows:

1. Compute the valid substitutions $\delta(F)$ by recursively decomposing F into parts.
2. Select those substitutions that have probability over the threshold.
3. For each substitution, find the sequence *sv* between start and end frame variables.
4. If a subsequence of *sv* is already in the result, continue to step 6.
5. Otherwise, add *sv* to the result.
6. Repeat steps 3–5.

The *answer_af* algorithm uses a recursive approach – based on Definition 14 – to compute the subset of $\delta(F)$ which contains only those elements with probabilities above the threshold p_t (we denote this subset by $\delta(F)|_{p_t}$). The algorithm optimizes the search for substitutions in three ways. The first is based on the observation that a conjunction of multiple formulas allows us to combine the intermediate results for each formula in any order we may choose; the *heuristic_order* algorithm (omitted for reasons of space) is used to give an ordering that is computationally optimal.

The second optimization uses the fact that we can eliminate those intermediate results that cannot possibly lead to solutions with probabilities higher than the threshold. The *pruning* method removes substitutions that would yield results under the given threshold. Consider the case of computing the substitution set for $F \wedge G$ with probability threshold p_t. A substitution $(\theta, p) \in \delta(F)$ can only be combined with substitutions in $\{(\theta', p') \in \delta(G) | p' \geq min_{\otimes}(p, p_t)\}$. If the cardinality of this set is small, the number of operations in computing $\delta(F \wedge G)$ is greatly reduced.

The third optimization separates the set of frame comparison predicates (e.g., $t_1 < t_2$) from the rest (lines 1, 13). The number of frames is much larger than the number of objects in a video, hence frame variables have a huge number of possible substitutions. We delay searching for substitutions for frame variables until the end of the method because pruning may make this costly operation unnecessary.

Algorithm 3. OffPad(F, v, p_t, V_s, V_e))

Require: Activity formula F, video v, probability threshold p_t, variables V_s,V_e denoting start and end frames of result.
Ensure: Set of video subsequences satisfying F with probability above p_t

```
1:  R ← answer_af(F, v, p_t, pruning)
2:  R' ← ∅
3:  for all (θ, ℘) ∈ R do
4:      should_add ← true
5:      ⟨f_s, f_e⟩ ← ⟨V_s θ, V_e θ⟩
6:      for all [f, f'] ∈ R' do
7:          if [f, f'] ⊇ [f_s, f_e] then
8:              R' ← R' − {[f, f']}
9:          else if [f, f'] ⊆ [f_s, f_e] then
10:             should_add ← false
11:         end if
12:     end for
13:     if should_add then
14:         R' ← R' ∪ {[f_s, f_e]}
15:     end if
16: end for
17: S ← ∅
18: for all [f_s, f_e] ∈ R' do
19:     S ← S ∪ {subvideo(v, f_s, f_e)}
20: end for
21: return S
```

Algorithm 4. *answer_af*$(F, v, p_t, pruning)$

Require: Activity formula F, video v, probability threshold p_t, pruning method *pruning*. If M is a substitution set and F_A a set of boolean atoms, we denote by $M|^{F_A} = \{(\theta, p) \in M | \forall\ A\ \in F_A, A\theta$ *is true*$\}$. *pruning_method*(M_1, M_2, p_t) returns a pair of substitution sets $\langle M_1', M_2' \rangle$ such that $M_1' \subseteq M_1$ and $M_2' \subseteq M_2$ and $(M_1' \wedge M_2')|_{p_t} = (M_1 \wedge M_2)|_{p_t}$.
Ensure: $\delta(F)|_{p_t}$.

```
1: F_C ← ∅;
2: F_A ← {a atom in F | a involves frame variable comparisons}
3: if F is an atom then
4:     return δ(F)|_{p_t};
5: else if F is of type (∃ x)G then
6:     return answer_af(G, v, p_t, pruning)
7: else if F is of type (∀ x)G then
8:     for all o ∈ 𝒪_x do {𝒪_x is the set of all possible values of x}
9:         F_C ← F_C ∪ {G([x/o])}
10:    end for
11:    F_C ← F_C − F_A
12: else if F = G_1 ∧ ··· ∧ G_n then
13:    F_C ← {G_1, ···, G_n} − F_A
14: else if F = ¬G then
15:    M ← answer_af(G, v, 0, pruning)
16:    M' ← {(θ, p) | (θ, 1 − p) ∈ M} ∪ {(θ, 1) | ∄(σ, v) ∈ M s.t. θ ∪ σ solvable }
17:    return M'|_{p_t}
18: else if F = G ∨ H then
19:    return answer_af(G, v, p_t, pruning) ∪ answer_af(H, v, p_t, pruning)
20: end if
21: p ← p_t
22: for all F_i ∈ F_C do
23:    M_i ← answer_af(F_i, v, p, pruning)
24:    p ← min_⊗(max({p' | (θ, p') ∈ M_i}), p)
25: end for
26: T ← heuristic_order({M_1, ···, M_i, ···}, p_t)
27: while |T| > 1 do
28:    for all nodes M_i, M_j with the same parent do
29:        ⟨M_i', M_j'⟩ ← pruning(M_i, M_j, p_t)
30:        parent(M_i, M_j) ← (M_i' ∧ M_j')|_{p_t}
31:        if parent(M_i, M_j) = ∅ then
32:            return ∅
33:        end if
34:        remove M_i, M_j from T
35:    end for
36: end while
37: return root(T)|^{F_A}_{p_t}
```

In the case of formulas of type $\neg G$ (lines 14–17), the algorithm computes the entire set of possible substitutions for the variables in G; this is because the answer to $\neg G$ contains – for boolean predicates – all substitutions that *are not* in $\delta(G)$ which is very expensive. The OffPad algorithm takes the substitution set returned by *answer_af*, along with the start and end frame variables and, for each substitution returned, determines the corresponding video sub-sequence. In order to ensure the minimality condition, such sub-sequences are retained if and only if they have no sub-sequence that satisfies the activity definition with a probability above the threshold. The following theorems state correctness and complexity results for *answer_af* and OffPad. Proofs are omitted for reasons of space.

Lemma 1. *Let F be an activity formula, v be a video sequence and $p_t \in [0,1]$ be a probability threshold. Method answer_af$(F, v, p_t, -, -)$ returns $\{(\theta, p) \mid p \geq p_t\}$ independently of the pruning and order methods.*

Theorem 4 (OffPad correctness). *Let F be an activity formula, v be a video sequence and $p_t \in [0,1]$ be a probability threshold. Let F_s, F_e be two frame variables that appear in F. Let S be the set of video sub-sequences returned by OffPad. Then $\forall\, sv \in S$, ($\nexists\, sv' \in S$ s.t. sv' is a subsequence of sv) $\wedge$ ($\exists\, (\theta, p) \in \delta(F)$ s.t. $p \geq p_t$, $F_s\theta = sframe(sv,v)$ and $F_e\theta = eframe(sv,v)$)*[5].

Theorem 5 (OffPad complexity). *Let F be an activity formula, let v be a video sequence and let $p_t \in [0,1]$ be a probability threshold. Let $|F|$ be the number of atoms in F and let O be the set of reference objects. Let (ℓ, pl) be the labeling for the video v; we denote by $|\mathsf{pl}|$ the size of the domain of pl. Then OffPad is running in time $\mathcal{O}(max(|O|, |\ell|, |\mathsf{pl}|)^{2\cdot|F|})$.*

5.4 The OnPad Algorithm

While OffPad is very effective for labeled videos, its recursive search for substitutions cannot be performed unless the full labeling is available when the algorithm is executed. However, in many security surveillance scenarios, activities must be detected as the video is being captured. We now present a real-time algorithm that solves the *Online Identification* problem stated in Section 2.1. OnPad (Algorithm 5) is based on representing formulas and their substitution sets as trees that are incrementally expanded and updated as new labeling information becomes available. We start by defining the tree representation for activity formulas.

Definition 16 (activity formula tree). An activity formula tree is a tuple $T = \langle N, l, v, var, \rightarrow \rangle$, where:

(i) N is the set of nodes.

(ii) $l : N \rightarrow \{A, \neg, \forall, \wedge, \vee\}$ is a function that assigns a label to each node.

(iii) $v : N \rightarrow \mathscr{M}$ is a function that assigns a substitution set to each node.

(iv) $var : \{n \in N | l(N) = \forall\} \rightarrow \mathscr{V}$ is a function that assigns a variable name to all nodes that are labeled with $\forall$.

(v) $\rightarrow$ is a binary relation on N such that $(N, \rightarrow)$ is a tree and $n \in N$ is a leaf node iff $l(n) = A$. The transitive closure of $\rightarrow$ will be denoted by $\rightarrow^*$.

For $n, n' \in N$ such that $n \rightarrow^* n'$, we use $Path(n, n')$ to denote the set of nodes situated on the path from n to n', excluding n, n'. Let $d(n, n') = |Path(n, n')|$. We can easily see that for any activity formula we can construct an activity formula tree in which every leaf node is an atom in the formula; conversely, for any activity formula tree we can easily compute the associated activity formula. We now define an ordering relationship between activity formula trees, which models how an activity formula tree can be expanded when new frames are available for processing.

Definition 17 (activity formula tree ordering). Let $T_1 = \langle N_1, l_1, v_1, var_1 \rightarrow_1 \rangle$, $T_2 = \langle N_2, l_2, v_2, var_2, \rightarrow_2 \rangle$ be two activity formula trees. We write $T_1 \sqsubseteq T_2$ iff there is an injective function $f : N_1 \rightarrow N_2$ such that:

[5] $sframe(sv,v)$ and $eframe(sv,v)$ denote the start and end frames of sv in v.

(i) Paths are preserved, i.e., $\forall\, n_1, n_1' \in N_1$ s.t. $n_1 \rightarrow_1 n_1'$, $f(n_1) \rightarrow_2^* f(n_1')$.

(ii) The substitution set for a leaf node in T_1 is a subset of the substitution set for the corresponding node in T_2, i.e., $\forall\, n_1 \in N_1$ s.t. $l_1(n_1) = A$, $v(n_1) \subseteq v(f(n_1))$.

Intuitively, $T_1 \sqsubseteq T_2$ if T_2 results from modifying T_1 when new labeling elements are added. This suggests that the OnPad algorithm will produce a order-preserving sequence of activity formula trees, one at each step of the algorithm.

Activity formula trees provide an ideal way of representing the current state of OnPad and are updated with new frames and labeling elements. However, the activity formula trees do not provide a direct way of estimating the probability that an activity is about to occur. To provide useful feedback to the user, OnPad returns a numeric *alert level* between 0 and 1 representing an indirect measure of the probability that an activity is about to occur.

Formally, an alert function takes an activity formula tree $T = \langle N, l, v, var, \rightarrow \rangle$ and, based on its structure and/or contents, returns a real number $\mathscr{L} \in [0,1]$, such that:

(i) $\mathscr{L} = 1 \Leftrightarrow v(root((T)) \neq \emptyset$;

(ii) $\forall n \in T, v(n) = \emptyset \Rightarrow \mathscr{L} = 0$.

Example 7. Let $T = \langle N, l, v, var, \rightarrow \rangle$ be an activity formula tree and $S = \{n \in N \mid v(n) \neq \emptyset\}$ be the set of non empty nodes in the tree. The following is an example of *alert level* function, but many others functions are possible. One sanity check is that an alert function returns 1 iff the activity has been fully detected.

$$f_p(T) = \begin{cases} 1, \text{ if } v(root(T)) \neq \emptyset \\ \frac{|S|}{|N|} \text{ otherwise} \end{cases} \tag{1}$$

OnPad takes as input the current state represented by an activity formula tree and the latest frame in the video sequence and returns a level of alert that measures the probability that the activity will occur in the near future. We assume the activity formula has been pre-parsed into an activity formula tree T, which is initialized so that for each node $n \in N$, $v(n) \leftarrow \emptyset$. The algorithm starts by computing the labeling for the frame currently being processed on line 1, and then recomputing the set of all possible substitutions by instantiating variables to the new objects in frame fr. In practice, we have noticed that these operations can often be avoided as frames generally resemble prior frames in a video sequence. Lines 5–12 may add new subtrees to $\forall$-labeled nodes; the subtrees correspond to instances introduced in fr of the same type as the universally-quantified variable. Finally, the algorithm computes the new substitution sets in a bottom-up fashion – the nodes furthest from the root are computed first and then changes are propagated upwards. When this process is completed, T represents the new state of the algorithm and is returned together with the alert level. The following theorems state correctness and complexity results for OnPad. Proofs are omitted for reasons of space.

Theorem 6 (OnPad correctness). *Let F be an event description, let v be a video sequence consisting of frames $[0, fr]$ and let $p_t \in [0,1]$ be a probability threshold. Let $\langle T, \delta \rangle$ be the result of applying OnPad to the frames of v until frame $fr - 1$. Then $root(T) = \delta(F)|_{p_t}$.*

Algorithm 5. OnPad(F, T_s, fr, ℓ, pl, *alert*))

Require: Activity formula F, activity formula tree $T_s = \langle N_s, l_s, v_s, var_s, \rightarrow_s \rangle$ for F representing the current state and the current frame fr. (ℓ, pl) represents the labeling of the video up to the current frame. *alert* is an instance of *alert_level*. We assume the existence of two methods: $toAFTree(F)$, which returns an activity formula tree from an activity formula F and $fromAFTree(T)$ that returns the activity formula corresponding to activity formula tree T.

Ensure: Pair $\langle T, \delta \rangle$, where $\delta \in [0,1]$ representing the alert level and T represents the new state of the incremental algorithm.

1: $(\ell', \mathsf{pl}') \leftarrow$ (**compute** labeling for frame fr)
2: $\mathscr{O}' \leftarrow$ the set of new objects in fr
3: $\Theta \leftarrow$ **recompute** the set of all possible substitutions from $\mathscr{O}'$
4: $T(\langle N, l, v, var, \rightarrow \rangle) \leftarrow T_s$
5: **for all** $n \in T$ s.t. $l(n) = \forall$ **do**
6: $T' \leftarrow$ the subtree of T rooted at n
7: $F \leftarrow fromAFTree(T')$
8: **for all** $o \in \mathscr{O}$ s.t. o has the same type with $var(n)$ **do**
9: $T' \leftarrow toAFTree(F[var(n)/o])$
10: **add** T' as a child of n
11: **end for**
12: **end for**
13: $S \leftarrow \{n \in N | l(n) = A\}$
14: $h \leftarrow max_{n \in S}(d(n, root(T))$
15: $w \leftarrow 0$
16: **while** $w \leq h$ **do**
17: **for all** $n \in N$ s.t. $d(n, root(T)) = h - w$ **do**
18: **if** $l(n) = A$ **then**
19: $\gamma \leftarrow$ **compute** new substitutions for fromAFTree(n) from (ℓ', pl')
20: $v(n) \leftarrow v(n) \cup \gamma$
21: **else if** $l(n) = \wedge$ or $l(n) = \forall$ **then**
22: $v(n) \leftarrow \bigwedge_{m \in \{n' \in N | n' \rightarrow n\}} v(m)$
23: **else if** $l(n) = \neg$ **then**
24: $n' \leftarrow n' \in N$ *s.t.* $n' \rightarrow n$
25: $v(n) \leftarrow \{(\theta, 1-p) | (\theta, p) \in v(n')\} \cup \{(\theta, 1) |$ $\nexists (\sigma, x) \in v(n')$ s.t. $\theta \cup \sigma$ solvable $\}$
26: **else if** $l(n) = \vee$ **then**
27: $v(n) \leftarrow \bigcup_{m \in \{n' \in N | n' \rightarrow n\}} v(m)$
28: **end if**
29: **end for**
30: $w \leftarrow w + 1$
31: **end while**
32: **return** $\langle T, alert(T) \rangle$

Theorem 7 (OnPad complexity). *Let F be an activity formula, let v be a video sequence $[0, fr-1]$, let fr be the current frame and let (ℓ, pl) be the labeling for $[0, fr]$. We denote by O the set of possible objects in the video. Then OnPad is running in time $\mathscr{O}(max(|O|, |\ell|, |\mathsf{pl}|)^{|F|})$.*

6 Experimental Evaluation

In this section, we report the most relevant results of the experimental evaluation of the algorithms proposed in this chapter and refer the reader to [2, 4] for more details.

We used two publicly available datasets – the ITEA CANDELA dataset, a bank surveillance dataset – and a third dataset containing TSA airport tarmac footage. In addition, we evaluated MAGIC on (i) a synthetic dataset of 5 million observations; (ii) a third party dataset consisting of travel information such as hotel reservations, passport and flight information, containing approximately 7.5 million observations.

The **ITEA CANDELA** dataset (http://www.multitel.be/∼va/) consists of 16 videos, about 1 minute in length, depicting package exchanges or people picking up and dropping off packages. We defined 10 activities of increasing complexity and designed the corresponding activity formulas. The **TSA** dataset consists of approximately 118 minutes of tarmac footage. We used a set of 23 activity definition including flight take-off and landing, baggage handling and other maintenance operations. The **Bank** dataset [23] consists of 7 videos, 15–30 seconds in length, depicting staged bank attacks and daily bank operations. Figures 4(a) and 4(b) contain frames from video sequences depicting a staged bank attack and regular bank operations respectively. For the Bank dataset, we used the 5 activities of Example 1 and designed the associated Stochastic Automata, PPNs and activity formulas.

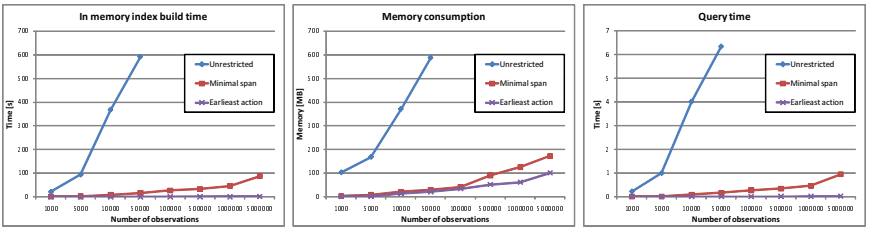
We compared the precision and recall of our algorithms against the ground truth provided by human reviewers as follows. Human reviewers received detailed explanations on the activity models, as well as sets of activity definitions and were asked to mark the starting and ending frame of each activity they encountered in the videos. An average over the reviewers was then used as the ground truth. In order to evaluate OnPad, for each activity definition, at uniformly sampled time points throughout the video, reviewers were asked to provide a number between 0 and 1 representing the likelihood that the activity was about to complete. We considered an average of the alert levels over the reviewers as the ground truth.

6.1 *MAGIC*

Figures 9.a and 9.b respectively show the time and memory taken to build the index for the synthetic dataset, w.r.t. to number of observations. As expected, the exponential nature of the unrestricted index makes the problem impractical for more than 50,000 observations. Instead, the proposed restrictions yield significantly better results. We also evaluated the average query time on the synthetic dataset. We generated multiple *evidence* and *identification* queries. Each query was run on 10 intervals generated uniformly at random encompassing between 1% and 75% of the data. Each *evidence* query was also run with different thresholds selected uniformly at random. The running times reported in Figure 9.c are an average over the entire set of queries. Query answering times are always below 2 seconds for any restrictions, showing that the MAGIC structure handles activity occurrences efficiently.

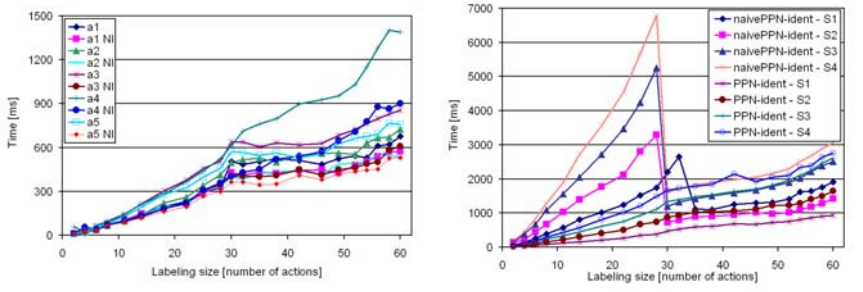
6.2 PPN-evidence, *naive* PPN-ident *and* PPN-ident

In the following, we briefly discuss the most significant results for PPN-evidence, naivePPN-ident and PPN-ident. We first measured the running times of the three algorithms for the five activity definitions described above while varying the the number of action symbols in the labeling (Figure 10). It is worth noting that naivePPN-ident exhibits a seemingly strange behavior – running time increases almost exponentially with labeling size, only to drop suddenly at a labeling size of 30. At this labeling size activity definitions first begin to be satisfied by the labeling.



(a) (b) (c)

Fig. 9 MAGIC (a) build time; (b) memory occupancy; and (c) query time

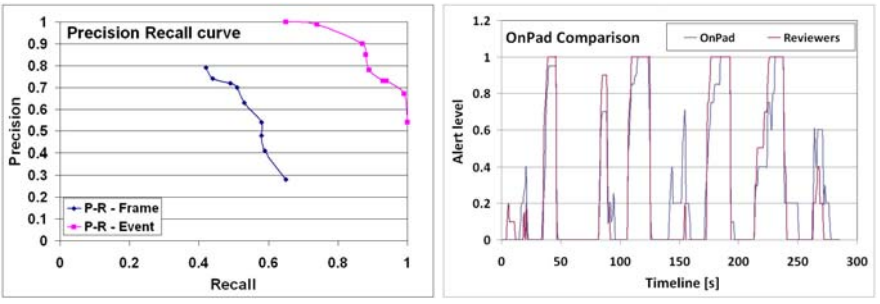


(a) (b)

Fig. 10 Typical execution times for (a) PPN-evidence, (b) naivePPN-ident and PPN-ident.

When no activity definition is satisfied, naivePPN-ident performs several iterations to decrease the threshold until it reaches 0, but it is comparable in terms of running time with PPN-ident when the activity is satisfied by at least one subsequence. Both PPN-evidence and PPN-ident perform linearly with the size of the input.

We then measured the precision and recall of PPN-evidence, naivePPN-ident and PPN-ident w.r.t. the human reviewers. We observed that the minimality condition used by the PPN-evidence algorithm poses an interesting problem – in almost all cases, humans will choose a sequence which is a superset of the minimal subsequence for an activity. In order to have a better understanding of true precision and recall, we compute two sets of measures: recall and precision at the *frame level* (R_f and P_f) and recall and precision at the event (or activity) level (R_e and P_e). At the event level we count as correct any subsequence returned by the algorithm that overlaps with a subsequence returned by a human reviewer. At the frame level we count as correct any frame returned by the algorithm that is also returned by a human reviewer. Details are omitted for reasons of space. However, it is worth mentioning that we observed a surprising behavior for frame recall of PPN-evidence, which appears to increase as the threshold increases. This can be explained considering that, for low thresholds, there is a relatively high number of small candidate subsequences, hence the minimality condition causes *fewer frames* to appear in the answer. This pattern disappears for higher threshold. We also investigated to what



(a) (b)

Fig. 11 (a) Precision/recall of OffPad; (b) Comparison between OnPad and human reviewers

extent the sequences returned by the reviewers and by PPN-evidence actually overlap, and found that overlap is above 70% in 80% of the cases.

6.3 *OffPad and OnPad*

We first measured the running time of OffPad for the three datasets and observed a high correlation (coefficient 0.95) between the degree of the formula and the time taken to find an answer. We also investigated the impact of the *heuristic_order* and *pruning* methods on the running time of OffPad. Without the *heuristic_order* method, the average running time was 1.35 times greater. Pruning had an even larger effect – running without the pruning method increased the running time 3.5 times, due primarily to the fact that substitutions for frame variables were analyzed even on search paths that could not lead to a viable result. Finally, we observed that running time is at most linear in the size of the labeling. We then evaluated precision and recall of OffPad w.r.t. the ground truth provided by human reviewers. The minimality condition used by OffPad poses the same problem discussed in Section 6.2, thus we computed recall and precision both at the *frame* and event level (Figure 11.a).

W.r.t. to OnPad, we first evaluated the average processing time per frame, and concluded that OnPad can process – in real-time – videos sampled at 4 frames per second. We then looked at how the alert levels returned by OnPad compare to those provided by human reviewers. In brief, the experiments show that OnPad approximates very well the behavior of the reviewers (Figure 11).b. However, we also observed that OnPad tends to be more "conservative" in its alert level while an activity is in its incipient stages, whereas humans tend to have a better intuition for what will happen, even just a few frames after the activity starts.

7 Conclusions and Future Directions

Significant progress has been made in recent years on several aspects of activity detection in videos. However, considerably less effort has been put towards the

definition of a framework where all the aspects of Semantic Video Content Analysis are integrated in a coherent and effective way.

In this chapter, we have presented a framework and a design methodology with the objective of bridging this gap. We have analyzed the typical requirements of a video analysis system from a user's perspective and identified three main classes of problems that such an integrated system should address. We have shown that, based on the nature and complexity of the activities being monitored, different formalisms may be used to model those activities, and different classes of algorithms can be designed to solve the above mentioned problems. Finally, we have shown in real experiments the effectiveness of the proposed design philosophy.

Although our work constitutes a first important step toward a unified framework for Semantic Video Content Analysis, there is still huge room for improvement. As part of the effort to reduce the gap between low-level primitives and semantic activities, we may explore the possibility of pre-processing the output of image processing algorithms and combine primitive actions into 'less than primitive' actions, in order to make the definition of high-level activities more intuitive.

References

1. Akdemir, U., Turaga, P., Chellappa, R.: An ontology based approach for activity recognition from video. In: Proc. of the 16th ACM Intl. Conf. on Multimedia (MM 2008), pp. 709–712 (2008)
2. Albanese, M., Chellappa, R., Moscato, V., Picariello, A., Subrahmanian, V.S., Turaga, P., Udrea, O.: A constrained probabilistic petri net framework for human activity detection in video. IEEE Transactions on Multimedia 10(8), 1429–1443 (2008)
3. Albanese, M., Moscato, V., Picariello, A., Subrahmanian, V.S., Udrea, O.: Detecting stochastically scheduled activities in video. In: Proc. of the 20th Intl. Joint Conf. on Artificial Intelligence (IJCAI 2007), pp. 1802–1807 (2007)
4. Albanese, M., Pugliese, A., Subrahmanian, V.S., Udrea, O.: MAGIC: A multi-activity graph index for activity detection. In: Proc. of the IEEE Intl. Conf. on Information Reuse and Integration (IRI 2007), pp. 267–278 (2007)
5. Avrahami-Zilberbrand, D., Kaminka, G., Zarosim, H.: Fast and complete symbolic plan recognition: Allowing for duration, interleaved execution, and lossy observations. In: Proc. of the AAAI Workshop on Modeling Others from Observations, MOO 2005 (2005)
6. Chen, D., Yang, J., Wactlar, H.D.: Towards automatic analysis of social interaction patterns in a nursing home environment from video. In: Proc. of the 6th ACM SIGMM Intl. Workshop on Multimedia Information Retrieval (MIR 2004), pp. 283–290 (2004)
7. Chen, S.M., Ke, J.S., Chang, J.F.: Knowledge representation using fuzzy petri nets. IEEE Transactions on Knowledge and Data Engineering 2(3), 311–319 (1990)
8. David, R., Alla, H.: Petri nets for modeling of dynamic systems a survey. Automatica 30(2), 175–202 (1994)
9. Duong, T.V., Bui, H.H., Phung, D.Q., Venkatesh, S.: Activity recognition and abnormality detection with the switching hidden semi-markov model. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2005), vol. 1, pp. 838–845 (2005)
10. Georis, B., Maziere, M., Brémond, F., Thonnat, M.: A video interpretation platform applied to bank agency monitoring. In: IEE Intelligent Distributed Surveilliance Systems (IDSS-04), pp. 46–50 (2004)

11. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing. Intl. Journal of Human-Computer Studies 43(5-6), 907–928 (1995)
12. Guler, S., Burns, J.B., Hakeem, A., Sheikh, Y., Shah, M., Thonnat, M., Bremond, F., Maillot, N., Vu, T.V., Haritaoglu, I., Chellappa, R., Akdemir, U., Davis, L.: An ontology of video events in the physical security and surveillance domain (2004), http://www.ai.sri.com/~burns/EventOntology/PhysicalSecurity1-30-2004.doc
13. Hakeem, A., Shah, M.: Ontology and taxonomy collaborated framework for meeting classification. In: Proc. of the 17th Intl. Conf. on Pattern Recognition (ICPR 2004), vol. 4, pp. 219–222. IEEE Computer Society, Los Alamitos (2004)
14. Hamid, R., Huang, Y., Essa, I.: ARGMode – activity recognition using graphical models. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshop (CVPRW 2003), vol. 4, pp. 38–43 (2003)
15. Hobbs, J., Nevatia, R., Bolles, B.: An ontology for video event representation. In: Proc. of the 2004 IEEE Conf. on Computer Vision and Pattern Recognition Workshops (CVPRW 2004), p. 119 (2004)
16. Luhr, S., Bui, H.H., Venkatesh, S., West, G.A.W.: Recognition of human activity through hierarchical stochastic learning. In: Proc. of the First IEEE Intl. Conf. on Pervasive Computing and Communications (PCC 2003), pp. 416–422 (2003)
17. Marhasev, E., Hadad, M., Kaminka, G.A.: Non-stationary hidden semi-markov models in activity recognition. In: Proc. of the AAAI Workshop on Modeling Others from Observations, MOO 2006 (2006)
18. Marsan, M.A., Balbo, G., Chiola, G., Conte, G., Donatelli, S., Franceschinis, G.: An introduction to generalized stochastic petri nets. Microelectronics and Reliability 31(4), 699–725 (1991)
19. Murata, T.: Petri nets: Properties, analysis and applications. Proc. of the IEEE 77(4), 541–580 (1989) (1989)
20. Nevatia, R., Zhao, T., Hongeng, S.: Hierarchical language-based representation of events in video streams. In: Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition Workshop (CVPRW 2003), vol. 4 (2003)
21. Petri, C.A.: Communication with automata. DTIC Research Report AD0630125 (1966)
22. Shoenfield, J.R.: Mathematical Logic. Addison Wesley, Reading (1967)
23. Vu, V.T., Brémond, F., Thonnat, M.: Automatic video interpretation: A novel algorithm for temporal scenario recognition. In: Proc. of the 18th Intl. Joint Conf. on Artificial Intelligence (IJCAI 2003), pp. 1295–1302 (2003)

Video Genre Inference Based on Camera Capturing Models

Ping-Hao Wu, Sanjay Purushotham, and C.-C. Jay Kuo

Abstract. On-line video collection is getting larger nowadays. It becomes difficult for users to go through the whole collection to find the video of their interest. To allow efficient browsing, search and retrieval, one intuitive solution is to cluster video clips according to their genres automatically. Then, users' choices can be narrowed down. Besides on-line video repositories, other applications include managing television broadcasting archives, video conferencing records, etc. The goal of video classification is to automatically place each video title in different categories, such as news, sports, etc. The classification process involves extracting the information from the video clips and classifying them into different classes. In this chapter, we first review related work in this field. Then, two novel features based on the camera shooting process is proposed for video genre classification. These new camera based features exploit the fact that a different genre tends to have different camera effects and user perception. Although a lot of work has been proposed with the consideration of cinematic principles, most extracted features are low-level features without much semantic information. We propose a feature that estimates the number of cameras used in a short time interval. Then, we propose another feature by calculating the distribution of the camera distance, which is approximated by the normalized foreground area of each frame. The block-based motion vector field is adopted to

Ping-Hao Wu
Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089
e-mail: pinghaow@usc.edu

Sanjay Purushotham
Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089
e-mail: spurusho@usc.edu

C.-C. Jay Kuo
Ming Hsieh Department of Electrical Engineering, University of Southern California, Los Angeles, CA 90089
e-mail: cckuo@sipi.usc.edu

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 177–201.
springerlink.com

reduce the complexity involved in foreground/background modeling. Preliminary experiment results show that the proposed features capture additional genre-related information so that the video genre can be inferred from the proposed features well.

1 Introduction/Motivation

The amount of video contents available to users is tremendous nowadays. The number of video files on the TV and the Internet is exponentially growing every day. Video files can be generated by anyone using a professional camera, a home camcorder or a camera-enabled mobile phone. The professional and the user-generated video contents make our video repositories extremely huge so that it is infeasible for human to handle it manually. To find a video file of interest for a user from a large video collection is a difficult task. Hence, there is a great need to automatically classify and index video contents for easy access and retrieval from large databases. Classifying video into various genres is not only helpful to video search but also for video recommendation systems.

1.1 What Is Video Genre Classification?

Video genre classification and categorization is an important yet basic module in the management of today's ever-growing video databases and video mining. It helps end users efficiently organize, browse, and search video clips in digital video libraries. Traditionally, classifying video into several pre-determined categories such as news, sports and commercials involves two steps. First, models are built for each genre from a set of training video clips. Second, video clips with an unknown genre are compared with the models of the pre-determined genres. In the first step, visual and/or audio features are extracted to represent each video clip. Learning methods are used to bridge the gap between low-level features and the video genre, which is a high-level semantic concept. In the second step, a proper similarity function is adopted to determine which genre the video belongs to.

Automatic video genre classification is an active challenging research topic today, and its significance is highlighted by the NIST TRECVid Video Retrieval competition which is held every year starting from 2001.

1.2 Organization

This chapter is organized as follows. In Section 2, we briefly describe different video genres and the genres considered in our work. In Section 3, different types of features for classification are reviewed, and their advantages and disadvantages are discussed. Typical classification tools are described in Section 4. Since the shooting scenario is inherently linked with the video genre, two new features that take the shooting process into account are proposed in Section 5. Preliminary experimental results are given and discussed in Section 6. We demonstrate that the proposed

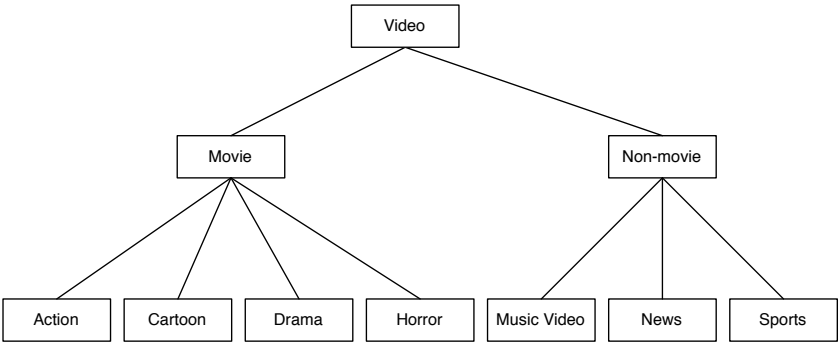


Fig. 1 Video genres considered in this chapter.

features can capture the additional genre-related information on top of traditional features, followed by the summary and future research directions in Section 7.

2 Video Genre Types

A lot of video files are generated every day. Some of them are created by professionals while others are created by ordinary users. They can be classified as follows.

a) Professional video
   Professional video files are created and edited by video/camera professionals. These include news video, sports video, motion picture video from the movie industry, commercial video, etc. They are carefully edited so that they tend to follow the practice of film theory. Professional video is generally shot using multiple professional cameras.
b) Personal video
   Personal video files are generated by ordinary users such as amateurs and hobbyists. They are recorded generally using a single camera such as a camcorder, a web cam, or a camera-based mobile phone. A large number of personal video files available in the Internet are unedited.

There are different ways of video genre classification. Apart from professional and personal video types, various video genres considered in this chapter are shown in Figure 1. Note that the genres of our consideration can be further classified. For example, the sports video may include basketball, baseball, soccer, football, etc. There are different kinds of cartoon contents, too.

3 Feature Extraction

Automatic video genre classification is a grand challenging problem in the multimedia information retrieval field. A large number of researchers have been working on

it for more than two decades. The task involves feature extraction and classification, and researchers have classified genres based on different information modalities (features) present in the video. For example, researchers use the visual modality frequently. Other modalities such as audio and text modalities have also been used. A few researchers use cinematic principles or concepts from film theory as additional features for video classification. Some researchers classify an entire video clip to a particular genre while others focus on classifying segments of video (*e.g.,* identifying violent scenes in a movie). Some classify a video file into one of board categories such as the movie genre while others classify a video file to a sub-category such as different types of sports video.

In this section, we examine different features such as visual, audio, text, and cinematic-based features that can be extracted from a video clip.

3.1 Low Level Visual Features

A visual modality based approach uses visual features extracted from the visual content of a frame, a shot or an entire scene. Although visual features are popular. However, they possess some difficulties such as the amount and dimensionality of the data can be huge and some of them are not easily extractable. As a result, visual-based features sometimes are used along with other modality features, including audio and text features.

3.1.1 Shot-Based Features

A shot is a continuous strip of video, made up of series of frames, generally between two cuts. A shot is detected by finding the transition between different shots. There are currently more than 100 different shot transitions and it is challenging to identify all of them correctly. Most common shot transitions include: hard cuts, fade in/fade out, and dissolves. It is important to identify shot transitions correctly to extract other features such as color and motion features from a shot. The shot-based features include: the shot transition length, the average number of shots and the type of shots. The shot transition length represents the number of frames in a particular shot while the average number of shots represents the mean number of shots in a scene or in a video clip. They have been used by Iyegar and Lippman [12], Jadon [13], Troung [30], among many others. Shot-based visual features are quite commonly used in video genre classification.

3.1.2 Frame-Based Features

Each video frame has several pixels and the color of each pixel is represented by values in the color space. The color content of a frame or the distribution of colors in a video frame can be represented by color-based features such as the color histogram. Color histograms are often used to compare two frames/two shots. For example, a horror scene has a darker color content than a comedy scene. Thus, color histograms of comedy and horror scenes are completely different. The most commonly

used color-based features include: the color histograms of video frames or regions in a video frame, the color intensity feature of video frames, the color Saturation feature (the amount of white light in the color) in video frames. Wan and Kuo [31] described an efficient way for the hierarchical color histogram representation.

Other frame-based features include: the edge-based and the texture-based features. The edge features represent the amount and the type of edges present in a video shot. For example, a basketball court has slanted edges in a diagonal long shot view. Texture features offer the texture of a surface seen in a shot. For example, the texture content in a soccer game can be a useful feature to distinguish it from a basketball game. Edge and texture features are often used in classifying sports video.

For recent work on robust video frame features, we refer to the work of Drew and Au [5], Fan *et. al* [6], Hauptmann [9], Iyegar [12] and Xu and Li [34].

3.1.3 Motion-Based Features

Generally speaking, there are two motion types; namely, the motion of objects in a video shot and the motion due to the movement of cameras. Commonly used motion-based features are: optical flow, motion vectors and pixel-based frame differencing features. Fischer [7], Kobla [16], Roach [28], Troung [30], Wang [32] and others have used motion-based features for video classification. The optical flow feature is the estimated velocity flow of pixels in video frames due to object or camera motion. It can be found by solving an optical flow equation under the smoothness constraint as described by Horn and Schunck [10] or using wavelets to measure the motion density as explained by Nam *et al.* [24]. The motion vector feature includes the average magnitude and the standard deviation of the motion, which is determined by the motion vector field. The pixel-based frame differencing feature was used by Roach *et al.* [27] in detecting the motion of foreground objects.

3.1.4 Object-Based Features

The object-based features include the number of objects, the color, size, texture, trajectory of the object in a video shot, and DCT-based features. Brezeale and Cook [3] and Lee [17] used the DCT based features exclusively in video genre classification. Identifying people in a video program using face recognition techniques provides another useful object-based feature. However, detecting and identifying objects and faces is a difficult problem, so that they are less commonly used for video genre classification.

3.2 Audio-Based Features

Audio-based features are very useful for video genre classification. The main advantage of audio-based features is that they can be easily extracted as compared with visual-based features. However, it was observed [2] that video classification with only audio-based features is not efficient. That is, the correct classification rate is

low. Thus, it is often to use the audio modality in combination with visual and text modality features to achieve a higher video genre classification rate.

According to [2], audio features can lead to three classes of audio understanding: low-level acoustics such as average frequency of a frame, midlevel feature classes such as the audio signature of the sound, a person speaking etc., and high-level scene classes such as background music playing in certain types of video scenes.The audio features can be classified into time-domain and frequency domain based features which are discussed below.

3.2.1 Time-Domain Based Features

The time-domain audio features include: RMS value of the audio signal energy, zero crossing rate (ZCR), silence ratio, etc. RMS of the signal energy feature measures the loudness/volume of the sound. For example, sports video have a different noise (sound) level content than a romantic video. Zero crossing rate is the number of amplitude changes of the audio signal in a video frame. ZCR feature can be used to identify the silence frames in the video shot. Silence ratio (feature) is the ratio of speech frames to silent frames in a video scene. The silence ratio for a news video is higher than commercials. Some researchers such as Zhang and Kuo [36] have done video segmentation based on audio based features.

3.2.2 Frequency-Domain Based Features

The most common frequency domain features are: the fundamental frequency, the frequency bandwidth and Mel frequency cepstral coefficients (MFCC). They have been used in [11, 17, 20]. The fundamental frequency is the lowest frequency in a musical note, and it approximates the pitch information, which helps distinguish male and female speakers. The bandwidth feature is a measure of the frequency range of audio signals. It is useful to distinguish speech from music since speech has narrower bandwidth than music. Thus, speech video such as a drama scene has narrower bandwidth than musical video. MFCCs are obtained by taking the logarithm of the spectral components and placing them in the Mel frequency bins. It is a perception-based feature, which is useful to identify speakers.

3.3 Text-Based Features

Reseachers such as Brezeale and Cook [3], Jasinchi and Louie [14], Lin and Hauptmann [19], Zhu *et al.* [38] used text-based features for video classification since the text feature containing a higher level semantic concept is easier to understand. However, since the text information may not be readily available from a video file, it is not as popular as the visual and the audio features.

Texts in a video file can be categorized as the viewable text and the transcript text. The viewable text includes: the scene text and the graphic text while the

transcript text includes transcript of dialogs and closed captions. The scene text refers to the text on objects in a video scene, for example, athlete's name on his jersey, the building name etc. The graphic text refers to the text embedded in a video shot such as scores of a sports event or subtitles in a movie. The viewable text can be extracted using the optical character recognition (OCR) technique while transcripts of dialogs can be extracted from speech in a video scene. Speech and language recognition techniques are used to extract the speech presented in video clip.

Closed captioning (similar to transcripts) is a method of displaying the text on a video shot. In addition to dialogs, closed captions have other information such as environment sounds (*e.g.*, bear growls [Grrrr], train passings [tututu]). The text-based features generally show a direct relationship between the text feature and a specific genre. To give an example, transcripts such as 'sport arena', 'Lakers', 'basketball' indicate a basketball sport genre. However, the character and speech recognition techniques to extract text-based features have high error rates. Moreover, text-based features are not present in many video clips, which makes the use of text-based features for video genre classification less practical. Thus, text based features are often used along with other features for video genre classification.

3.4 Cinematic Based Features

Cinematic-based features include: cinematic principles or concepts from film theory and camera based features such as camera motion and camera angles. Cinematic principles refers to the film grammar for different video shots such as the extreme long, the long , the medium long, the medium, the close up and the extreme close up shots. These cinematic principles can be used to interpret different genres differently. For example, a close up shot is commonly used in drama video while a long shot is used in a soccer sport video. The camera motion feature include panning, titling, zooming, etc, while the camera angle feature measures the camera angle of the video shot. These features are helpful in classifying video, for example, a horror video may have more slanted camera angles than a news video. Wei [33], used camera based features such as camera motion - panning, tilting, etc. in video genre classification. Rasheed and Shah [25] considered concepts from film theory.

3.5 Feature Integration

Table 1 lists the advantages and the disadvantages of each type of feature. Considering the drawbacks when they are used alone, researchers have used multiple features to better represent the underlying video. The integration of multiple features gives higher classification accuracy than the use of a single feature. Fischer *et al.* [7] used a three-step process to combine the visual, the audio and the cinematic features to get better video classification results. Huang *et al.* [11] used joint audio and visual features. Li *et al.* [18] did a thorough review on movie content analysis based on

Table 1 Feature Comparison

Feature Type	Advantages	Disadvantages
Text-based features • Viewable text such as scene text, graphic text • Transcript text such closed captions, transcript of dialogs	Relation between features and specific genre is easy to understand	• High OCR, ASR error rates • Text based features not present in all videos • Higher computational cost for large dialogs
Audio-based features • Time-domain features such as ZCR, silence ratio, RMS of audio signal • Frequency-domain features such as fundamental frequency, freq. bandwidth, MFCCs	Computational complexity lower than visual features	• Difficult to distinguish between different environment sounds
Visual-based features • Color-based such as color histogram, edge, texture • Motion-based such as motion vectors, optical flow • Shot-based such as avg. no. of shots, shot length • Object-based such as the object size, texture, color	• Color-based features are simple and easy to implement • Visual features are commonly used since they are present in all video contents	• Shot-based features such as shot transition detection are difficult to identify • Object recognition is difficult and computationally expensive

joint features. Rasheed and Shah [25] used audio, visual, and cinematic principles. A basket ball sports video was analyzed by Zhou *et al.* [37] using integrated motion, color and edge features.

There is however no standard approach to combine different features. Some combined all features into a single feature vector while others trained classifiers for each feature type and used another classifier to make final decision [2]. Furthermore, it is possible to study the correlation between different features to reduce complexity and improve classification accuracy. Feature fusion is still in its infancy, and it is a growing area in machine learning. It remains to be one of the challenging and active research areas in the video information retrieval field.

4 Classification Rules

Video classification involves classification of input features into different genres. A training set is used in finding the parameters of the classification model

(or rule) while a testing set is used for video classification. Commonly used classifiers include: k-nearest neighbors, the Guassian mixture model (GMM), the hidden Markov model (HMM) and the support vector machine (SVM), etc. In this section, we briefly discuss commonly used classification tools in the context of video genre classification.

4.1 K-Nearest Neighbors (kNN)

KNN is one of the simplest and commonly used machine learning algorithms for classifying features into multiple classes. A feature vector is classified to a particular class based on the majority vote of its neighbors. Sometimes, a weighted contribution of neighbors is adopted for classification.

4.2 Gaussian Mixture Models (GMM)

Features under the modeling of GMM is generated by a probability density distribution (pdf), which is expressed as the weighted sum of a set of Gaussian pdfs. Using the Expectation Maximization (EM) algorithm, the optimum set of GMM parameters can be identified in an iterative manner. In our context, the pdf of the input feature vector is modeled by GMM, and the classification result is the target video genre.

4.3 Support Vector Machine (SVM)

SVM is a state-of-the-art machine learning tool, which has been used by researchers to classify video into various genres. SVM is trained by features extracted from video shots or clips. SVM views the input data instances as two sets of vectors in an n-dimensional space, and constructs a separating hyper-plane that maximizes the margin between the two data sets. Thus, it is also known as the maximum margin classifier.

Mathematically, given a training set of instance-label pairs (x_i, y_i), $i = 1, \cdots, l$, where $x_i \in R^n$ and $y \in \{-1, 1\}$ (*i.e.,* two classes), SVM requires the solution to the following optimization problem:

$$\min_{w,b,\xi} \frac{1}{2} w^T w + C \sum_{i=1}^{l} \xi_i, \tag{1}$$
$$\text{subject to } y_i \left(w^t \phi(x_i) + b \right) \geq 1 - \xi_i, \text{ and } \xi \geq 0.$$

Here, training vectors x_i are mapped into a higher (maybe infinite) dimensional space by function ϕ. Then, SVM finds a linear separating hyperplane with the maximal margin in this higher dimensional space. $C > 0$ is a penalty parameter of the error term.

We can define a kernel function in form of

$$K(x_i, x_j) \equiv \phi^T(x_i)\phi(x_j). \tag{2}$$

Several other kernal functions are given below.

- Linear kernel: $K(x_i, x_j) = x_i^T x_j$,
- Polynomial kernel: $K(x_i, x_j) = (\gamma x_i^T x_j + r)^d$, $\gamma > 0$,
- Radial basis functions (RBF): $K(x_i, x_j) = \exp(-\gamma \|x_i - x_j\|^2)$, $\gamma > 0$, and
- Sigmoid: $K(x_i, x_j) = \tanh(\gamma x_i^T x_j + r)$, $\gamma > 0$,

where γ, r and d are kernel parameters.

4.4 Other Classification Tools

There are other classifiers used for video genre classification such as the hidden Markov model (HMM), the linear threshold unit (LTU), the decision tree, linear discriminant analysis, the neural network, the Bayesian network, etc.

5 Camera Capturing Model

Traditional methods use low-level features such as the shot length, color, motion etc. Here, a novel approach based on an intermediate video capturing model is introduced to bridge the gap between low-level features and the video genre. The idea is that the video genre is inherently linked with the video capturing scenario. For example, only a single camera is involved for personal video while multiple cameras and editing are employed for producing professional video. Another example is the dialogue scene, which happens frequently in romance and other character movies. In dialog scenes, two cameras are often used to capture the faces of two actors.

Two novel features for video genre classification are proposed based on the video capturing scenario. The first one is the number of cameras. By comparing the similarity between key frames of shots, we can roughly determine how many scenes there are within a certain time window. The second feature is the distance of the subject to the camera. If the subject is farther away from the camera, it appears smaller in the frame and could be of less importance. On the contrary, if the subject is closer to the camera, the subject is bigger and could be more important. Since the features are extracted by taking the filming process into account, most commonly used classifiers can successfully label video clips into different genres.

This section started by discussing the basis knowledge in filming, especially the effects of different usages of cameras. Then, two features are proposed.

5.1 Filming Basics

There are many ways to use cameras depending on how you want to shoot and present the scene. The camera characteristics can be roughly categorized by different

subject distances, camera angles (including horizontal and vertical positions), focal lengths, or camera levels as explained below.

- Subject distances: extreme long shot, long shot, full shot, medium shot, head and shoulders close-up, close-up, big close-up, extreme close-up.
- Horizontal position: front angle, profile angle, rear angle.
- Vertical position: bird's eye angle, high angle, neutral angle, low angle, worm's eye angle.
- Lens or focal lengths: wide-angle lens, telephoto lens.
- Camera levels: normal, dutch angle (tilting the camera off to the side so that the shot is composed with the horizon at an angle to the bottom of the frame).

Different uses of cameras can have different presentations for the same scene, and convey different information. For example, close-ups indicate the importance of a subject being filmed and create impacts; longer shots make the scene less intense; high angles make the viewer feel more powerful than the subject while low angles suggest the powerfulness of the subject; and dutch angles are often used to create tension or uneasiness. Therefore, it is reasonable to argue that there could be different distributions of camera distances or angles for video of a different genre.

For fiction video that wants to tell some stories usually sticks to a moderate position or angle, because they do not want to call attention to themselves. This is especially true for video like comedy, which usually tries to avoid big close-ups. The drama is similar except that tighter angles tend to be chosen to increase emotional intensity. As for action movies, a lot of dutch angles are used. Sometimes, unusual angles are used for some special effects. For example, a sudden bird's eye view or an extreme close-up can startle the audience. However, these special angles are not used often.

As for non-fiction video such as news, sports and documentaries, the important thing is neutrality. Therefore, camera setups tend to be neutral. The camera angle types range from the long shot to the loose close-up. Extreme setups like the wide angle, extreme close-up, or off-level angles are often avoided when filming video of these genres. For example, on-camera reporters are usually covered in medium shots. Even for the news shots in the field, similar rules also apply.

Long shots or telephoto lens are used often in sports videos, since directors would like the camera to be off the playing field. Sometimes, closer shots on the players, either on the field or the bench, coaches, or even the spectators may be desirable. In these occasions, some close-ups may be used. Commercial and music video contents usually contain a lot of editing or shooting effects. In some ways, more extreme angles are preferred to create as dramatic and/or striking feeling as possible.

5.2 Camera Distance

The distance between the camera and the objects being shot can be classified into the following several types.

- Long shot (LS) (sometimes wide shot): Long shots typically show the entire subject (for example, human figures) and usually include a large portion of the surroundings to provide a comprehensive view. There are also the extreme long shot (XLS) and the medium long shot (MLS). The extreme long shot is obtained when the camera is at the furthest position from the subject. It usually shows the outside of a building or a landscape. The medium long shot is a distance somewhere between long shot and medium shot. In the case of a human figure, it usually cuts off the feet and ankles.
- Medium shot (MS): In the medium shot, the subject and its surrounding occupy about the same areas in the frame. In the case of the human figure, a medium shot is from the knees or waist up. It is usually used for dialogue scenes. It is good at showing body languages but lacks the ability to show facial expressions.
- Close-up (CU): Close-up shots show a very small part of background. They concentrate on the detail, such as a character's face, which usually fills almost the whole frame. Variations include the medium close-up (MCU), the extreme close-up (XCU), etc. A medium close-up includes the head and the shoulder in the case of a human figure. On the other hand, an extreme close-up magnifies the subjects beyond what we usually experience in a real world. In the case of a human figure, it shows the head, usually from the forehead to the chin.

For the shot distance, the main difference is the object size, which gets bigger and bigger from the long shot to the close-up. Therefore, by measuring the ratio of the foreground object area to the background area, the relative shot distance can be estimated.

Extraction/detection of foreground or moving objects is an important step for many applications, such as object tracking and identification in video surveillance systems. In some applications, the background information is available for all frames, for example, when the background is static. Instead of modeling foreground objects, the background information allows the detection of the foreground by "learning" and "subtracting" the background from the video frame. For example, in [8], the frame difference of three consecutive frames are used and the background is modeled from several seconds of video. The edge map of the frame difference is used in [15]. The background registration technique is used to construct a reliable background map in [4]. The mixture of Gaussian (MoG) model is often used in background modeling [29, 21].

However, when the background is not static, that is, dynamic background, or when the camera is moving, the modeling and detection of background becomes a very challenging problem [22, 1, 26]. Background modeling and subtraction cannot be applied directly to these cases. Usually, motion compensation has to be applied to video frames first to compensate the movement caused by the moving camera. To this end, motion parameters of a camera motion model are estimated, usually based on optical flow, or motion vectors of certain feature points. These techniques assume that the adopted camera motion models are accurate enough so that video frames can be well compensated and aligned with others. Nevertheless, motion vectors are usually noisy, which means accurate camera motion reconstruction is generally difficult, and the estimation of the camera motion parameters is usually a complicated

process if certain accuracy is required. In addition, unlike surveillance video, shot change occurs in most movies, after which the background model would have to be reset. This makes the foreground/background extraction even more challenging.

Instead of the cumbersome procedure of estimating the camera motion, motion compensation of video frames, background modeling, and background subtraction, the idea of human visual attention model can be used to identify the foreground and background. Foreground objects, usually with more motion, attract more human attention then the background. By examining the motion vector field, it is possible to identify the regions that attract more attention.

A director usually moves the camera to track the movements of objects. Therefore, motion vectors should all be approximately equal to a global motion (v_1, v_2), which can be roughly estimated by the mean of the motion vector field:

$$\overline{v_i} = \frac{1}{M \times N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} v_i(x,y), \quad i = 1,2, \tag{3}$$

where (M,N) are the number of blocks in column and row, respectively, and $v_i(x,y)$ is the motion vector at position (x,y).

Foreground objects are identified as regions with motion vectors different from the global motion. Let $V(x,y)$ be a map denoting background by 0 and foreground by 1, then it can be obtained by

$$F(x,y) = \begin{cases} 1, \text{ if } |v_i(x,y) - \overline{v_i}| > \sigma_i, \\ 0, \text{ otherwise} \end{cases} \tag{4}$$

where σ_i is the threshold between the foreground and background. If the current motion vector is too much deviated from the mean motion vector, the current block is labeled as foreground. The threshold is selected as the standard deviation of the motion vector field:

$$\sigma_i^2 = \frac{1}{M \times N} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} (v_i(x,y) - \overline{v_i})^2, \quad i = 1,2. \tag{5}$$

However, when the background has some homogeneous or periodic content, the estimated motion vector could be wrong, resulting in some background blocks to be wrongly labeled as foreground blocks. To remedy this, for each 4×4 block in the current frame, a close neighborhood of the corresponding position in the previous frame is checked. If there is a block from the previous frame that is similar enough to the current block, the current block should be labeled as background. The similarity of two blocks is measured by the sum of absolute difference (SAD) of the luminance normalized by the sum of the luminance of the current block.

$$D_{\Delta x,\Delta y} = \frac{\sum_{i=0}^{3} \sum_{j=0}^{3} |I_t(x+i,y+j) - I_{t-1}(x+i+\Delta x, y+j+\Delta y)|}{\sum_{i=0}^{3} \sum_{j=0}^{3} I_t(x+i,y+j)}, \tag{6}$$

where $I(x,y)$ is the luminance component of the current frame, and Δx and Δy define the local neighborhood. If $D_{\Delta x,\Delta y}$ is smaller than a threshold, the current block is labeled as background; otherwise, the foreground/background is determined by $F(x,y)$. The camera distance is then estimated by the ratio of the foreground area to the background area, or to the entire frame.

The frame differencing is also often used in video classification. It measures the amount of motion between frames. If the camera is still, frame differencing can capture the movement of foreground objects. However, if the camera is moving, the content of the entire frame is generally changing and thus difficult to capture the movement of the foreground. In our approach, since regions that are consistent with the global motion are excluded, the foreground objects can be identified and the camera distance can be approximated.

5.3 Camera Number

Many existing methods use the average duration of shots or number of shots as one of the feature to classify video. It is based on the observation that action video usually has shorter and more shots to create the intense feeling for viewers, like car chasing, fighting or explosion scene. On the other hand, drama or romance video tends to have longer shots to develop the characters or scenes. Note that a longer duration also implies fewer shots in a fix interval.

However, drama movies sometimes would also have shorter shot durations, not necessarily as short as in action movies, but short enough to cause ambiguity. Take dialogue scenes for example, which happen a lot in drama movies. Two cameras are used to capture the faces of the two persons that are in the dialogue. There could be more than 10, or even almost 20 shot changes during an one minute conversation between two people, depending on the content of the conversation. The action movies sometimes have around 20 shot changes in one minute. Therefore, it is more important to determine the number of cameras used during a period of time rather than counting the number of shot changes.

Shot boundaries are determined via computing certain similarity (or distance) measure between adjacent frames. If the similarity (distance) is below (above) some threshold, a shot boundary is declared at the current frame. The same idea can be used to determine the number of cameras. If the camera is not moving or changing focus, the frames that are shot by the same camera should look similar given that they are close in time. The simplest way is to compare every pair of frames during a time period. However, this would demand heavy computations. The alternative is to extract key frames. First, shot change detection is applied first. Then, for each shot, the first frame is used as the key frame [23]. Although there exist other more complicated algorithms for determining the key frame of each shot, this simple approach should suffice for the purpose of determining the camera number.

To find out the number of cameras, we can simply compare the key frames extracted. If two key frames are similar enough according to a certain distance measure, they are labeled with the same camera index. Note that it makes no sense to

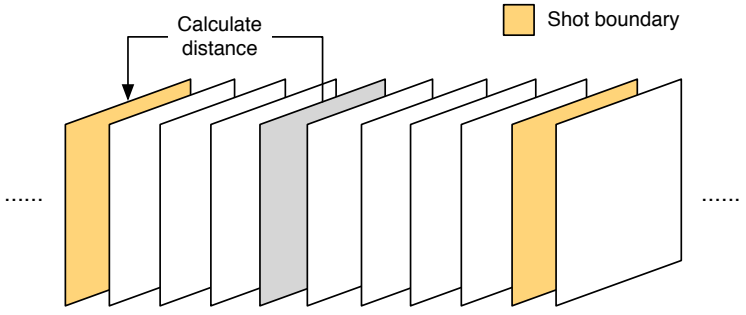


Fig. 2 The process of determining new camera frames.

the numbers of cameras in a long video since when the scene is changed, the camera setup is also changed as well. Therefore, the number of cameras should be calculated in a short time period only, for example, in one minute.

Furthermore, since camera motion is pretty common with which one key frame is insufficient to represent a shot, more than one key frames should be extracted from a shot. To achieve this, the approach presented in [35] is adopted, which compares each frame with the current key frame in the shot. If the distance between them exceeds a threshold, the current frame is selected as the next (new) key frame. This is illustrated in Figure 2.

The proposed algorithm for determining the number of cameras in a short time window is detailed below.

1. Compute the color histogram for the incoming (current) frame i.
2. Compute the distance between the current and the previous frames. If the distance exceeds a threshold, T_1, the current frame is chosen as a new key frame.
3. For each new key frame, compute the distances with previous key frames. Assign the camera index of the most similar key frames (should be below certain threshold T_3) to the new key frame.
4. If not, compute the distance between the current frame and the previous key frame. If the distance exceeds a threshold, T_2, the current frame is chosen as a new key frame but with a camera index of the key frame corresponding to the previous shot boundary.
5. Proceed to the next frame by increment frame index i by one.

6 Experimental Results and Discussions

Several video programs with different genres are collected from YouTube. They are collected for movie and non-movie classes. The movie class contains action, cartoon, drama, and horror, while the non-movie class contains music video, news, and sports, as shown in Fig. 1. Video programs were first divided into segments of approximately equal duration of 1 min. Then, they were all encoded by the H.264

video format. There are 20 1min clips in each genre, 140 in total, except that action08.mp4 is 20 seconds. The total length was around 2 hour and 20 minutes.

Note that this amount of video is not enough for both training and testing. In addition, to form a complete representation of each genre, more features that take other aspects such as color into consideration should be included as well. The results shown here only serve as preliminary results that demonstrate the feasibility of the proposed two new features. All thresholds that are used to extract the two features are determined empirically.

6.1 Camera Distance

The camera distance is approximated by the ratio of the foreground to the entire frame. Fig. 3 shows some examples of the foregrounds extracted. The idea is that a different camera distance can achieve a different visual effect and different genres should have different distributions of the camera distance. Therefore, the foreground to background ratio should not be averaged over the whole clip. Instead, the histogram of the ratio should be measured during a short time window, say, 1 minute. The ratio of the foreground to the whole frame is normalized and quantized to the

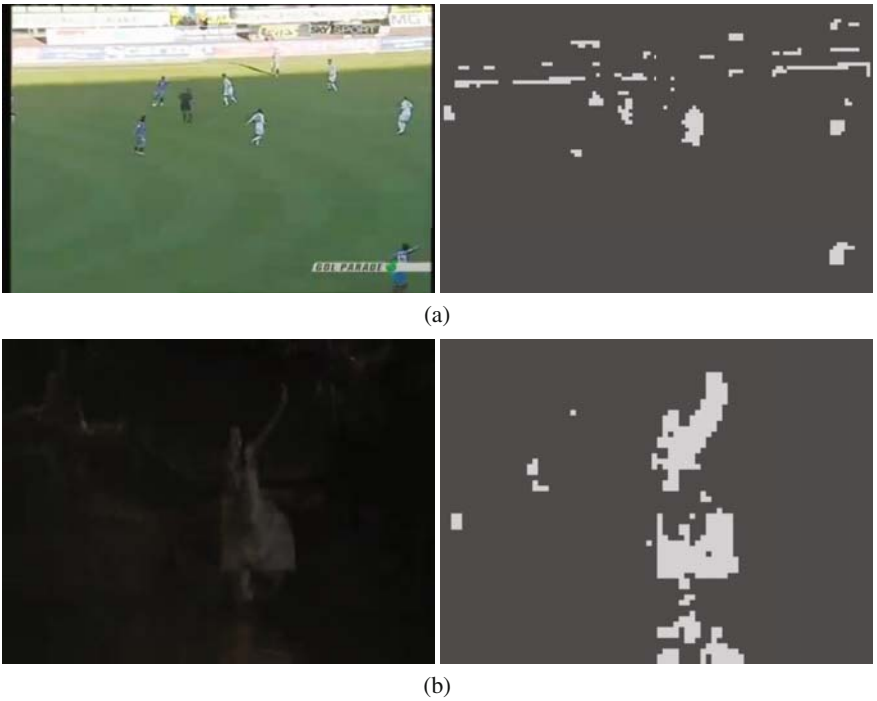


Fig. 3 Several foreground maps.

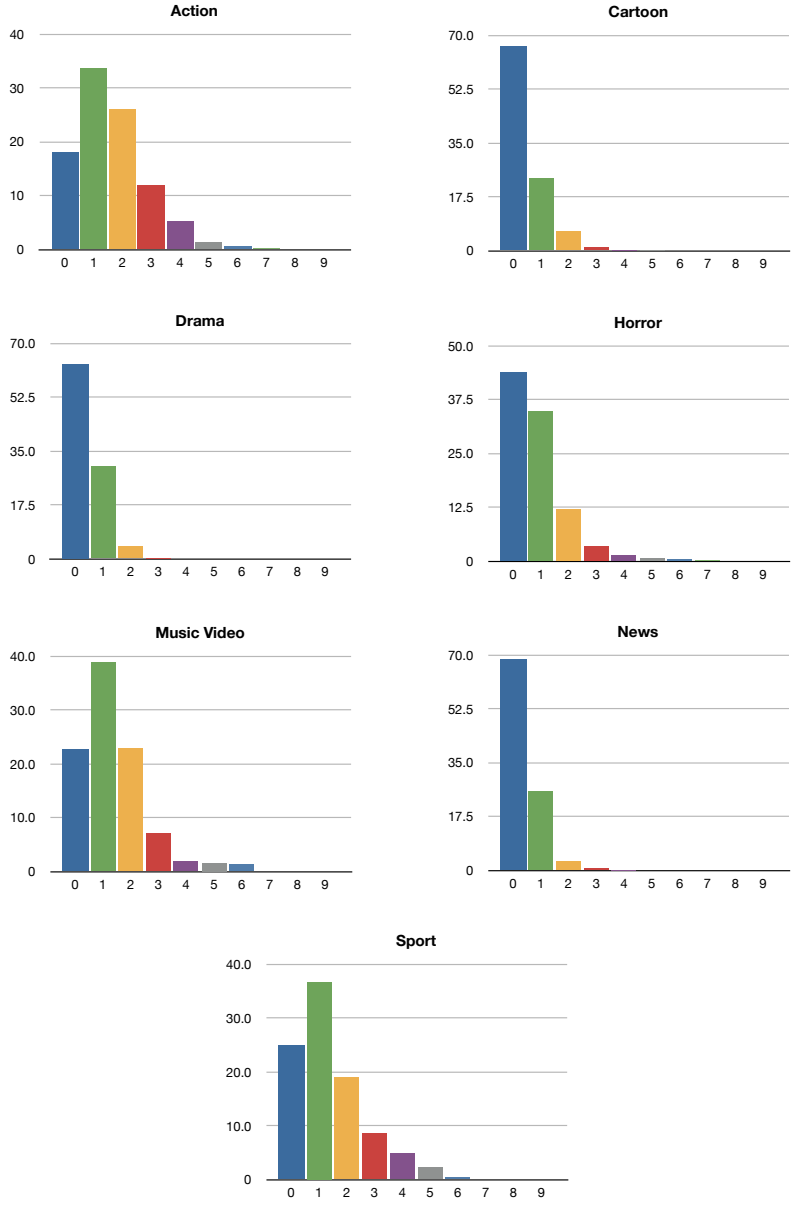


Fig. 4 The camera distance histogram for each genre.

range of 0 to 9. The histograms in every genre are averaged, respectively, as shown in Fig. 4.

We see that the action movie has a more distinct histogram as compared with others. The histograms for music video and sports video are similar to the one for action, but with fewer frames having larger values (3 6). Horror forms another group itself, which is a little like action, but actually quite different considering it has larger value at bin 0. Cartoon, drama, and news are the third group. The common point of these three genres is that they all have static background often. In addition, the scenes of a news presenter bear strong resemblance to dialog scenes in dramas. As for cartoon, when one object is moving, other objects tend not to move because this is easier for animators to create animation, which makes the scene has similar portion of foreground as the scenes in news or drama.

Note that since the proposed algorithm is based on the motion vector field, the foreground object would not be detected if it is not moving at all. That is the reason that many frames have value 0. Thus, this can be viewed as a metric related to motion as well. However, unlike frame difference methods which cannot extract the foreground when the camera is moving, the proposed algorithm is able to measure the portion that foreground objects take and, as shown by the results, some information about the video genre can be inferred from the distribution of the foreground ratio.

To conclude, the proposed camera distance measure does have different distributions among different genres. However, other features should be used as well in order to obtain a more precise description of the video genre. For example, more accurate measurement of the motion information can be obtained if it is combined with the frame difference method.

6.2 Camera Number

As mentioned earlier, the average shot number is often used in video genre classification, but it fails to consider the fact that there are sometimes fewer scenes than shots. The dialog scene, which happens often in movies that need character development, is such an example. Although there might be many shot changes during a conversation, there is actually only 2 cameras covering the two persons.

Fig. 5 shows the first frame of each shot in drama03.mp4, which is a short clip in the movie *No Country for Old Men*. This clip contains a typical dialog scene, where an establishing shot shooting the outside of a store is used to point out the location of the following event. A second shot is one person inside the store followed by another person walking into the store. Then, a conversation between them happens. As seen in the figure, the shot is alternating between the two persons in the conversation. There are 15 shot changes (with 16 shots in total) during this 1 minute clip. However, there are only 3 cameras in this video. The number of cameras (3) is more meaningful than the number of shots (16) in this case. The proposed algorithm successfully picks the first three frames shown in Fig. 5 to represent the cameras used

Fig. 5 Shots of drama03.mp4.

in this clip, which is an establishing shot, a shot of the first person and a shot of the second person.

Fig. 6 gives another example. There are 20 shots in drama07.mp4, which is a short clip from the movie *Shawshank Redemption*. However, by examining extracted shots, we see that there are only 6 different camera angles, which can be represented by the first 5 and the last key frames. Most of the time, the director just alternated between two cameras to capture face expressions and body movements of the subjects. Fig. 7 shows the frames that are extracted by the proposed algorithm to represent the cameras used in this clip. A total of 7 frames is extracted. The first 6 are correct frames, while the last one is a false alarm caused by the sudden motion in the scene where one person ripped off a wallpaper on the wall very quickly.

However, the proposed algorithm does not work well in every situation. Sometimes, the camera position changes when being inactive, which cause the frame content to change and can be regarded as a new camera. For example, in drama06.mp4, there are only 4 different cameras but 8 are detected, which is caused by the fact that there are several different close-ups for subjects besides a normal medium shot.

Figure 8 shows the numbers of shots and cameras for each video genre. In action movies, the number of shots is generally large since shot change occurs frequently to make the scene more intense. Sometimes, the number of cameras is much less than

Fig. 6 Shots of drama07.mp4.

Fig. 7 Cameras of drama07.mp4.

the number of shots in some cases. The reason is that, to make the scene intense, the director sometimes quickly alternates between two cameras. Although several action movie clips have about the same (or only a little bit less) number of cameras as the number of shots, it is safe to say that, if the number of cameras is less than the number of shots to a certain degree and the number of shots exceeds a certain threshold, the video clip is likely to be an action movie.

Music video also has many shot changes as well. The difference is the number of cameras is almost always about the same as the number of shots. Cartoon and sport videos also have about the same number of shots and cameras, but with fewer number of shots than the action movie or music video. For news video, the number of shots is significantly less than other genres. The number of cameras depends on what type of the segment is. If it is inside studio, the number of camera is less than the number of shots since the cameras are alternating between each other. However, if the news segment contains shots in the field, the number of cameras tend to be about the same as the number of shots. Horror movies have more shots than drama movies, but they both have a significantly less number of cameras then the number of shots. In drama movies, the camera number is few because of frequent dialogue scenes. As to horror movies, cameras sometimes switch quickly between several to create intense feeling. In addition, scenes in horror movies are usually very dark, which makes the differentiation more difficult than brighter scenes.

Recall that three groups can be formed by inspecting camera distance histogram. Consider the first group, which contains action, music video, and sports. Action movie and music video have similar number of shots, but action movies tend to have

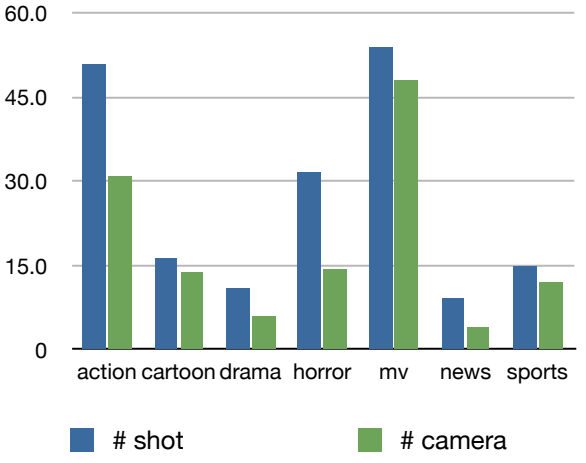


Fig. 8 Number of shots and number of cameras for each genre

less number of shots comparing to music video. On the other hand, sports video has both less number of shot and number of camera than action and music video. As for cartoon, drama, and news, which belong to the second group, cartoon has more shots and about the same number of camera; drama and news both have fewer number of shots comparing to their number of shots, but news video typically has less number of shot. Therefore, using the number of shots and the camera number, it is easy to distinguish inside each group.

To conclude, instead of using the number of shot changes alone, the number of cameras should also be considered. By jointly consider the number of shots and the number of cameras, much more information about the shooting process can be inferred. Note also that the number of cameras should only be calculated in a short interval, such as 1 min in the demonstrated cases.

7 Conclusion

In this chapter, we first reviewed techniques for video genre classification. Then, we introduced two new features; namely, the camera number feature and the camera distance feature based on the camera capturing model. It was shown by preliminary experimental results that, given the proposed features, we are able to infer the shooting process to capture additional video genre-related information on top of traditional visual features.

There is ample space for future research in video genre classification. Some of them are given below.

- Feature fusion *i.e.*, combining different multimodal features;
- Finding new features that are characteristic of a specific genre;
- Finding new classification rules for better video categorization;

- Managing and classifying large video database;
- Sub-genre video classification;
- New computational techniques to make feature extraction easier.

Video genre classification has many real world applications. It is closely related to other research areas such as video indexing, retrieval, and summarization. In some sense, automatic video genre classification is the first step to the management of a large video dataset. It helps end users efficiently organize, browse, and search video clips in video libraries.

References

1. Araki, S., Matsuoka, T., Takemura, H., Yokoya, N.: Real-time tracking of multiple moving objects in moving camera image sequences using robust statistics. In: International Conference on Pattern Recognition, vol. 2, p. 1433 (1998), http://doi.ieeecomputersociety.org/10.1109/ICPR.1998.711972
2. Brezeale, D., Cook, D.: Automatic video classification: A survey of the literature. IEEE Transactions on Systems, Man, and Cybernetics, Part C: Applications and Reviews 38(3), 416–430 (2008), doi:10.1109/TSMCC.2008.919173
3. Brezeale, D., Cook, D.J.: Using closed captions and visual features to classify movies by genre. In: 7th Int. Workshop Multimedia Data Min, MDM/KDD (2006)
4. Chien, S.Y., Ma, S.Y., Chen, L.G.: Efficient moving object segmentation algorithm using background registration technique. IEEE Transactions on Circuits and Systems for Video Technology 12(7), 577–586 (2002), doi:10.1109/TCSVT.2002.800516
5. Drew, M.S., Au, J.: Video keyframe production by efficient clustering of compressed chromaticity signatures (poster session). In: MULTIMEDIA 2000: Proceedings of the eighth ACM international conference on Multimedia, pp. 365–367. ACM, New York (2000), http://doi.acm.org/10.1145/354384.354534
6. Fan, J., Luo, H., Xiao, J., Wu, L.: Semantic video classification and feature subset selection under context and concept uncertainty, pp. 192–201 (2004), doi:10.1109/JCDL.2004.1336120
7. Fischer, S., Lienhart, R., Effelsberg, W.: Automatic recognition of film genres. In: MULTIMEDIA 1995: Proceedings of the third ACM international conference on Multimedia, pp. 295–304. ACM, New York (1995), http://doi.acm.org/10.1145/217279.215283
8. Haritaoglu, I., Harwood, D., Davis, L.: W4: real-time surveillance of people and their activities. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 809–830 (2000), doi:10.1109/34.868683
9. Hauptmann, A., Yan, R., Qi, Y., Jin, R., Christel, M., Derthick, M., Chen, M.Y., Baron, R., Lin, W.H., Ng, T.D.: Video classification and retrieval with the informedia digital video library system. In: Text Retrieval Conf., TREC 2002 (2002)
10. Horn, B.K., Schunck, B.G.: Determining optical flow. Tech. rep., Cambridge, MA, USA (1980)
11. Huang, J., Liu, Z., Wang, Y., Chen, Y., Wong, E.: Integration of multimodal features for video scene classification based on hmm, pp. 53–58 (1999), doi:10.1109/MMSP.1999.793797
12. Iyengar, G., Lippman, A.: Models for automatic classification of video sequences. In: Storage and Retrieval for Image and Video Databases (SPIE), pp. 216–227 (1998)

13. Jadon, R.S., Chaudhury, S., Biswas, K.K.: Generic video classification: An evolutionary learning based fuzzy theoretic approach. In: Indian Conf. Comput. Vis. Graph. Image Process, ICVGIP (2002)
14. Jasinschi, R., Louie, J.: Automatic tv program genre classification based on audio patterns, pp. 370–375 (2001), doi:10.1109/EURMIC.2001.952477
15. Kim, C., Hwang, J.N.: Fast and automatic video object segmentation and tracking for content-based applications. IEEE Transactions on Circuits and Systems for Video Technology 12(2), 122–129 (2002), doi:10.1109/76.988659
16. Kobla, V., DeMenthon, D., Doermann, D.: Identifying sports videos using replay, text, and camera motion features. In: Proc. SPIE Conf. Storage Retrieval Media Databases, pp. 332–343 (2000)
17. Lee, M., Nepal, S., Srinivasan, U.: Edge-based semantic classification of sports video sequences, vol. 1, pp. I-157– I-160 (2003), doi:10.1109/ICME.2003.1220878
18. Li, Y., Lee, S.H., Yeh, C.H., Kuo, C.C.J.: Techniques for movie content analysis and skimming: tutorial and overview on video abstraction techniques. IEEE Signal Processing Magazine 23(2), 79–89 (2006), doi:10.1109/MSP.2006.1621451
19. Lin, W.H., Hauptmann, A.: News video classification using svm-based multimodal classifiers and combination strategies. In: MULTIMEDIA 2002: Proceedings of the tenth ACM international conference on Multimedia, pp. 323–326. ACM, New York (2002), doi:10.1145/641007.641075
20. Liu, Z., Wang, Y., Chen, T.: Audio feature extraction and analysis for scene segmentation and classification. J. VLSI Signal Process. Syst. 20(1-2), 61–79 (1998)
21. Morellas, V., Pavlidis, I., Tsiamyrtzis, P.: Deter: detection of events for threat evaluation and recognition. Mach. Vision Appl. 15(1), 29–45 (2003), http://dx.doi.org/10.1007/s00138-003-0121-6
22. Murray, D., Basu, A.: Motion tracking with an active camera. IEEE Transactions on Pattern Analysis and Machine Intelligence 16(5), 449–459 (1994), doi:10.1109/34.291452
23. Nagasaka, A., Tanaka, Y.: Automatic video indexing and full-video search for object appearances. In: Proceedings of the IFIP TC2/WG 2.6 Second Working Conference on Visual Database Systems II, pp. 113–127. North-Holland Publishing Co, Amsterdam (1992)
24. Nam, J., Alghoniemy, M., Tewfik, A.: Audio-visual content-based violent scene characterization, vol. 1, pp. 353–357 (1998), doi:10.1109/ICIP.1998.723496
25. Rasheed, Z., Shah, M.: Movie genre classification by exploiting audio-visual features of previews, vol. 2, pp. 1086–1089 (2002)
26. Ren, Y., Chua, C.S., Ho, Y.K.: Motion detection with nonstationary background. Mach. Vision Appl. 13(5-6), 332–343 (2003), doi: http://dx.doi.org/10.1007/s00138-002-0091-0
27. Roach, M., Mason, J., Pawlewski, M.: Motion-based classification of cartoons, pp. 146–149 (2001), doi:10.1109/ISIMP, 925353
28. Roach, M., Mason, J., Pawlewski, M.: Video genre classification using dynamics. In: IEEE International Conference on Acoustics, Speech, and Signal Processing, 2001. Proceedings (ICASSP 2001), vol. 3, pp. 1557–1560 (2001), doi:10.1109/ICASSP.2001.941230
29. Stauffer, C., Grimson, W.E.L.: Learning patterns of activity using real-time tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 22(8), 747–757 (2000), doi: http://doi.ieeecomputersociety.org/10.1109/34.868677
30. Truong, B.T., Dorai, C.: Automatic genre identification for content-based video categorization. In: Proceedings of 15th International Conference on Pattern Recognition, vol. 4, pp. 230–233 (2000), doi:10.1109/ICPR.2000.902901

31. Wan, X., Kuo, C.C.J.: A new approach to image retrieval with hierarchical color clustering. IEEE Transactions on Circuits and Systems for Video Technology 8(5), 628–643 (1998), doi:10.1109/76.718509
32. Wang, P., Cai, R., Yang, S.Q.: A hybrid approach to news video classification multimodal features, vol. 2, pp. 787–791 (2003), doi:10.1109/ICICS.2003.1292564
33. Wei, G., Agnihotri, L., Dimitrova, N.: Tv program classification based on face and text processing, vol. 3, pp. 1345–1348 (2000), doi:10.1109/ICME.2000.871015
34. Xu, L.Q., Li, Y.: Video classification using spatial-temporal features and pca, vol. 3, pp. III-485–III-488 (2003)
35. Zhang, H., Wu, J., Zhong, D., Smoliar, S.W.: An integrated system for content-based video retrieval and browsing. Pattern Recognition 30(4), 643–658 (1997)
36. Zhang, T., Kuo, C.C.J.: Audio content analysis for online audiovisual data segmentation and classification. IEEE Transactions on Speech and Audio Processing, 9(4), 441–457 (2001), doi:10.1109/89.917689
37. Zhou, W., Vellaikal, A., Kuo, C.C.J.: Rule-based video classification system for basketball video indexing. . In: MULTIMEDIA 2000: Proceedings of the 2000 ACM workshops on Multimedia, pp. 213–216. ACM, New York (2000), http://doi.acm.org/10.1145/357744.357941
38. Zhu, W., Toklu, C., Liou, S.P.: Automatic news video segmentation and categorization based on closed-captioned text. In: IEEE International Conference on Multimedia and Expo., ICME 2001, pp. 829–832 (2001)

Visual Concept Learning from Weakly Labeled Web Videos

Adrian Ulges, Damian Borth, and Thomas M. Breuel

Abstract. Concept detection is a core component of video database search, concerned with the automatic recognition of visually diverse categories of objects ("airplane"), locations ("desert"), or activities ("interview"). The task poses a difficult challenge as the amount of accurately labeled data available for supervised training is limited and coverage of concept classes is poor. In order to overcome these problems, we describe the use of videos found on the web as training data for concept detectors, using tagging and folksonomies as annotation sources. This permits us to scale up training to very large data sets and concept vocabularies.

In order to take advantage of user-supplied tags on the web, we need to overcome problems of label weakness; web tags are context-dependent, unreliable and coarse. Our approach to addressing this problem is to automatically identify and filter non-relevant material. We demonstrate on a large database of videos retrieved from the web that this approach – called *relevance filtering* – leads to significant improvements over supervised learning techniques for categorization. In addition, we show how the approach can be combined with active learning to achieve additional performance improvements at moderate annotation cost.

1 Introduction

Recent technological developments like high-speed internet and large-scale storage devices have made it possible for private users to generate, publish, and share large

Adrian Ulges
German Research Center for Artificial Intelligence (DFKI),
D-67663 Kaiserslautern, Germany
e-mail: adrian.ulges@dfki.de

Damian Borth
University of Kaiserslautern, D-67663 Kaiserslautern, Germany
e-mail: d_borth@cs.uni-kl.de

Thomas M. Breuel
University of Kaiserslautern, D-67663 Kaiserslautern, Germany
e-mail: tmb@cs.uni-kl.de

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 203–232.
springerlink.com

amounts of data. This has led to a break-through of digital video, which is now not only broadcasted by TV stations but is also produced, streamed and stored on a private basis. Web video portals like YouTube, blinkx, or myspace[1] have become essential sources of information and entertainment to millions of users, and it is fair to say that digital video is part of our everyday life, with massive amounts of content being viewed and stored [34, 44].

While video content is fairly simple to produce, finding the desired information becomes a difficult challenge as video databases grow larger and larger. The most comfortable way for users to express their information needs remains a text-based approach on the basis of keywords. This, however, requires an *indexing* that links the video content in a database to semantic concepts (or *tags*) appearing in it, like objects ("airplane"), scene types ("cityscape"), and activities taking place ("interview"). The challenge of creating such an index has been referred to as the *semantic gap* [36], the discrepancy between a video's low-level content on the one hand and the viewer's high-level interpretation on the other.

So far, the only reliable bridge over the semantic gap remains human perception. This means that – to build an accurate textual index for video search – human operators are required to manually label video content with concepts appearing in it. For many large-scale practical applications, however, this approach is simply too time-consuming. As a scalable alternative to complement human labeling, *concept detection* systems have been developed that infer the presence of tags automatically from the content of a video [7, 6, 46, 49]. Though such detectors do not reach a precision comparable to human annotators, they have been demonstrated to be extraordinarily useful in a video search context [38].

While concept detection is considered an approach of high potential for video search and has been realized in several research prototypes [7, 6, 19], it has not been widely applied in practical large-scale settings yet. One reason for this is that the supervised machine learning techniques underlying concept detection require video content labeled with target concepts for training. Currently, this training information is acquired manually, i.e. human operators label data with respect to concept presence. The quality of the resulting training material is high in a sense that the annotated concepts are carefully selected with respect to feasibility and usefulness [27], that clear and restrictive definitions of concepts are predefined, and that fine-grain annotation is done on shot level.

On the downside, the effort associated with such a time-consuming acquisition restricts concept detection in several ways: first, it limits the number of concepts to be learned, such that the size of current detector vocabularies is far from optimal [17]. Second, detectors have been reported to overfit to small training sets and generalize poorly [51]. Third, keeping track of dynamic changes of users' information needs is infeasible as new concepts of interest emerge (such as "President Obama" or "Olympics 2008").

Given this scalability problem, the question arises whether explicit manual annotations – which are precise but difficult to acquire – can be substituted with weaker

[1] http://www.youtube.com, http://www.blinkx.com, http://vids.myspace.com

Fig. 1 Sample frames from YouTube clips tagged with "basketball". While some frames do show basketball (top), other *non-relevant* content is not visually related to the concept (bottom).

label information that can be obtained more easily (or is even freely available). One source of such information is *web video*, which is publicly available at a large scale from portals such as YouTube and comes with tags indicating the presence of concepts in a clip. If we could utilize this tag information as class labels in a concept learning framework, systems could automatically harvest training material from the web. This way, detectors could perform a more autonomous learning, scale up to thousands of concepts, and keep track of trends in user interest.

Two key aspects are important: first, the described label information is significantly easier to acquire at a large scale, as the annotations used have already been made by a large community of YouTube users. Second, labels are *weak*, i.e. content annotated with a target concept *may* show the concept but does not necessarily do so. An illustration is given in Fig. 1, which shows representative keyframes from web videos tagged with "basketball". While the concept is present in some frames, others are not visually related to it at all. We will refer to the first kind of frames as *relevant*, while calling the latter *non-relevant*.

It should be noted that non-relevant content can be caused by different reasons: for once, tags are coarse and indicate *that* a concept appears in a video, but not *when* it appears. A second reason is that labels are inherently *unreliable* - for example, the tag "Steven Spielberg" does not necessarily indicate that Steven Spielberg appears in a clip but might just hint to a news report on the Academy Awards.

In the following, we will refer to training content where positive labels are only coarse and unreliable indicators of concept presence as *weakly labeled*. Obviously, training a concept detection system on such data is a difficult challenge: typically, for each target concept a binary classification problem is cast of differentiating concept presence from concept absence, and a statistical model is learned from a set of labeled training samples (here, keyframes). When applying such supervised learning to web video, non-relevant content causes *false positives* in the training set, and it is to be expected (and will be demonstrated later) that concept detection performance degrades with increasing influence of non-relevant content.

In this chapter, this setup of training concept detectors on weakly labeled web video data is studied. We will show that the tag information associated with web

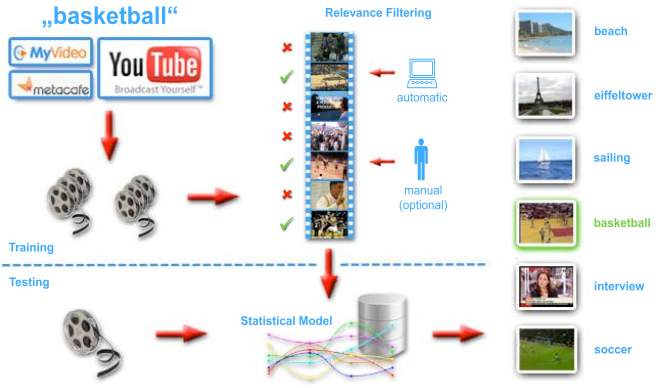


Fig. 2 Our concept detection system learning from web video. Clips downloaded from online platforms like YouTube are used for concept learning. Since such videos are only weakly labeled, relevance filtering identifies and discards non-relevant content. The resulting statistical model is applied to detect the learned concepts in previously unseen videos.

video is in fact unreliable, and that non-relevant material in the training set can degrade the performance of standard detectors severely (Sect. 3). To overcome this problem, we present a framework for learning from weakly labeled web video data called *relevance filtering* (Sect. 4). This probabilistic approach views the given labels as weak indicators of true latent class labels, which are inferred during concept detector training. This corresponds to a filtering of non-relevant material, which can be applied as a wrapper around well-known supervised learning techniques. Two such extensions are presented, one for a generative approach (kernel density estimation) and one for a discriminative one (support vector machines [SVMs]). It is shown in quantitative experiments (Sect. 5) that relevance filtering can successfully identify non-relevant content and give significant improvements over standard supervised learning.

As outlined so far, relevance filtering works without manual supervision and identifies non-relevant material automatically based on its distribution in feature space. Beyond this, we also demonstrate that the approach can be extended with *active learning*. In this framework, the system requests labels for a few samples from a user (Sect. 6). By selecting the most *informative* query samples, concept models can be improved further at moderate annotation cost.

Overall, our contributions constitute an approach for concept learning from web video. As illustrated in Fig. 2 for the concept “basketball”, concepts are learned by acquiring a raw dataset from web portals like YouTube. Relevance filtering - which can be optionally enriched with thoroughly selected manual annotations – is used for a joint model learning and content filtering. The resulting concept detector can then be applied to previously unseen videos.

2 Related Work

In this section, related work in the context of learning from weakly labeled videos is outlined. We will omit a general review of concept detection (for more information on this topic, please refer to the literature [17, 27, 35, 39, 46, 49] or to a recent survey [37]) and focus on the aspect of weak label information instead. This setup is viewed from two different perspectives – the first tackles the machine learning aspects of the problem and discusses *semi-supervised* learning (Sect. 2.1). The second perspective is domain-specific and focuses on learning from noisy image and video content. (Sect. 2.2 and Sect. 2.3).

2.1 Semi-supervised Learning

Semi-supervised learning refers to a class of machine learning techniques designed for dealing with incomplete label information. In this setup, a (usually small) set of training samples $X_L = \{x_1, .., x_l\}$ with class labels $y_1, .., y_l$ is assumed to be given. A second (usually large) set of samples $X_U = \{x_{l+1}, .., X_N\}$ is available as well, but the associated labels are unknown (or *latent*). Semi-supervised learning can be seen as a borderline case between supervised learning (where all training data is labeled, i.e. $X_U = \varnothing$) and unsupervised learning (where no labels are given at all, i.e. $X_L = \varnothing$).

Semi-supervised learning is attractive in application areas where lots of unlabeled training samples can be obtained easily, but the acquisition of label information is associated with considerable effort (as it is the case for concept detection). While supervised methods in such setups learn only from a small set of labeled examples, semi-supervised techniques can exploit further information in form of unlabeled content (which can be viewed as evidence on the overall sample distribution $p(x)$). To leverage this information, a variety of strategies has been proposed (for a survey of the field, please refer to [9, 52]).

One simple semi-supervised learning strategy is to infer the labels of unlabeled samples and treat the resulting labeled samples in a supervised framework. This **self-training** is an iterative wrapper around a base classifier, in which samples are iteratively classified and the training set is automatically expanded with a selection of the newly labeled data (usually the ones for which the classifier is most confident). As an extension, **co-training** [5] has been suggested, where multiple classifiers are trained on different feature subsets of the data and "teach" each other.

Another technique called **Expectation Maximization** (EM) [11] casts semi-supervised learning in a probabilistic setting. Model parameters are fitted by maximizing the data likelihood, whereas a marginalization over latent class labels is done. This leads to a search in parameter space in which alternately label posteriors are inferred, and based on these estimates the system parameters are updated.

An alternative strategy follows from the insight that decision boundaries are usually situated in low-density areas of $p(x)$, and should correspondingly lie far away from data points. If cast in a maximum-margin setting, this approach leads to

transductive SVMs [20], which estimate unseen labels together with a separating hyperplane. Finally, **graph-based** methods have been proposed, which view samples as nodes in a graph and estimate node labels via label propagation or regularization [9, Ch. 11].

2.2 Learning from Web Images

Web image content (as it can be acquired via text-based image search engines or from portals like Flickr) is a data source similar to web video content in a sense that label information is weak and large parts of the retrieved training data may be junk. For Google Image Search, Fergus et al. [14] have reported a label precision between 18% and 77% for 7 object categories. Schroff et al. [31] have measured an average precision of 39% over 18 categories.

To overcome this label weakness, a variety of approaches have been suggested [4, 31, 40, 50] targeted at a content-based refinement of raw web image sets. Usually, a three-step procedure is applied: first, a raw set of images is acquired from the web. Second, a subset of "good" candidate images for concept presence is selected, which can be done using manual annotation [4] or an analysis of text and meta-data surrounding the image [31, 50]. Finally, a statistical model of concept presence (a support vector machine [31], a region-level annotation model [3], or a mining procedure based on a saliency measure [40]) is trained on the refined image set and used to re-rank all web images. Similar to the work in this chapter, this approach is targeted at a refinement of training sets. However, it does not cover the actual learning of concept models. In contrast to this, we tackle a joint training set refinement and model learning, and our focus is on the performance of the resulting detectors.

Other related work follows an approach more similar to ours and combines training data refinement with model learning. Fergus et al. [14] learn visual models of object categories from Google's image search using a topic model. The key assumption of the approach is that images showing the target object accumulate in a single cluster (or *topic*), which is then used for object recognition. The OPTIMOL system by Li et al. [24] follows an incremental approach instead: a training set is agglomerated while learning an object model in parallel. The approach works in a self-training fashion, starting from an initial highly accurate set of sample images. Iteratively, a topic model is trained and the pool of training data is expanded using a Bayesian decision. The approach has been demonstrated to outperform Fergus' system [14]. Yet, a problem remains in the initialization with *good* training samples, which has been reported to be a crucial factor [26].

In contrast to the incremental OPTIMOL system [24], Wnuk and Soatto [48] follow a filtering approach. A measure of *strangeness* is defined based on a nearest neighbor analysis in feature space, and content with high strangeness values is filtered out. We follow this general idea and extend it to a probabilistic setting called *relevance filtering*, which can be integrated with a variety of supervised learning techniques (Sect. 4).

2.3 Learning from Weakly Labeled Videos

Only few previous contributions have been made with respect to learning from video data with weak labels. Gargi and Yagnik [15] point out that label information in videos may be coarse, which they refer to as the *label resolution problem*. They rely on a feature selection using Adaboost to achieve robustness with respect to non-relevant content. Gu et al. [16] cast concept detection as a multiple instance problem and propose to adapt the kernel function in an SVM framework. Both methods, however, do not model non-relevant content explicitly.

A contribution closer to the one presented here has been made by Wang et al. [47], who study concept detection in a semi-supervised setup (where only a few initial labeled samples are given). A kernel density model is extended such that the contribution of each training sample is weighted by its class posterior, and an iterative fitting algorithm is proposed to match unlabeled content to classes. Performance improvements over supervised learning from a few initial samples are demonstrated.

In previous work, we have already addressed the problem of concept learning from weakly labeled web videos [43] and proposed a model similar to the one by Wang et al. In this chapter, we will extend this idea further and demonstrate that it can be integrated with a variety of supervised learning techniques.

3 Concept Learning on Web Video

In a first experiment, we study web video as a data source for concept detector training. First, we present manual annotation results demonstrating that the tag information coming with web videos is only an unreliable indicator of concept presence, such that web video training sets contain significant amounts of non-relevant content (Sect. 3.1). Second, we study how standard concept detection techniques are influenced by this non-relevant content (Sect. 3.2) and show that significant performance loss is to be expected.

3.1 The Precision of Web Video Tags

In a first experiment, we study the precision of web video tags when used as class labels in a concept learning framework. Therefore, keyframes are sampled from YouTube videos and serve as positive training samples (if the video is tagged with the target concept) or as negative ones (if it is not). Since tags are coarse and unreliable, we expect that only a certain fraction of positive training samples is truly relevant. This *relevance fraction* is denoted with α in the following:

$$\alpha := \frac{\text{number of positive training samples showing the concept}}{\text{number of positive training samples}}$$

α can be seen as the *precision* of label information. It is close to 100% if annotations are accurate (as it is usually assumed in concept detector training). For web video,

Table 1 A manual annotation of training material downloaded from YouTube indicates that the label precision α of web video training sets is low (in most cases below 50%).

Concept	Raw Query*	Refined Query*	Concept	Raw Query*	Refined Query*
basketball	20.5	40.6	helicopter	14.6	38.1
beach	15.6	44.3	sailing	16.4	26.2
cats	47.6	50.1	soccer	25.3	43.7
desert	11.4	19.0	swimming	23.4	60.0
eiffeltower	21.4	39.7	tank	14.5	24.3
			average	**21.1**	**38.6**

* *values indicate fraction of relevant training content α (%)*

however, we expect α to be significantly lower and also to vary between concepts: while for some concepts high-quality training sets may be obtained, others may be used as tags often but *appear* only infrequently.

To get a deeper insight into the quality of tags as training annotations, we conducted an annotation experiment. Ten test concepts were chosen from the YouTube-22concepts dataset[2] with respect to a good coverage of concepts, including objects ("cats", "eiffeltower"), locations ("beach", "desert"), and sports ("basketball", "golf"). For each concept, 1,000 clips were downloaded from YouTube using two different queries to the YouTube API (the overall length of the dataset is about 100 hours):

1. **Raw Queries:** The query consists of a single tag describing the concept, like "beach". This may be the case if a concept detection system is given only a vocabulary of tags and crawls YouTube fully automatically for training material.
2. **Refined Queries:** Querying the YouTube API with a single tag must be expected to give very noisy results. For example, the query "beach" does not only return scenes of beaches, but also music videos by the "Beach Boys" and scenes of Daytona Beach. While these may be valid annotations to the video owner, they must be considered non-relevant when it comes to learning a specific concept like beach sceneries. Therefore, two refinements are made. First, the fact is used that videos at YouTube are organized in categories like "Pets&Animals" or "Autos&Vehicles". The download is restricted to a canonical category (like "Travel&Places" for "beach", which excludes music videos). Second, queries are refined according to a brief analysis of the first YouTube results page. For example, the query "beach" is replaced with "walk on the beach", which usually rules out city names.

For each concept, a canonical definition was formulated (which can be found in the Appendix and is publicly available[3]), and over 1,000 keyframes sampled from YouTube clips tagged with the concepts were manually assessed according

[2] http://www.dfki.uni-kl.de/~ulges/youtube-22concepts/
[3] http://www.dfki.uni-kl.de/~ulges/VSM-testconcepts/

to these definitions. Results of the annotation process are given in Table 1. They indicate that YouTube labels are in fact weak – the downloaded content contains significant fractions (in most cases more than 50%) of non-relevant material. It can also be seen that α is particularly low for raw queries (21.1% on average), whereas a manual refinement leads to better results (38.6%). Finally, the percentage of relevant material varies strongly between concepts: for example, the label precision ranges from 26.2% ("sailing") to 60% ("swimming") for refined queries.

These results correspond to similar observations made previously for the image domain: for datasets based on image search, a precision of 39% has been reported for object category recognition [31]. For Flickr images, Kennedy et al. [21] have observed an accuracy of 50% for the domain of New York sights. These precisions are slightly higher than our results, which can be attributed to the fact that for video the *coarseness* of labels in the time domain poses an additional problem.

Yet, it should be noted that this does not necessarily mean that video is a worse source for visual learning than images. Rather, it is to be expected that the preferred training modality depends on the concepts: a wide variety of concepts are action-related or video-specific (for example, think of "soccer" or "interview"). For such concepts, video-based training material will be more appropriate than images.

3.2 Concept Learning from Web Video

In the last section, YouTube datasets have been demonstrated to contain significant amounts of non-relevant content. The next key question is how this influences concept detectors when trained on web video using standard methods. Intuitively, it can be expected that material similar to false positives in the training set will be classified incorrectly, such that detection performance degrades. This is validated in the following experiment.

Data. The experiment is conducted on the same YouTube data and annotations used in the last section. According to our ground truth labels, we randomly compiled training sets of varying noise ratio α as illustrated for the concept "desert" and $\alpha = 60\%$ in Fig. 3: negative samples – which can be obtained easily from videos not tagged with the concept – are drawn for the background class. Positive samples consist of 60% true positives (which were manually assessed to show the target concept) and 40% non-relevant frames, which were again drawn randomly from YouTube videos *not* tagged with the concept. Further, test sets with known ground truth labels were sampled:

for $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5, 0.7, 1.0\}$:

1. *// sample training set*
   - sample 1000 non-relevant frames with label -1
   - sample $(1-\alpha) \cdot 500$ non-relevant frames with label 1 ("false positives")
   - sample $\alpha \cdot 500$ relevant frames with label 1 ("true positives")
2. *// sample test set*
   - sample 500 relevant frames with label 1
   - sample 1500 non-relevant frames with label -1

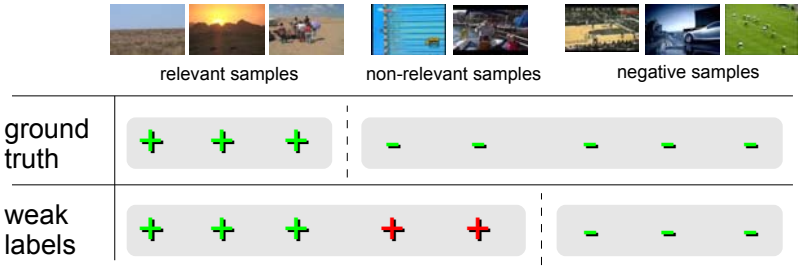


Fig. 3 Sampling a random training set for the concept "desert" and $\alpha = 60\%$. Non-desert content models the background class (right). Positive samples are mixed of 60% desert frames and 40% non-desert frames. The latter are incorrectly labeled as relevant. This weakly labeled setup (top) is compared with learning from correct labels (bottom).

To avoid overfitting, it was made sure that no material from the same video clip was assigned to training and testing at the same time. Also, it should be noted that only the training set is weakly labeled, while the test set uses ground truth to assure a precise evaluation.

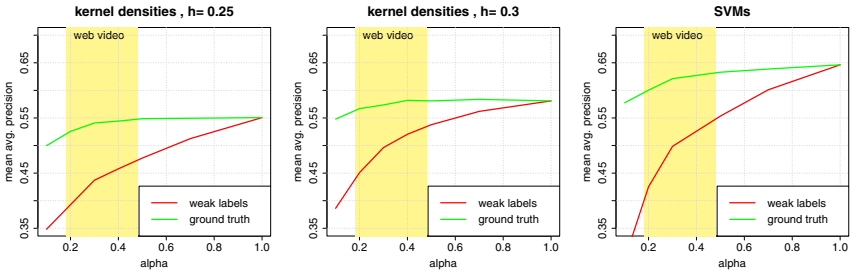
Features. Frames are represented by bag-of-visual-words features [33], which have previously been demonstrated to give a good performance in a variety of recognition tasks including concept detection [45] or object category recognition [13]. For each frame, a feature is extracted by regularly sampling about 3,600 SIFT patches [25] at several scales. These are matched with a 2,000-dimensional visual codebook learned previously on a large dataset of 81 concepts. A dimensionality reduction is applied to the resulting visual word histograms using PLSA [18], obtaining a 64-dimensional feature vector per frame. This dimensionality reduction is done for efficiency purposes and has previously been validated to give comparable results to the high-dimensional visual word histograms.

Models. Tests were run for two standard supervised learning approaches: a generative model (kernel densities) [12, Ch. 4] and a discriminative one (SVMs) [30]. Given training samples $x_1,..,x_n$ with labels $y_1,..,y_n \in \{-1,1\}$, the **kernel density** approach models class-conditional densities of concept presence and absence:

$$p^1(x) = \frac{1}{Z} \sum_{i:y_i=1} K_h(x;x_i),$$
$$p^0(x) = \frac{1}{Z'} \sum_{i:y_i=-1} K_h(x;x_i). \tag{1}$$

A test frame x is scored using Bayes' rule (the class prior – which does not influence the ranking of test items – is assumed to be uniform):

$$P(y=1|x) = \frac{p^1(x)}{p^1(x)+p^0(x)}$$



(a) (b) (c)

Fig. 4 Comparing concept detection when trained on ground truth labels (green) and on weak labels (red). The mean average precision over all 10 test concepts is plotted against the label precision α. The two results on the left represent the kernel density system for bandwidths 0.25 (a) and 0.3 (b), the result on the right is for SVMs (c).

As a kernel function, the well-known Epanechnikov kernel with Euclidean distance function is used:

$$K_h(x;x') = \frac{3}{4} \cdot \left(1 - \frac{||x-x'||^2}{h^2}\right) \cdot 1_{(||x-x'|| \leq h)}$$

This choice is made mostly for efficiency reasons: as the Epanechnikov kernel has only local support, it can be evaluated efficiently when combined with methods for fast nearest neighbor search such as kd-trees [28]. The kernel bandwidth h is a free parameter of the system. It has been reported previously to have a strong influence on the resulting kernel densities [42], with high values of h leading to a smoother density in feature space. For the experiment presented here, several choices of the kernel bandwidth $h \in \{0.225, 0.25, 0.275, 0.3, 0.325\}$ were tested, and results are reported for a representative low value ($h = 0.25$) and a high one ($h = 0.3$).

As a discriminative approach, **Support Vector Machines** (SVMs) [30] were tested, which are a popular choice for concept detection [45, 49, 51]. An RBF kernel is used, whereas the smoothness σ and the cost C are evaluated in a grid search cross-validation (for more information on these parameters, please refer to [8]). For efficiency reasons, no complete search was done for each run, but the values $C = 5$ and $\sigma = 25$ are used, which were validated to give stable good results. SVM scores were mapped to class posterior estimates using the LIBSVM implementation [8].

Results. Both systems – kernel densities and SVMs – were tested in two setups (as illustrated in Fig. 3):

- **weak labels:** this setup corresponds to the practical situation of concept learning from weakly labeled web content. Only a fraction α of positive training samples is truly related to the concept.
- **ground truth:** this is a control run with an oracle providing ground truth labels (which are not available in practice). The run indicates how well concept learning would work if non-relevant content content was filtered out.

As a performance measure, average precision is used, i.e. the area under the recall-precision curve over the ranked list of test frames. By averaging over all 10 test concepts, we obtain the *mean average precision* (MAP), which is a standard choice for concept detector evaluation [22]. Quantitative results are given in Fig. 4 (all values were obtained by averaging over 5 runs). Both systems were tested for varying fractions of relevant content α, and the system performance on the test data is plotted against α.

We now study how concept detection behaves when varying the noise level in the training set. A first observation is that the influence of non-relevant material on the oracle-based control run (green) is negligible, such that performance remains almost constant when varying α. This is intuitively correct, since non-relevant samples (which become more frequent with lower values of α) are assigned their correct negative labels. A lower performance for low relevance fractions $\alpha \approx 10\%$ can be attributed to a lower absolute number of positive training samples.

When comparing the ground truth runs with systems trained on weakly labeled data, we can see that in the absence of noise ($\alpha = 1$) both systems give the same performance (which is trivial, as no false positives exist and training labels are identical otherwise). However, when decreasing α the performance of the weakly supervised system degrades significantly: for example, for training sets with 70% non-relevant material ($\alpha = 0.3$) and a bandwidth of 0.25, the kernel density estimation trained on weakly labeled data gives a performance of 43.7%, while training on the correct labels gives 54.1%. The more noise in the training data, the stronger the gap between the weakly supervised run and the control run becomes. This observation can be made for the generative model (Fig. 4(b) and 4(a)) as well as the discriminative one (Fig. 4(c)).

We can now match these results with the annotations in Table 1, which indicate that the label precision of web video data is typically in the range of 20% (for raw queries) to 50% (for refined queries). This range is highlighted in yellow in all plots. If we focus on this area, we can see that performance degradation due to weak labels is significant, ranging from 4% to 19%.

4 Relevance Filtering

Our results in the last experiment indicate that concept learning on web video could be improved significantly if we were able to filter out non-relevant content in the training set. In this section, we follow this strategy and present a framework in which the statistical models underlying concept detection are adapted such that non-relevant content is automatically identified and filtered during training. The approach is based on a formulation of concept learning as a *weakly supervised learning* problem, in which the given labels (here: YouTube tags) are only weak indicators of true class labels. These true class labels are inferred during concept learning.

This approach will be referred to as *relevance filtering* in the following. Its core assumption is that relevant content forms clusters in feature space, while non-relevant material comes as outliers that can be identified and relabeled. The approach can be combined with a variety of well-known supervised learning

Table 2 An overview of the basic concepts and notation used in Sect. 4. Concept detection is viewed as a *weakly supervised learning problem*, in which given labels are only weak indicators of true, latent ones.

x	feature vector representing a test keyframe
$y \in \{-1,1\}$	absence/presence of target concept in x
$P(y=1\|x)$	keyframe score (to be estimated)
$x_1,..,x_n$	feature vectors representing training frames
$\tilde{y}_1,...,\tilde{y}_n \in \{-1,1\}$	weak labels of concept presence in training frames (observed)
$y_1,.,y_n \in \{-1,1\}$	actual absence/presence of target concept in training frames (unknown)
$\beta_j := P(y_i=1\|x_i,\tilde{y}_i)$	*relevance score*: the probability of a training frame being relevant (unknown)
$\alpha := P(y_i=1\|\tilde{y}_i=1)$	*relevance prior*: assumed fraction of truly relevant training frames among potentially relevant ones

techniques. Two such combinations are presented for the models used in the last experiment (namely, kernel density estimation and Support Vector Machines).

4.1 Basic Concepts

In the following, a video is represented by keyframes, such that concept detection is effectively conducted on keyframe level. Each frame is associated with a feature vector $x \in \mathbb{R}^d$. The presence of the target concept is denoted with a label y, such that $y=1$ indicates concept presence and $y=-1$ concept absence. The goal of concept detection is to estimate the concept *score* $P(y=1|x)$. Training data is also represented by keyframes (or associated features) $x_1,..,x_n \in \mathbb{R}^d$. For each training frame x_i, a weak indicator of concept presence is given that tells us whether the concept *may* appear in the frame (in practice, this is a tag given to the corresponding web video clip). This information is denoted by a *weak label* $\tilde{y}_i \in \{-1,1\}$. The *actual* presence of the target concept, however, is latent. It is denoted with $y_i \in \{-1,1\}$. Concept detection is now cast as a binary classification problem (see Table 2 for an overview of the notation used):

Definition 1. ***Weakly Labeled Binary Classification Problem***
Given training data in form of samples $x_1,..,x_n \in \mathbb{R}^d$ with labels $\tilde{y}_1,..,\tilde{y}_n \in \{-1,1\}$, learn a scoring function $\phi : \mathbb{R}^d \to [0,1]$ such that $\phi(x) \approx P(y=1|x)$. Thereby, training labels are assumed to be *weak* indicators of *true* labels $y_1,..,y_n$ such that:

1. If the weak label is negative ($\tilde{y}_i=-1$), the true label is negative as well ($y_i=-1$).
2. If the weak label is positive ($\tilde{y}_i=1$), the sample *may* belong to the positive class, but does not necessarily do so, i.e. the true label y_i is unknown.
3. A prior for weakly labeled samples being truly positive is assumed to be given, which is denoted with $\alpha := P(y_i=1|\tilde{y}_i=1)$.

In this setup, true latent class labels are separated from given ones. They can thus be estimated during learning, such that the model ϕ is effectively trained on the

estimated true class labels instead of the weak labels. It should also be noted that – while we model false positives (i.e. it is possible that $\tilde{y}_i = 1$ and $y_i = -1$) – false negatives ($\tilde{y}_i = -1$ and $y_i = 1$) are not taken into account. Strictly speaking, this is not true (for example, there might be videos showing "basketball" that the user has simply forgotten to label). According to our observations made on web video, however, the percentage of such false negatives is negligible compared to the one of false positives.

Let us compare the weakly labeled classification problem with other learning setups. First, when compared with standard *supervised learning*, two key differences are that only weak indicators of the true class labels are given, and that an additional assumption is made (in form of α) on how much of the weakly labeled material does in fact show the target concept. Particularly, the supervised setting can be seen as a special case of the weakly supervised one, where α equals 100%.

Compared with the *semi-supervised learning* setup, the above definition can be seen as a degenerate special case. This is because weakly labeled samples $\{x_i : \tilde{y}_i = 1\}$ can be viewed as *unlabeled* (their true label y_i is not known). This leads to an extremely *imbalanced* problem: while semi-supervised learning usually assumes a few initial labels of either class to be given, in our setup we are confronted with many samples from class -1 (simply because content not labeled with a concept can be obtained easily) but no sample of class 1 (since indicators of concept presence are weak). This renders a straightforward application of many semi-supervised algorithms impossible, since these would require an initialization with a few reliable samples of both classes.

Finally, the weakly labeled learning setup strongly resembles several approaches for visual learning from noisy image sources like Google's image search [14, 24, 48]. The work in this chapter follows a strategy similar to these approaches (particularly to the one by Wnuk and Soatto [48], who also propose a distribution-based filtering of training sets). Yet, several differences remain. First (and obviously), the web video domain addressed here differs from images delivered by web search engines. Second (and more importantly), we do not cover a single statistical model, but view relevance filtering as a wrapper than can be applied around a variety of supervised learning techniques. For both a generative and a discriminative base model, relevance filtering extensions will be presented in the following.

4.2 The Generative Case: Kernel Density Estimation

In this section, relevance modelling is used as a wrapper around a generative model for concept detection, namely *kernel density estimation* [12, Ch. 4]. Thereby, the relevance of training content is modeled as a latent random variable that is inferred during the learning procedure.

Class-conditional Densities and Scoring. Class-conditional densities of relevant and non-relevant content are modeled by the following weighted kernel densities p^1_β and p^0_β:

$$p^1_\beta(x) = \frac{1}{Z} \cdot \sum_{i=1}^{n} \beta_i \cdot K_h(x; x_i),$$
$$p^0_\beta(x) = \frac{1}{Z'} \cdot \sum_{i=1}^{n} (1-\beta_i) \cdot K_h(x; x_i), \tag{2}$$

where $Z = \sum_i \beta_i$ and $Z' = n - Z$ are normalization constants. Compared to the fully supervised setup from Equation (1), the key difference is that p^1 and p^0 are now parameterized by a vector $\beta = (\beta_1, .., \beta_n)$. This vector consists of *relevance scores* $\beta_i := P(y_i|\tilde{y}_i, x_i)$, which means that for p^1_β each training sample is weighted by its probability of being relevant (correspondingly, for the distribution of non-relevant content p^0_β this weight is $1 - \beta_i$). Consequently, if a training sample is likely to be relevant, it has a strong influence on the distribution of relevant samples p^1_β but low influence on p^0_β. In this way, the uncertainty of label information is taken into account (a similar model has been used in a semi-supervised setup before [47]).

Note that if we set the relevance scores according to the weak labels:

$$\beta_i = \begin{cases} 1, & \tilde{y}_i = 1 \\ 0, & \tilde{y}_i = -1 \end{cases}$$

the system degenerates to the standard supervised case (Equation (1)) in which all positively labeled samples are assumed to be relevant.

Training. To compute the class-conditional densities p^1_β and p^0_β, the vector of relevance scores β must be inferred in system training. The input consists of features $x_1, .., x_n$, weak labels $\tilde{y}_1, .., \tilde{y}_n$, and the relevance prior α. For each training frame x_i, three situations may occur:

1. $\tilde{y}_i = -1$ (*negative*): if x_i is *not* labeled with the concept, it is assumed to be non-relevant, i.e. $\beta_i = 0$.
2. $\tilde{y}_i = y_i = 1$ (*true positive*): x_i is labeled with the concept and is in fact relevant. Accordingly, β_i should be high.
3. $\tilde{y}_i = 1, y_i = -1$ (*false positive*): x_i is labeled with the concept but is not relevant. Such noise samples may occur, since labels $\tilde{y}_i$ are only weak indicators of concept presence. For them, β_i should be low.

Let us assume that m training samples are weakly labeled with the concept, and that training samples are sorted such that $\tilde{y}_1 = .. = \tilde{y}_m = 1$ and $\tilde{y}_{m+1} = .. = \tilde{y}_n = -1$. While we know that $\beta_{m+1} = .. = \beta_n = 0$, the relevance scores $\beta_1, ., , \beta_m$ need to be estimated, i.e. training must divide potentially relevant frames into actually relevant ones and non-relevant ones. Therefore, the parameter vector β is restricted to the non-zero entries $\beta = (\beta_1, .., \beta_m)$.

Our strategy to estimate β is based on a simple fixpoint iteration in parameter space. First, relevance scores are initialized with the relevance prior: $\beta^0 = (\alpha, .., \alpha)$. Then, the parameter vector β^k is iteratively updated to a new version β^{k+1} by plugging the current parameter estimate β^k into the class-conditional densities $p^1_{\beta^k}$ and

$p^0_{\beta^k}$ (Equation (2)). From these densities, new estimates of relevance scores can be obtained using Bayes' rule:

$$\begin{aligned}\beta_i^{k+1} &:= P(y_i = 1|x_i, \tilde{y}_i = 1) \\ &= \frac{p(y_i = 1, x_i|\tilde{y}_i = 1)}{p(y_i = 1, x_i|\tilde{y}_i = 1) + p(y_i = -1, x_i|\tilde{y}_i = 1)} \\ &\approx \frac{P(y_i = 1|\tilde{y}_i = 1) \cdot p(x_i|y_i = 1)}{P(y_i = 1|\tilde{y}_i = 1) \cdot p(x_i|y_i = 1) + P(y_i = -1|\tilde{y}_i = 1) \cdot p(x_i|y_i = -1)} \\ &\approx \frac{\alpha \cdot p^1_{\beta^k}(x_i)}{\alpha \cdot p^1_{\beta^k}(x_i) + (1-\alpha) \cdot p^0_{\beta^k}(x_i)}\end{aligned}$$

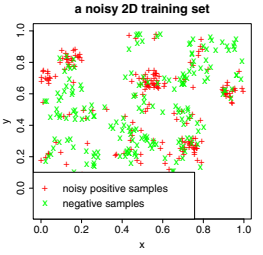
This process is repeated for a fixed number of iterations. Intuitively, the algorithm identifies regions in feature space where positively labeled samples concentrate and assigns high relevance scores to them. Outliers similar to negative content are given low relevance scores. The approach resembles the well-known Expectation Maximization (EM) scheme [11], which maximizes the data likelihood by alternating so-called "E" steps (in which posteriors for latent variables are estimated) and "M" steps (in which system parameters are updated according to this knowledge by maximizing the expected log-likelihood of training data). If we compare the EM scheme to the fixpoint iteration used here, the relevance scores β_i resemble posteriors for latent variables in the EM scenario (namely the true labels y_i). However, since the parameters of the class-conditional densities are equal to the relevance scores β_i and the framework is non-parametric otherwise, no "M" step is required.

The approach is also similar to the training procedure used by Wang et al. [47], but the system is constrained in a different way: while Wang et al. addressed a semi-supervised setup – where initial reliable training samples for all classes are available – we cannot rely on such information in our weakly supervised setup. Instead, we constrain the system with a certain prior of the label precision α. Note that if we choose this *relevance prior* to be $\alpha = 1$, it follows that $\beta_1 = \beta_2 = .. = \beta_m = 1$, such that the model degenerates to the supervised case (Equation (1)).

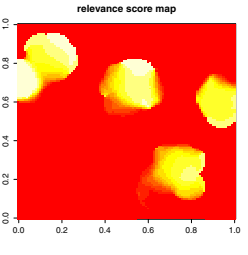
A Sample Problem In the following, an illustration of relevance filtering for kernel densities is given in a small experiment. A two-dimensional weakly labeled dataset is generated such that samples from the positive class contain a certain amount of incorrectly labeled false positives. For two classes (representing concept presence and absence), random prototypes are drawn from $[0,1]^2$. Samples are drawn from the surrounding of these prototypes according to kernel densities p^1 and p^0 with bandwidth $h = 0.05$, obtaining a training set with 200 noisy positive samples (which are again compiled of positive and negative content):

$$x_1, .., x_{200} \sim \alpha \cdot p^1 + (1-\alpha) \cdot p^0.$$

The fraction of relevant samples is varied such that $\alpha \in \{0.2,\ 0.6,\ 1.0\}$, i.e. we use one clean training set without false positives ($\alpha = 1.0$), one with moderate noise



(a)



(b)

error rate (%)			
	$\hat{\alpha}$		
α	0.2	0.6	1.0
0.2	27.7	32.2	35.5
0.6	13.0	12.1	15.8
1.0	11.0	10.9	4.1

(c)

Fig. 5 (a) A 2D sample training set. Positive samples (red) concentrate in 5 peaks, but contain 40% outliers. (b) A learned relevance map shows that relevant content is identified at the correct five peaks. (c) Classification error rates on synthetic sample sets, whereas the fraction of relevant content α and its estimate $\hat{\alpha}$ are varied. A choice of $\hat{\alpha} \approx \alpha$ gives the best classification results.

($\alpha = 0.6$) and one with lots of noisy samples ($\alpha = 0.2$). For each training set, negative training samples are drawn from $p^0(x)$ and added. Finally, a test set of equal size is sampled from the same distribution as the training set. This experiment is repeated 100 times, whereas for each run the relevance filtering framework is tested with a relevance prior of $\hat{\alpha} \in \{0.2, 0.6, 1.0\}$. Note that the true relevance fraction is unknown in practice, which is why we distinguish between the true value α and the relevance prior we expect (which is denoted with $\hat{\alpha}$ in the following).

A typical dataset used in this experiment is illustrated in Fig. 5(a). It can be seen that positive samples (red) concentrate near five prototypes of class 1, but many red outliers (false positives) occur. The result of relevance filtering is also illustrated: a *relevance map* plots the relevance score β over feature space (Fig. 5(b)). It can be seen that high relevance scores are assigned to samples accumulating near the five prototypes, while outliers close to negative samples are assigned low relevance scores. Classification results when applying the kernel density model with relevance filtering are reported in Table 5(c). Two observations can be made: first – and not surprisingly – the overall error rate of classification increases with the amount of noise material in the training set. The second observation is that the actual noise level and the optimal choice of the relevance prior are correlated, i.e. the lowest error rate is achieved for $\hat{\alpha} \approx \alpha$. For example, for the clean training set ($\alpha = 1$) the supervised system ($\hat{\alpha} = 1$) performs best, while for $\alpha = 0.2$ the best performance is achieved for $\hat{\alpha} = 0.2$. Generally, this result indicates that relevance filtering can improve kernel density classification on weakly labeled training sets.

4.3 The Discriminative Case: Support Vector Machines

While in the last section a generative technique was adapted for weakly labeled concept detection, a similar extension will be presented for discriminative models in the following. The approach can be applied as a wrapper around a variety of

Table 3 Weakly Supervised Discriminative Training: Samples are iteratively refined in a self-training fashion by learning a discriminative classifier, scoring training content, and relabeling the samples most likely to be false positives.

1. for i=1,.,n: set $\beta_i = \begin{cases} 1, & \tilde{y}_i = 1 \\ 0, & \tilde{y}_i = -1 \end{cases}$
2. randomly split $X = \{x_1, .., .x_n\}$ into five folds $X_1, .., X_5$
3. until $\frac{1}{p}\sum_{i=1}^{p} \beta_i \leq \alpha$:
   - for $k = 1, .., 5$:
     - train a classifier on $X \setminus X_k$
     - apply the classifier to X_k, obtaining *scores* σ
     - for the N_f samples $x_i \in X_k$ with $\beta_i = 1$ and lowest scores $\sigma(x_i)$: set $\beta_i = 0$

discriminative base classifiers. The only requirement on the base model is that it delivers a posterior-like *score* σ. As a sample classifier, SVMs are used (which can be considered a standard choice for concept detection [45, 49, 51]).

The basic idea of relevance filtering for discriminative methods is similar to a semi-supervised self-training but works in a filtering fashion instead of an incremental one: iteratively, the base classifier is trained and used to identify false positives in the training set. Samples that the classifier identifies as most likely to be false positives are relabeled (i.e., their relevance scores β_i are set from 1 to 0), and training is repeated. This way, the weakly labeled positive samples are iteratively filtered and refined. The whole process is repeated until the estimated relevance prior $\frac{1}{p}\sum_i \beta_i$ (which constantly decreases due to relabeling) reaches the expected relevance prior α. The whole training procedure is outlined in Table 3 (note that filtering is done in a cross-validation fashion to avoid overfitting).

Let us compare the approach with the generative relevance filtering from the last section. Generally, both techniques follow the same idea, namely to estimate the relevance of training content using the distribution in feature space and a relevance prior. However, two key differences can be identified. First, while the generative approach relied entirely on the distribution of content in feature space, the discriminative technique involves a classification method, such that the quality of filtering results is inherently bound to the classifier used. Second, the discriminative relevance filtering approach is not probabilistic: the scores σ used for filtering may be interpretable as relevance posteriors but do not necessarily have to be. Also, no soft assignment is used (as for kernel densities), but a complete relabeling of samples from the positive to the negative class takes place.

5 Experiments with Automatic Relevance Filtering

In the last section, relevance filtering has been proposed as a strategy to overcome label unreliability in concept detection training sets, and based on this idea extensions of two standard techniques (kernel densities and SVMs) have been presented.

In practice, however, such an automatic filtering – which is entirely based on the distribution of content in feature space – is not 100% accurate. Therefore, we need to investigate how well relevant content be separated from non-relevant one, and whether the performance of concept detection can be improved this way. In the following, it is demonstrated that the filtering of non-relevant content is possible (though far from perfect), and performance improvements of up to 9% compared with an equivalent supervised system are validated when false positives are drawn from an overall "world" distribution (Sect. 5.1). After this, the relevance filtering framework is trained on raw web video content downloaded from YouTube (where non-relevant material is correlated with the concept), and it is shown that relevance filtering still gives performance improvements in the range of $2-5\%$ over the supervised case (Sect. 5.2).

5.1 Controlled Setup

The purpose of this experiment is to study relevance filtering in a controlled scenario with known relevance fraction α. The setup is almost identical to the one used in Sect. 3: the same randomly sampled training sets and test sets are used, results are averaged over 5 runs, the feature representation remains the same (visual words, followed by a dimensionality reduction using PLSA), and the same statistical models are tested (namely kernel density estimation and Support Vector Machines). The only difference is that – besides the control runs used in Sect. 3.2 – additional results for relevance filtering extensions are presented. The following approaches are tested:

1. **ground truth**: the control run from Sect. 3.2 trained on ground truth labels.
2. **weak labels**: supervised learning from Sect. 3.2 trained on weak labels.
3. **relevance filtering – kernel densities**: the relevance filtering extension of the generative kernel density approach from Sect. 4.2. The number of training iterations is set to 100. The relevance prior is set to the correct fraction of relevant material, i.e. $\hat{\alpha} := \alpha$ (the behavior when varying this parameter will be studied later). The run appears in Figs. 7(a) and 7(b).
4. **relevance filtering – SVMs**: the relevance filtering extension of the discriminative approach from Sect. 4.3 using SVMs as base classifiers. Ten false positives are filtered in each training iteration. The same smoothness parameter $\sigma = 25$ and cost parameter $C = 5$ are used as in Sect. 3. The run appears in Fig. 7(c). Again, we set $\hat{\alpha} := \alpha$.

We first visualize the effects of relevance filtering in Fig. 6 to find out what content is actually identified as non-relevant by the system. Positive training frames are ranked by their score β_i, and the images with highest scores and lowest scores are displayed in Fig. 6 (a training set with $\alpha = 0.3$ was used, a bandwidth of 0.275, and a relevance prior of 0.3). At the top, we see the content identified to be most relevant, i.e. the highest scores β were assigned. Below this, material is illustrated that was labeled with the concept but was identified to be non-relevant by our system. Obviously, the content identified as relevant is in fact very likely to be visually

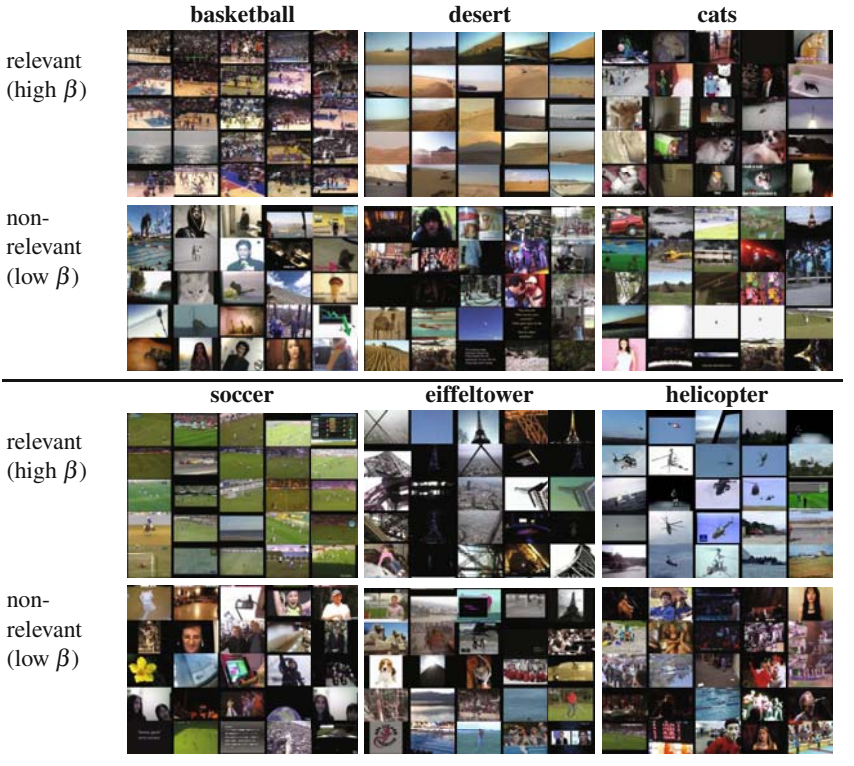
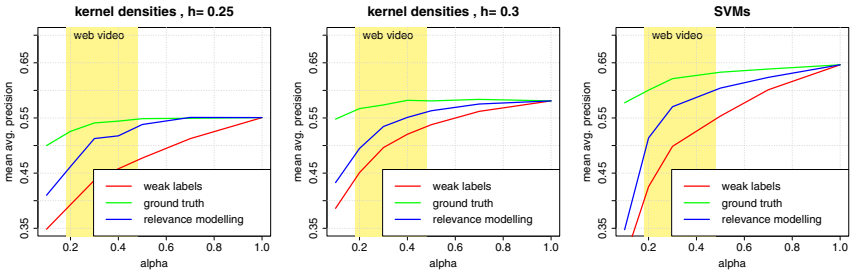


Fig. 6 Results of relevance filtering (using the generative approach): for six concept, the frames are displayed that the relevance filtering approach learns to be most relevant (top) and least relevant (bottom). Relevance filtering works in general, and non-relevant content – though labeled with the concept – can be identified. However, the quality of filtering seems strongly related to concept difficulty: for example, compare "cats" (top right) with "soccer" (bottom left).

related to the concept, and non-relevant material – though labeled with the target concept – tends to be identified successfully.

Quantitative results of the experiment are illustrated in Figs. 7(a) and 7(b) (for the generative model) and in Fig. 7(c) (for the discriminative one). Performance is plotted against the label precision α in a similar fashion as in Sect. 3. In contrast to earlier results, however, relevance filtering is included.

One first observation is that for $\alpha = 1$ all methods perform equally well (which can be observed both for kernel densities and SVMs). This is trivial as the systems are all trained on the same labels – no false positives exist and no filtering takes place. However, with decreasing α – i.e. with increasing non-relevant content in the training set – differences can be observed. It can be seen that relevance filtering (though not reaching the performance of the oracle-based control run) significantly outperforms standard supervised learning. For example, for a bandwidth $h = 0.25$ and a relevance fraction of $\alpha = 0.3$, relevance filtering gives an improvement from



(a) (b) (c)

Fig. 7 Results of relevance filtering for kernel densities (a,b) and SVMs (c). Performance is plotted against the relevance fraction α. It can be seen that relevance filtering (blue) – though not achieving the performance of a hypothetical perfect relevance filter (green) – gives significant improvements over its standard supervised equivalent (red).

44% to 51%. For a higher bandwidth of 0.3 this improvement is lower, which can be explained by the fact that the supervised baseline is more competitive due to a stronger smoothing.

When comparing the results for kernel densities with the ones for the SVM approach, similar observations can be made for the discriminative model: due to inherent errors of filtering, relevance filtering does not reach the performance of the oracle-based control run, but it significantly outperforms its standard supervised counterparts.

Finally, the experiment also indicates for which label precisions relevance filtering is the most promising. If the training set is extremely noisy ($\alpha \leq 10\%$), a fully automatic relevance filtering becomes difficult. This can be observed in Fig. 7(c), where for the leftmost point ($\alpha = 10\%$) the improvement by relevance filtering is only weak. On the other hand, for high values of α the supervised baseline is already quite competitive. For moderate values of $0.2 \leq \alpha \leq 0.5$, the benefits of relevance filtering are most prominent. According to this result, relevance filtering is of particular interest for web content, which comes with noise ratios in the same range. Here, performance improvements in the range of $3-9\%$ are achieved.

5.2 Raw Web Video Content

The purpose of the last experiment was to give a proof-of-concept for relevance filtering in a controlled setup, where the ratio of relevant material is known. In this case, relevance filtering was demonstrated to outperform supervised models significantly.

In the following, we test relevance filtering on training sets of real-world web video content. In contrast to the controlled setup studied in the last section, there are two key differences. First, the fraction of relevant content is not known a priori when downloading raw material from YouTube. A simple workaround for this is to set $\hat{\alpha}$ to a "reasonable" value like 0.5, which will be demonstrated to give comparable

Fig. 8 Non-relevant content from web videos labeled with "Eiffel Tower", which indicate that noise content in real-world training sets is correlated with the target concept. This renders a fully automatic relevance filtering based only on the distribution in feature space a difficult challenge.

results to using the true relevance prior. The second issue is related to the non-relevant samples themselves: while the proposed approach assumes such false positives to be drawn from an overall "world" distribution, non-relevant content in practice depends strongly on the concept. For example, non-relevant material in "basketball" videos tends to show scenes of a cheering crowd, while non-relevant material for the concept "eiffeltower" contains many urban scenes of Paris (a few typical false positives from "Eiffel Tower" videos are displayed in Fig. 8). Note that – since relevance filtering is entirely based on the fact that relevant content forms peaks in feature space – non-relevant material forming similar peaks (for example "shots of Paris") may be difficult to separate from truly relevant content.

We use a similar setup as in previous experiments, i.e. training sets of known noise ratios are randomly compiled (the same sample numbers are used as described in the last section). The key difference is that false positives – which were previously sampled from videos *not* labeled with the concepts – are now drawn from clips tagged with the concept (but are still non-relevant according to manual annotation). Correspondingly, negative test samples now consist of 500 frames from videos tagged with the concept and $1,000$ frames from other videos.

Figure 9 also tackles the question how to estimate the relevance prior α. It suggests a very simple solution, namely to set it to a "reasonable" choice such as 0.5 (which corresponds to a typical value for web-based training sets as shown in Sect. 3). Figure 9 compares relevance filtering when using the true prior $\hat{\alpha} = \alpha$ and when using $\hat{\alpha} = 0.5$. It can be seen that by simply setting $\hat{\alpha} = 0.5$ (i.e. by filtering half of all positive training samples) a stable performance can be obtained that is comparable to the true relevance fraction, at least for the range of $\alpha = 0.2 - 0.5$ typical for web video. For the SVM approach and very noisy training sets ($\alpha < 0.2$ in Fig. 9(c)), this choice even outperforms a more aggressive filtering as for $\hat{\alpha} = \alpha$.

6 Relevance Filtering with Active Learning

In previous experiments, it was shown that automatic relevance filtering improves concept detection performance by up to 9% for real-world web video material when run in an controlled setup. When trained on raw web video content as downloaded from YouTube, it was shown again that automatic relevance filtering gives performance improvements. These improvements, however, are lower than for the controlled setup, which can be explained by the fact that non-relevant content tends

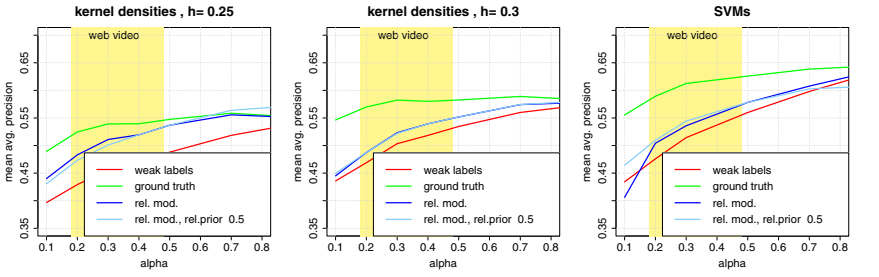


Fig. 9 Comparing relevance filtering on raw web video training sets when using the correct relevance prior ($\hat{\alpha} = \alpha$, dark blue) with a default choice of $\hat{\alpha} = 0.5$ (light blue). It can be seen that a simple choice of $\hat{\alpha} = 0.5$ leads to comparable results.

to be concept-dependent in practice (for example, noise material tagged with "Eiffel Tower" tends to show urban scenes of Paris, as illustrated in Fig. 8). Such content forms clusters in feature space similar to relevant material and cannot be separated easily.

This raises the question whether a better filtering could be achieved using a little manual supervision, i.e. by requiring human operators to provide a few selected labels. Here, active learning techniques – where the system selects informative examples for the user to annotate [32] – is an interesting extension to the current relevance filtering framework.

The approach has already proven to be successful in the large-scale concept detection evaluation TRECVID [2], where training examples for a concept of interest are accumulated from a completely unknown video database. This setup (which usually starts from few reliable initial labels [1, 2, 10]) differs from the one studied in this chapter, as we focus on a *refinement* of large and only partly non-relevant training sets. Despite this difference, however, such an extension fits quite elegantly into the proposed learning framework: whenever a user annotates a training sample x_i, the relevance score β_i is adapted and fixed to the given label, and learning is re-iterated.

In this section, an active learning extension to relevance filtering is presented. Also, we compare several active learning sample selection strategies in an experiment and show that the approach – if integrated with the kernel density version of relevance filtering – leads to significant improvements of concept learning from web video at moderate annotation effort.

6.1 Basic Concepts

In Sect. 4.2, a generative approach using kernel density estimation has been extended such that relevance scores β_i capture the uncertainty of the given label information. To reduce this uncertainty, we propose an iterative manual refinement of selected samples and successive retraining. Such a relevance feedback mechanism

Table 4 Active Learning Extension: Wrapped around relevance filtering, active learning selects informative samples for refinement by a human operator. Once the label is given, its relevance score is set to either 0 or 1, the system is re-trained, and the remaining relevance scores are adapted.

1. for $j = 1,.,m$ do:
   - train relevance filtering and get relevance scores $\beta^j = \{\beta_i^j\}$
   - select sample s^* according to an *active learning* criterion Q:

$$s^* := \arg\max_{i:\tilde{y}_i=1} \; Q(\beta_i^j)$$

   - annotate manually, obtaining the true label y_{s^*}
   - fix the relevance score $\beta_{s^*}^{j+1,..,m} = \begin{cases} 1, & y_{s^*} = 1 \\ 0, & y_{s^*} = -1 \end{cases}$

can be placed as a wrapper around automatic relevance filtering. The procedure is illustrated in Table 4: iteratively, relevance filtering training is applied, obtaining relevance scores β. Based on these scores, the most informative weakly labeled sample is selected for manual annotation (note that – as according to Definition 1 only positive labels are unreliable – we focus on the positive samples). After manual labeling of the selected sample s^*, we can fix $\beta_{s^*}^j$ to either 0 or 1 depending on the received label. This new information is used for retraining relevance filtering, providing new relevance scores β_i^{j+1} for the next iteration of sample selection. With increasing iterations of such *active learning*, the procedure separates relevant content from non-relevant one more reliably.

6.2 Active Learning Methods

Different sample selection strategies for active learning have been proposed in the literature (see [1, 10, 41] for work in the video retrieval domain and [32] for a more complete survey). We test a few of the most popular ones in combination with relevance learning. These strategies select samples based on their a class posterior (which in our case corresponds to the *relevance score* β).

1. **random sampling:** samples are selected randomly (serves as a baseline).
2. **most relevant sampling:** samples are selected which are most likely to be relevant and are therefore associated with a maximum relevance score β. This approach was first introduces in information retrieval [29] but has also been proven to be a good option in a concept detection setup [1, 2]:

$$Q_{REL}(\beta) := \beta$$

3. **uncertainty sampling:** samples are selected for which the relevance filtering method is most inconfident, i.e. $\beta \approx 0.5$ [23]:

$$Q_{UNC}(\beta) := 1 - |\beta - 0.5|$$

6.3 Experiments with Active Relevance Filtering

In the following, we run an experiment on raw web video similar to Sect. 5.2 and apply the active learning extension to relevance filtering with the goal to improve concept detection performance further.

The dataset used is equal to the one in Sect. 5.2. The kernel bandwidth is fixed to $h = 0.275$. The key difference to Sect. 5.2 is that the relevance fraction is fixed at $\alpha = 0.2$, which poses a difficult challenge to automatic relevance filtering as the majority of content is non-relevant.

Results averaged over 5 runs are illustrated in Fig. 10. In contrast to the previous experiments, performance is plotted against the number of manually annotated samples ($\alpha = 0.2$). Also, the results from previous experiments in Sect. 5.2 can be found in the plot as horizontal lines: *no relevance filtering*, *automatic relevance filtering*, and *ground truth* training.

Now, we study how concept detection performance increases if we iteratively replace weak labels – potentially associated with false positives – with true labels provided by human annotations. As seen in Fig. 10 the performance of the different active learning methods lies within a corridor bounded by automatic relevance filtering (bottom) and the ground truth run (top). Starting with no refined samples, the performance equals the one of automatic relevance filtering. With more manual annotations, performance increases and finally converges to the ground truth case.

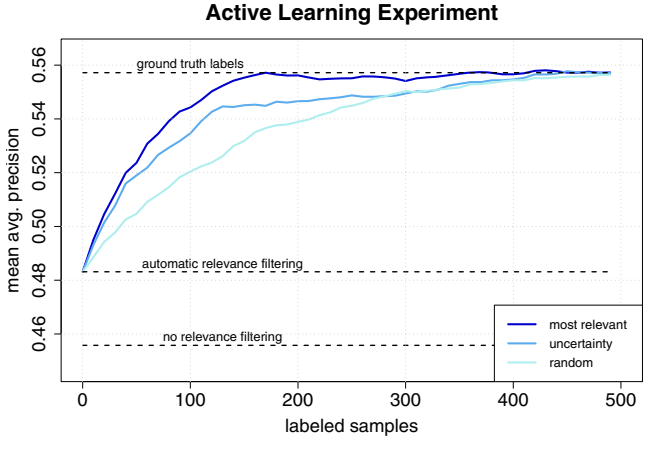


Fig. 10 Results of active learning for relevance filtering with generative kernel densities. The label precision α is fixed at 0.2. Performance is plotted against the number of manually annotated training samples. It can be seen that – if using a proper sample selection – it is sufficient to annotate only $30 - 40$ weakly positive training samples to achieve a significant performance improvement.

When comparing the different active learning strategies, it can be seen that they both outperform random sampling significantly, and that the best performance is achieved by *most relevant sampling*, which mines the dataset for truly relevant samples. This can be explained by the fact that relevance filtering relies strongly on correct relevance weights. Consider the example of "eiffeltower" and assume that the illustrated false positives in Fig. 8 all belong to the same cluster. If the system misleadingly assigns high relevance score to this cluster, one of the samples will be selected early for manual refinement. The false positives will be identified, corrected, and further iterations of relevance filtering will propagate this new information among the cluster in giving neighboring samples lower relevance scores.

Overall, it can be seen that with active learning we can improve performance at moderate annotation cost. For example, with as few as 40 annotations, a performance increase of about 4% is achieved. When continuing with annotation, we can see that concept detection performance converges to the ground truth case at $100-150$ iterations (which corresponds to only $25-30\%$ of the overall dataset). Concluding, relevance filtering – if combined with appropriate active learning strategies – can improve concept learning on the difficult domain of raw web video content.

7 Conclusions

In this chapter, the visual learning of concept detectors from weakly labeled videos has been addressed. Such data offers a scalable alternative to the conventional manual acquisition of training data, as label information can be acquired without manual overhead. On the other hand, class information becomes unreliable, and labels are only weak indicators of concept presence. We have demonstrated for the domain of web video that typical training sets include significant amounts of non-relevant noise material, with a resulting performance degradation of up to 20%.

To overcome this problem without additional manual overhead, a framework called *relevance filtering* has been proposed. A binary classification problem is cast for each target concept, whereas true (latent) concept presence is inferred during system training. Based on this idea, two relevance filtering extensions of well-known supervised techniques were presented, one for a generative approach (kernel densities) and one for a discriminative one (Support Vector Machines). In experiments on real-world web video material, it was demonstrated that relevant content can be separated from non-relevant one in general, and that the performance of concept detection can be improved by up to 9% in a controlled setup. When trained on raw web video content as downloaded from YouTube, relevance filtering still improves over the supervised case though by a lower margin, which is because non-relevant content shows a tendency to be concept-dependent. To handle such conditions, an extension to relevance filtering was introduced, where minimal manual supervision adapts the scores of the weakly labeled training samples. In particular, by utilizing active learning methods for sample selection, the system could improve by up to 8% by refining only $25-30\%$ of weak positive labels from the training set.

As the approach in this chapter focuses entirely on a *visual* learning of concepts, one interesting extension of the framework would be the additional use of tag information coming with web video clips for relevance filtering. While we currently make only very limited use of such meta information, we envision our future system to use tag information directly in the refinement process. Particularly, *deep tagging* – where users provide detailed tags at certain time stamps in a video – might be an interesting clue to overcome the *label coarseness* problem.

Acknowledgements. This work was supported by the German Research Foundation (DFG), project MOONVID (BR 2517/1-1).

References

1. Ayache, S., Quenot, G.: Evaluation of active learning strategies for video indexing. Signal Processing: Image Communication 22(7-8), 692–704 (2007)
2. Ayache, S., Quenot, G.: Video Corpus Annotation using Active Learning. In: Proc. Europ. Conf. on Information Retrieval, pp. 187–198 (March 2008)
3. Barnard, K., Duygulu, P., Forsyth, D., de Freitas, N., Blei, D., Jordan, M.: Matching Words and Pictures. J. Mach. Learn. Res. 3, 1107–1135 (2003)
4. Berg, T., Forsyth, D.: Animals on the Web. In: Proc. Int. Conf. Computer Vision and Pattern Recognition, pp. 1463–1470 (June 2006)
5. Blum, A., Mitchell, T.: Combining Labeled and Unlabeled Data with Co-training. In: Proc. Ann. Conf. on Computational Learning Theory, pp. 92–100 (July 1998)
6. Snoek, C., et al.: The MediaMill TRECVID 2007 Semantic Video Search Engine. In: Proc. TRECVID Workshop (unreviewed workshop paper) (November 2007)
7. Campbell, M., Haubold, A., Liu, M., Natsev, A., Smith, J., Tesic, J., Xie, L., Yan, R., Yang, J.: IBM Research TRECVID-2007 Video Retrieval System. In: Proc. TRECVID Workshop (unreviewed workshop paper) (November 2007)
8. Chang, C.-C., Lin, C.-J. (LIBSVM): A Library for Support Vector Machines (2001), Software available at http://www.csie.ntu.edu.tw/~cjlin/libsvm
9. Chapelle, O., Schölkopf, B., Zien, A. (eds.): Semi-supervised Learning. MIT Press, Cambridge (2006)
10. Chen, M., Christel, M., Hauptmann, A., Wactlar, H.: Putting Active Learning into Multimedia Applications: Dynamic Definition and Refinement of Concept Classifiers. In: Proc. Int. Conf. on Multimedia, pp. 902–911 (November 2005)
11. Dempster, A., Laird, N., Rubin, D.: Maximum Likelihood from Incomplete Data via the EM Algorithm. Journal of the Royal Statistical Society, Series B 39(1), 1–38 (1977)
12. Duda, R., Hart, P., Stork, D.: Pattern Classification, 2nd edn. Wiley Interscience, Hoboken (2000)
13. Everingham, M., Van Gool, L., Williams, C., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge 2008 (VOC2008) Results (October 2008)
14. Fergus, R., Fei-Fei, L., Perona, P., Zisserman, A.: Learning Object Categories from Google's Image Search. Computer Vision 2, 1816–1823 (2005)
15. Gargi, U., Yagnik, J.: Solving the Label Resolution Problem in Supervised Video Content Classification. In: Proc. Int. Conf. on Multimedia Retrieval, pp. 276–282 (October 2008)
16. Gu, Z., Mei, T., Hua, X.-S., Tang, J., Wu, X.: Multi-layer Multi-instance Kernel for Video Concept Detection. In: Proc. Int. Conf. on Multimedia, pp. 349–352 (September 2007)

17. Hauptmann, A., Yan, R., Lin, W.: How many High-Level Concepts will Fill the Semantic Gap in News Video Retrieval? In: Proc. Int. Conf. Image and Video Retrieval, pp. 627–634 (July 2007)
18. Hofmann, T.: Unsupervised Learning by Probabilistic Latent Semantic Analysis. Machine Learning 42, 177–196 (2001)
19. Yuan, J., et al.: THU and ICRC at TRECVID 2007. In: Proc. TRECVID Workshop (unreviewed workshop paper) (November 2007)
20. Joachims, T.: Transductive Inference for Text Classification using Support Vector Machines. In: Int. Conf. Machine Learning, pp. 200–209 (June 1999)
21. Kennedy, L., Chang, S.-F., Kozintsev, I.: To Search or to Label?: Predicting the Performance of Search-based Automatic Image Classifiers. In: Int. Workshop Multimedia Information Retrieval, pp. 249–258 (October 2006)
22. Kraaij, W., Over, P.: TRECVID-2007 High-Level Feature Task: Overview. In: Proc. TRECVID Workshop (November 2007)
23. Lewis, D., Gale, W.: A Sequential Algorithm for Training Text Classifiers. In: Proc. Int. Conf. Research and Development in Information Retrieval, pp. 3–12 (July 1994)
24. Li, L.-J., Wang, G., Fei-Fei, L.: OPTIMOL: automatic Object Picture collecTion via Incremental MOdel Learning. In: Proc. Int. Conf. Computer Vision and Pattern Recognition, pp. 57–64 (June 2007)
25. Lowe, D.: Object Recognition from Local Scale-Invariant Features. In: Int. Conf. Computer Vision, pp. 1150–1157 (September 1999)
26. Morsillo, N., Pal, C., Nelson, R.: Semi-supervised Visual Scene and Object Analysis from Web Images and Text. In: Scene Understanding Symposium (February 2008)
27. Naphade, M., Smith, J., Tesic, J., Chang, S., Hsu, W., Kennedy, L., Hauptmann, A., Curtis, J.: Large-Scale Concept Ontology for Multimedia. IEEE MultiMedia 13(3), 86–91 (2006)
28. Paredes, R., Perez-Cortes, A.: Local Representations and a Direct Voting Scheme for Face Recognition. In: Proc. Workshop on Pattern Rec. and Inf. Systems, pp. 71–79 (July 2001)
29. Salton, G., Buckley, C.: Improving Retrieval Performance by Relevance Feedback. Journal of the American Society for Information Science 41(4), 288–297 (1990)
30. Schölkopf, B., Smola, A.: Learning with Kernels: Support Vector Machines, Regularization, Optimization, and Beyond. MIT Press, Cambridge (2001)
31. Schroff, F., Criminisi, A., Zisserman, A.: Harvesting Image Databases from the Web. In: Proc. Int. Conf. Computer Vision, pp. 1–8 (October 2007)
32. Settles, B.: Active Learning Literature Survey. Computer Sciences Technical Report 1648, University of Wisconsin–Madison (2009)
33. Sivic, J., Zisserman, A.: Video Google: Efficient Visual Search of Videos. In: Toward Category-Level Object Recognition, pp. 127–144. Springer, New York, Inc. (2006)
34. Smeaton, A.: Techniques Used and Open Challenges to the Analysis, Indexing and Retrieval of Digital Video. Inf. Syst. 32(4), 545–559 (2007)
35. Smeaton, A., Over, P., Kraaij, W.: Evaluation Campaigns and TRECVID. In: Int. Workshop Multimedia Information Retrieval, pp. 321–330 (October 2006)
36. Smeulders, A., Worring, M., Santini, S., Gupta, A., Jain, R.: Content-Based Image Retrieval at the End of the Early Years. IEEE Trans. Pattern Analysis and Machine Intelligence 22(12), 1349–1380 (2000)
37. Snoek, C., Worring, M.: Concept-based Video Retrieval. Foundations and Trends in Information Retrieval 4(2), 215–322 (2009)
38. Snoek, C., Worring, M., de Rooij, O., van de Sande, K., Yan, R., Hauptmann, A.: VideOlympics: Real-Time Evaluation of Multimedia Retrieval Systems. IEEE MultiMedia 15(1), 86–91 (2008)

39. Snoek, C., Worring, M., Huurnink, B., van Gemert, J., van de Sande, K., Koelma, D., de Rooij, O.: MediaMill: Video Search using a Thesaurus of 500 Machine Learned Concepts. In: 1st Int. Conf. Sem. Dig. Media Techn (Posters and Demos.) (2006)
40. Sun, Y., Shimada, S., Taniguchi, Y., Kojima, A.: A Novel Region-based Approach to Visual Concept Modeling using Web Images. In: Int. Conf. Multimedia, pp. 635–638 (2008)
41. Tong, S., Chang, E.: Support Vector Machine Active Learning for Image Retrieval. In: Proc. Int. Conf. on Multimedia, pp. 107–118 (September 2001)
42. Turlach, B.: Bandwidth Selection in Kernel Density Estimation: A Review. In: CORE and Institut de Statistique, pp. 23–49 (1993)
43. Ulges, A., Schulze, C., Keysers, D., Breuel, T.: Identifying Relevant Frames in Weakly Labeled Videos for Training Concept Detectors. In: Proc. Int. Conf. Image and Video Retrieval, pp. 9–16 (July 2008)
44. YouTube Serves up 100 Million Videos a Day Online. USA Today (Garnnett Company, Inc.) (July 2006), http://www.usatoday.com/tech/news/2006-07-16-youtube-views_x.htm (retrieved, September 2008)
45. van de Sande, K., Gevers, T., Snoek, C.: A Comparison of Color Features for Visual Concept Classification. In: Proc. Int. Conf. Image and Video Retrieval, pp. 141–150 (July 2008)
46. Wang, D., Liu, X., Luo, L., Li, J., Zhang, B.: Video Diver: Generic Video Indexing with Diverse Features. In: Proc. Int. Workshop Multimedia Information Retrieval, pp. 61–70 (September 2007)
47. Wang, M., Hua, X.-S., Song, Y., Yuan, X., Li, S., Zhang, H.-J.: Automatic Video Annotation by Semi-supervised Learning with Kernel Density Estimation. In: Proc. Int. Conf. on Multimedia, October 2006, pp. 967–976 (2006)
48. Wnuk, K., Soatto, S.: Filtering Internet Image Search Results Towards Keyword Based Category Recognition. In: Proc. Int. Conf. Computer Vision and Pattern Recognition, pp. 1–8 (June 2008)
49. Yanagawa, A., Chang, S.-F., Kennedy, L., Hsu, W.: Columbia University's Baseline Detectors for 374 LSCOM Semantic Visual Concepts. Technical report, Columbia University (2007)
50. Yanai, K., Barnard, K.: Probabilistic Web Image Gathering. In: Int. Workshop on Multimedia Inf. Retrieval, November 2005, pp. 57–64 (2005)
51. Yang, J., Hauptmann, A.: (Un)Reliability of Video Concept Detection. In: Proc. Int. Conf. Image and Video Retrieval, July 2008, pp. 85–94 (2008)
52. Zhu, X.: Semi-supervised Learning Literature Survey. Technical Report 1530, Computer Sciences, University of Wisconsin, Madison (2005)

Appendix

This section gives a description of the concepts used in our experiments. We have defined canonical definitions of each concept and performed a manual annotation of web-based material. Table 5 provides the definitions as well as information on how video data was downloaded from YouTube.

Table 5 Meta Information regarding the 10 test concepts used in the experiments

Concept	Description	YT Query*	YT Category*
basketball	Scenes showing people playing basketball. Includes streetball if recognizable as such.	basketball basketball nba basketball dunking basketball best moves basketball dunks	sports
beach	Scenes showing a beach. Water does not have to be visible (if anything else qualifies the scene as showing a beach). Shots from a distance qualify as well, but only if the coastline is clearly a beach.	walk on the beach beach sunbath beach hawaii beach panorama beach malibu day	travel&places
cats	Scenes showing one or multiple cats. Closeups qualify as well as full body shots.	cats cats funny cats pets animals cats playing cats eating	pets&animals
desert	Scenes showing desert landscape. Panoramic shots involving significant amounts of sky are allowed (as long as some desert landscape is visible at the bottom). Things like plants, rocks, canyons, cars, etc. are allowed, but the landscape should show desert.	desert egypt driving through desert desert panorama desert sahara desert trip	travel&places
eiffeltower	Scenes showing the Eiffel Tower. Views from top of the tower qualify only if you see a part of tower (like parts of the steel construction). Night shots qualify. Closeups showing only parts of the steel construction qualify (if the tower can be identified) as well as panoramic shots from the distance. Shots with people in the foreground and the tower in the background count as well.	tour eiffel, eiffel tower, eiff.t. paris france, eiffelturm paris	travel&places
helicopter	Scenes showing a helicopter (airborne or on the ground). Views from inside the helicopter are allowed if they can be identified as such. Only instruments or the pilot are not sufficient. Shots of toy helicopters qualify as well.	helicopter, helicoptero, helicopter flying, helicopter landing	autos&vehicles
sailing	Scenes showing sailing ships/boats on the water/in the harbor. Panoramic views from onside a boat qualify if you see a part of the boat (like sails). Catamarans qualify as sailing ships, but surf boards or tankers do not (generally, everything with a sail qualifies).	sailing, sailing trip, sailing boat, sailing holiday, sailing mediterranean	travel&places
soccer	Shots showing a soccer match. Actions only distantly related to soccer do not qualify (like people doing soccer tricks in the street). Close-ups of players are allowed as well as global shots (if clearly identifiable as soccer). Soccer fields without action qualify as well. Shots of a cheering crowd do not qualify.	soccer bundesliga, soccer goals, soccer match, soccer game outdoor, fussball spiel	sports
swimming	Scenes showing somebody swimming. A swimming pool counts too (even if nobody is swimming inside it). Also includes swimming objects (fish, bottles).	swimming, swimming pool -clean, swimming technique, sw. competition, swimming olympics, sw. championship	sports
tank	Scenes showing a tank, i.e. a heavily armored vehicle. Any scene qualifies if a part of the tank is visible such that the tank is identifiable. Other sorts of military ground vehicles qualify.	tanques, tank, tank battle, panzer, tank fire -flashpoint	autos&vehicles

* *Values used for YouTube API calls*

Part IV
Personalized Video

Face Recognition and Retrieval in Video

Caifeng Shan

Abstract. Automatic face recognition has long been established as one of the most active research areas in computer vision. Face recognition in unconstrained environments remains challenging for most practical applications. In contrast to traditional still-image based approaches, recently the research focus has shifted towards video-based approaches. Video data provides rich and redundant information, which can be exploited to resolve the inherent ambiguities of image-based recognition like sensitivity to low resolution, pose variations and occlusion, leading to more accurate and robust recognition. Face recognition has also been considered in the content-based video retrieval setup, for example, character-based video search. In this chapter, we review existing research on face recognition and retrieval in video. The relevant techniques are comprehensively surveyed and discussed.

1 Introduction

Automatic face recognition has long been established as one of the most active research areas in computer vision [119, 70, 61, 99, 1]. This is mainly due to its wide range of applications such as person identification, law enforcement, smart environment, visual surveillance, human-computer interaction, and image/video retrieval. After three decades of intense research, the state-of-the-art approaches can achieve high recognition rate under controlled settings [86]. However, face recognition in unconstrained real-life environments remains challenging for most practical applications.

Faces are probably the most common cue used by humans for identifying people. Face recognition has mainly been studied for biometric identification. Biometrics refers to the measurement and analysis of physical or behavioral characteristics of humans; various visual traits, such as fingerprint, face, iris, gait and hand geometry,

Caifeng Shan
Philips Research, Eindhoven, The Netherlands
e-mail: caifeng.shan@philips.com

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 235–260.
springerlink.com

have been explored for biometric recognition [54]. Among different biometric modalities, face recognition allows unobtrusive identification in uncontrolled scenarios without requiring user cooperation, for example, in low quality surveillance footage.

Numerous approaches have been developed for face recognition in the last three decades. Traditionally, face recognition research has mainly focused on recognizing faces from still images [103, 15]. However, single-shot based recognition is hard because of the well-known problems such as illumination change, pose variation, facial expression, and occlusion. The face image differences caused by these factors often exceed those due to identity changes. A single image might not provide enough information for reliable classification. Another problem with image-based face recognition is that it is possible to use a prerecorded face photo to confuse the recognition system to take it as a live subject [101]. Recently, the research focus has shifted more and more towards video-based approaches. The advent of inexpensive video cameras and increased processing power makes it viable to capture, store and analyze face videos. Video inputs provide rich and redundant information in the form of multiple frames, for example, normally in video people show a lot of pose and expression variations. It is widely believed that, by properly extracting the additional information, video-based recognition could resolve the inherent ambiguities of image-based recognition, such as sensitivity to low resolution, pose variations and partial occlusion, leading to more accurate and robust face recognition. Furthermore, video inputs allow to capture facial dynamics that are useful for face identification [60, 83].

Although most of the existing research has been focused on the biometric identification paradigm, recently face recognition has been considered in the content-based video retrieval setup [11, 96]. Face information is important in different kinds of videos, especially in news programs, dramas, and movies. Face recognition could be used for video content description, indexing and mining, e.g., rapid browsing or retrieval of scenes based on the presence of specific actors. Increasing amount of video content on the web is marking a new phase of how users consume information, where users often look for specific people related video content. Current video search engines mainly rely on the keywords that appear in descriptions or in the surrounding web page content. Face recognition enables more accurate video search by focusing on the content of videos.

In this chapter, we review existing research on face recognition in video. The relevant techniques are comprehensively surveyed and discussed. We also introduce recent work on face retrieval in video. The chapter is organized as follows. In Section 2, we briefly introduce face detection and face tracking, the two important components in face recognition and retrieval research. Section 3 discusses video-based face recognition technologies. Specifically, three main categories, including key-frame based, temporal model based, and image-set matching based, are described respectively. In Section 4, we present recent research on face retrieval in video. Challenges and future research directions are discussed in Section 5. Finally Section 6 concludes the chapter.

2 Face Detection and Tracking

Face detection and face tracking are important components in face recognition systems, which automatically detect or locate the face region in the input frames. Normally from the located face region, the relevant features are extracted and subsequently served as input to the face recognizer. In this section, we briefly introduce existing work in these two areas.

2.1 Face Detection

Face detection plays a crucial role in face-related vision applications. Due to its practical importance, numerous techniques have been proposed for face detection (see [113] for a survey). In most of existing methods, appearance features such as edge, intensity, and color, are extracted to locate the faces using statistical or geometric models. The real-time face detection scheme proposed by Viola and Jones [104, 105] is arguably the most commonly employed front face detector, which consists of a cascade of classifiers trained by AdaBoost employing Harr-wavelet features. AdaBoost [34, 92] is one of the most successful machine learning techniques applied in computer vision, which provides a simple yet effective approach for stagewise learning of a nonlinear classification function. Later their approach was extended with rotated Harr-like features and different boosting algorithms [76]. In [71], by incorporating Floating Search into AdaBoost, FloatBoost was proposed for improved performance on multi-view face detection.

Many other machine learning techniques such as Neural Network and Support Vector Machine (SVM) have also been introduced for face detection. In [77], the Bayes classifier was adopted with discriminating feature analysis for frontal face detection. The input image, its 1D Harr wavelet representation, and its amplitude projections are combined to derive a discriminating feature vector. Later the features were extended and combined with a SVM-based classifier [94]. SVM was also used with the spectral histogram features for face detection [107]. To improve the detection efficiency, Garcia and Delakis [37] designed a convolutional neural network for face detection, which performs simple convolutional and subsampling operations. More recently, the approach in [77], Viola and Jones's approach [104, 105], and the approach in [37] are modified and combined for a fast and robust face detector [22]. Overall, face detection technology is fairly mature and a number of reliable face detectors have been built based on existing approaches.

2.2 Face Tracking

Most of face detectors can only detect faces in the frontal or near-frontal view. To handle large head motion in video sequences, face or head tracking is usually adopted. In some work [120, 55], face tracking and recognition are integrated in one framework, for example, Zhou *et al.* [122, 120] proposed a probabilistic approach for simultaneous tracking and recognition.

Fig. 1 Examples of face tracking using the online boosting algorithm [44].

Visual tracking of objects has been intensively studied in computer vision, and different approaches have been introduced, for example, particle filtering [13] and mean shift [19, 23]. Accurate face tracking is difficult because of face appearance variations caused by the non-rigid structure, 3D motion, occlusions, and environmental changes (e.g., illumination). Therefore, adaptation to changing target appearance and scene conditions is a critical property a face tracker should satisfy. Ross *et al.* [88] represented the target in a low-dimensional subspace which is adaptively updated using the images tracked in the previous frames. In [44], Grabner *et al.* introduced the online boosting for tracking, which allows online updating of the discriminative features of the target object. Some face tracking results of their approach are shown in Fig. 1. Compared to the approaches using a fixed target model such as [14], these adaptive trackers are more robust to face appearance changes in video sequences.

One main drawback of these adaptive approaches is their susceptibility to drift, i.e., gradually adapting to non-targets, because the target model is built from the previous tracked results. To address this problem, a mechanisms for detecting or correcting drift should be introduced [55], by adding global constraints on the overall appearance of the target. For faces, such constraints could be learned from a set of generic well-cropped/aligned face images that span possible variations in pose, illumination, and expression. Specifically, two constraint terms were introduced in [55]: (1) a set of facial pose subspaces (or manifolds), each of which represents a particular out-of-plane pose, and (2) a SVM based goodness-of-crop discriminator whose confidence score indicates how well the cropped face is aligned. Grabner *et al.* [45] introduced an online semi-supervised boosting to alleviate the drifting problem. They formulated the update process in a semi-supervised fashion which uses the labeled data as a prior and the data collected during tracking as unlabeled samples.

After the face detection and tracking stages, faces are only roughly localized and aligned. Face registration methods can be adopted to deal with the effect of varying pose, for example, by utilizing the characteristic facial points (normally locations of the mouth and eyes). In some work (e.g., [59]), face recognition is performed directly on faces roughly localized, close to the conditions given by typical surveillance systems.

3 Face Recognition in Video

Recent years have witnessed more and more studies on face recognition in video. Zhang and Martinez [114, 115] investigated whether the methods, defined to recognize faces from a single still image, perform better if they could work with multiple images or video sequences. By extending their probabilistic appearance-based approach [80], they showed that regardless of the feature extraction method used, the recognition results improve considerably when using a video sequence rather than a single image. It is also observed in [47] that the spatial-temporal representation derived from video outperforms the image-based counterpart. Video-based recognition provides a setting where weak evidence in individual frames can be integrated over time, potentially leading to more reliable recognition in spite of the difficulties such as pose variation and facial expression.

We partition the existing research into three categories: (1) key-frame (or exemplar) based approaches, (2) temporal model based approaches, and (3) image-set matching based approaches. The first class [47, 115, 98, 101] considers the problem as a recognition problem from still images by independently using all or a subset of the face images. Usually a voting rule is used to come up with a final result. In most of cases, only a subset of representative face images is used for recognition, where ad hoc heuristics are used to select key-frames. The second class [75, 123, 64, 79, 49] makes use of all face images together with their temporal order in the video. By taking into account temporal coherence, face dynamics (such as non-rigid facial expressions and rigid head movements) within the video sequence are modeled and exploited to enforce recognition. The third class [110, 93, 10, 59, 106] also uses all face images, but does not assume temporal coherence between consecutive images; the problem was treated as an image-set matching problem. The distributions of face images in each set are modeled and compared for recognition, and the existing work can be further divided into statistical model-based and mutual subspace-based methods (see Section 3.3 for details). Both the second and third categories integrate the information expressed by all the face images into a single model. The categorization of relevant techniques are summarized in Table 1. In the following sections, we discuss each category in details.

3.1 Key-Frame Based Approaches

The approaches in this category treat each video as a collection of images, and perform face recognition by comparing all or a subset of individual face images

Table 1 Categorization of video-based face recognition techniques.

Category	Method
Key-frame based Approaches	[90], [40], [47], [114], [100], [17], [115], [31], [78], [85], [98], [101], [118]
Temporal Model based Approaches	[74], [73], [72], [75], [18], [24], [67], [69], [68], [122], [120], [123], [121], [64], [65], [66], [79], [55], [2], [43], [50], [49]
Image-Set Matching based Approaches	
Statistical model-based	[93], [4], [96], [7], [10], [6], [9]
Mutual subspace-based	[110], [90], [35], [82], [108], [56], [57], [5], [58], [59], [106], [38]

with training data using image-based recognition techniques. They neither make any assumption on the underlying distributions of input face images, nor use their temporal coherence. They are based on premise that similarity of the test video with the training data, which could be still images or videos, is reflected by the similarity of the images from the testing video and training data. For example, Satoh [90] matches two face sequences based on face matching between a closest pair of face images across two sequences. These approaches may fail as they do not take into account of the effect of outliers [59]. If requiring a comparison of every pair of samples drawn from the input video and training data, such methods could be time consuming.

Normally a subset of "good" or representative frames (called key-frames or exemplars) is selected to perform image-based recognition. In [40], face recognition is performed based on tracking the nose and eyes. Their locations are used to decide whether the face is suitable for recognition. If they form an equilateral triangle, image-based recognition is performed; otherwise, the tracking continues until an appropriate frame occurs. Berrani and Garcia [17] proposed to select good-quality face images using robust statistics. Specifically, by considering it as an outlier detection problem, they utilized RobPCA to filter out the noisy face images (e.g., not well-centered, non-frontal). Their experiments on two face image databases show that the filtering procedure improves the recognition rate by 10% to 20%. In [85], three face matchers are fused for face recognition in video, where the estimated face poses and the detected motion blur are used for adaptive fusion, e.g., frames with motion blur are not considered for recognition. Experimental results on the CMU Face-In-Action database [39] with 204 subjects show that their approach achieves consistent improvements.

It is argued in [63] that the best exemplars are those which minimize the expected distance between the video frames and the exemplars; radial basis functions are applied to select exemplars. Hadid and Pietikäinen [47] proposed to extract the most representative faces by applying K-Means clustering in the low-dimensional space derived by Locally Linear Embedding [89]. They adopted a probabilistic voting to combine image-based recognition over video frames for final decision. In [31],

following the Isomap algorithm [102], the geodesic distances between face images are estimated, based on which a hierarchical agglomerative clustering algorithm is applied to derive local face clusters of each individual. The authors argued that using a single exemplar for each cluster may not fully characterize the image variability, and proposed to construct two subspaces to characterize intra-personal and extra-personal variations for each local cluster. Given a test image, the angle between its projections onto the two subspace is used as a distance measure to the cluster. Experiments on a video data set of 40 subjects demonstrate their approach produces promising results compared to previous methods. Zhao and Yagnik [118] presented an approach for large scale face recognition in web videos. For each face set derived by facial feature tracking, key faces are selected by clustering; the face sets are further clustered to get more representative key faces and remove duplicate key faces. A combination of majority voting and probabilistic voting is adopted for final decision. Evaluated on large-scale web videos, their approach achieves 80% top-5-precision on tested persons.

Tang and Li [100, 101] proposed to align video sequences of different subjects based on the audio signal in video, i.e., frames with similar facial expressions are synchronized according to the associated speech signal. A number of distinctive frames are selected from the synchronized video sequences. Key fiducial points on each face image are further aligned, and Gabor wavelet features are extracted from these points for facial representation. For matching the spatial and temporal synchronized video sequences, they developed a multi-level discriminant subspace analysis algorithm. They also integrated the frame-based classification using the majority voting or sum rule. In [78], Liu *et al.* proposed a synchronized frame clustering method which incrementally outputs aligned clusters across all video sequences, and adopted a Bayesian method to select key-frames. A Nonparametric Discriminant Embedding is introduced for learning spatial embedding. With the spatial-temporal embedding of video sequences, they presented a statistical classification solution, which uses a probabilistic voting strategy to combine the recognition confidences in each frame. Encouraging results on the XM2VTS database [81] are reported in these studies [100, 101, 78].

Stallkamp *et al.* [98] presented a real-time video-based face recognition system which recognizes people entering through the door of a laboratory. As shown in Fig. 2, without user cooperation, the captured video data contains difficult situations arising from pose variations, facial expressions, occlusions due to accessories and hair, illumination changes due to the time and weather conditions and light switched on/off. Their approach combines the individual frame-based classification results to one score per sequence. With DCT-based appearance features, individual frame scores are generated using a k-nearest neighbor classifier and a set of Gaussian mixture models learned from training sequences. They proposed two measures to weight the contribution of each individual frame: *distance-to-model* (DTM) and *distance-to-second-closest* (DT2ND). DTM takes into account how similar a test sample is to the representatives of the training set. Test samples that are very different from the training data are more likely to cause a misclassification, so DTM is used to reduce the impact of these samples on the final score. Based on the idea that

Fig. 2 The real-world video data used in [98], which shows a variety of different lighting, pose and occlusion conditions.

reliable matching requires the best match to be significantly better than the second-best match, DT2ND is used to reduce the impact of frames which deliver ambiguous classification results. Their experiments on a database of 41 subjects show both measures have positive effects on the classification results. Despite promising results, the need for parameter tuning and heuristic integration schemes may limit the generalization of this approach.

3.2 Temporal Model Based Approaches

Other than the multitude of still frames, video allows to characterize faces based on the inherent dynamics which is not possible with still images. Facial dynamics include the non-rigid movement of facial features (e.g., facial expressions) and the rigid movement of the whole face (e.g., head movements). Psychological studies [60, 83, 95] indicate that facial dynamics play an important role in the face recognition process, and both static and dynamic facial information are used in the human visual system to identify faces. Facial dynamics are even more crucial under degraded viewing conditions such as poor illumination, low resolution, and recognition at distance. Many of these points have been verified in computer vision research [48]. For example, Gorodnichy [41, 42] illustrated the lack of resolution can be compensated by the dynamic information coming from the time dimension. Many approaches have been proposed to utilize the temporal continuity inherent in videos for face recognition [20].

Li *et al.* [74, 73, 72, 75] proposed to model facial dynamics by constructing facial identity structures across views and over time, referred to identity surfaces (shown in Fig. 3), in the Kernel Discriminant Analysis feature space. Dynamic face recognition is performed by matching the face trajectory computed from a video input and a set of model trajectories constructed on the identity surfaces. The trajectory encodes the spatio-temporal dynamics of moving faces, while the trajectory distance accumulates recognition evidence over time. Experimental results on video sequences of 12 subjects were reported with a recognition rate of 93.9%. Similarly, in [18], each image sequence of a rotating face is projected into the eigen-space using Principal Component Analysis (PCA) and represented as a trajectory in the space; face recognition is performed as the trajectory matching. They reported excellent recognition rates on a data set of 28 subjects.

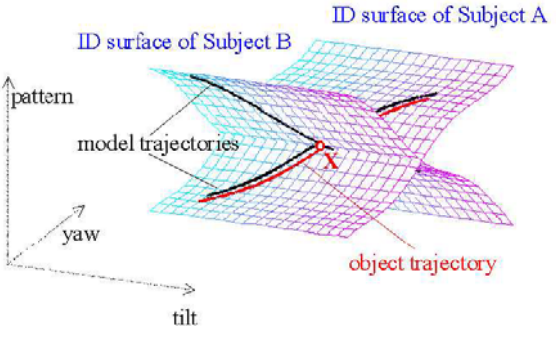


Fig. 3 Identity surfaces for dynamic face recognition [73].

Edwards *et al.* [24] learnt how individual faces vary through video sequences by decoupling sources of image variations such as expression, lighting and pose. The trained statistical face model is used to integrate identity evidence over a sequence, which is more stable and robust than a model trained from a single image. Li and Chellappa [67, 69] presented an approach to simultaneous tracking and verification in video, based on posterior density estimation using sequential Monte Carlo methods; verification is realized through hypothesis testing using the estimated posterior density. By rectifying each face template onto the first frame of the testing video, the approach has been applied to face verification in video. They [68] also introduced a method for face verification using the motion parameters of tracked facial features, where the features are extracted using Gabor filters on a regular 2D grid. Their method produces encouraging results on a data set with 19 subjects.

Following [67, 69], Zhou *et al.* [122, 120] proposed a probabilistic approach for simultaneous tracking and recognition in video. They used a time-series state space model which is parameterized by a tracking state vector (continuous) and a identity variable (discrete), in order to simultaneously characterize the evolving kinematics and identity. The joint posterior probability is approximated and propagated using the Sequential Importance Sampling algorithms, and the marginal distribution of the identity variable is estimated to provide the identity result. In the still-to-video setting, where the gallery consists of still images and the probe consists of videos, the approach was evaluated on two data sets with 12 subjects and 30 subjects respectively. In [123], to consider the video-to-video setting (i.e., generalizing the gallery to videos), they adopted an exemplar learning method [63] to select representatives from the gallery videos, serving as still templates in the still-to-video scenario. Their approach was tested on the MoBo database [46]. Later the simultaneous tracking and recognition approach was improved by incorporating appearance-adaptive models [121]. The appearance changes between input frames and gallery images are modeled by constructing the intra- and extra-personal spaces. Experiments on a data set of 29 subjects illustrate the approach can handle appearance changes caused by pose and illumination variations, leading to improved tracking and recognition performance.

To address continues head pose changes in video, Lee *et al.* [64] proposed to learn a low-dimensional appearance manifold of faces, which is approximated by piecewise linear subspaces (named pose manifolds). To construct the representation, exemplars are first sampled from videos by finding frames with the largest distance to each other, which are further clustered using K-means clustering. Each cluster models face appearance in nearby poses, represented by a linear subspace computed by PCA. The dynamics among pose manifolds are encoded as transition probabilities, learned from training video sequences. They presented a maximum a posteriori formulation for face recognition, which integrates the likelihood that the input image comes from a particular pose manifold and the transition probability to this pose manifold from the previous frame. Their approach was extended for simultaneously tracking and recognizing faces in video [65], achieving the recognition rate of 98.8% on a data set of 20 subjects. However, the appearance model in these works was learned by a batch training process from short video clips, which is not practical for large number of lengthy video sequences. In [66], an online learning algorithm was introduced to incrementally construct a person-specific appearance manifold using an initial generic prior and successive frames from the video of a subject. Experimental results demonstrate the approach constructs an effective representation for face tracking and recognition.

Liu and Chen [79] introduced to use the adaptive Hidden Markov Model (HMM) for video-based face recognition. In the training phase, a HMM is created for each individual to learn the statistics and temporal dynamics using the eigen-face image sequence. During the recognition process, the test sequence is analyzed over time by the HMM corresponding to each subject; its identity is determined by the model providing the highest likelihood. In case of face recognition with low-quality images, the HMM-based method was shown to produce better results than image-based methods [47]. Considering PCA features may not be sufficiently discriminative among multiple head poses, Kim *et al.* [55] proposed to use Linear Discriminant Analysis (LDA) coupled with the modeled pose dynamics in the HMM framework. By fusing pose-discriminant and person-discriminant features over the video sequence, their approach leads to superior performance, e.g., recognition rate of over 70% on a YouTube video set containing 35 celebrities in 1500 video sequences.

Aggarwal *et al.* [2] posed video-based face recognition as a dynamical system identification problem. A moving face is modeled as a linear dynamical system, and each frame is regarded as the output of the system. They adopted an autoregressive and moving average (ARMA) model to represent such a system. The similarity between gallery and probe video sequences is computed using principal angle based metrics. Their approach performs well on the data set of 16 subjects and the UCSD/Honda database [64]. Gorodnichy [43] proposed to use the neuro-associative principle for face recognition, according to which both memorization and recognition are done based on a flow of frames. The temporal dependence between consecutive images is considered by adding extra neurons. This approach achieves

Fig. 4 Example frames from video sequences in the IIT-NRC database.

recognition rate of over 95% on the IIT-NRC database[1] of 11 subjects. Some example frames of this database are shown in Fig. 4.

Recently texture-based spatiotemporal representations have been exploited for analyzing faces in video. In [117], volume Local Binary Patterns (LBP) based description was applied to facial expression recognition. Hadid *et al.* [50, 49] proposed to use local volumetric spatio-temporal features for face recognition, by considering a face sequence as a selected set of volumes from which local histograms of extended volume LBP are extracted. They adopted the boosting scheme to learn the discriminative local volumetric features, where all combination of sequence pairs are considered as the intra- and extra-classes. Experimental results on three public databases demonstrate their approach provides superior recognition performance. We compare the reported recognition performance on several public datasets in Table 2.

3.3 Image-Set Matching Based Approaches

While in some cases temporal dynamics could be exploited, in a more general scenario, the extracted face images may not be temporally consecutive. This could be due to the practical limitations in current face acquisition process, e.g., it is difficult to continuously detect or track face from every video frame, or the images are the sparse and unordered observations collected over an long periods of time and from multiple viewpoints. It is then not possible to model facial dynamics for face recognition. Image-set matching based approaches formulate face recognition as matching a probe image set against all the gallery image sets each of which representing one subject, without assuming temporal coherence between consecutive images. They can be applied to image sets containing ordered observations collected over consecutive time steps, and also image sets acquired by multiple independent still shots (e.g., photo collections) or long-term discontinuous observations.

In [59], relevant approaches to image set classification are divided into two categories: non-parametric sample-based and parametric model-based. Non-parametric sample-based approaches are based on matching pair-wise samples in the image

[1] http://synapse.vit.iit.nrc.ca/db/video/faces/cvglab.

Table 2 Recognition results on several public databases in the literature.

Database	Subjects/Sequences	Approach	Result(%)
Honda/UCSD[64]	20/40	Probabilistic appearance manifold [64]	93.2
		Exemplar-based probabilistic voting [47]	86.5
		System identification[2]	90.0
		Extended probabilistic appearance manifold [65]	98.8
		HMM (LDA+LandMark Template) [55]	100
		Boosted extended volume LBP [49]	96.0
		Manifold-Manifold Distance [106]	96.9
MoBo[46]	24/96	Adaptive HMM [79]	98.8
		Exemplar-based probabilistic voting [47]	90.8
		Boosted extended volume LBP [49]	97.9
		Manifold-Manifold Distance [106]	93.6
XM2VTS[81]	295/1180	Multi-level discriminant subspace analysis [101]	99.3
		Multi-classifier (voting or sum) [101]	99.3
		Spatial-temporal embedding [78]	99.3

sets. To recognize the two image sets as the same class, a solution would be to find the representative images from each image set and then measure their similarity. These approaches have been discussed in Section 3.1. Parametric model-based approaches tend to represent each image set by a parametric distribution and then measure the similarity between two distributions [93, 7, 10]. This is based on the assumption that images are drawn from some distributions on the underlying face pattern manifold, and normally statistical learning algorithms are adopted to model the distribution. Recently, following the mutual subspace method [110], many approaches build a compact model of the distribution by representing each image set as a linear subspace, and measure their similarity using the canonical angles [110, 82, 59]. In the following sections, we discuss these two groups of approaches: statistical model-based and mutual subspace-based, respectively.

3.3.1 Statistical Model-Based Approaches

Shakhnarovich *et al.* [93] cast face recognition from an image set as a statistical hypothesis testing problem, with the assumption that images are independently and identically (i.i.d) drawn samples from a probability density function (pdf). They proposed to classify sets of face images by comparing the probability distributions of the probe set and the gallery sets. Specially, they estimated the face appearance distribution by a multivariate Gaussian, and used the Kullback-Leibler (KL) divergence, which quantifies how well a particular pdf describes samples from another

pdf, to measure the similarity. Evaluation on two data sets of frontal face images, with 29 subjects and 23 subjects respectively, demonstrates their approach achieves equal or improved recognition performance compared to image-based recognition methods and the mutual subspace method. However, to make the divergence computation tractable, they made a crude assumption that face patterns are normally distributed, which may not be true [108]. Arandjelovic and Cipolla [4, 7] argued against the use of KL divergence due to its asymmetry, and proposed to use the Resistor-Average Distance (RAD) as the dissimilarity measure between distributions. They adopted kernel PCA to solve the closed-form expression of RAD between two Gaussian distributions, which also allows for expressive modeling of nonlinear face manifolds. In addition, a data-driven approach was used for stochastic manifold repopulating, in order to generalize unseen modes of variation. Their approach achieves recognition rate of 98% on a database of 100 individuals collected under varying imaging conditions, outperforming KL divergence based approaches and the mutual subspace method.

To deal with nonlinear variations in face appearance due to illumination and pose changes, Arandjelovic *et al.* [10] model the face appearance distribution as Gaussian Mixture Models (GMMs) on low-dimensional manifolds. The KL divergence was adopted to measure the similarity between the estimated distributions. The advantage of this approach over the previous kernel method [4, 7] lies in its better modeling of distributions confined to nonlinear manifolds; however this benefit comes at the cost of increased difficulty of divergence computation. The KL divergence, which for GMMs cannot be computed in the closed form, is evaluated by a Monte-Carlo algorithm. They evaluated the proposed method on a data set with 100 subjects, and obtained the average performance of 94%. In [6], they derived a local manifold illumination invariant, and formulated the face appearance distribution as a collection of Gaussian distributions corresponding to clusters obtained by K-means. The image set matching was performed by pair-wisely comparing all clusters of two manifolds and the maximal of cluster similarities is chosen as the overall manifold similarity. To compare two Gaussian clusters, they proposed to find the most probable mode of mutual variations between the two clusters. Recently they [9] proposed to decompose each face appearance manifold into three Gaussian pose clusters describing small face motion around different head poses. Given two manifolds, the corresponding pose clusters are compared, and the pair-wise comparisons are combined to measure the similarity between manifolds. To achieve illumination invariant recognition, they considered coarse region-based Gamma correlation with fine illumination manifold-based normalization. Their approach demonstrated consistently superior recognition performance on a database with 60 individuals.

Statistical model-based approaches make strong assumptions about the underlying distributions. The main drawbacks are that they need to solve the difficult parameter estimation problem and they easily fail when the training sets and the testing sets have weak statistical relationships, for example, when they are from different ranges of poses, expressions or illumination changes.

3.3.2 Mutual Subspace-Based Approaches

The distribution of a set of face images can be compactly represented by a lower-dimensional linear subspace. Yamaguchi *et al.* [110] introduced the Mutual Subspace Method (MSM), where each image set is represented by the linear subspace spanned by the principal components of the images. The similarity between image sets is measured by the smallest principal angles between subspaces. Principal angles [51] are the minimal angles between vectors of two subspaces, which reflect the common modes of variation of two subspaces. Canonical correlations, which are cosines of principal angles, are often used as the similarity measure. Later, to make it insensitive to variations such as pose and illumination changes, MSM was extended to the Constrained Mutual Subspace Method (CMSM) [35]. In CMSM, the test subspace and the reference subspace are projected onto a constraint subspace, where each subspace exhibits small variance and the two subspaces could be better separated. A real-time system implemented using CMSM was demonstrated in [62]. In [82], the authors further introduced the Multiple Constrained Mutual Subspace Method (MCMSM), which generates multiple constraint subspaces by using the ensemble learning algorithms (Bagging and Boosting). The similarities on each constraint subspace are combined for recognition. They conducted experiments on a database of 50 subjects and a database of 500 subjects, and experimental results show improved recognition performance compared to MSM and CMSM.

An attractive feature of MSM-based methods is their computational efficiency [56]: principal angles between linear subspaces can be computed rapidly, while the estimation of linear subspaces can be performed in an incremental manner. However, the simplistic modeling using a linear subspace cannot sufficiently model complex and nonlinear face appearance variations, and is sensitive to particular data variations. It is also argued in [93, 106] that MSM-based methods could not consider the entire probabilistic model of face variations, since the eigenvalues corresponding to the principal components, as well as the means of the samples, are disregarded.

There have been some attempts to extend MSM-based methods for nonlinear subspaces or manifolds [108, 56, 106]. Wolf and Shashua [108] introduced to compute principal angles between nonlinear manifolds using the kernel trick. However, as in all kernel approaches, finding the optimal kernel function is a difficult problem. Kim *et al.* [56, 57] argued that MSM-based methods have two shortcomings: the limited capability of modeling nonlinear pattern variations and the ad hoc fusion of information contained in different principal angles. They extended the concept of principal angles to nonlinear manifolds by combining global manifold variations with local variations, where the locally linear manifold patches are obtained using mixtures of Probabilistic PCA. The similarity between manifolds is computed as a weighted average of the similarity between global modes of data variation and the best matching local patches. They further adopted AdaBoost to learn the application-optimal principal angle fusion. Experiments on a database with 100 subjects demonstrate the above two nonlinear manifold modeling approaches both achieve superior performance to the basic MSM. By decomposing the nonlinear manifold as a collection of local linear models, each depicted by a linear subspace, Wang *et al.* [106]

introduced to compute the Manifold-Manifold Distance, which is defined as the distance of the closest subspace pair from two manifolds. Regarding the subspace-subspace distance, they argued principal angles mainly reflect the common modes of variation between two subspaces while ignoring the data itself, so they proposed to also consider the sample means in the local models to measure the local model similarity. Experiments on two public databases demonstrate their approach produces superior performance to the MSM method.

Using canonical correlations as the distance measure of image sets, Kim *et al.* [58, 59] proposed a discriminative learning method for image set classification. They developed a linear discriminant function that maximizes canonical correlations of within-class sets and minimizes canonical correlations of between-class sets. Image sets transformed by the discriminant function are then compared by the canonical correlations. There approach was evaluated on various object recognition tasks, achieving consistently superior recognition performance. In [38], a loss based Regularized LDA is introduced for face image set classification using canonical correlations.

4 Face Retrieval in Video

Face recognition could be used for video content description, indexing, and retrieval [25, 96]. The dramatic increase of videos demands more efficient and accurate access to video content. Finding a specific person in videos is essential to understand and retrieve videos [91]. Face information is an important cue for person identification in many types of videos such as news programs, dramas, movies, and home-made videos. Face retrieval enables many new functionality, e.g., rapid browsing, where only shots containing a specific character are chosen to play. Face recognition also allows character-based video search, which receives growing interest due to huge amount of video content online (e.g., YouTube).

Face retrieval in general is to retrieve shots containing particular persons/actors given one or more query face images. In context of videos captured in real-life scenarios (e.g., news, programs, and films), lighting variations, scale changes, facial expressions and varying head pose drastically change the appearance of faces. Partial occlusions, because of the objects in front of faces or resulting from hair style changes also cause problems. Artefacts caused by motion blur and low resolution are also common. In brief, the uncontrolled imaging conditions makes face retrieval very challenging [5]. Some example faces in films are shown in Fig. 5. The existing studies mainly address two kinds of applications: person retrieval and cast listing. In this section, we review relevant approaches for these applications.

Person Retrieval — Everingham and Zisserman [29, 28, 30] addressed finding particular characters in situation comedies, given a number of labeled faces (for each character) as training data. They [29] used a skin color model with multi-scale blob detection to detect candidate face regions in the input frame. To deal with pose variations, a coarse 3D ellipsoid head model with multiple texture maps was used to render faces from the train data at the same pose as the target face. The identity of

Fig. 5 Faces in films exhibits a great variability depending on the extrinsic imaging conditions.

the target face is determined by comparing it with the rendered views. The texture maps of the model can be automatically updated as new poses and expressions are detected. In [30], rendered images are used to train a discriminative tree-structured classifier, which detects the individual and estimates the pose over a range of scale and pose. The identity is verified using a generative approach. Their approach was evaluated on 4,400 key-frames (1,500 key-frames in [29]) from a TV situation comedy for detecting three characters. They [28] proposed to synthesize additional training data from a single training image by fitting the 3D model to the person's head. The parts-based constellation models are trained, which propose candidate detections in the input frame. The 3D model is aligned to the detected parts, and the global appearance is considered for recognition. Sivic *et al.* [96] presented a video shot retrieval system based on faces in the shot. Instead of matching single faces, they proposed to match sets of faces for each person, where sets of faces (called face-tracks) are gathered automatically in shots by tracking. Each face in the face-track is described by a collection of five SIFT descriptors around salient facial features. The entire face-track is represented as a distribution (i.e., histogram) over vector quantized face exemplars, resulting in a single feature vector. Face-tracks are matched by comparing the vectors using the chi-square statistics.

Arandjelovic and Zisserman [11, 12] built a system to retrieve film shots based on the presence of specific characters, given one or multiple query face images. To address the variations on scale, pose, and illumination, as well as occlusion, encountered in films, they proposed to obtain a *signature image* by a cascade of processing steps for each detected face. In the first step, facial feature are detected by SVMs that are trained on image patches surrounding eyes and mouth. Face images are then affine warped to have salient facial features aligned. Considering the bounding box typically contains background clutter, the face is segmented from the surrounding background using the face outline, which is detected by combining a learnt prior on the face shape and a set of measurements of intensity discontinuity. Finally illumination effects are removed by band-pass filtering. The end signature image is insensitive to illumination changes, pose variations, and background clutter, and mainly depends on the person's identity (and expression). Signature images are matched using a distance measure for person retrieval. Evaluations on several feature-length films demonstrate that their system achieves recall rates (over 92%) whilst maintaining good precision (over 93%). In [90], Satoh presented comparative evaluation of face sequence matching methods in drama videos.

Face information has been combined with text information for face or person retrieval in video [21, 111, 53, 112, 84]. In [111], multimodal content in videos, including names occurred in the transcript, face information, anchor scenes, and

the timing pattern between names and appearances of people, are exploited to find specific persons in broadcast news videos. However, face information was given very small weight in their system. Ozkan and Duygulu [84] also integrated text and face information for person retrieval. Specifically, they limit the search space for a query name by choosing the shots around which the name appears. To find the most-frequently occurring faces in the space, they construct a graph with nodes corresponding to all faces in the search space, and edges corresponding to the similarity of the faces; the problem is transformed into finding the densest component in the graph. A limitation of their approach is that it cannot find a person in parts of the video where his/her name is not mentioned. Zhao and Yagnik [118] presented a large scale system that can automatically learn and recognize faces in web videos by combining text, image, and video. To address the difficulty of manually labeling training data for a large set of people, they used the text-image co-occurrence in the web as a weak signal of relevance, and proposed consistence learning to learn the set of face models from the very large and nosy training set.

Cast Listing — An interesting face retrieval problem is automatically determining the cast of a feature-length film. Cast listing has been mainly based on the recognition of faces, as faces being the most repeatable cue in this setting [5], although others cues, such as clothes, can be used as additional evidence. It is challenging because the cast size is unknown, with great face appearance changes caused by extrinsic imaging factors (e.g., illumination, pose, expression). Fitzgibbon and Zisserman [33] made an earlier attempt to this problem using affine invariant clustering. They developed a distance metric that is invariant to affine transformations. Two classes of priors were considered in the distance metric: deformation priors between any pair of frames, and temporal coherence prior between continuous frames. To address the lighting variations, they utilized a simple bandpass filter to pre-process the detected faces. Their approach was tested on the film 'Groundhog Day' with a principal cast of around 20. Arandjelovic and Cipolla [5] introduced an approach based on clustering over face appearance manifolds, which correspond to sequences of moving faces in a film. They temporally segment the video into shots, and obtain face tracks by connecting face detections through time. The CMSM method [35] is adopted for pair-wise comparisons of face tracks (i.e., face manifolds). To allow unsupervised discriminative learning on an unlabeled set of video sequences, their approach starts from a generic discriminative manifold and converges to a data-specific one, automatically collecting within-class data. Evaluation on a situation comedy illustrates the effectiveness of their method. However, it remains challenging to obtain a small number of clusters per character without merging multiple characters into a single cluster. In [36], normalized Graph Cuts is adopted to cluster face tracks for cast indexing.

In the above cast listing systems, all faces of a particular character are collected into one or a few clusters, which are then assigned a name manually. Everingham *et al.* [26, 27] addressed the problem of automatically labeling faces of characters in TV or film materials with their names. Similar to the "Faces in the News" labeling in [16], where detected frontal faces in news images are tagged with names

appearing in the news story text, they proposed to combine visual cues (face and cloth) and textual cues (subtitle and transcript) for assigning names. Regarding face processing [3], face detections in each frame are linked to derive face tracks, and each face is represented by local appearance descriptors computed around 13 facial features. Clothing appearance was also considered as additional cues. They align the transcripts with subtitles using dynamic time warping to obtain textual annotation, and use visual speaker detection to resolve the ambiguities, i.e., only associating names with face tracks where the face is detected as speaking. A nearest neighbor [26] or SVM [27] classifier, trained on labeled tracks, is used to classify the unlabeled face tracks. Their approach has demonstrated promising performance on three 40 minute episodes of a TV serial. In [97], the approach was further extended for improved coverage by character detection and recognition in profile views. Considering there are not enough temporally local name cues in the subtitle and script for local face-name matching, Zhang *et al.* [116] proposed to perform a global matching between the clustered face tracks and the names extracted from the film script. A graph matching method is utilized to build face-name association between a face affinity network and a name affinity network. Experiments on ten films demonstrate encouraging results.

Ramanan *et al.* [87] introduced a semi-supervised method for building large labeled face datasets by leveraging archival video. They implemented a system for labeling archival footage spanning 11 years from the television show Friends. The dataset they compiled consists of more than 600,000 faces, containing appearance variations due to age, weight gain, changes in hairstyles, and other factors. In their system, at the lowest level, detected frontal faces are clustered and tracked using the color histogram of the face, hair and torso. By part-based color tracking, faces with extreme pose changes are also collected in the clusters. The resulting face tracks are reliable since body appearance is stable over short time scales. At the scene level, face tracks are grouped by an agglomerative clustering procedure based on body appearance, since people tend to ware consistent clothes within a scene. They manually labeled the clusters from a single reference episode, which are used to label the dataset using a nearest-neighbors framework.

5 Challenges and Future Directions

With camera sensors become pervasive in our society, video data has been one of the main sensory inputs in electronic systems. Meantime, huge amounts of video content have been generated. Face recognition in video, which has many applications such as biometric identification, video search and retrieval, visual surveillance, and human-computer interaction, has received much interest in the last decade. Although much progress has been made, the problem remains difficult for videos captured in unconstrained settings. Some major challenges that should be addressed in future research are considered here:

- **Databases:** Most of the existing public datasets [81, 46, 64] were collected under controlled (laboratory) conditions, with limited number of subjects, covering

limited face appearance variations. It is believed that databases built from "the wild" are important for training and evaluating real-world recognition systems. Not only does lack of the large realistic database prevent suitable evaluation of the existing techniques, but also provides little encouragement to novel ideas. Recently the "Labeled Faces in the Wild" database [52] has been collected for unconstrained face recognition. The database contains more than 13,000 labeled face photographs collected from the web, spanning the range of conditions encountered in real life. However, only frontal face images are included in this database. We believe a large database consisting of face videos "in the wild" is necessary for future research. How to build a comprehensive face video database with low manual effort should be investigated [87].

- **Low-quality Video Data:** In many real-life applications, the video data is of low quality (e.g., limited imaging resolution or low frame rate), such as video footage from surveillance cameras and videos captured by consumers via mobile or wearable cameras. The low-quality data could be caused by the poor sensor performance, motion blur, environmental factors such as illumination, or video compression. The sensors in the non-visible spectrum (e.g. Near-Infrared) also generate low-resolution videos with much noise. Existing face recognition approaches mainly focuses on good-quality video data, which cannot be directly applied to low-quality video data. To enable practical applications, it is necessary to investigate face recognition techniques for low-quality video data. Super-resolution could be a solution [8], and temporal information in the video could be used to compensate for the lost spatial information.
- **Computational Cost:** Many applications of face recognition can be foreseen on platforms with limited computing power, for example, video retrieval in mobile devices, smart cameras for video surveillance, user interface of consumer electronics (e.g., toy robotics). The processing power on these devices is limited for traditional video processing tasks like face recognition. Although advanced hardware design and algorithm optimization could be helpful to certain extent [32], most of existing video-based face recognition approaches require high computation, which prevents them for wide applications in low-performance systems. Therefore, there is a great need to investigate low-cost algorithms for face recognition in video.

6 Conclusions

Face recognition in video has been an active topic in computer vision, due to many potential applications. This chapter brings a comprehensive review on existing research in this area. Different types of approaches to this problem are discussed. We also introduce recent work on face retrieval. Finally, some challenges and research directions are discussed. Although we mainly focus on video-based approaches, recent years have witnessed some interesting still-image based approaches (e.g., [109]), which could be helpful for face recognition in video.

References

1. Abate, A.F., Nappi, M., Riccio, D., Sabatino, G.: 2d and 3d face recognition: A survey. Pattern Recognition Letters 28(14), 1885–1906 (2007)
2. Aggarwal, G., Roy-Chowdhury, A.K., Chellappa, R.: A system identification approach for video-based face recognition. In: International Conference on Pattern Recognition (ICPR), pp. 175–178 (2004)
3. Apostoloff, N., Zisserman, A.: Who are you? - real-time person identification. In: British Machine Vision Conference (BMVC), pp. 509–518 (2007)
4. Arandjelović, O., Cipolla, R.: Face recognition from image sets using robust kernel resistor-average distance. In: International Workshop on Face Processing in Video, p. 88 (2004)
5. Arandjelović, O., Cipolla, R.: Automatic cast listing in feature-length films with anisotropic manifold space. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 1513–1520 (2006)
6. Arandjelović, O., Cipolla, R.: Face set classification using maximally probable mutual modes. In: International Conference on Pattern Recognition (ICPR), vol. 1, pp. 511–514 (2006)
7. Arandjelović, O., Cipolla, R.: An information-theoretic approach to face recognition from face motion manifolds. Image and Vision Computing 24(6), 639–647 (2006)
8. Arandjelović, O., Cipolla, R.: A manifold approach to face recognition from low quality video across illumination and pose using implicit super-resolution. In: IEEE International Conference on Computer Vision, ICCV (2007)
9. Arandjelović, O., Cipolla, R.: A pose-wise linear illumination manifold model for face recognition using video. Computer Vision and Image Understanding 113(1), 113–125 (2009)
10. Arandjelović, O., Shakhnarovich, G., Fisher, G., Cipolla, J.R., Zisserman, R.,, A.: Face recognition with image sets using manifold density divergence. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 581–588 (2005)
11. Arandjelović, O., Zisserman, A.: Automatic face recognition for film character retrieval in feature-length films. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 860–867 (2005)
12. Arandjelović, O., Zisserman, A.: On film character retrieval in feature-length films. In: Hammoud, R. (ed.) Interactive Video: Algorithms and Technologies. Springer, Heidelberg (2006)
13. Arulampalam, M., Maskell, S., Gordon, N., Clapp, T.: A tutorial on particle filters for online nonlinear/non-gaussian bayesian tracking. IEEE Transactions on Signal Processing 50(2), 174–189 (2002)
14. Avidan, S.: Support vector tracking. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(8), 1064–1072 (2004)
15. Belhumeur, P.N., Hespanha, J.P., Kriegman, D.J.: Eigenfaces vs. fisherfaces: Recognition using class specific linear projection. IEEE Transactions on Pattern Analysis and Machine Intelligence 19(7), 711–720 (1997)
16. Berg, T.L., Berg, A.C., Edwards, J., Maire, M., White, R., Teh, Y.W., Learned-Miller, E., Forsyth, D.A.: Names and faces in the news. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 848–854 (2004)
17. Berrani, S.A., Garcia, C.: Enhancing face recognition from video sequences using robust statistics. In: IEEE International Conference on Advanced Video and Signal-Based Surveillance (AVSS), pp. 324–329 (2005)

18. Biuk, Z., Loncaric, S.: Face recognition from multi-pose image sequence. In: International Symposium on Image and Signal Processing and Analysis, pp. 319–324 (2001)
19. Bradski, G.: Computer vision face tracking for use in a perceptual user interface. Intel Technology Journal (Q2) (1998)
20. Chellappa, R., Aggarwal, G.: Video biometrics. In: International Conference on Image Analysis and Processing, pp. 363–370 (2007)
21. Chen, M.Y., Hauptmann, A.: Searching for a specific person in broadcast news video. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 1036–1039 (2004)
22. Chen, Y.N., Han, C.C., Wang, C.T., Jeng, B.S., Fan, K.C.: A cnn-based face detector with a simple feature map and a coarse-to-fine classifier. IEEE Transactions on Pattern Analysis and Machine Intelligence (2010)
23. Comaniciu, D., Ramesh, V., Meer, P.: Real-time tracking of non-rigid objects using mean shift. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 142–149 (2000)
24. Edwards, G., Taylor, C., Cootes, T.: Improving identification performance by integrating evidence from sequences. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1486–1491 (1999)
25. Eickeler, S., Wallhoff, F., Lurgel, U., Rigoll, G.: Content based indexing of images and video using face detection and recognition methods. In: IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP), pp. 1505–1508 (2001)
26. Everingham, M., Sivic, J., Zisserman, A.: "hello! my name is.. buffy" automatic naming of characters in tv video. In: British Machine Vision Conference (BMVC), pp. 889–908 (2006)
27. Everingham, M., Sivic, J., Zisserman, A.: Taking the bite out of automated naming of characters in tv video. Image and Vision Computing 27(5), 545–559 (2009)
28. Everingham, M., Zisserman, A.: Automated visual identification of characters in situation comedies. In: International Conference on Pattern Recognition (ICPR), pp. 983–986 (2004)
29. Everingham, M., Zisserman, A.: Automatic person identification in video. In: International Conference on Image and Video Retrieval (CIVR), pp. 289–298 (2004)
30. Everingham, M., Zisserman, A.: Identifying individuals in video by combining 'generative' and discriminative head models. In: IEEE International Conference on Computer Vision (ICCV), vol. 2, pp. 1103–1110 (2005)
31. Fan, W., Yeung, D.Y.: Locally linear models on face appearance manifolds with application to dual-subspace based classification. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 2, pp. 1384–1390 (2006)
32. Farrugia, N., Mamalet, F., Roux, S., Yang, F., Paindavoine, M.: Fast and robust face detection on a parallel optimized architecture implemented on fpga. IEEE Transactions on Circuits and Systems for Video Technology 19(4), 597–602 (2009)
33. Fitzgibbon, A., Zisserman, A.: On affine invariant clustering and automatic cast listing in movies. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2352, pp. 289–298. Springer, Heidelberg (2002)
34. Freund, Y., Schapire, R.E.: A decision-theoretic generalization of on-line learning and an application to boosting. Journal of Computer and System Sciences 55(1), 119–139 (1997)
35. Fukui, K., Yamaguchi, O.: Face recognition using multi-viewpoint patterns for robot vision. In: International Symposium of Robotics Research, pp. 192–201 (2003)
36. Gao, Y., Wang, T., Li, J., Du, Y., Hu, W., Zhang, Y., Ai, H.: Cast indexing for videos by ncuts and page ranking. In: International Conference on Image and Video Retrieval (CIVR), pp. 441–447 (2007)

37. Garcia, C., Delakis, M.: Convolutional face finder: A neural architecture for fast and robust face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(11), 1408–1423 (2004)
38. Geng, Y., Shan, C., Hao, P.: Square loss based regularized lda for face recognition using image sets. In: IEEE CVPR Workshop on Biometrics, pp. 99–106 (2009)
39. Goh, R., Liu, L., Liu, X., Chen, T.: The cmu face in action (fia) database. In: IEEE International Workshop on Analysis and Modeling of Faces and Gestures (AMFG), pp. 255–263 (2005)
40. Gorodnichy, D.O.: On importance of nose for face tracking. In: IEEE International Conference on Automatic Face & Gesture Recognition (FG), pp. 181–186 (2002)
41. Gorodnichy, D.O.: Facial recognition in video. In: Kittler, J., Nixon, M.S. (eds.) AVBPA 2003. LNCS, vol. 2688, pp. 505–514. Springer, Heidelberg (2003)
42. Gorodnichy, D.O.: Recognizing faces in video requires approaches different from those developed for face recognition in photographs. In: Workshop on Enhancing Information System Security through Biometrics (2004)
43. Gorodnichy, D.O.: Video-based framework for face recognition in video. In: International Workshop on Face Processing in Video, pp. 330–338 (2005)
44. Grabner, H., Grabner, M., Bischof, H.: Real-time tracking via on-line boosting. In: British Machine Vision Conference (BMVC), pp. 47–56 (2006)
45. Grabner, H., Leistner, C., Bischof, H.: Semi-supervised on-line boosting for robust tracking. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part I. LNCS, vol. 5302, pp. 234–247. Springer, Heidelberg (2008)
46. Gross, R., Shi, J.: The cmu motion of body (mobo) database. Tech. rep., Robotics Institute, Carnegie Mellon University (2001)
47. Hadid, A., Pietikäinen, M.: From still image to video-based face recognition: An experimental analysis. In: IEEE International Conference on Automatic Face & Gesture Recognition (FG), pp. 813–818 (2004)
48. Hadid, A., Pietikäinen, M.: An experimental investigation about the integration of facial dynamics in video-based face recognition. Electronic Letters on Computer Vision and Image Analysis 5(1), 1–13 (2005)
49. Hadid, A., Pietikäinen, M.: Combining appearance and motion for face and gender recognition from videos. Pattern Recognition 42(11), 2818–2827 (2009)
50. Hadid, A., Pietikäinen, M., Li, S.: Learning personal specific facial dynamics for face recognition from videos. In: IEEE International Workshop on Analysis and Modeling of Faces and Gestures (AMFG), pp. 1–15 (2007)
51. Hotelling, H.: Relations between two sets of variates. Biometrika 8, 321–377 (1936)
52. Huang, G., Ramesh, M., Berg, T., Learned-Miller, E.: Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Tech. Rep. 07-49, University of Massachusetts, Amherst (2007)
53. Ikizler, N., Duygulu, P.: Person search made easy. In: Leow, W.-K., Lew, M., Chua, T.-S., Ma, W.-Y., Chaisorn, L., Bakker, E.M. (eds.) CIVR 2005. LNCS, vol. 3568, pp. 578–588. Springer, Heidelberg (2005)
54. Jain, A., Ross, A., Prabhakar, S.: An introduction to biometric recognition. IEEE Transactions on Circuits and Systems for Video Technology 14(1), 4–20 (2004)
55. Kim, M., Kumar, S., Pavlovic, V., Rowley, H.: Face tracking and recognition with visual constraints in real-world videos. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2008)
56. Kim, T., Arandjelović, O., Cipolla, R.: Learning over sets using boosted manifold principal angles (bomva). In: British Machine Vision Conference (BMVC), vol. 2, pp. 779–788 (2005)

57. Kim, T.K., Arandjelović, O., Cipolla, R.: Boosted manifold principal angles for image set-based recognition. Pattern Recognition 40(9), 2475–2484 (2007)
58. Kim, T.K., Kittler, J., Cipolla, R.: Learning discriminative canonical correlations for object recognition with image sets. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3953, pp. 251–262. Springer, Heidelberg (2006)
59. Kim, T.K., Kittler, J., Cipolla, R.: Discriminative learning and recognition of image set classes using canonical correlations. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(6), 1005–1018 (2007)
60. Knight, B., Johnson, A.: The role of movement in face recognition. Visual Cognition 4(3), 265–273 (1997)
61. Kong, S., Heo, J., Abidi, B., Paik, J., Abidi, M.: Recent advances in visual and infrared face recognition - a review. Computer Vision and Image Understanding 97(1), 103–135 (2005)
62. Kozakaya, T., Nakaia, H.: Development of a face recognition system on an image processing lsi chip. In: International Workshop on Face Processing in Video, p. 86 (2004)
63. Krueger, V., Zhou, S.: Exemplar-based face recognition from video. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2353, pp. 732–746. Springer, Heidelberg (2002)
64. Lee, K.C., Ho, J., Yang, M.H., Kriegman, D.: Video-based face recognition using probabilistic appearance manifolds. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 313–320 (2003)
65. Lee, K.C., Ho, J., Yang, M.H., Kriegman, D.: Visual tracking and recognition using prababilistic appearance manifolds. Computer Vision and Image Understanding 99(3), 303–331 (2005)
66. Lee, K.C., Kriegman, D.: Online learning of probabilistic appearance manifolds for video-based recognition and tracking. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), vol. 1, pp. 852–859 (2005)
67. Li, B., Chellappa, R.: Simultaneous tracking and verification via sequential posterior estimation. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 110–117 (2000)
68. Li, B., Chellappa, R.: Face verification through tracking facial features. Journal of Optical Society of America 18(12), 2969–2981 (2001)
69. Li, B., Chellappa, R.: A generic approach to simultaneous tracking and verification in video. IEEE Transactions on Image Processing 11(5), 530–544 (2002)
70. Li, S.Z., Jain, A.K. (eds.): Handbook of Face Recognition. Springer, New York (2005)
71. Li, S.Z., Zhang, Z.: Floatboost learning and statistical face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 26(9), 1–12 (2004)
72. Li, Y.: Dynamic face models: Construction and applications. Ph.D. thesis, Queen Mary, University of London (2001)
73. Li, Y., Gong, S., Liddell, H.: Recognising trajectories of facial identities using kernel discriminant analysis. In: British Machine Vision Conference (BMVC), pp. 613–622 (2001)
74. Li, Y., Gong, S., Liddell, H.: Video-based online face recognition using identity surfaces. In: IEEE Workshop on Recognition, Analysis and Tracking of Faces and Gestures in Real-time Systems, pp. 40–46 (2001)
75. Li, Y., Gong, S., Liddell, H.: Constructing facial identity surfaces for recognition. International Journal of Computer Vision 53(1), 71–92 (2003)
76. Lienhart, R., Kuranov, D., Pisarevsky, V.: Empirical analysis of detection cascades of boosted classifiers for rapid object detection. In: DAGM 25th Pattern Recognition Symposium, Madgeburg, Germany, pp. 297–304 (2003)

77. Liu, C.: A bayesian discriminating features method for face detection. IEEE Transactions on Pattern Analysis and Machine Intelligence 25(6), 725–740 (2003)
78. Liu, W., Li, Z., Tang, X.: Spatial-temporal embedding for statistical face recognition from video. In: Leonardis, A., Bischof, H., Pinz, A. (eds.) ECCV 2006. LNCS, vol. 3952, pp. 374–388. Springer, Heidelberg (2006)
79. Liu, X., Chen, T.: Video-based face recognition using adaptive hidden markov models. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 340–345 (2003)
80. Martinez, A.M.: Recognizing imprecisely localized, partially occluded and expression variant faces from a single sample per class. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(6), 748–763 (2002)
81. Messer, K., Matas, J., Kittler, J., Lüttin, J., Matitre, G.: Xm2vtsdb: The extended m2vts database. In: International Conference on Audio and Video-Based Person Authentication (AVBPA), pp. 72–77 (1999)
82. Nishiyama, M., Yamaguchi, O., Fukui, K.: Face recognition with the multiple constrained mutual subspace method. In: Kanade, T., Jain, A., Ratha, N.K. (eds.) AVBPA 2005. LNCS, vol. 3546, pp. 71–80. Springer, Heidelberg (2005)
83. O'Toole, A., Roark, D., Abdi, H.: Recognizing moving faces: A psychological and neural synthesis. Trends in Cognitive Sciences 6(6), 261–266 (2002)
84. Ozkan, D., Duygulu, P.: Finding people frequently appearing in news. In: International Conference on Image and Video Retrieval (CIVR), pp. 173–182 (2006)
85. Park, U., Jain, A.K., Ross, A.: Face recognition in video: Adaptive fusion of multiple matchers. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2007)
86. Phillips, P.J., Flynn, P.J., Scruggs, T., Bowyer, K.W., Chang, J., Hoffman, K., Marques, J., Min, J., Worek, W.: Overview of the face recognition grand challenge. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 947–954 (2005)
87. Ramanan, D., Baker, S., Kakade, S.: Leveraging archival video for building face datasets. In: IEEE International Conference on Computer Vision (ICCV), pp. 1–8 (2007)
88. Ross, D., Lim, J., Lin, R.S., Yang, M.H.: Incremental learning for robust visual tracking. International Journal of Computer Vision 77(1-3), 125–141 (2008)
89. Roweis, S.T., Saul, L.K.: Nonlinear dimensionality reduction by locally linear embedding. Science 290, 2323–2326 (2000)
90. Satoh, S.: Comparative evaluation of face sequence matching for content-based video access. In: IEEE International Conference on Automatic Face & Gesture Recognition (FG), pp. 163–168 (2000)
91. Satoh, S., Nakamura, Y., Kanade, T.: Name-it: Naming and detecting faces in news videos. IEEE Transactions on Multimedia 6(1), 22–35 (1999)
92. Schapire, R.E., Singer, Y.: Improved boosting algorithms using confidence-rated predictions. Maching Learning 37(3), 297–336 (1999)
93. Shakhnarovich, G., Fisher, J.W., Darrel, T.: Face recognition from long-term observations. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2352, pp. 851–868. Springer, Heidelberg (2002)
94. Shih, P., Liu, C.: Face detection using discriminating feature analysis and support vector machine. Pattern Recognition 39(2), 260–276 (2006)
95. Sinha, P., Balas, B., Ostrovsky, Y., Russell, R.: Face recognition by humans: Nineteen results all computer vision researchers should know about. Proceedings of the IEEE 94(11), 1948–1962 (2006)

96. Sivic, J., Everingham, M., Zisserman, A.: Person spotting: Video shot retrieval for face sets. In: International Conference on Image and Video Retrieval (CIVR), pp. 226–236 (2005)
97. Sivic, J., Everingham, M., Zisserman, A.: "who are you?" - learning person specific classifiers from video. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1145–1152 (2009)
98. Stallkamp, J., Ekenel, H.K., Stiefelhagen, R.: Video-based face recognition on real-world data. In: IEEE International Conference on Computer Vision (ICCV), pp. 1–8 (2007)
99. Tan, X., Chen, S., Zhou, Z.H., Zhang, F.: Face recognition from a single image per person: A survey. Pattern Recognition 39(9), 1725–1745 (2006)
100. Tang, X., Li, Z.: Frame synchronization and multi-level subspace analysis for video based face recognition. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 902–907 (2004)
101. Tang, X., Li, Z.: Audio-guided video-based face recognition. IEEE Transactions on Circuits and Systems for Video Technology 19(7), 955–964 (2009)
102. Tenenbaum, J.B., Silva, V., Langford, J.C.: A global geometric framework for nonlinear dimensionality reduction. Science 290, 2319–2323 (2000)
103. Turk, M., Pentland, A.P.: Face recognition using eigenfaces. In: IEEE Conference on Computer Vision and Pattern Recognition, CVPR (1991)
104. Viola, P., Jones, M.: Rapid object detection using a boosted cascade of simple features. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 511–518 (2001)
105. Viola, P., Jones, M.: Robust real-time face detection. International Journal of Computer Vision 57(2), 137–154 (2004)
106. Wang, R., Shan, S., Chen, X., Gao, W.: Manifold-manifold distance with application to face recognition based on image set. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1–8 (2008)
107. Waring, C., Liu, X.: Face detection using spectral histograms and svms. IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics 35(3), 467–476 (2005)
108. Wolf, L., Shashua, A.: Learning over sets using kernel principal angles. Journal of Machine Learning Research 4(10), 913–931 (2003)
109. Wright, J., Yang, A., Ganesh, A., Sastry, S., Ma, Y.: Robust face recognition via sparse representation. IEEE Transactions on Pattern Analysis and Machine Intelligence 31(2), 210–227 (2009)
110. Yamaguchi, O., Fukui, K., Maeda, K.: Face recognition using temporal image sequences. In: IEEE International Conference on Automatic Face & Gesture Recognition (FG), pp. 318–323 (1998)
111. Yang, J., Hauptmann, A., Chen, M.Y.: Finding person x: Correlating names with visual appearances. In: International Conference on Image and Video Retrieval (CIVR), pp. 270–278 (2004)
112. Yang, J., Rong, Y., Hauptmann, A.: Multiple-instance learning for labeling faces in broadcasting news videos. In: ACM International Conference on Multimedia, pp. 31–40 (2005)
113. Yang, M.H., Kriegman, D., Ahuja, N.: Detecting faces in images: A survey. IEEE Transactions on Pattern Analysis and Machine Intelligence 24(1), 34–58 (2002)
114. Zhang, Y., Martinez, A.M.: From static to video: Face recognition using a probabilistic approach. In: International Workshop on Face Processing in Video, pp. 78–78 (2004)
115. Zhang, Y., Martinez, A.M.: A weighted probabilistic approach to face recognition from multiple images and video sequences. Image and Vision Computing 24(6), 626–638 (2006)

116. Zhang, Y.F., Xu, C., Lu, H., Huang, Y.M.: Character identification in feature-length films using global face-name matching. IEEE Transactions on Multimedia 11(7), 1276–1288 (2009)
117. Zhao, G., Pietikäinen, M.: Dynamic texture recognition using local binary patterns with an application to facial expressions. IEEE Transactions on Pattern Analysis and Machine Intelligence 29(6), 915–928 (2007)
118. Zhao, M., Yagnik, J.: Large scale learning and recognition of faces in web videos. In: IEEE International Conference on Automatic Face & Gesture Recognition (FG), pp. 1–7 (2008)
119. Zhao, W., Chellappa, R., Phillips, P.J., Rosenfeld, A.: Face recognition: A literature survey. ACM Computing Surveys 35(4), 399–458 (2004)
120. Zhou, S., Chellappa, R.: Probabilistic human recognition from video. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002. LNCS, vol. 2352, pp. 681–697. Springer, Heidelberg (2002)
121. Zhou, S., Chellappa, R., Moghaddam, B.: Visual tracking and recognition using appearance-adaptive models in particle filters. IEEE Transactions on Image Processing 13(11), 1491–1506 (2004)
122. Zhou, S., Krueger, V., Chellappa, R.: Face recognition from video: A condensation approach. In: IEEE International Conference on Automatic Face & Gesture Recognition (FG), pp. 221–226 (2002)
123. Zhou, S., Krueger, V., Chellappa, R.: Probabilistic recognition of human faces from video. Computer Vision and Image Understanding 91(1), 214–245 (2003)

A Human-Centered Computing Framework to Enable Personalized News Video Recommendation

Hangzai Luo and Jianping Fan

Abstract. In this chapter, an interactive framework is developed to enable personalized news video recommendation and allow news seekers to access large-scale news videos more effectively. First, multiple information sources (audio, video and closed captions) are seamlessly integrated and synchronized to achieve more reliable news topic detection, and the inter-topic contextual relationships are extracted automatically for characterizing the interestingness of the news topics more effectively. Second, topic network (i.e., news topics and their inter-topic contextual relationships) and hyperbolic visualization are seamlessly integrated to achieve more effective navigation and exploration of large-scale news videos at the topic level, so that news seekers can have a good global overview of large-scale collections of news videos at the first glance. Through a hyperbolic approach for interactive topic network visualization and navigation, large amounts of news topics and their contextual relationships are visible on the display screen, and thus news seekers can obtain the *news topics of interest* interactively, build up their mental search models easily and make better search decisions by selecting the visible news topics directly. Our system can also capture the search intentions of news seekers implicitly and further recommend the most relevant news videos according to their importance and representativeness scores. Our experiments on large-scale news videos (10 TV news programs for more than 3 months) have provided very positive results.

1 Introduction

According to the CIA world factbook, there are more than 30,000 television stations in the world. These stations broadcast a large number of TV news programs

Hangzai Luo
Software Engineering Institute, East China Normal University, Shanghai, China
e-mail: hzluo@sei.ecnu.edu.cn

Jianping Fan
Dept. of Computer Science, UNC-Charlott, NC 28223, USA
e-mail: jfan@uncc.ed

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 261–281.
springerlink.com

(news videos) every day. Different organizations and individuals utilize these broadcast news videos for different purposes, such as presidential candidates' debat for public assessment, economic performance analysis and prediction, sports and crime reports. People watch the news videos (TV news programs) to understand what is happening now and predict what might happen in the near future, so that they can make better daily decisions.

Due to the large number of broadcast channels and TV news programs, finding news videos of interest is not a trivial task: (a) Most existing content-based video retrieval (CBVR) systems assume that news seekers can formulate their information needs precisely either in terms of keywords or example videos. Unfortunately, news seekers may not be able to know what is happening now (i.e., if they know it, it is not a news), thus it is very hard for them to find the suitable keywords or example videos to formulate their news needs precisely without obtaining sufficient knowledge of the available news topics of interest. Thus there is an urgent need to develop new techniques for detecting news topics of interest from large-scale news videos to assist news seekers on finding news videos of interest more effectively. (b) Because the same news topic can be discussed in many TV channels and news programs, topic-based news search may return large amounts of news videos and thus simple news search via keyword matching of news topics may bring the serious problem of information overload to news seekers. (c) Most existing CBVR systems treat all the news seekers equally while completely ignoring the diversity and rapid change of their search interests. Besides the rapid growth of broadcast TV channels and news programs, we have also observed different scenarios of news needs from different people, thus it is very difficult to come up with a *one size fits all* approach for accessing large-scale news videos. (d) The keywords for news topic interpretation may not be expressive enough for describing the rich details of video content precisely and using only the keywords may not be able to capture the search intentions of news seekers effectively. Thus visualization is becoming a critical component of personalized news video recommendation system [1-2, 9-12]. (e) The objectives for personalized video recommendation and content-based video retrieval are very different, which make it unsuitable to directly apply the existing CBVR techniques for supporting personalized video recommendation. Thus supporting personalized news video recommendation is becoming one important feature of news services [3-4].

There are some existing approaches to support personalized video recommendation by using only the associated text terms such as the titles, tags, and comments [3-4], and the relevant videos are recommended according to the matching between the associated text terms for video content description and the users' profiles. Unfortunately, the text terms, which are associated with the videos, may not have exact correspondence with the underlying video content. In addition, a sufficient collection of users' profiles may not be available for recommendation purpose. Thus there is an urgent need to develop new frameworks for supporting personalized news video recommendation, which may not completely depend on the users' profiles and the associated texts for video content description.

Context between the news topics is also very important for people to make better search decisions, especially when they are not familiar with the available news topics and their search goals or ideas are still fuzzy. The inter-topic context can give a good approximation of the interestingness of the news topics (i.e., like PageRank for characterizing the importance of web pages [17]). Thus it is very attractive to integrate topic network (i.e., news topics and their inter-topic contextual relationships) for characterizing the interestingness of the news topics, assisting news seekers on making better search decisions and suggesting the future search directions.

To incorporate topic network for supporting user-adaptive topic recommendation, it is very important to develop new algorithm for large-scale topic network visualization, which is able to provide a good balance between the local detail and the global context. The local detail is used to help news seekers focus on the news topics of interest in current focus. The global context is needed to tell news seekers where the other news topics are and their contextual relationships with the news topics in current focus, such global context can effectively suggest the new search directions to news seekers. Thus supporting visualization and interactive navigation of the topic network is becoming a complementary and necessary component for personalized news video recommendation system and it may lead to the discovery of unexpected news videos and guide the future search directions effectively.

On the other hand, the search criteria are often poorly defined or depend on the personal preferences of news seekers. Thus supporting interactive visualization, exploration and assessment of the search results are very important for allowing news seekers to find the news videos of interest according to their personal preferences. Information retrieval community has also recognized that designing more intuitive system interface for search result display may have significant effects on assisting users to understand and assess the search results more effectively [13]. To incorporate visualization for improving news search, effective techniques for intelligemt news video analysis should be developed to discover the meaningful knowledge from large-scale news videos.

Several researchers have used the ontology (i.e., video concepts and their simple inter-topic contextual relationships such as “IS-A" and “part-of") to assistant visual content anslysis and retrieval [23-24]. Because the news content are highly dynamic, the inter-topic contextual relationships cannot simply be characterized by using “IS-A" or “part-of", which are used for ontology construction. Thus it is unacceptable to incorporate the ontology for supporting personalized news video recommendation. On the other hand, automatic video understanding is still an open problem for computer vision community [25-31].

In this chapter, an interactive approach is developed to enable personalized news video recommendation, and our approach has significant differences from other existing work: (a) Rather than performing semantic video classification for automatic news video understanding, we have integrated multiple information sources to achieve more reliable news topic detection. (b) The associations among the news topics (i.e., inter-topic contextual relationships) are determined automatically and an interestingness score is automatically assigned to each news topic via statistical analysis, and such interestingness scores are further used to select the news topics

of interest and filter out the less interesting news topics automatically. (c) A hyperbolic visualization tool is incorporated to inform news seekers with a better global overview of large-scale news videos, so that they can make better search decisions and find the most relevant news videos more effectively. (d) A novel video ranking algorithm is developed for recommending the most relevant news videos according to their importance and representativeness scores.

The chapter is organized as follows. Section 2 briefly reviews some related work on news topic detection and personalized information recommendation; Section 3 introduces our work on integrating topic network and hyperbolic visualization to enable user-adaptive topic recommendation; Section 4 introduces our new scheme on news video ranking for supporting personalized news video recommendation; Section 5 summarizes our work on algorithm and system evaluation; We conclude in Section 6.

2 Related Work

To enable personalized news video recommendation, one of the most important problems is to extract *news topics of interest* automatically from large-scale news videos. This problem is becoming very critical because of the following reasons: (a) The amount of news topics could be very large; (b) Different news topics may have different importance and interestingness scores, such importance and interestingness scores may also depend on the personal preferences of news seekers. In this section, we have provided a brief review of some existing work which are critical for developing personalized news video recommendation system: (1) automatic news topic detection; (2) news visualization; (3) personalized video recommendation.

Topic extraction refers to the identification of individual stories or topics within a broadcast news video by detecting the boundaries where the topic of discussion changes. News topics may be of any length and consist of complete and cohesive news report on one particular topic. Each broadcast channel has its own peculiarities in terms of program structures and styles, which can be integrated for achieving more accurate detection of news topics and their boundaries [11-12]. News topics can also be detected by using some existing techniques for named-entity extraction [5-6].

There are two well-accepted approaches for supporting personalized information retrieval [20-22]: *content-based filtering* and *collaborative filtering*. Because the profiles for new users are not available, both the collaborative filtering approach and the content-based filtering approach cannot support new users effectively. Thus there is an urgent need to develop more effective frameworks for supporting personalized news video recommendation.

Visualization is widely used to help the users explore large amount of information and find interesting parts interactively [9-12]. Rather than recommending the most interesting news topics to news seekers, all of these existing news visualization systems disclose all the available news topics to them, and thus news seekers have to dig out the news topics of interest by themselves. When large-scale news collections come into view, the number of the available news topics could be very

large and displaying all of them to news seekers may mislead them. Thus it is very important to develop new algorithms for characterizing the interestingness of news topics and reducing the number of news topics to enable more effective visualization and exploration of large-scale news videos.

3 User-Adaptive News Topic Recommendation

In this chapter, a novel scheme is developed by incorporating *topic network* and hyperbolic visualization to recommend the *news topics of interest* for assisting news seekers on accessing large-scale news videos more effectively. To do this, an automatic scheme is developed to construct the topic network for representing and interpreting large-scale news videos at the topic level. In addition, a hyperbolic visualization technique is developed to enable interactive topic network navigation and recommend the news topics of interest according to the personal preferences and timely observations of news seekers, so that they can make better search decisions.

3.1 News Topic Detection

For TV news programs, there are three major information sources (audio, video and closed captions) that can be integrated and synchronized to enable more reliable

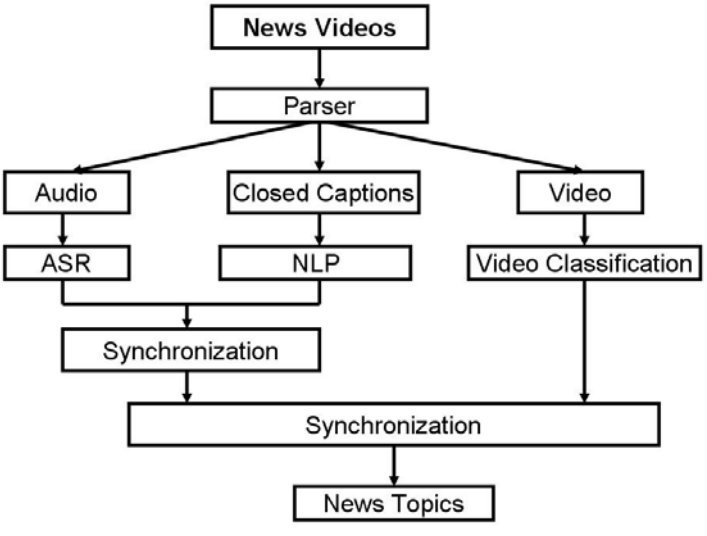


Fig. 1 The flowchart for synchronizing multiple sources for news topic detection, where automatic speech recognition (ASR), natural language processing (NLP), and semantic video classification are seamlessly integrated.

news topic detection. We have developed a new scheme for automatic news topic detection by taking the advantage of multiple information sources (cross-media) as shown in Fig. 1. First, automatic speech recognition (ASR), natural language processing (NLP), and semantic video classification are performed on these three information sources parallelly to determime the keywords for news topic description from both the audio channel and the closed captions and detect the video concepts from the video channel. Second, the audio channel is synchronized with the closed caption channel, and the video channel is further synchronized with the audio channel and the closed caption channel. Finally, the detection results of news topics from these three information sources are integrated to boost the performance of our news topic detection algorithm.

The closed captions of news videos can provide abundant information and such information can be used to detect the news topics of interest and their semantic interpretations with high accuracy. To do this, the closed captions are first segmented into a set of sentences, and each sentence is further segmented into a set of keywords. In news videos, some special text sentences, such as "*somebody*, CNN, *somewhere*" and "ABC's *somebody* reports from *somewhere*", need to be processed separately. The names for news reporters in those text sentences are generally not the content of news report. Therefore, they are not appropriate for news semantics interpretation and should be removed. Because there have some clear and fixed patterns for these specific sentences, we have designed a context-free syntax parser to detect and mark this information. By incorporating 10-15 syntax rules, our parser can detect and mark such specific sentences in high accuracy. Standard text processing techniques are used to remove the stop words automatically.

Most named entity detectors may fail in processing all-capital strings because initial capitalization is very important to achieve accurate named entity recognition. One way to resolve this problem is to train a detector with ground truth from the text documents of closed captions. However, it's very expensive to obtain the manually marked text material. Because English has relatively strict grammar, it's possible to parse the sentences and recover the most capital information by using part-of-speech (POS) and lemma information. TreeTagger [7] is used to perform the part-of-speech tagging. Capital information can be recovered automatically by using the TreeTagger parsing results.

After such specific sentences are marked and the capital information is recovered, an open source text analysis package LingPipe [8] is used to perform the named entity detection and resolve co-reference of the named entities. The named entities referring to the same entity are normalized to the most representative format to enable statistical analysis, where the news model of LingPipe is used and all the parameters are set to default value. Finally, the normalized results are parsed again by TreeTagger to extract the POS information and resolve the words to their original formats. For example, TreeTagger can resolve "better" to "well" or "good" according to its POS tag.

We have defined a set of over 4000 elemental topics, each keyword represents an elemental topic, and all these detected news stories that consist of one particular keyword are assigned to the corresponding cluster of news topic. Our multi-task

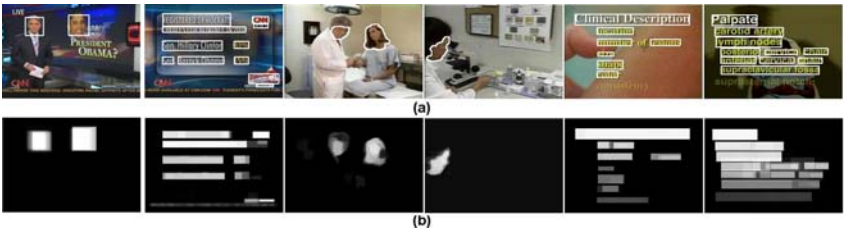


Fig. 2 Integrating confidence map for salient object detection: (a) original images and the detected salient objects; (b) confidence maps for the salient objects.

learning algorithm is performed to learn the topic detectors from a given corpus by exploiting the inter-topic correlation [25-27]. Once we have a set of topic detectors, they are used to determine the most topic-similar clusters for the new piece of news videos.

For TV news videos, the video shots are the basic units for video content representation, and thus they can be treated as one of the semantic items for news topic detection. Unlike the keywords in text documents, the re-appearance of video shots cannot be detected automatically via simple comparison of their visual properties. For news videos, video objects, such as text areas and human faces, may provide important clues about news stories of interest. Text lines and human faces in news videos can be detected automatically by using suitable computer vision techniques [28]. Obviously, these automatic detection functions may fail in some cases. Thus the results that are detected by using a single video frame may not be reliable. To address this problem, the detection results on all the video frames within the same video shot are integrated and the corresponding confidence maps for the detection results are calculated as shown in Fig. 2 [27]. The video concepts associated with the video shots can provide valuable information to enable more accurate news topic detection, and semantic video classification is one potential solution to detect such video concepts [27]. To detect the video concepts automatically, we have adopted our previous work reported in [25-28].

Unfortunately, the closed captions may not synchronize with the video channel accurately and have a delay of a few seconds in general. Thus the news topics that are detected from the closed captions cannot directly be synchronized with the video concepts that are detected from the news videos. On the other hand, the closed captions have good synchronization with the relevant audios. Therefore, they can be integrated to take advantage of cross-media to clarify the video content and remove the redundant information. Even the audio channel generally synchronizes very well with the video channel, the accuracy of most existing techniques for automatic speech recognition (ASR) is still low. By integrating the results for automatic speech recognition with the topic detection results from the closed captions, we can synchronize the closed captions with the video content in higher accuracy. After the closed captions are synchronized with the news videos, we can assign the video shots to the most relevant news topics that are accurately detected from the closed captions. Thus all the video shots, which locate between the start time and the end

time of a given new topic that has been detected from the closed captions, are assigned to the given news topic automatically.

3.2 Topic Association Extraction

The contextual relationships among these significant news topics are obtained automatically, where both the semantic similarity and the co-occurrence probability for the relevant news topics are used to define a new measurement for determining the inter-topic associations effectively. The inter-topic association (i.e., inter-topic contextual relationship) $\phi(C_i, C_j)$ is determined by:

$$\phi(C_i, C_j) = -\alpha \cdot \log \frac{d(C_i, C_j)}{2L} + \beta \cdot \frac{\psi(C_i, C_j)}{\log \psi(C_i, C_j)}, \quad \alpha + \beta = 1 \qquad (1)$$

where the first part denotes the semantic similarity between the news topics C_j and C_i, the second part indicates their co-occurrence probability, α and β are the weighting parameters, $d(C_i, C_j)$ is the length of the shortest path between the news topics C_i and C_j by searching the relevant keywords for news topic interpretation from WordNet [23], L is the maximum depth of WordNet, $\psi(C_i, C_j)$ is the co-ocurrence probability between the relevant news topics. The co-occurrence probability $\psi(C_i, C_j)$, between two news topics C_j and C_i, is obtained in the news topic detection process. Obviously, the value of the inter-topic association $\phi(C_i, C_j)$ increases with the strength of the contextual relationship between the news topics C_i and C_j.

Thus each news topic is automatically linked with multiple relevant news topics with the higher values of the associations $\phi(\cdot, \cdot)$. One portion of our large-scale topic network is given in Fig. 3, where the news topics are connected and organized according to the strength of their associations, $\phi(\cdot, \cdot)$. One can observe that such a topic network can provide a good global overview of large-scale news videos and

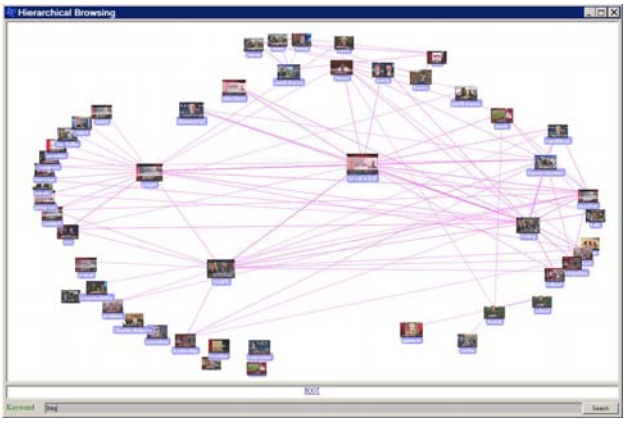


Fig. 3 One portion of our topic network for organizing large-scale news videos.

can precisely characterize the interestingness of the relevant news topics, and thus it can be used to assist news seekers on making better search decisions.

To integrate the topic network for supporting user-adaptive topic recommendation, it is very attractive to achieve graphical representation and visualization of the topic network, so that news seekers can obtain a good global overview of large-scale news videos at the first glance and make better search decisions in the process of interactive topic network exploration and navigation. Unfortunately, visualizing large-scale topic network in a 2D system interface with a limited screen size is not a trivial task. To achieve more effective visualization of large-scale topic network, we have developed multiple innovative techniques: (a) highlighting the news topics according to their interestingness scores for allowing news seekers to obtain the most important insights at the first glance; (b) integrating hyperbolic geometry to create more space for large-scale topic network visualization and exploration.

3.3 Interestingness Scores of News Topics

We have integrated both the popularity of the news topics and the importance of the news topics to determine their interestingness scores. The popularity of a given news topic is related to the number of TV channels or news programs which have discussed or reported the given news topic. If one news topic is discussed or reported by more TV channels or news programs, it tends to be more interesting. The importance of a given news topic is also related to its linkage structure with other news topics on the topic network. If one news topic is related to more news topics on the topic network, it tends to be more interesting [17]. For example, the news topic for “roadside bond in Iraq" may relate to the news topics of “gap price increase" and “stock decrease". Thus the interestingness score $\rho(C_i)$ for a given news topic C_i is defined as:

$$\rho(C) = \lambda \cdot \log(m(C_i) + \sqrt{m^2(C_i) + 1}) + \gamma \cdot \log(k(C_i) + \sqrt{k^2(C_i) + 1}), \lambda + \gamma = 1 \quad (2)$$

where $m(c_i)$ is the number of TV channels or news programs which have discussed or reported the given news topic C_i, $k(c_i)$ is the number of news topics linked with the given news topic C_i on the topic network. Thus the interestingness score for a given news topic increases adaptively with both the number of the relevant TV channels or news programs and the number of the linked news topics. Such interestingness scores can be used to highlight the most interesting news topics and eliminate the less interesting news topics for reducing the visual complexity for large-scale topic network visualization and exploration.

3.4 Hyperbolic Topic Network Visualization

Supporting graphical representation and visualization of the topic network can provide an effective solution for exploring large-scale news videos at the topic level and recommend the *news topics of interest* interactively for assisting news seekers to make

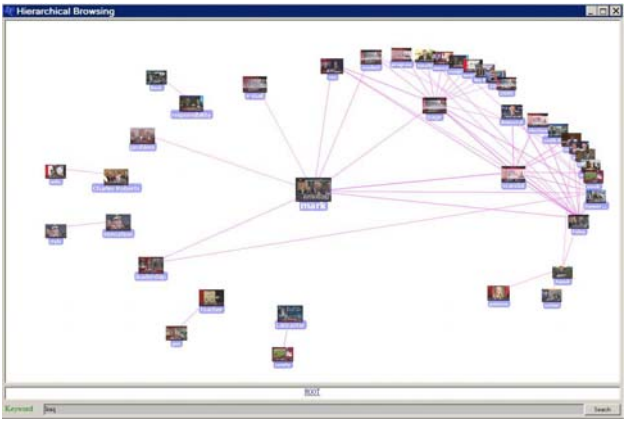


Fig. 4 One view of hyperbolic visualization of our topic network.

better search decisions. However, visualizing large-scale topic network in a 2D system interface with a limited screen size is a challenging task. We have investigated multiple solutions to tackle this challenge task: (a) A string-based approach is incorporated to visualize the topic network with a nested view, where each news topic node is displayed closely with the most relevant news topic nodes according to the values of their associations. The underlying inter-topic contextual relationships are represented as the linkage strings. (b) The geometric closeness of the news topic nodes is related to the strength of their inter-topic contextual relationships, so that such graphical representation of the topic network can reveal a great deal about how these news topics are connected. (c) Both geometric zooming and semantic zooming are integrated to adjust the levels of visible details automatically according to the discerning constraint on the number of news topic nodes that can be displayed per view.

Our approach for topic network visualization exploits hyperbolic geometry [14-16]. The hyperbolic geometry is particularly well suited for achieving graph-based layout of the topic network, and it has “more space" than Euclidean geometry. The essence of our approach is to project the topic network onto a hyperbolic plane according to the inter-topic contextual relationships, and layout the topic network by mapping the relevant news topic nodes onto a circular display region. Thus our topic network visualization scheme takes the following steps: (a) The news topic nodes on the topic network are projected onto a hyperbolic plane according to their inter-topic contextual relationships, and such projection can usually preserve the original contextual relationships between the news topic nodes. (b) After such context-preserving projection of the news topic nodes is obtained, Poincaré disk model [14-16] is used to map the news topic nodes on the hyperbolic plane to a 2D display coordinate. Poincaré disk model maps the entire hyperbolic space onto an open unit circle, and produces a non-uniform mapping of the news topic nodes to the 2D display coordinate.

Our approach for topic network visualization relies on the representation of the hyperbolic plane, rigid transformations of the hyperbolic plane and mappings of

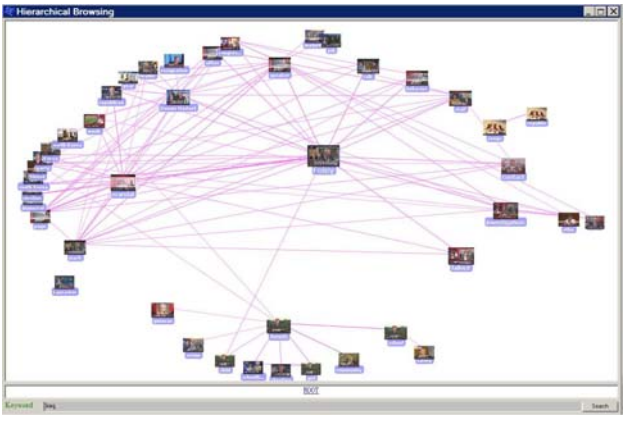


Fig. 5 Another view of hyperbolic visualization of our topic network.

the news topic nodes from the hyperbolic plane to the unit disk. Internally, each news topic node on the graph is assigned a location $z = (x, y)$ within the unit disk, which represents its Poincaré coordinates. By treating the location of the news topic node as a complex number, we can define such a mapping as the linear fractional transformation [14-16]:

$$z_t = \frac{\theta z + P}{1 + \bar{P}\theta z} \tag{4}$$

where P and θ are the complex numbers, $|P| < 1$ and $|\theta| = 1$, and $\bar{P}$ is the complex conjugate of P. This transformation indicates a rotation by θ around the origin following by moving the origin to P (and $-P$ to the origin).

To incorporate such transformation for topic network visualization, the layout routine is structured as a recursion that takes a news topic node and a wedge in which to lay out the news topic node and its relevant news topic nodes. It places the news topic node at the vertex of the wedge, computes a wedge for each relevant news topic node and recursively calls itself on each relevant news topic node. The relevant news topic nodes are placed in the middle of their subwedges at a distance computed by the formula:

$$d = \sqrt{\left(\frac{(1-s^2)sin(a)}{2s}\right)^2 + 1} - \frac{(1-s^2)sin(a)}{2s} \tag{5}$$

where a is the angle between the midline and the edge of the subwedge and s is the desired distance between a relevant news topic node and the edge of its subwedge. In our current implementation, we set $s = 0.18$. The result, d, is the necessary distance from current news topic node to its relevant news topic node. If the value of d is less than that of s, we set d to s for maintaining a minimum space between the current news topic node and the relevant news topic node. Both s and d are represented as the hyperbolic tangent of the distance in the hyperbolic plane.

3.5 Personalized Topic Network Generation

After the hyperbolic visualization of the topic network is available, it can be used to enable interactive exploration and navigation of large-scale news videos at the topic level via *change of focus*. The *change of focus* is implemented by changing the mapping of the news topic nodes from the hyperbolic plane to the unit disk for display, and the positions of the news topic nodes in the hyperbolic plane need not to be altered during the focus manipulation. As shown in Fig. 4 and Fig. 5, news seekers can change their focuses of the news topics by clicking on any visible news topic node to bring it into the focus at the screen center, or by dragging any visible news topic node interactively to any other screen location without losing the contextual relationships between the news topics, where the rest of the layout of the topic network transforms appropriately. In such interactive topic network navigation and exploration process, news seekers can obtain the *news topics of interest* interactively, build up their mental search models easily and make better search decision effectively by selecting the visible news topics directly. Because our hyperbolic visualization framework can assign more spaces for the news topic node in current focus and ignore the details for the residual news topic nodes on the topic network, it can theoretically avoid the overlapping problem by supporting change of focus and thus it can spporting large-scale topic network visualization and navigation.

On the other hand, such interactive topic network exploration process has also provided a novel approach for capturing the search interests of news seekers automatically. We have developed a new algorithm for generating personalized topic network automatically from the current search actions of news seeker while the new seeker navigates the topic network. Thus the *personalized interestingness score* for a given news topic C_i on the topic network is defined as:

$$\rho(C_i) = \rho^{org}(C_i) + \rho^{org}(C_i) \left\{ \alpha \frac{e^{v(C_i)} - e^{-v(C_i)}}{e^{v(C_i)} + e^{-v(C_i)}} + \beta \frac{e^{s(C_i)} - e^{-s(C_i)}}{e^{s(C_i)} + e^{-s(C_i)}} + \delta \frac{e^{d(C_i)} - e^{-d(C_i)}}{e^{d(C_i)} + e^{-d(C_i)}} \right\} \tag{6}$$

where $\alpha + \beta + \delta = 1$, $\rho^{org}(C_i)$ is the original interestingness score for the given news topic C_i as defined in Eq. (2), $v(C_i)$ is the visiting times of the given news topic C_i from the particular news seeker, $s(C_i)$ is the staying seconds on the given news topic C_i from the particular news seeker, $d(C_i)$ is the interaction depth for the particular user to interact with the news topic C_i and the relevant news videos which are relevant to the given news topic C_i, α, β and δ are the weighting factors. The visiting times $v(C_i)$, the staying seconds $s(C_i)$, and the interaction depth $d(C_i)$ can be captured automatically in the user-system interaction procedure. Thus the personalized interestingness scores of the news topics are determined immediately when such user-system interaction happens, and they will increase adaptively with the visiting times $v(C_i)$, the staying seconds $s(C_i)$, and the interaction depth $d(C_i)$.

After the personalized interestingness scores for all these news topics are learned from the current search actions of news seeker, they can further be used to weight the news topics for generating a *personalized topic network* to represent the user profiles (i.e., search preferences of news seeker) precisely. Thus the news topics

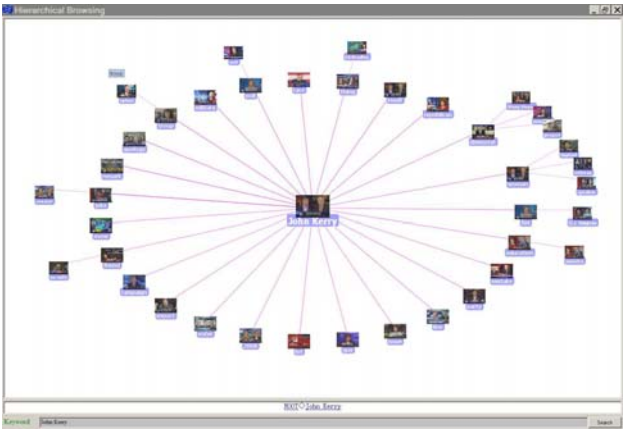


Fig. 6 The most relevant news topics for interestingness propagation.

with smaller values of the personalized interestingness scores can be eliminated automatically from the topic network, so that each news seeker can be informed by the most interesting news topics according to his/her personal preferences.

The search interests of news seeker may be changed according to his/her timely observations of news topics, and one major problem for integrating the user's profiles for topic recommendation is that the user's profiles may over-specify the search interests of news seeker and thus they may hinder news seeker to search other interesting news topics on the topic network. Based on this observation, we have developed a novel algorithm for propagating the search preferences of news seeker over other relevant news topics on the topic network, i.e., the news topics which have stronger correlations with the news topics which have been accessed by the particular news seeker. Thus the personalized interestingness score $\upsilon(C_j)$ for the news topic C_j to be propagated is determined as:

$$\chi(C_j) = \rho(C_j)\bar{\phi}(C_j), \quad \bar{\phi}(C_j) = \sum_{l \in \Omega} \phi(C_l, C_j) \tag{7}$$

where Ω is the set of the accessed news topics linked with the news topic C_j to be propagated as shown in Fig. 6 and Fig. 7, $\phi(C_l, C_j)$ is the inter-topic association between the news topic C_j and the news topic C_l which is linked with C_j and has been accessed by the particular news seeker, and $\rho(C_j)$ is the interestingness score for the news topic C_j to be propagated. Thus the news topics, which have larger values of the personalized interestingness scores $\chi(\cdot)$ (strongly linked with some accessed news topics on the topic network), can be propagated adaptively.

By integrating the inter-topic correlations for automatic propagation of the preferences of news seeker, our proposed framework can precisely predict his/her hidden preferences (i.e., search intentions) from his/her current actions. Thus the user's profiles can be represented precisely by using the personalized topic network, where the interesting news topics can be highlighted according to their personalized

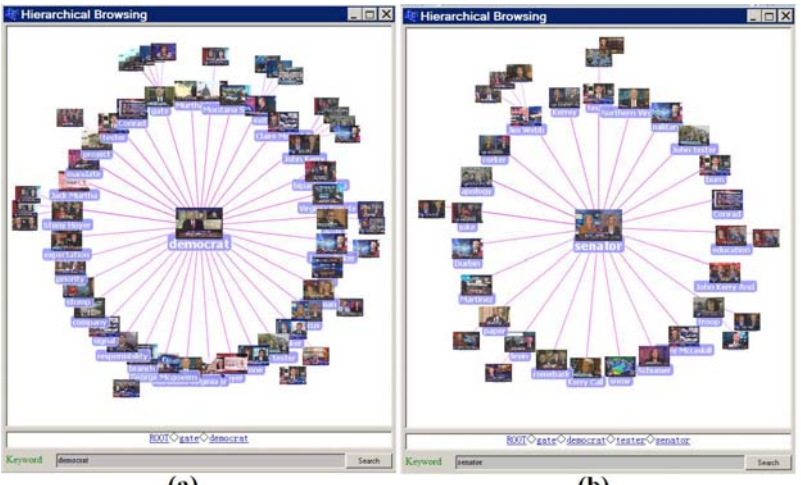


Fig. 7 The most relevant news topics for interestingness propagation.

interestingness scores as shown in Fig. 6 and Fig. 7. Such personalized topic network can further be used to recommend the news topics of interest for news seekers to make better future search decisions.

4 Personalized News Video Recommendation

Because the same news topic may be discussed many times in the same TV news program or be discussed simultaneously by multiple TV news programs, the amount of news videos under the same topic could be very large. Thus topic-based news search via keyword matching may return large amount of news videos which are relevant to the same news topic. To reduce the information overload, it is very important to develop new algorithms for ranking the news videos under the same news topic and recommending the most relevant news videos according to their importance and representativeness scores [18-19].

The news videos, which are relevant to the given news topic C_j, are ranked according to their importance and representiveness scores. For the given news topic C_j, the importance and representativeness score $\varrho(x|C_j)$ for one particular news video x is defined as:

$$\varrho(x|C_j) = \alpha e^{-\Delta t} + (1-\alpha)\left\{\beta \frac{e^{v(x|C_j)} - e^{-v(x|C_j)}}{e^{v(x|C_j)} + e^{-v(x|C_j)}} + \gamma \frac{e^{r(x|C_j)} - e^{-r(x|C_j)}}{e^{r(x|C_j)} + e^{-r(x|C_j)}}\right.$$
$$\left. + \eta \frac{e^{q(x|C_j)} - e^{-q(x|C_j)}}{e^{q(x|C_j)} + e^{-q(x|C_j)}}\right\} \tag{8}$$

where $\beta + \lambda + \eta = 1$, Δt is the time difference between the time for the TV news programs to discuss and report the given news topic C_j and the time for the news seeker to submit their searches, $v(x|C_j)$ is the visiting times of the given news video x from all the news seekers, $r(x|C_j)$ is the rating score of the given news video x from all the news seekers, $q(x|C_j)$ is the quality of the given news video.

We separate the time factor from other factors for news video ranking because the time factor is more important than other factors for news video ranking (i.e., one topic can be treated as the news because it is new and tell people what is happening now or what is discussing now). The quality $q(x|C_j)$ is simply defined as the frame resolution and the length of the given news video x. If a news video has higher frame resolution and longer length (be discussed for longer time), it should be more important and representative for the given news topic.

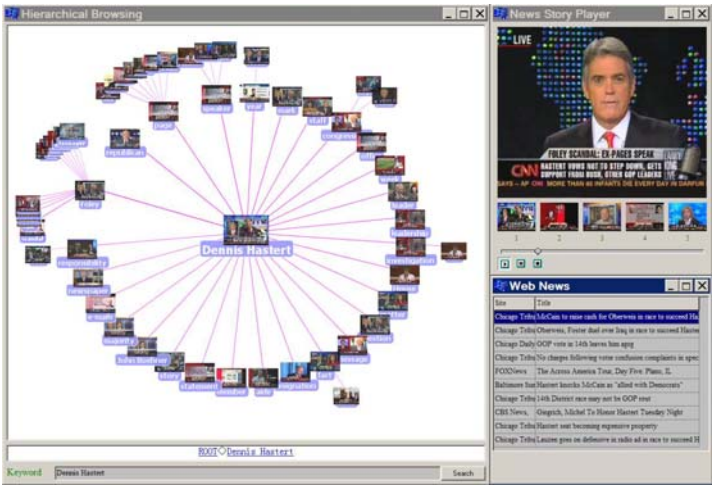


Fig. 8 Our system interface for supporting multi-modal news recommendation.

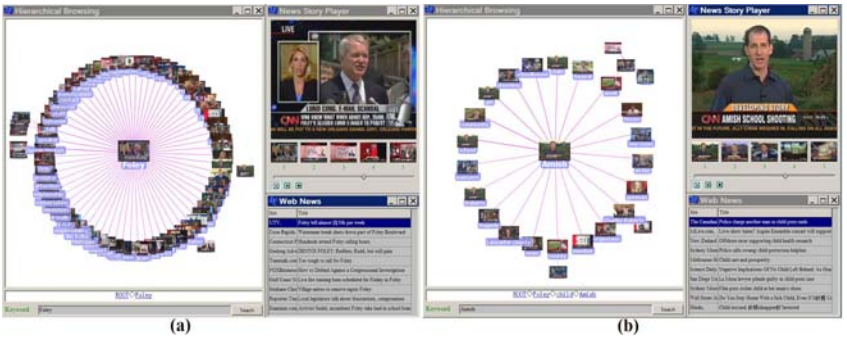


Fig. 9 Two examples for supporting multi-modal news recommendation.

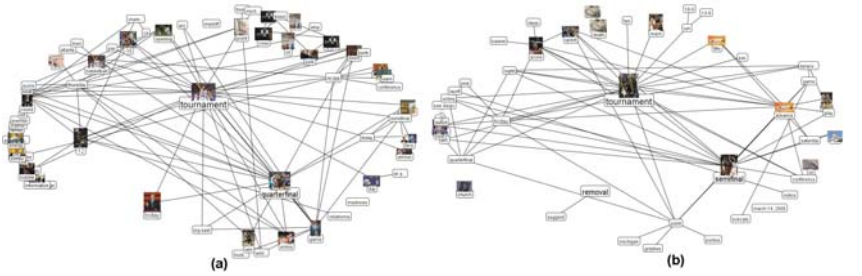


Fig. 10 Our system for supporting online news recommendation: (a) topic network for March 13; (b) topic network for March 14.

After the search goals pf news seekers (i.e., which are represented by the accessed news topics) are captured interactively, our personalized news video recommendation system can: (a) recommend top 5 news videos according to their importance and representative scores; (b) recommend the news topics of interest on the topic network which are most relevant to the accessed news topic and suggest them as the future search directions according to the current preferences of news seekers, where the accessed news topic is set as the current focus (i.e., center of the topic network); (c) recommend the most relevant online text news which are relevant with the accessed news topic, so that news seekers can also read the most relevant online web news; (d) record the search history and preferences of news seekers for generating more reliable personalized topic network to make better recommendation in the future. Some experimental results are given in Fig. 8 and Fig. 9, one can conclude that our personalized news video recommendation system can effectively support multi-modal news recommendation from large-scale collections of news videos.

We have also extended our multi-modal news analysis tools to support personalized online news recommendation. First, the news topics of interest are extracted from large-scale online news collections and the inter-topic similarity contexts are determined for topic network generation as shown in Fig. 10, one can observe that

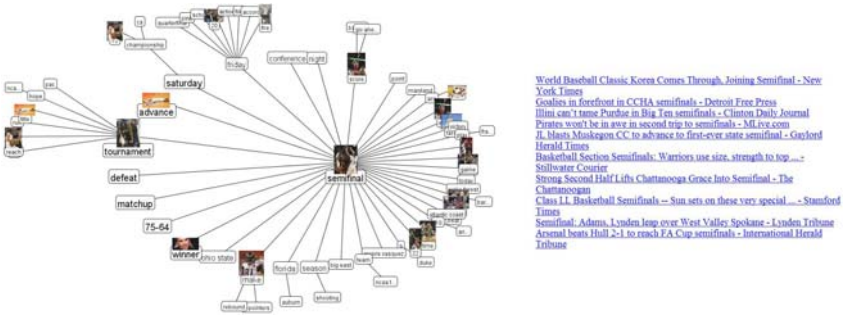


Fig. 11 Our system for supporting online news recommendation: personalized topic network and the relevant online news sources.

such the topic network can represent the news topics of interest and their inter-topic similarity contexts effectively. By incorporating the inputs of news seekers for online news recommendation, the accessed news topic is set as the current focus and the most relevant news sources are recommended as shown in Fig. 11.

5 Algorithm Evaluation

We carry out our experimental studies by using large-scale news videos. The topic network which consists of 4000 most popular news topics is learned automatically from large-scale news videos. Our work on algorithm evaluation focus on: (1) evaluating the performance of our news topic detection algorithm and assessing the advantages for integrating multiple information sources for news topic detection; (2) evaluating the response time for supporting change of focus in our system, which is critical for supporting interactive navigation and exploration of large-scale topic network to enable user-adaptive topic recommendation; (3) evaluating the performance (efficiency and accuracy) of our system for allowing news seekers to look for some particular news videos of interest (i.e., personalized news video recommendation).

Automatic news topic detection plays an important role in our personalized news video recommendation system. However, automatic topic detection is still an open problem in natural language processing community. On the other hand, automatic video understanding via semantic classification is also an open issue in computer vision community. In this chapter, we have integrated multiple information sources (audio, video and closed captions) to exploit the cross-media advantages for achieving more reliable news topic detection.

Based on this observation, our algorithm evaluation for our automatic news topic detection algorithm focuses on comparing its performance difference by combining different information sources for news topic detection. We have compared three combination scenarios for news topic detection: (a) only the closed captions are used for news topic detection; (b) the closed captions and the audio channel are integrated and synchronized for news topic detection; (c) the closed captions, the audio channel and the video channel are seamlessly integrated and synchronized for news topic detection. As shown in Fig. 12, integrating multiple information sources for news topic detection can enhance the performance of our algorithm significantly.

One critical issue for evaluating our personalized news video recommendation system is the response time for supporting change of focus to enable interactive topic network navigation and exploration, which is critical for supporting user-adaptive topic recommendation. In our system, the change of focus is used for achieving interactive exploration and navigation of large-scale topic network. The *change of focus* is implemented by changing the Poincaré mapping of the news topic nodes from the hyperbolic plane to the display unit disk, and the positions of the news topic nodes in the hyerbolic plane need not to be altered during the focus manipulation. Thus the response time for supporting change of focus depends on two components: (a) The computational time T_1 for re-calculating the new Poincaré mapping of large-scale topic network from a hyperbolic plane to a 2D display unit

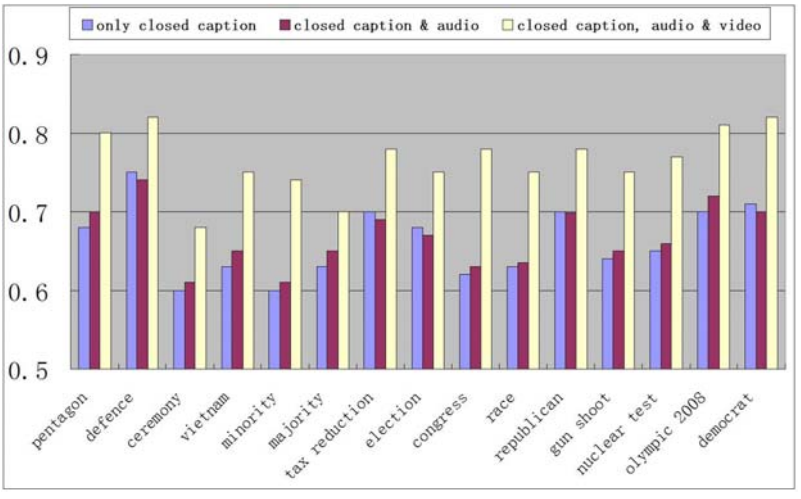


Fig. 12 The comparision results of our automatic news topic detection algorithm by integrating different sources.

disk, i.e., re-calculating the Poincaré position for each news topic node; (b) The visualization time T_2 for re-layouting and re-visualizing the new Poincaré mapping of large-scale topic network on the display disk unit. As shown in Fig. 13, one can find that the computational time T_1 is not sensitive to the number of news topics, and thus re-calculating the Poincaré mapping for large-scale topic network can almost be achieved in real time. We have also evaluated the empirical relationship between the visualization time T_2 and the number of news topic nodes. In our experiments, we have found that re-visualization of large-scale topic network is not sensitive to the number of news topics, and thus our system can support re-visualization of large-scale topic network in real time.

When the news topics of interest are recommended, our system can further allow news seekers to look for the most relevant news videos according to their importance and representative scores. For evaluating the effeciency and the accuracy of our personalized news video recommendation system, the *benchmark metric* includes *precision* P and *recall* R. The precision P is used to characterize the accuracy of

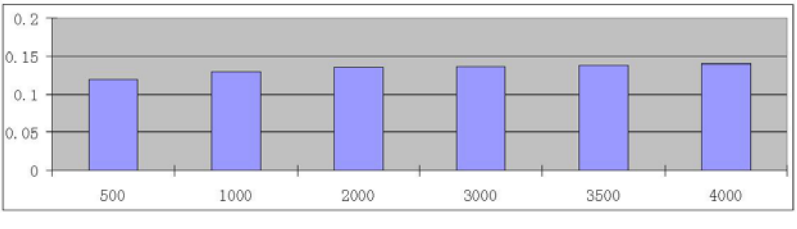


Fig. 13 The empirical relationship between the computational time T_1 (seconds) and the number of news topic nodes.

Table 1 The precision and recall for supporting personalized news video recommendation.

news topics	policy	pentagon	change	insult
precision/recall	95.6% /97.3%	98.5% /98.9%	100% /99.2%	92.8% /93.6%
news topics	implant	wedding	haggard	bob
precision/recall	90.2% /93.5%	96.3% /94.5%	96.5% /92.8%	90.3% /97.4%
news topics	gate	steny hoyer	democrat	urtha
precision/recall	95.9% /96.8%	96.5% /96.2%	96.3% /97.1%	93.6% /94.3%
news topics	majority	leader	confirmation	defence
precision/recall	99.2% /98.6%	93.8% /99.3%	94.5% /93.8%	100% /99.6%
news topics	secretary	veterm	ceremony	service
precision/recall	100% /98.8%	99.8% /99.2%	99.3% /96.6%	91.2% /93.2%
news topics	honor	vietnam	lesson	submit
precision/recall	91.2% /93.5%	98.8% /96.7%	90.3% /91.6%	91.2% /91.5%
news topics	minority	indonesia	president	trent lott
precision/recall	100% /99.6%	96.8% /97.7%	100% /96.8%	92.5% /92.3%
news topics	o.j. sinpson	trial	money	book
precision/recall	95.6% /99.4%	90.5% /90.3%	100% /90.6%	96.8% /93.6%
news topics	john kerry	military	race	mandate
precision/recall	100% /96.5%	100% /93.2%	100% /97.8%	92.6% /92.5%
news topics	election	leadship	school gun shoot	execution
precision/recall	100% /95.5%	92.8% /90.3%	100% /96.7%	90.6% /91.3%
news topics	responsibility	sex	message	congress
precision/recall	92.1% /91.5%	97.5% /98.2%	88.3% /87.6%	100% /96.3%
news topics	north korea	japan	china	white house
precision/recall	100% /99.3%	98.5% /95.6%	97.3% /95.2%	100% /94.8%
news topics	nuclear test	republican	amish	gun shoot
precision/recall	100% /97.6%	91.6% /92.8%	99.5% /91.6%	100% /99.8%
news topics	teacher	conduct	program	olmypic 2008
precision/recall	93.8% /94.5%	87.92% /88.3%	83.5% /90.2%	100% /99.3%
news topics	beijing	child	tax reduction	shooting
precision/recall	99.2% /97.3%	91.3% /91.5%	98.5% /96.9%	99.6% /98.4%
news topics	safety	investigation	ethic	committee
precision/recall	94.5% /94.8%	93.3% /96.5%	93.3% /95.6%	91.8% /95.2%
news topics	scandal	dennis hastert	preseident candidates	matter
precision/recall	96.6% /97.3%	95.3% /88.3%	98.5% /97.3%	85.2% /85.3%

our system for finding the particular news videos of interest, and the recall R is used to characterize the efficiency of our system for finding the particular news videos of interest. They are defined as:

$$P = \frac{TP}{TP + FP}, \quad R = \frac{TP}{TP + TN} \tag{9}$$

where TP is the set of true positive news videos that are relevant to the need of news seeker and are recommended correctly, FP is the set of fause positive news

videos that are relevant to the need of news seeker and are not recommended, and TN is the set of true negative news videos that are relevant but are recommended incorrectly. Table 1 gives the precision and recall of our personalized news video recommendation system. From these experimental results, one can observe that our system can support personalized news video recommendation effectively, thus news seekers are allowed to search for some particular news videos of interest effectively.

6 Conclusions

In this chapter, we have developed an interactive framework to support personalized news video recommendation and allow news seekers to access large-scale news videos more effectively. To allow news seekers to obtain a good global overview of large-scale news videos at the topic level, topic network and hyperbolic visualization are seamlessly integrated to achieve user-adaptive topic recommendation. Thus news seekers can obtain the *news topics of interest* interactively, build up their mental search models easily and make better search decisions by selecting the visible news topics directly. Our system can also capture the search intentions of news seekers implicitly and further recommend the most relevant news videos according to their importance and representativeness scores. Our experiments on large-scale news videos have provided very positive results.

References

1. Marchionini, G.: Information seeking in electronic environments. Cambridge University Press, Cambridge (1997)
2. Fan, J., Keim, D., Gao, Y., Luo, H., Li, Z.: JustClick: Personalized Image Recommendation via Exploratory Search from Large-Scale Flickr Images. IEEE Trans. on Circuits and Systems for Video Technology 19(2), 273–288 (2009)
3. Yang, B., Mei, T., Hua, X.-S., Yang, L., Yang, S.-Q., Li, M.: Online video recommendation based on multimodal fusion and relevance feedback. In: ACM Conf. on Image and Video Retrieval (CIVR 2007), pp. 73–80 (2007)
4. Yang, H., Chaisorn, L., Zhao, Y., Neo, S.-Y., Chua, T.-S.: VideoQA: question answering on news video. ACM Multimedia, 632–641 (2003)
5. Borthwick, A., Sterling, J., Agichtein, E., Grishman, R.: NYU: Description of the MENE named entity system as used in MUC-7. In: Proc. of the Seventh Message Understanding Conf., MUC-7 (1998)
6. McDonald, D., Chen, H.: Summary in context: Searching versus browsing. ACM Trans. Information Systems 24(1), 111–141 (2006)
7. Schmid, H.: Probabilistic part-of-speech tagging using decision trees. In: Intl. Conf. on New Methods in Language Processing (1994)
8. A.I. Inc., "Lingpipe", http://www.alias-i.com/lingpipe/
9. Swan, R.C., Allan, J.: TimeMine: visualizing automatically constructed timelines. In: ACM SIGIR (2000)
10. Havre, S., Hetzler, B., Whitney, P., Nowell, L.: ThemeRiver: Visualizing thematic changes in large document collections. IEEE Trans. on Visualization and Computer Graphics 8(1), 9–20 (2002)

11. Luo, H., Fan, J., Yang, J., Ribarsky, W., Satoh, S.: Large-scale new video classification and hyperbolic visualization. In: IEEE Symposium on Visual Analytics Science and Technology (VAST 2007), pp. 107–114 (2007)
12. Luo, H., Fan, J., Yang, J., Ribarsky, W., Satoh, S.: Exploring large-scale video news via interactive visualization. In: IEEE Symposium on Visual Analytics Science and Technology (VAST 2006), pp. 75–82 (2006)
13. van Wijk, J.: Bridging the gaps. IEEE Computer Graphics and Applications 26(6), 6–9 (2006)
14. Walter, J.A., Ritter, H.: On interactive visualization of high-dimensional data using the hyperbolic plane. In: ACM SIGKDD (2002)
15. Lamping, J., Rao, R.: The hyperbolic browser: A focus+content technique for visualizing large hierarchies. Journal of Visual Languages and Computing 7, 33–55 (1996)
16. Furnas, G.W.: Generalized fisheye views. In: ACM CHI, pp. 16–23 (1986)
17. Brin, S., Page, L.: The anatomy of a large-scale hypertextual web search engine. In: WWW (1998)
18. Wang, J., Chen, Z., Tao, L., Ma, W.-Y., Liu, W.: Ranking user's relevance to a topic through link analysis on web logs. In: WIDM, pp. 49–54 (2002)
19. Lai, W., Hua, X.-S., Ma, W.-Y.: Towards content-based relevance ranking for video search. ACM Multimedia, 627–630 (2006)
20. Teevan, J., Dumais, S., Horvitz, E.: Personalized search via automated analysis of interests and activities. In: ACM SIGIR (2005)
21. Goldberg, K., Roeder, T., Gupta, D., Perkins, C.: Eigentaste: A constant time collaborative filtering algorithm. Information Retrieval 4(2), 133–151 (2001)
22. Mooney, R., Roy, L.: Content-based book recommending using learning for text categorization. In: ACM Conf. on Digital Libraries, pp. 195–204 (2000)
23. Fellbaum, C.: WordNet: An Electronic Lexical Database. MIT Press, Boston (1998)
24. Naphade, M., Smith, J.R., Tesic, J., Chang, S.-F., Hsu, W., Kennedy, L., Hauptmann, A., Curtis, J.: Large-scale concept ontology for multimedia. IEEE Multimedia (2006)
25. Fan, J., Gao, Y., Luo, H.: Integrating concept ontology and multi-task learning to achieve more effective classifier training for multi-level image annotation. IEEE Trans. on Image Processing 17(3) (2008)
26. Fan, J., Gao, Y., Luo, H., Jain, R.: Mining multi-level image semantics via hierarchical classification. IEEE Trans. on Multimedia 10(1), 167–187 (2008)
27. Fan, J., Luo, H., Gao, Y., Jain, R.: Incorporating concept ontology to boost hierarchical classifier training for automatic multi-level video annotation. IEEE Trans. on Multimedia 9(5), 939–957 (2007)
28. Fan, J., Yau, D.K.Y., Elmagarmid, A.K., Aref, W.G.: Automatic image segmentation by integrating color edge detection and seeded region growing. IEEE Trans. on Image Processing 10(10), 1454–1466 (2001)
29. Wactlar, H., Hauptmann, A., Gong, Y., Christel, M.: Lessons learned from the creation and deployment of a terabyte digital video library. IEEE Computer 32(2), 66–73 (1999)
30. Adams, W.H., Iyengar, G., Lin, C.-Y., Naphade, M.R., Neti, C., Nock, H.J., Smith, J.R.: Semantic indexing of multimedia content using visual, audio and text cues. EURASIP JASP 2, 170–185 (2003)
31. Naphade, M.R., Huang, T.S.: A probabilistic framework for semantic video indexing, filtering, and retrival. IEEE Trans. on Multimedia 3, 141–151 (2001)

Part V
Video Mining

A Holistic, In-Compression Approach to Mining Independent Motion Segments for Massive Surveillance Video Collections

Zhongfei (Mark) Zhang and Haroon Khan

Abstract. This chapter describes a large scale surveillance video data mining approach for those segments that contain independently moving targets. Given the typical scenario where the video data collections are massive in size, We propose a holistic, in-compression approach, called Linear System Consistency Analysis (LSCA), to efficient video data mining for those independent motion segments. By efficient, we mean that the mining speed is close to or even faster than real-time in "normal" platforms (we do not assume using special hardware or any parallel machines) while still maintaining a good mining quality. Theoretical and experimental analyses demonstrate and validate this holistic, in-compression approach to solving for video mining problem for temporal independent motion segmentation.

1 Introduction

This chapter develops a large scale surveillance video data unsupervised segmentation technique regarding whether there is a presence of independent motion. Since processing time is critical in summarizing and/or segmenting large scale surveillance video, we must provide efficient processing. Consequently, we focus on developing a technique aiming at efficient processing of MPEG video stream data. By *efficient*, we mean that the processing speed is close to or even faster than real-time in "normal" platforms (we do not assume using special hardware or any parallel machines) while still maintaining a good quality segmentation.

We target surveillance applications. Based on the ultimate goal of efficient video segmentation, we propose a holistic, in-compression approach, and we demonstrate

Zhongfei (Mark) Zhang
SUNY Binghamton, Binghamton, NY, USA
e-mail: zhongfei@cs.binghamton.edu

Haroon Khan
SUNY Binghamton, Binghamton, NY, USA
e-mail: haroon.khan@gmail.com

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 285–303.
springerlink.com

the approach by focusing on a specific video segmentation task — independent motion detection. By independent motion, we mean that the target in the surveillance scene moves independently regardless of whether or not the camera is in motion. Consequently, if the camera is still, independent motion detection trivially becomes the standard motion detection in the scene. On the other hand, if the camera is in motion, every pixel in a frame may contain motion. For those background pixels, the motion reflected in the image domain corresponds to the 3D camera motion; for those pixels corresponding to independently moving objects in a frame, their motion corresponds to the combination of the 3D camera motion and their own independent motion in the 3D space. In this case, simple frame based differencing does not work [16], and certain sophisticated techniques must be applied to separate the independent motion from the camera motion, which is also called the background motion. This problem becomes even more complicated when there is 3D motion parallax involved. In this case, a 3D motion model must be applied in order to robustly and accurately separate the independent motion from the camera motion. Therefore, independent motion detection aims at detecting the target motion instead of the camera motion. In the surveillance applications, it is not unusual that the surveillance cameras are in motion. For example, the surveillance conducted by an unmanned aerial vehical (UAV) is in such a scenario where the cameras are always in motion, which is actually the scenario that motivates this research and is the scenario where the experimental data are collected.

Since this research is motivated by the UAV surveillance scenario, for a typical UAV surveillance, there are multiple UAVs performing the surveillance at the same time, and for each UAV, there are multiple cameras shooting different directions simultaneously. For each camera in surveillance, the collected data are transmitted to a ground station for archive. Clearly, for a typical UAV surveillance task, the volume of the archived surveillance data transmitted from UAV cameras is massive. The current practice is to have image analysts manually examine the archived surveillance data to determine where attention needs to be paid and thus what subsequent actions need to be taken. Since mornally only those surveillance scenes that contain independent motion need to pay attention to whereas typically the majority of the surveillance scenes do not contain any independent motion, the problem now is reduced to mining a massive collection of UAV surveillance video data to filter out the majority of the data that do not contain independent motion and to only identify and return those scenes that contain independent motion. Given the massive data collection, the current manual practice for image analysts is very expensive. Consequently, we are motivated to develop an automated approach to identifying and returning the scenes with independent motion. In addition, this also justifies why the solution we have developed here must be efficient.

In fact, with a developed efficient independent motion detection solution, we may apply this solution to the UAV surveillance scenario in two modes, the online mode and the offline mode. In the online mode, each surveillance camera is equipped with the solution while in surveillance. Thus, the detection is performed in real-time and only those detected independent motion scenes need to be transmitted to the ground station for image analysts to have further examinations and the rest of the

surveillance video data do not need to be transmitted to the ground station at all. In the offline mode, all the surveillance data from all the surveillance cameras are transmitted to a ground station and the ground station database is equipped with the solution. The independent motion detection solution works on all the archived data to filter out those data that do not contain independent motion and only return those data that contain independent motion for image analysts to have further examinations.

There are two scenarios related to independent motion detection. Given a video stream or an image sequence, one scenario refers to the detection in which a *temporal* segmentation is conducted into those subsequences (called shots) that contain the scene in which one or more independently moving objects are present, in addition to a *spatial* segmentation and delineation of each of the independently moving objects in each of the frames of these shots. The other scenario, on the other hand, refers to the detection in which only the *temporal* segmentation is conducted to return those shots that contain independent motion; no spatial segmentation is performed to identify the independently moving objects in each frame. Clearly, the focus of this chapter is primarily in the latter approach. We argue that it is not necessary for spatial segmentation in the video frames in the UAV surveillance as well as other related applications. This is due to the following two reasons. (i) In these applications, the time issue, i.e., the expectation of an efficient solution, is always an important concern. Obviously the additional spatial domain segmentation requires more processing time and thus reduces the mining speed. (ii) Even if the independently moving objects are all segmented and identified in each frame, given the current status of computer vision and artificial intelligence in general, it is *not possible* to have a fully automated capability to interpret whether the segmented and identified independent motion in the frames indicates any specific significance such that special attention needs to be paid without interaction with human expertise. Therefore, these detected shots must be sent to the users (i.e., image analysts) for further analysis anyway, *regardless* of whether or not the independently moving targets in the frames are segmented and identified in these shots.

It is also observed that in the literature, most of the existing techniques for independent motion detection are based on image sequences, as opposed to compressed video streams. In other words, given a surveillance video, these methods require that the video be first fully decompressed to recover an image sequence before these methods can be applied. This restriction (or assumption) significantly hinders these techniques from practical applications, as in today's world, information volume grows explosively, and all the video sequences are archived in compressed forms. This is particularly true in the surveillance applications, in which the data volume is *massive* and they must be archived in a compressed form, such as MPEG.

Even if this work only focuses on mining massive collections of surveillance video data in temporal segmentation for the scenes with independent motion without further spatial segmentation, there are still many challenging issues that need to be addressed. The specific challenges include the expected efficient computation and the related computation within the compressed domain in order to develop such an efficient solution, the mining accuracy, the possible poor quality of the data, the

possible multiple independently moving targets in the scene, and the possibly significantly varying sizes of the independently moving targets in the scene. The main contribution of this work is the development of a holistic, in-compression approach to mining massive collections of surveillance video data that addresses all the above challenging issues, which is validated by both theoretical analysis and experimental evaluations.

2 Related Work

Motion analysis has been a focused topic in computer vision and image understanding research for many years [36, 15, 13, 11, 25]. Due to the difficult nature of the problems in this topic, it is still considered as an open topic and many research efforts are still being developed in this topic [9, 10]. Independent motion analysis, on the other hand, deals with multiple motion components simultaneously, and therefore, is presumably more challenging.

The earliest work in independent motion detection may be dated back to the early 80's. Jain [18] proposed a solution assuming that the camera was under a translation. Adiv [1] assumed the availability of optical flow and used the flow to group regions based on the rigidity constraint over two frames. Nelson [26] proposed two methods based on velocity constraints to detect independently moving objects. Thompson et al [34] used a similar approach based on the rigidity constraint. Bouthemy and Francois [8] treated the problem of independent motion detection as a statistical regularization problem and attempted to use the Markov Random Field model to solve for the problem. Ayer et al [6] used robust statistical regression techniques to detect independent motion. Smith and Brady [32] used geometric constraints for independent motion segmentation. Sharma and Aloimonos [30] provided a solution to this problem based on the normal flow field — the spatiotemporal derivatives of the image intensity function, as opposed to the typical optical flow field. Irani and Anandan [16] proposed a three-frames constraint based on a general 3D motion parallax model to detect independent motion. Argyros et al [4, 2, 3, 5] and Lourakis et al [23] used stereo camera streams to detect independent motion. Their techniques were essentially the combination of applying the normal flow field to the stereo streams and using robust statistical regression. Fejes and Davis [12] developed a low-dimensional, projection-based algorithm to separate independent motion using the epipolar structure of rigid 3D motion flow fields. Torr [35] proposed a method based on model selection and segmentation for separating multiple 3D motion components. Pless et al [27] provided a solution to the problem in a special case in which the scene may be approximated as a plane, which is valid for typical aerial surveillance. Their method is based on spatiotemporal intensity gradient measurements to directly compute an exact background motion model, and then the independent motion is detected based on the constraint violation for the mosaics developed over many frames. Sawhney et al [29] proposed a method that simultaneously exploits both constraints of epipolar and shape constancy over multiple frames. This method is based on the previous work on plane-plus-parallax decomposition [19, 28, 31],

and thus requires explicitly estimating the epipolar and the homography between a pair of frames. The other two pieces of related work are Ma et al [24] and Liu et al [22].

3 LSCA Approach

We propose a holistic, in-compression approach to solving for the unsupervised segmentation problem for the independent motion detection based on the linear system consistency analysis theory, and thus, we call this approach as **LSCA**. We use a 3D to 2D affine model to approximate the video camera imaging system. For a typical surveillance video, where the local changes in the ground are much smaller than the depth from a sensor to the ground, this affine model is sufficiently accurate for the mapping from 3D scenes to 2D images. Our experiments also show that this model even works well for some of the non-surveillance video such as movies (see Fig. 3 for an example).

Given a 3D point P and its corresponding 2D point p, a 3D to 2D affine transform is a linear transform, and is defined as [17]:

$$p = AP + t \tag{1}$$

where A is a 2 by 3 matrix with six independent parameters, and t is a 2D vector with another two independent parameters.

Assume that the camera motion between two arbitrary frames is an arbitrary 3D motion, which can be represented as a 3 by 3 rotation matrix R, and a 3D translation vector T.

$$P' = RP + T \tag{2}$$

where P' is the same point of P after the camera motion in the 3D space. The displacement of the point P in the 3D space with respect to time after the motion is:

$$\dot{P} = P' - P = (R - I)P + T \tag{3}$$

where I is the identity matrix. From Eq. 1 and Eq. 3, it is clear:

$$\dot{p} = A\dot{P} = A(R - I)P + AT \tag{4}$$

Let $P = (X, Y, Z)^T$ and $p = (x, y)^T$. Given each image point p, Eqs. 4 and 1 give rise to four independent equations. Eliminating P, we obtain a linear constraint for each image point p in a video frame:

$$\dot{x} + \theta\dot{y} + \alpha x + \beta y + \gamma = 0 \tag{5}$$

where the variables α, β, γ, and θ are functions of the motion parameters R, T between the two frames and the sensor parameters A, t with the following relationship:

$$\alpha = \frac{f(m_{ij})}{g(m_{ij})} \tag{6}$$

where m_{ij} are the motion parameters (the elements of R and T) and/or the sensor parameters (the elements of A and t), and f, g are both quadratic functions. Similar expressions exist for β, θ, γ.

When a pair of neighboring frames is determined, the motion parameters R, T are constants for all the image points in the frames. This indicates that for each point in a frame, there is a linear constraint represented in Eq. 5. Given n points, we have a linear system:

$$D\xi = b \tag{7}$$

$$D = \begin{pmatrix} \dot{y}_1 & x_1 & y_1 & 1 \\ \dots & & & \dots \\ \dot{y}_n & x_n & y_n & 1 \end{pmatrix} \quad \xi = \begin{pmatrix} \theta \\ \alpha \\ \beta \\ \gamma \end{pmatrix} \quad b = \begin{pmatrix} -\dot{x}_1 \\ \dots \\ -\dot{x}_n \end{pmatrix}$$

with the following theorem:

Theorem 1. Given n points represented in the linear system of Eq. 7, if there is no independent motion with any of these points, then the linear system is consistent.

This means that the consistency of the linear system is the necessary condition of no independent motion in the n points. In general, given $n > 4$, the rank of D is 4; consequently, the consistency of Eq. 7 means that there is a unique solution to this linear system. From Theorem 1, it is clear that if the linear system Eq. 7 is not consistent, there must be independent motion involved. However, the linear consistency of the system Eq. 7 is not the sufficient condition for detecting any independent motion of the n points. Nevertheless, we can still use it to detect the independent motion. This may be subject to a false negative. Similarly, if a large computation noise occurs (e.g., the image point localization errors, the displacement vector estimation errors), a consistent linear system could turn out to be inconsistent. In this case, a false positive would be generated. In general, a few false positives are allowed while the number of false negatives must be guaranteed to a minimum.

Now the question is, given n points in two frames, how to determine whether the linear system Eq. 7 is consistent. By linear algebra theory [20], Eq. 7 is consistent *iff*

$$Rank(D) = Rank(D_b) \tag{8}$$

where D_b is the augmented matrix of Eq. 7. In order to determine the rank of the above two matrices, we apply singular value decomposition (SVD) to both D and D_b, and define

$$R = \frac{\sigma_{min}(D)}{\sigma_{min}(D_b)} \tag{9}$$

where $\sigma_{min}(D)$ and $\sigma_{min}(D_b)$ are the smallest singular values of D and D_b, respectively. Clearly, if Eq. 8 exactly holds true, $\sigma_{min}(D) > 0$ and $\sigma_{min}(D_b) = 0$, leading to $R \to \infty$. In practice, Eq. 7 is consistent *iff* R is above a threshold, following the theory and practice of [39, 40, 38]. Note that from the definition of Eq. 9, $R \geq 1$.

Recall Eqs. 1 and 4. Instead of applying them to a set of feature points in a frame, we now apply them to *every* points of a region of m points in the frame. Thus, we have

$$\sum_{i=1}^{m} \dot{p}_i = A(R-I)\sum_{i=1}^{m} P_i + mAT \qquad \sum_{i=1}^{m} p_i = A\sum_{i=1}^{m} P_i + mt \tag{10}$$

Define

$$\bar{p} = \frac{1}{m}\sum_{i=1}^{m} p_i = (\bar{x},\bar{y})^T \qquad \bar{\dot{p}} = \frac{1}{m}\sum_{i=1}^{m} \dot{p}_i = (\bar{\dot{x}},\bar{\dot{y}})^T \qquad \bar{P} = \frac{1}{m}\sum_{i=1}^{m} P_i = (\bar{X},\bar{Y},\bar{Z})^T$$

we obtain

$$\bar{\dot{p}} = A(R-I)\bar{P} + AT \qquad \bar{p} = A\bar{P} + t \tag{11}$$

If we take each MPEG macroblock as such a region, then m becomes a constant (i.e., $m = 256$) over the whole frame. Therefore, we have a similar linear constraint for *each macroblock* of a frame:

$$\bar{\dot{x}} + \theta\bar{\dot{y}} + \alpha\bar{x} + \beta\bar{y} + \gamma = 0 \tag{12}$$

and consequently, given n macroblocks, we can build a similar linear system

$$D_m = \xi_m b_m \tag{13}$$

Thus, we have a similar theorem:

Theorem 2. Given n macroblocks in an MPEG video frame represented in the linear system of Eq. 13, if there is no independent motion with any of these macroblocks, then the linear system is consistent.

In the MPEG compression standard, for each macroblock in a frame, if this macroblock is inter-coded, there is a motion vector available. We approximate $\bar{\dot{p}}$ with the motion vector, and $\bar{p}$ is the center of the macroblock. Since the macroblock information (including the motion vector and the center coordinates) can be easily obtained directly from a compressed MPEG video stream, we have a linear system Eq. 13 that can directly work on the MPEG compressed data without having to depend on a specific algorithm to compute the correspondence or optical flow between the two frames, simultaneously eliminating the two potential problems mentioned above with Eq. 7. If the macroblock is intra-coded, we just exclude this macroblock from the linear system of Eq. 13. If the frame is an I-frame in which all the macroblocks are intra-coded, we can obtain the motion vector of a macroblock by predicting it from the one in the previous B-frame.

While the displacement vector may be approximated by the motion vector of a macroblock, this may create another problem, i.e., how accurate this approximation is. It is known [33] that the motion vector estimation in MPEG is subject to errors, and how large the errors are depends on the specific implementation of the motion vector estimation algorithm under the MPEG standard [7]. The theoretic relationship between the errors in motion vector estimation in MPEG and the detection accuracy is shown in the Appendix. Here we provide a tentative solution to

this problem based on the normal flow computation to attempt to lower the potential errors for the motion estimation. Research shows [37] that the normal flow is more reliable than the standard optical flow. Assuming that the intensity function of a frame is $I(x,y)$, the normal flow n_p at the point $p = (x,y)^T$ is defined as the dot product between the normalized gradient of the point p and the displacement vector at this point:

$$n_p = \frac{\partial I}{\partial x}\dot{x} + \frac{\partial I}{\partial y}\dot{y} \tag{14}$$

Since in the compressed MPEG video stream we only have the motion vectors for each macroblock as opposed to each point, we must extend this point-based normal flow definition to the macroblock based one. Let $\nabla I(p)$ be the normalized gradient of the intensity function I at a point p. Given a macroblock M, the *macroblock gradient* $\nabla I(M)$ is defined as:

$$\nabla I(M) = \frac{1}{m}\sum_{i=1}^{m} \nabla I(p_i) \tag{15}$$

where p_i is a point of M, and m is the total number of points in M. In MPEG, $m = 256$.

Now the question is how to estimate the gradient of a macroblock without decompressing the video data. Lee et al [21] showed a method of estimating the approximated gradient for a whole block only using a few low frequency AC coefficients of the DCT of the block in MPEG. This is essentially to approximate the original DCT AC coefficients AC_{uv} with the corresponding "continuous" versions $\tilde{AC}_{uv}$:

$$AC_{uv} \approx \tilde{AC}_{uv} = C(u)C(v)\int_0^8 \int_0^8 cos\frac{xu\pi}{8} cos\frac{yv\pi}{8} I(x,y)dxdy \tag{16}$$

where $C(u)$ and $C(v)$ are the scale factors of the standard DCT definition [33]. Given a few limited lower frequency terms of AC_{uv}, we can explicitly solve for the block edge orientation, the block edge offset, and the block edge strength [21]. The question, however, is how many such lower AC coefficients would suffice an accurate estimate of the block gradient. Reported research [21] shows that in order to estimate the block gradient, only the five lowest frequency AC coefficients are necessary to recover the information (i.e., $AC_{01}, AC_{10}, AC_{20}, AC_{11}, AC_{02}$). Consequently, the majority of the AC coefficients as well as the DC component are not required. This shows that it is still not necessary to decompress the video stream in order to recover the block gradient; the method can directly work on the compressed MPEG stream to extract the small piece of the "essential" information (i.e., the motion vector of a macroblock and the five low frequency AC components of a block) without having to decompress the video stream.

Once we have the block gradient vectors available for all the four blocks of a macroblock, the macroblock gradient is computed by averaging the four block gradient vectors based on the definition. Finally, the normal flow value of a macroblock, $n(M)$, is defined similar to that of a point in Eq. 14 by taking the dot product between

the macroblock gradient vector, $\nabla I(M)$, and the motion vector of this macroblock, $\mathbf{V}(M)$

$$n(M) = \nabla I(M) \cdot \mathbf{V}(M) \tag{17}$$

When we have the normal flow value computed for a macroblock, we can make a decision regarding whether this macroblock should be incorporated into the linear system of Eq. 13. In Appendix, we show the theoretic error bound for the detection based on **LSCA** on the motion vector errors. It is shown that as long as the motion vector errors are within the given bound, **LSCA** is capable of detecting the independent motion video segments. Furthermore, even with outliers larger than the error bound but the outliers are sporadic in distribution, **LSCA** can still filter them out and result in a correct segmentation. This observation is further verified in the experiments in Sec. 4. On the other hand, this analysis also indicates that **LSCA** is not appropriate for dense outliers larger than the given error bound, in which case, the outliers would be considered as independent motion. As demonstrated by the real data experiments, we argue that in many applications it is rare to have dense outliers larger than the given error bound.

Now the **LSCA** algorithm is summarized in Algorithm 1, which takes four parameters: the normal flow threshold T_n, the scan window width r, the R statistic threshold T_R, and the defined minimum number of frames T_f of a segment that contains independent motion.

Note that **LSCA** is based on the assumption of constant camera model in terms of the sensor parameters A and t. In real applications, it is possible that the camera

Input: a video stream in compressed MPEG. **Output:** All the video segments containing independent motion. **Method:**

for Every pair of neighboring frames **do**
  Start to build up the linear system Eq. 13.
  for Each macroblock M of the first frame l of the pair **do**
    Estimate the normal flow $n(M)$ of M.
    if $n(M) > T_n$ **then**
      Incorporate M into Eq. 13 based on Eq. 12.
    end if
  end for
  Compute R of the linear system Eq. 13.
  Compute the median filtered $\bar{R}$ over a window of r frames.
end for
if $\bar{R} - 1 > T_R$ **then**
  Label l as a frame with no independent motion.
else
  Label l as a frame with independent motion.
end if
Any independent motion segment with frame number $> T_f$ is retrieved.

Algorithm 1. LSCA Algorithm

internal parameters change during the surveillance (e.g., zoom in/out). Since **LSCA** only focuses on two neighboring frames, given the current video frame rate (about 30 frames/second), if the change is slow, we can ignore the change and still use the algorithm to compute the R statistic between the two frames; if the change is fast, the computed R value between the two frames may be wrong, which will lead to a false positive or negative. However, in this case, there will be only a few frames subject to the error of R values, and they will be shown as outliers of a temporal window and will then typically be filtered out by **LSCA**.

Based on the analysis given above, it is clear that as a novel unsupervised video segmentation solution to independent motion detection, **LSCA** has the following distinctive advantages as compared with the existing methods in the literature:

1. No camera calibration is required or necessary in order to apply **LSCA**.
2. The statistics computed in **LSCA** are stable due to the low condition number in the linear system.
3. **LSCA** is very fast because it directly works with compressed stream.

4 Experimental Evaluations

In this section, we first report a simulation based analysis on estimating the detection false positives and false negatives, as well as the detectability bounds for **LSCA**. We then present the real data experimental evaluations to demonstrate the robustness and effectiveness of **LSCA**.

Both analyses on false positives and false negatives are the *sensitivity* analysis for **LSCA**. This may be achieved by testing the *stability* of the statistic R of **LSCA** under different levels of noise through simulation.

The simulated data consist of 10 3D points with no independent motion and another 3D point with independent motion all projected to a 100×120 pixel image. Thus, 1 pixel deviation of Gaussian noise approximately corresponds to 1% of the *whole* effective image dimension.

The corrupted image coordinates and the displacement vectors are input into **LSCA**, and the R statistic is computed for each noise level. Since there is no independent motion involved in this scenario, the R value should be high. Under the corruption of the noise, however, the R value degrades as the noise level increases. Fig. 1(a) shows the logarithm of the R values averaged over 1000 runs with different seeds under each Gaussian noise level parameterized by the standard deviation in terms of the number of pixels when there is no independent motion.

From the Figure, if the noise level is controlled under 2 pixels, the R value always stabilizes somewhere statistically significantly higher than 1 (above 2). Note that considering the effective image dimension as 100 by 120, 2 pixels' noise is significantly large in practice. This shows that **LSCA** is very robust in rejecting false positives in independent motion detection.

The simulation scenario continues when the point of independent motion is added into the original 10 point set. This time the R value always stays at 1 regardless

of what level the noise is, indicating **LSCA** is effective in detecting independent motion.

While false positives and false negatives are the probabilities describing an event of a detection failure of **LSCA**, detectability is an issue of how significant an independent motion should be such that **LSCA** is able to detect it. *Detectability* is a different but a related concept, which is defined as the smallest independent motion that **LSCA** can detect. Fig. 1(b) to (g) show the detectabilities in different scenarios of independent motion. A qualitative examination of these simulation results reveals that the performance of **LSCA** appears less sensitive to the independent motion related to the Z axis (rotation about the axis or translation along the axis) than to the independent motion related to the other axes; quantitatively, based on this simulation, the detectability is related to the noise levels, and a higher noise level increases detectability. The reason is that a higher level of noise increases the false positives, which help increase detectability. Take the noise level of 1.5 pixel deviation for example. If the threshold value is set as 2 for R, the detectability is under 1 unit for all the translations, and under 0.1 degree for all the rotations. If the threshold of R decreases to 1.5, the detectability for translations along X, Y, and Z axes is above 8, 10, and 20 units, respectively, for rotations about X, Y, and Z axes is above 0.4, 0.8, and 3.0 degrees, respectively. Note that these parameters are obtained from this specific set of simulation only. However, since in the simulation data we know the ground truth of the displacement vectors, we observe that the R statistic errors propagated from the displacement vector errors confirm well with the theoretic bound in Eq. 29.

We have implemented **LSCA** as a stand alone version in a Windows2000 platform with Pentium III 800 MHz CPU and 512 MB memory. Fig. 2(a) and (b) show two surveillance video clips for the two scenarios with and without independent motion, and Fig. 2(c) and (d) show the R statistics computed at every frames for the two shots from two surveillance videos in Fig. 2(a) and (b), respectively. The statistics are obvious to tell whether and where there is independent motion in the video. The first shot containing 264 frames describes an independent motion of an airplane landing to its destination. The mean of the R statistics is 1.012 and the deviation is 0.0083 over the 264 frames. The second shot containing 1024 frames surveys an area of ground terrain with no independent motion. The mean of the R statistics is 1.389 and the deviation is 0.169 over the 1024 frames.

In order to give a meaningful evaluation, we make an assumption that a reliable independent motion shot should last at least 30 frames (i.e., $T_f = 30$), which corresponds at least about one second presence of independent motion in the video. This assumption ensures that any sporadic detection false positives due to motion estimation outliers and/or sensor parameter changes will be removed. Since **LSCA** performs frame-based independent motion detection, it is reasonable to define the *detection rate* as the percentage of the number of truthed independent motion frames detected by **LSCA** of the total number of detected independent motion frames, and to define the *detection false alarm* as the percentage of the number of falsely detected independent motion frames reported by **LSCA** of the total number of truthed independent motion frames in a video. Based on these definitions, we have run

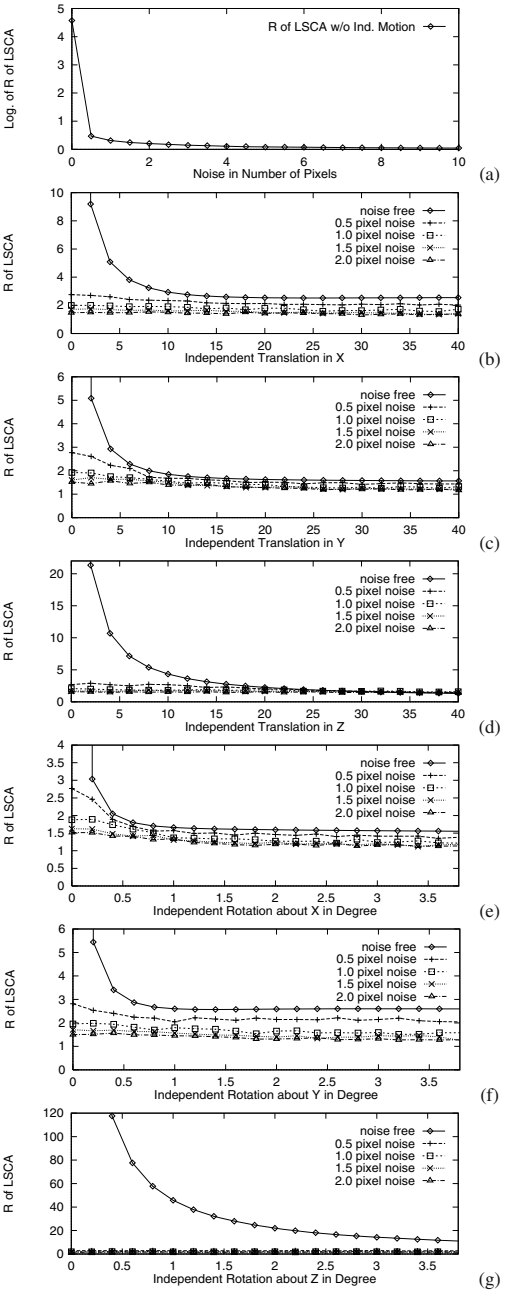


Fig. 1 (a) Logarithm of the R statistic of **LSCA** under different Gaussian noise levels when no independent motion involved. (b)-(d) Detectabilities of **LSCA** w.r.t. independent translations along X, Y, and Z axes, respectively. (e)-(g) Detectabilities of **LSCA** w.r.t. independent rotations about X, Y, and Z axes, respectively.

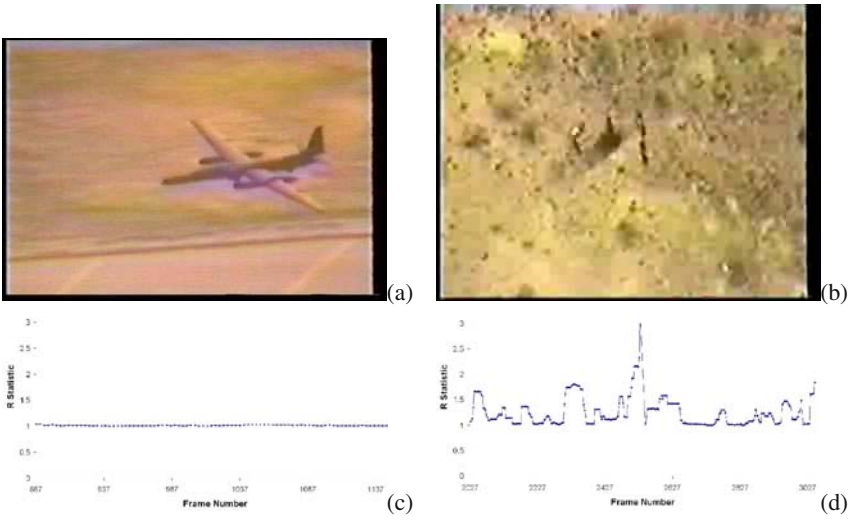


Fig. 2 (a) An example of a shot containing an independently moving object (an airplane) (b) An example of a shot containing no independent motion (terrain) (c) The R statistics computed for the shot in (a) (264 frames) (d) The R statistics computed for the shot in (b) (1024 frames).

LSCA on a video testbed which collects different shots of surveillance video of total 10602 frames. The overall detection rate is 94.9% and the false alarm is 3.07% with the threshold of R as 1.2, which is obtained in a combination of the empirical observation of the experimental data and the simulation-based analysis.

Since for the purpose of reporting the detection rate and the false alarm quantitatively, we must manually truth every frame in the testbed which is rather expensive, we are unable to evaluate **LSCA** using a larger, ground-truthed testbed. However, in addition to the 10602-frame testbed that we used for reporting the detection rate and the false alarm, we have also tested **LSCA** in a much larger data collection that is not ground-truthed. A comparably good performance is observed with different types of independent motion (such as single target independent motion, multiple targets independent motion, targets independently moving in different directions simultaneously, independently moving targets in different sizes, and substantially varying background when an independent motion occurs).

To show that **LSCA** is not only valid for the typical surveillance scenario where the camera is far away from the scene, but also could be valid for the scenario where the camera is relatively close to the scene, Fig. 3 demonstrates an experimental result of **LSCA** in which we take a movie with an independent motion very close to the camera, and split the spatial domain into the left and the right halves such that the left video does not contain independent motion while the right one does. The result clearly shows that **LSCA** is robust even under the situation where the camera is close to the scene.

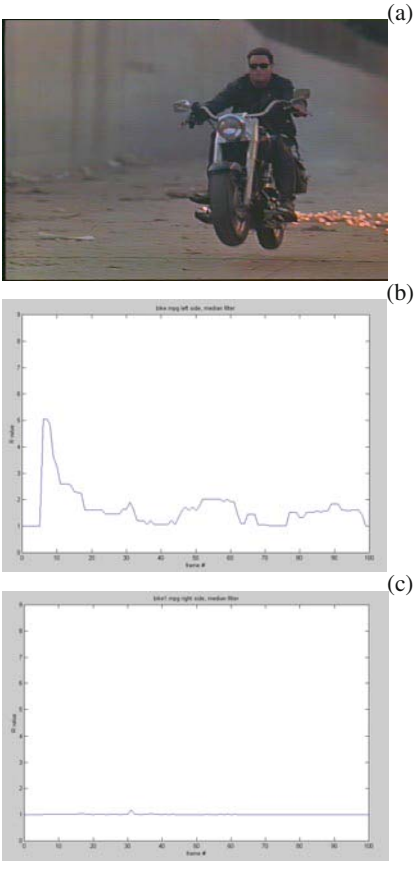


Fig. 3 (a) A frame of a movie (100 frames) (b) The *R* statistics for the left halves of the video (with no independent motion) (c) The *R* statistics for the right halves of the video (with independent motion).

Since **LSCA** essentially just needs to compute the *R* value for each frame, and since in each frame there is typically a very limited number of macroblocks, the complexity of **LSCA** is very low. The current prototype of **LSCA** scans a compressed MPEG video with a typical frame resolution of 240 by 350 at the speed of 35 frames/second under the current platform, which is faster than real-time. Note that this implementation is just for proof of the concept and the code has not been optimized yet. This shows that **LSCA** holds a great promise in the future applications in both proposed scenarios: real time surveillance data scanning equipped with the sensors and efficient data mining for an archived database of surveillance video.

5 Conclusions

This chapter focuses on developing a large scale surveillance video data unsupervised segmentation technique regarding whether there is a presence of independent motion. We propose a holistic, in-compression approach **LSCA** to efficient video segmentation. By efficient, we mean that the processing speed is close to or even faster than real-time in "normal" platforms (we do not assume using special hardware or any parallel machines) while still maintaining a good quality segmentation. Theoretical and experimental analyses demonstrate and validate the **LSCA** technique in solving for the video segmentation for independent motion detection problem.

Appendix: Theoretic Detection Bound of LSCA Method

To simplify the notations, in the rest of the chapter, we will drop the subscript m for the matrices D_m, $D_m b_m$, and the vectors ξ_m, b_m in Eq. 13, and $\bar{p} = (\bar{x}, \bar{y})^T \Longrightarrow p = (x, y)^T$, and $\bar{\dot{p}} = (\bar{\dot{x}}, \bar{\dot{y}})^T \Longrightarrow \dot{p} = (\dot{x}, \dot{y})^T$. Define $\tilde{\dot{p}} = \dot{p} + \triangle\dot{p} = (\dot{x}, \dot{y})^T + (\triangle\dot{x}, \triangle\dot{y})^T$ as the motion vector given in the MPEG streams, which is assumed here to be decomposed into the true motion vector $\dot{p} = (\dot{x}, \dot{y})^T$ and the error $\triangle\dot{p} = (\triangle\dot{x}, \triangle\dot{y})^T$. Thus,

$$(D + \triangle D)\xi = b + \triangle b \tag{18}$$

where

$$D = \begin{pmatrix} \dot{y}_1 & x_1 & y_1 & 1 \\ \dots & & & \dots \\ \dot{y}_n & x_n & y_n & 1 \end{pmatrix} \quad \triangle D = \begin{pmatrix} \triangle\dot{y}_1 & 0 & 0 & 0 \\ \dots & & & \dots \\ \triangle\dot{y}_n & 0 & 0 & 0 \end{pmatrix} \quad b = \begin{pmatrix} -\dot{x}_1 \\ \dots \\ -\dot{x}_n \end{pmatrix} \quad \triangle b = \begin{pmatrix} -\triangle\dot{x}_1 \\ \dots \\ -\triangle\dot{x}_n \end{pmatrix}$$

Define the augmented matrix

$$H = D_b = \begin{pmatrix} \dot{y}_1 & x_1 & y_1 & 1 & -\dot{x}_1 \\ \dots & & & & \dots \\ \dot{y}_n & x_n & y_n & 1 & -\dot{x}_n \end{pmatrix}$$

and further define

$$H' = H + \triangle H = (D + \triangle D)|(b + \triangle b)$$

resulting in

$$\triangle H = \begin{pmatrix} \triangle\dot{y}_1 & 0 & 0 & 0 & -\triangle\dot{x}_1 \\ \dots & & & & \dots \\ \triangle\dot{y}_n & 0 & 0 & 0 & -\triangle\dot{x}_n \end{pmatrix}$$

We introduce the following notations. For a matrix B, we denote $\lambda_i(B)$ as the ith eigenvalue of the matrix B, and $\sigma_i(B)$ as the ith singular value of the matrix B. In particular, we denote $\lambda_{min}(B)$ as the smallest eigenvalue of the matrix B, and $\sigma_{min}(B)$

as the smallest singular value of the matrix B. We assume that for a matrix B, all the eigenvalues or the singular values are sorted from the largest to the smallest.

From the perturbation theory of singular value decomposition [14], we have

$$\|\sigma_{min}(H+\triangle H)-\sigma_{min}(H)\| \leq \sigma_1(\triangle H) \tag{19}$$

$$\|\sigma_{min}(D+\triangle D)-\sigma_{min}(D)\| \leq \sigma_1(\triangle D) \tag{20}$$

From the relationship between the eigenvalues and the singular values of the corresponding matrices [14], we have

$$\sigma_1^2(\triangle H) = \lambda_1(\triangle H^T \triangle H) \tag{21}$$

$$\sigma_1^2(\triangle D) = \lambda_1(\triangle D^T \triangle D) \tag{22}$$

By explicitly solving for the eigenvalues of the matrices $\triangle H^T \triangle H$ and $\triangle D^T \triangle D$, we have

$$\sigma_1(\triangle H) = \sqrt{\frac{\sum_{i=1}^n \triangle \dot{y}_i{}^2 + \sum_{i=1}^n \triangle \dot{x}_i{}^2 + \sqrt{(\sum_{i=1}^n \triangle \dot{y}_i{}^2 - \sum_{i=1}^n \triangle \dot{x}_i{}^2)^2 + 4(\sum_{i=1}^n \triangle \dot{x}_i \triangle \dot{y}_i)^2}}{2}} \tag{23}$$

$$\sigma_1(\triangle D) = \sqrt{\sum_{i=1}^n \triangle \dot{y}_i{}^2} \tag{24}$$

Based on the definition in Eq. 9, now we have

$$\begin{aligned}
\triangle R &= \textstyle\sum_{i=1}^n \frac{\partial R}{\partial \dot{x}_i}\triangle \dot{x}_i + \sum_{i=1}^n \frac{\partial R}{\partial \dot{y}_i}\triangle \dot{y}_i \\
&= \textstyle\sum_{i=1}^n \frac{\frac{\partial \sigma_{min}(D)}{\partial \dot{x}_i}\sigma_{min}(H) - \frac{\partial \sigma_{min}(H)}{\partial \dot{x}_i}\sigma_{min}(D)}{\sigma_{min}^2(H)}\triangle \dot{x}_i \\
&\quad + \textstyle\sum_{i=1}^n \frac{\frac{\partial \sigma_{min}(D)}{\partial \dot{y}_i}\sigma_{min}(H) - \frac{\partial \sigma_{min}(H)}{\partial \dot{y}_i}\sigma_{min}(D)}{\sigma_{min}^2(H)}\triangle \dot{y}_i \\
&= \textstyle\frac{1}{\sigma_{min}(H)}\sum_{i=1}^n (\frac{\partial \sigma_{min}(D)}{\partial \dot{x}_i}\triangle \dot{x}_i + \frac{\partial \sigma_{min}(D)}{\partial \dot{y}_i}\triangle \dot{y}_i) \\
&\quad - \textstyle\frac{\sigma_{min}(D)}{\sigma_{min}^2(H)}\sum_{i=1}^n (\frac{\partial \sigma_{min}(H)}{\partial \dot{x}_i}\triangle \dot{x}_i + \frac{\partial \sigma_{min}(H)}{\partial \dot{y}_i}\triangle \dot{y}_i)
\end{aligned} \tag{25}$$

Based on the calculus theory,

$$\|\sum_{i=1}^n (\frac{\partial \sigma_{min}(D)}{\partial \dot{x}_i}\triangle \dot{x}_i + \frac{\partial \sigma_{min}(D)}{\partial \dot{y}_i}\triangle \dot{y}_i)\| \simeq \|\sigma_{min}(D+\triangle D)-\sigma_{min}(D)\| \tag{26}$$

and

$$\|\sum_{i=1}^n (\frac{\partial \sigma_{min}(H)}{\partial \dot{x}_i}\triangle \dot{x}_i + \frac{\partial \sigma_{min}(H)}{\partial \dot{y}_i}\triangle \dot{y}_i)\| \simeq \|\sigma_{min}(H+\triangle H)-\sigma_{min}(H)\| \tag{27}$$

Consequently, from Eq. 25, we have

$$\begin{aligned}\|\triangle R\| \leq & \|\frac{1}{\sigma_{min}(H)}\Sigma_{i=1}^{n}(\frac{\partial\sigma_{min}(D)}{\partial\dot{x}_i}\triangle\dot{x}_i+\frac{\partial\sigma_{min}(D)}{\partial\dot{y}_i}\triangle\dot{y}_i)\| \\ & +\|\frac{\sigma_{min}(D)}{\sigma_{min}^2(H)}\Sigma_{i=1}^{n}(\frac{\partial\sigma_{min}(H)}{\partial\dot{x}_i}\triangle\dot{x}_i+\frac{\partial\sigma_{min}(H)}{\partial\dot{y}_i}\triangle\dot{y}_i)\|\end{aligned} \tag{28}$$

Since $\sigma_{min}(D) > 0, \sigma_{min}(H) > 0$, from Eqs. 19, 20, 23, 24, 26, and 27, we have the theoretic error bound for R:

$$\begin{aligned}\|\triangle R\| \leq & \frac{\sqrt{\Sigma_{i=1}^{n}\triangle\dot{y}_i{}^2}}{\sigma_{min}(H)} \\ & +\frac{\sigma_{min}(D)}{\sigma_{min}^2(H)}\sqrt{\frac{\Sigma_{i=1}^{n}\triangle\dot{y}_i{}^2+\Sigma_{i=1}^{n}\triangle\dot{x}_i{}^2+\sqrt{(\Sigma_{i=1}^{n}\triangle\dot{y}_i{}^2-\Sigma_{i=1}^{n}\triangle\dot{x}_i{}^2)^2+4(\Sigma_{i=1}^{n}\triangle\dot{x}_i\triangle\dot{y}_i)^2}}{2}}\end{aligned} \tag{29}$$

Experimental data (see Sec 4) have shown that this error bound is consistent very well with the error distributions in the data.

Acknowledgements. This work is supported in part by NSF through grants IIS-0535162 and IIS-0812114, and AFOSR through grant F49620-03-1-0007. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the funding agencies. Stoyan Kurtev of International University – Bremen, Germany, Xunyin Wang and Abhay Garg at SUNY Binghamton, USA, helped in part of the implementation and the experiments for the early versions of the systems developed in this work.

References

1. Adiv, G.: Determining 3D motion and structure from optical flows generated by several moving objects. IEEE Trans. Pattern Analysis and Machine Intelligence 7(4), 384–401 (1985)
2. Argyros, A.A., Lourakis, M.I.A., Trahanias, P.E., Orphanoudakis, S.C.: Fast visual detection of changes in 3D motion. In: Proc. IAPR Workshop on Machine Vision Applications (1996)
3. Argyros, A.A., Lourakis, M.I.A., Trahanias, P.E., Orphanoudakis, S.C.: Independent 3D motion detection through robust regression in depth layers. In: Proc. British Machine Vision Conference (1996)
4. Argyros, A.A., Lourakis, M.I.A., Trahanias, P.E., Orphanoudakis, S.C.: Qualitative detection of 3D motion discontinuities. In: Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems (1996)
5. Argyros, A.A., Orphanoudakis, S.C.: Independent 3D motion detection based on depth elimination in normal flow fields. In: Proc. International Conference on Computer Vision and Pattern Recognition. IEEE Computer Society Press, Los Alamitos (1997)
6. Ayer, S., Schroeter, P., Bigun, J.: Segmentation of moving objects by robust motion parameter estimation over multiple frames. In: Proc. European Conference on Computer Vision (1994)
7. http://mpeg.telecomitalialab.com/

8. Bouthemy, P., Francois, E.: Motion segmentation and qualitative dynamic scene analysis from an image sequence. International Journal of Computer Vision 10(2), 157–182 (1993)
9. Cai, Q., Aggarwal, J.K.: Tracking human motion in structured environments using a distributed-camera system. IEEE Trans. Pattern Analysis and Machine Intelligence 21(11), 1241–1247 (1999)
10. Cutler, R., Davis, L.S.: Robust real-time periodic motion detection, analysis, and applications. IEEE Trans. Pattern Analysis and Machine Intelligence 22(8), 781–796 (2000)
11. Faugeras, O.: Three-Dimensional Computer Vision: A Geometric Viewpoint. MIT Press, Cambridge (1993)
12. Fejes, S., Davis, L.S.: What can projections of flow fields tell us about the visual motion. In: Proc. International Conf. Computer Vision (1998)
13. Forsyth, D., Mundy, J.L., Zisserman, A., Coelho, C., Heller, A., Rothwell, C.: Invariant descriptors for 3-D object recognition and pose. IEEE Trans. Pattern Analysis and Machine Intelligence 13(10), 971–991 (1991)
14. Golub, G.H., Loan, C.F.V.: Matrix Computations, 2nd edn. The Johns Hopkins University Press, Baltimore (1989)
15. Huang, T.S., Lee, C.H.: Motion and structure from orthographic views. IEEE Trans. Pattern Analysis and Machine Intelligence 11, 536–540 (1989)
16. Irani, M., Anandan, P.: A unified approach to moving object detection in 2D and 3D scenes. In: Proc. of IUW (1996)
17. Jacobs, D.W.: Recognizing 3-D Objects Using 2-D Images. Ph.D. Dissertation, MIT AI Lab. (1992)
18. Jain, R.C.: Segmentation of frame sequences obtained by a moving observer. IEEE Trans. Pattern Analysis and Machine Intelligence 7(5), 624–629 (1984)
19. Kumar, R., Anandan, P., Hanna, K.: Direct recovery of shape from multiple views: a parallax based approach. In: Proc. International Conf. Pattern Recognition (1994)
20. Lang, S.: Linear Algebra, 3rd edn. Springer, Heidelberg (1987)
21. Lee, S.-W., Kim, Y.-M., Choi, S.W.: Fast scene change detection using direct feature extraction from MPEG compressed videos. IEEE Trans. Multimedia 2(4), 240–254 (2000)
22. Liu, Q., Hua, Z., Zang, C., Tong, X., Lu, H.: Providing on-demand sports video to mobile devices. In: Proc. ACM Multimedia (2005)
23. Lourakis, M.I.A., Argyros, A.A., Orphanoudakis, S.C.: Independent 3D motion detection using residual parallax normal flow fields. In: Proc. International Conference on Computer Vision. IEEE Computer Society Press, Los Alamitos (1998)
24. Ma, Y.-F., Lu, L., Zhang, H.-J., Li, M.: A user attention model for video summarization. In: Proc. ACM Multimedia (2002)
25. Maybank, S.: Theory of Reconstruction from Image Motion. Springer, Heidelberg (1993)
26. Nelson, R.C.: Qualitative detection of motion by a moving observer. In: Proc. of CVPR. IEEE, Los Alamitos (1991)
27. Pless, R., Brodsky, T., Aloimonos, Y.: Detecting independent motion: the statistics of temporal continuity. IEEE Trans. Pattern Analysis and Machine Intelligence 22(8), 768–773 (2000)
28. Sawhney, H.S.: 3D geometry from planar parallax. In: Proc. International Conference on Computer Vision and Pattern Recognition (1994)
29. Sawhney, H.S., Guo, Y., Kumar, R.: Independent motion detection in 3D scenes. IEEE Trans. Pattern Analysis and Machine Intelligence 22(10), 1191–1199 (2000)
30. Sharma, R., Aloimonos, Y.: Early detection of independent motion from active control of normal image flow patterns. IEEE Trans. SMC 26(1), 42–53 (1996)

31. Shashua, A., Navab, N.: Relative affine structure: theory and application to 3D reconstruction from perspective views. In: Proc. International Conference on Computer Vision and Pattern Recognition (1994)
32. Smith, S.M., Brady, J.M.: ASSET-2: real time motion segmentation and shape tracking. IEEE Trans. Pattern Analysis and Machine Intelligence 17(8) (1995)
33. Tekalp, A.M.: Digital Video Processing. Prentice-Hall, Englewood Cliffs (1995)
34. Thompson, W., Lechleider, P., Stuck, E.: Detecting moving objects using the rigidity constraint. PAMI 15 (1993)
35. Torr, P.H.S.: Geometric motion segmentation and model selection. Philosophical Trans. Royal Soc. A, 1321–1340 (1998)
36. Ullman, S.: The Interpretation of Visual Motion. MIT Press, Cambridge (1979)
37. Verri, A., Poggio, T.: Motion field and optical flow: qualitative properties. IEEE Trans. Pattern Analysis and Machine Intelligence 11(5), 490–498 (1989)
38. Zhang, Z.: 3D Reconstruction under Varying Constraints on Camera Geometry for Robotic Navigation Scenarios. PhD thesis, CMPSCI 96-08, University of Massachusetts/Amherst (1996)
39. Zhang, Z., Weiss, R., Hanson, A.R.: Qualitative obstacle detection. In: IEEE International Conference on CVPR. IEEE Computer Society Press, Los Alamitos (1994)
40. Zhang, Z., Weiss, R., Hanson, A.R.: Obstacle detection based on qualitative and quantitative 3D reconstruction. PAMI 19(1) (1997)

Video Repeat Recognition and Mining by Visual Features

Xianfeng Yang and Qi Tian

Abstract. Repeat video clips such as program logos and commercials are widely used in video productions, and mining them is important for video content analysis and retrieval. In this chapter we present methods to identify known and unknown video repeats respectively. For known video repeat recognition, we focus on robust feature extraction and classifier learning problems. A clustering model of visual features (e.g. color, texture) is proposed to represent video clip and subspace discriminative analysis is adopted to improve classification accuracy, which results in good results for short video clip recognition. We also propose a novel method to explore statistics of video database to estimate nearest neighbor classification error rate and learn the optimal classification threshold. For unknown video repeat mining, we address robust detection, searching efficiency and learning issues. Two detectors in a cascade structure are employed to efficiently detect unknown video repeats of arbitrary length, and this approach combines video segmentation, color fingerprinting, self-similarity analysis and Locality-Sensitive Hashing (LSH) indexing. A reinforcement learning approach is also adopted to efficiently learn optimal parameters. Experiment results show that very short video repeats and long ones can be detected with high accuracy. Video structure analysis by short video repeats mining is also presented in results.

1 Introduction

Video repeats which refer to copies of a video clip ubiquitously exist in broadcast and web videos, and their distributions embed abundant structural information both in program level and web database scale. The most common video repeats are those short video clips from a few seconds to several minutes such as TV commercials, station logo or program logo, etc. To discover and locate video repeats from large video database or video streams robustly and efficiently is very

Xianfeng Yang
Faculty of Engineering, National University of Singapore, Singapore
xianfengy@gmail.com

Qi Tian
Institute for Infocomm Research, Singapore
tianqig@gmail.com

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 305–326.
springerlink.com

important for video content analysis and retrieval. For example, by detecting video repeats in unlabeled video data, we can find correlation of different video parts and discover structural video elements used for syntactic segmentation purpose, hence video structure model can be effectively constructed and applied to video syntactical segmentation ([1][2]). Video repeat mining also has many other prospective applications, such as commercial monitoring ([3][4][5]), video copy detection ([6][7]), web video multiplicity estimation ([8]), video content summary, personalization as well as lossless video compression ([9]).

Video repeat mining tasks can be divided into two categories: known video repeat mining and unknown video repeat mining. For known video repeat mining, we often construct a feature vector set from prototype videos and use nearest neighbor (NN) classifier to recognize copies of prototype videos from video collections or streams. In this part we focus on the feature representation and classifier learning problems. Since video copies located in different video sources have different formats, e.g. different frame sizes, frame rates as well as bitrates, so diverse distortions pose a big challenge to video copy recognition. So far many research efforts on video identification have been dedicated to extraction of distinct and robust video features from color or geometry field ([10][11][12]), called video hashing, with the aim to map video object to a unique hash code that could also be robust to kinds of distortions. However, finding a general robust yet distinct video hash code is very difficult, so the question is: if video features do not show good identification performance under certain video distortions, can they be transformed to a better one? In this chapter we examine commonly used visual features (e.g. color histogram, texture) and improve their video recognition performance under significant distortions through subspace discriminative analysis. Subspace discriminative analysis is extensively used in face recognition and text classification ([13][14][15]), and we will show it also results in very promising results in video copy recognition ([16]).

To obtain the minimum error classifier, we propose a novel method to explore statistics of prototype video database in order to estimate error rate of threshold NN classifier and learn the optimal threshold. Three types of 'sample-to-database' distances are defined, and error rate is exactly estimated from the three distance distributions. Compared to 'sample-to-sample' distance ([17]), 'sample-to-database' distance is naturally related to the feature distribution of video database, thus making database statistics and error rate estimation more reasonable.

Unknown repeat mining task is usually implemented on video collections from the same source, e.g. broadcast videos in different days, to analyze video structure, and the challenge is that prior knowledge about video repeats such as their content, length and location, is not known in advance, moreover video repeats in different locations may also have distortions, e.g. caption overlay, partial repeats. In unknown repeat mining section we will address robust detection, searching efficiency and learning issues. The approach we proposed combines video segmentation, color fingerprinting, self-similarity analysis, cascaded detection, LSH indexing and reinforcement learning. Compared to other media repeat pattern identification methods ([3][9][18]), our approach can detect very short repeats (e.g. those less than 1 second) along with long ones, and high accuracy has been

achieved in our experiments. Methods by Cheung et al. ([3]) and Herley ([18]) both use a fixed time window to do feature extraction and comparison, so those repeats significantly shorter than the window are very likely be missed. The method by Pua et al. ([9]) is able to identify repeated shots but can not identify partially repeated shots, while our approach can identify even small portion of a shot or clip by adopting segmentation with granularity smaller than the shot. Another novelty of our approach is that a reinforcement learning approach is adopted to train the video repeat detectors, and this approach demonstrates efficiency in parameter learning, which makes the repeat mining system manageable and easy to train.

The remainder of this chapter is arranged as follows: In section 2, we present the known video repeat recognition approach and results. In section 3, the unknown video repeat mining method and results are presented. In section 4, the concluding remarks are discussed.

2 Known Video Repeat Recognition

In this section we first propose a model clustering visual features to represent a video clip, and adopt Oriented PCA (OPCA) approach to transform this video feature to subspace representation in order to improve video model separability while suppressing distortions. We also propose a novel method to explore statistics of video database to estimate error rate of threshold NN classifier and learn the optimal classification threshold. Recognition performance is evaluated under significant video distortions and different video length. Results show that recognition error rate below 5% has been achieved under significant distortions, and subspace representation lead to a large reduction of error rate compared to using original feature, especially for very short video clips (e.g.5s).

2.1 Video Feature Extraction

2.1.1 Color and Texture Feature Model

Since a video clip consists of a group of images, to reduce video data and remove redundancy, the frames are sampled every half second. For each sample frame RGB color histogram and texture feature are calculated. R, G, B channels are each divided into 8 bins, thus color histogram is a 512 dimensional feature vector. Texture feature extraction adopts the statistical texture analysis method based on concurrence gray matrix ([19]). In this method four gray level concurrence matrices are first computed, which corresponds to four neighborhood directions, namely horizontal, vertical, left-down diagonal 45° and right-down diagonal 45°. Totally 13 texture components are computed from each gray level concurrence matrix, including Angular Second Moment, Contrast, Variance, Relevance Coefficient, Entropy etc. Complete computation of the 13 texture components please refer to ([19]).

Texture components computed from the four gray level concurrence matrices are averaged to form one mean texture feature vector. Since texture components have different physical meanings and value ranges, each component is normalized by Gaussian normalization approach to make them equally contribute to feature distance computation. Based on the normalized color and texture features extracted from sample frames, unsupervised clustering approach (e.g. K-Means clustering) is employed to get typical feature model of the video clip. Color feature vector and texture feature vector are clustered separately, and feature distance measure adopts Euclidean distance. The number of clusters for color feature and texture feature are set as the same value, so the video clip's feature model F is as follows,

$$F = [F_c^1, ..., F_c^{K_1}; F_T^1, ..., F_T^{K_2}] \tag{1}$$

Where F_c^i represents the ith color cluster center, F_T^i represents the ith texture cluster center, K_1 and K_2 are the number of clusters. Advantage of this representation is that a video clip can be represented by a fixed dimensional feature vectors, and it is robust to feature distortion of individual frames, as well as frame dropping.

2.1.2 Subspace Discriminative Analysis by OPCA

In above feature representation, dimension of color feature is 512x K_1, and that of texture feature is 13x K_2. If video is matched in this space, computation load will be heavy, and storage need is high, moreover, prototype videos may not be well separated regarding to Euclidean distance. So it is necessary to reduce feature's dimensionality and find its optimal representation in subspace.

In our approach, video clips with different contents means different video classes, and each class is represented by one prototype video, hence. a video database with N classes consists of N prototype feature vectors represented by set $X = \{X_i\}_{i=1,...N}$, $X_i \in R^D$, where X_i is feature vector of the ith proto-video, which is treated as the signal vector, D is the dimension of original feature space. Thereafter, vector is defined as a row vector. The vectors of distorted proto-videos are included in set $\hat{X} = \{\hat{X}_1^1, \hat{X}_1^2, \cdots, \hat{X}_N^m\}$, $\hat{X}_i^k \in R^D$, where $\hat{X}_i^k$ represents the kth distorted vector of the ith proto-video. Difference vector between vectors X_i and $\hat{X}_i^k$ is $Z_i^k = X_i - \hat{X}_i^k$, which is treated as the noise vector. The set of difference vectors is denoted by $Z = \{Z_1^1, Z_1^2, \cdots, Z_N^m\}$.

Given original prototype feature set X and difference vector set Z, Oriented PCA is adopted to compute feature's optimal subspace projection with the aim to maximize signal-to-noise ratio in this subspace ([17] [20]).

Let one of the unit projection vector be denoted by $\vec{n}$, OPCA is to maximize the following generalized Rayleigh quotient:

$$q = \frac{\vec{n} C_X \vec{n}^T}{\vec{n} R_Z \vec{n}^T} \tag{2}$$

$$\text{where } C_X = E(X_i - \bar{X})^T (X_i - \bar{X})$$

$$R_Z = E(Z_i^T Z_i)$$

Where C_X is covariance matrix of feature vectors in *X*, while R_Z is correlation matrix of difference vectors in *Z*. The nominator of (2) is the variance of prototype vectors' projected values on direction $\vec{n}$, while denominator is correlation of difference vectors' projected values on the projection axis. Therefore maximizing *q* will make proto-vectors separate while making difference vectors shrink as much as possible on this projection direction. Here correlation matrix of difference vectors is computed instead of covariance matrix, because their mean value not just variance should be compressed on the projections. To compute projection directions, let $\nabla q = 0$, then the solution of $\vec{n}$ becomes solving the following generalized eigenvector problem,

$$C_X \cdot \vec{n}^T = q \cdot R_Z \cdot \vec{n}^T \tag{3}$$

If the dimension of subspace is set to D_1, then unit vectors corresponding to the D_1 largest generalized eigenvalues are used as OPCA projection directions. Solving (3) can first take Cholesky decomposition of R_Z and transform it to the normal eigenvector problem. Different from PCA, OPCA projection vectors are not necessarily orthogonal to each other, and not necessarily unit ones.

2.2 Statistical Analysis of Video Database

A video model database generally consists of a lot of prototype feature vectors, so NN classifier will be an efficient way to recognize video copies. A test video is recognized as the closest proto-video if the distance is below a threshold θ, otherwise the test video will not belong to any proto-video.

In order to estimate error rate of threshold NN classifier and obtain optimal classification threshold θ_{opt}, it is necessary to know about statistics of video model database. Since classification is based on feature discrimination, we define three types of feature distances to explore video database statistics. Distances are defined as follows:

1) The first type of distance is within-class distance d_w between distorted proto-videos and model database. Let the set of prototype vectors be denoted by $O = \{O_i\}_{i=1,\ldots,N}$, and its distorted vector set be $\hat{O} = \{\hat{O}_1^1, \hat{O}_1^2, \cdots, \hat{O}_N^m\}$, where $\hat{O}_i^k$ represents a distorted vector of the *i*th proto-video, so the within-class distance between $\hat{O}_i^k$ and *O* is defined as,

$$d_w(\hat{O}_i^k, O) = d(\hat{O}_i^k, O_i) \tag{4}$$

Where $d(\hat{O}_i^k, O_i)$ is the feature distance function.

2) The second type of distance is the minimum between-class distance between distorted proto-video and the database, denoted by d_{bi},

$$d_{bi}(\hat{O}_i^k, O) = \min_{j \neq i} d(\hat{O}_i^k, O_j) \tag{5}$$

3) The third type of distance is the minimum distance between non-prototype video and the database, denoted by d_{bo}, where non-prototype video means the video that does not belong to any class in database.

$$\text{If } Q \notin O \cup \hat{O},\ d_{bo}(Q, O) = \min_i d(Q, O_i) \tag{6}$$

Illustration of d_w, d_{bi} and d_{bo} in feature space is shown as Fig.1.

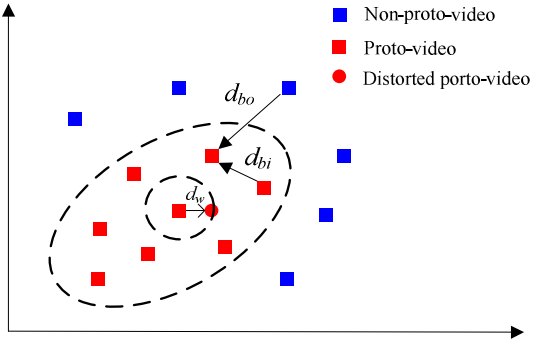


Fig. 1 Illustration of d_w, d_{bi} and d_{bo}

Distributions of d_w and d_{bi} are built-in statistics of model database, which reflect the variation between proto-videos and their distorted copies, and the discrimination between proto-videos. Distribution of d_{bo} is not only dependent on model database, but also related to distribution of non-prototype videos.

If we get the distributions of the above three distances, error rate of threshold NN classifier can be exactly computed. When the number of video models is greater than 2, recognition error comes from the following three sources: ① distorted proto-video is classified as non-prototype video; ② distorted copy of one proto-video is recognized as another proto-video; ③ non-prototype video is recognized as a proto-video.

If proto-video vector set is denoted as O, and video class label set be denoted by $\tilde{O}$, prototype vectors and their distorted vectors are included in set $\bar{O} = O \cup \hat{O}$, q is test video, $r(q)$ is the class label recognized, while $\bar{r}(q)$ is its

true class label, then the probability that q be wrongly classified is computed as follows:

$$\begin{aligned} P_e &= P(q \in \bar{O}, r(q) \notin \tilde{O}) \\ &\quad + P(q \in \bar{O}, r(q) \in \tilde{O}, r(q) \neq \bar{r}(q)) + P(q \notin \bar{O}, \bar{r}(q) \in \tilde{O}) \end{aligned} \tag{7}$$

If a unified threshold θ is adopted, (7) will become as,

$$\begin{aligned} P_e &= P(q \in \bar{O}) \cdot P(d_w > \theta, d_{bi} > \theta \mid q \in \bar{O}) \\ &\quad + P(q \in \bar{O}) \cdot P(d_{bi} \le \theta, d_w > d_{bi} \mid q \in \bar{O}) \\ &\quad + P(q \notin \bar{O}) \cdot P(d_{bo} \le \theta \mid q \notin \bar{O}) \end{aligned} \tag{8}$$

Suppose normalized distance is continuous in [0,1], and the density functions of d_w, d_{bi} and d_{bo} are $p_1(x)$, $p_2(x)$ and $p_3(x)$ respectively. It is also assumed that random variables d_w, d_{bi} are independent to each other, which is reasonable because for one feature point d_w, d_{bi} are computed with reference to two non-overlapped subsets of proto-vectors. Given assumption above, if the prior probability of proto-videos and their distorted copies is $P(q \in \bar{O}) = \eta$, and the prior for non-prototype videos is $P(q \notin \bar{O}) = 1 - \eta$, then (8) will become as,

$$P_e(\theta) = \eta \cdot (\int_{\theta}^{1} p_1(x) \cdot dx \int_{\theta}^{1} p_2(y) \cdot dy + \int_{0}^{\theta} p_2(x) \cdot dx \int_{x}^{1} p_1(y) \cdot dy) + (1-\eta) \cdot \int_{0}^{\theta} p_3(x) \cdot dx$$

Since it is a function of threshold θ, its minimum value can be obtained by setting $P_e'(\theta) = 0$, which is,

$$P_e'(\theta) = -\eta \cdot p_1(\theta) \int_{\theta}^{1} p_2(x) dx + (1-\eta) p_3(\theta) = 0$$

If $\eta = 0.5$,

$$p_1(\theta) \int_{\theta}^{1} p_2(x) dx = p_3(\theta) \tag{9}$$

Since $\int_{\theta}^{1} p_2(x) dx \le 1$, so $p_1(\theta) \ge p_3(\theta)$, the optimal threshold lies on the left of the intersection point of $p_1(x)$ and $p_3(x)$.

2.3 Results

In experiment we built a prototype video database which consists of 1000 short video clips with length from 15 to 90s, most of which are commercials and film trailers. Video format is: frame size 720x576, 25fps. Distorted copies of these

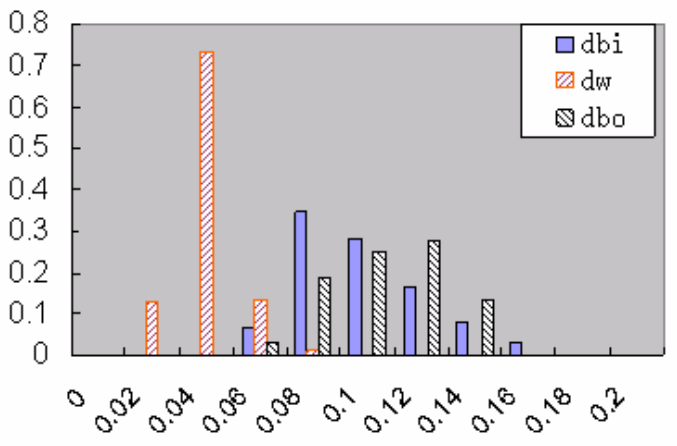


Fig. 2 Histograms of d_w, d_{bi} and d_{bo} (350 models)

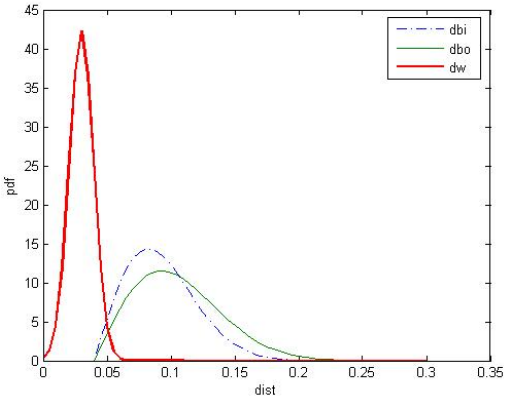


Fig. 3 Density functions of d_w, d_{bi} and d_{bo} (350 models)

proto-videos are produced by transcoding operations combining frame downsizing from 720x576 to 352x288 with frame rate reduction from 25fps to 15fps, which is the common distortion lying between broadcast video and web video copies. Video length is set to 10s when computing the feature vectors. The number of texture feature clusters is 5 while that of color feature is 1. Then OPCA is adopted to compute the 64 subspace projections.

This method is evaluated under two database sizes, one has all the 1000 proto-videos, and the other has 350 randomly chosen proto-videos. Density functions of d_w and d_{bi} are estimated from proto-vectors and their distorted vectors in subspace, and another 670 non-prototype videos are used to compute distribution of d_{bo}. Histograms of the three distances for 350 models are shown as Fig.2, and

their density functions are shown as Fig.3, where d_w is approximated by normal distribution, while d_{bi}, d_{bo} are approximated by Rayleigh distribution.

From Fig.2, 3 we can see that distributions of d_w and d_{bi} or d_{bo} are well separated in subspace. Optimal threshold θ_{opt} is chosen as the intersection point of d_w and d_{bi} which is about 0.05, and corresponding training error rate is a quite low value 1.63%, as shown in Table 1.

Table 1 Minimum training error rates

Error rate / Method	*L*=5s	*L*=10s	*L*=15s
Subspace feature (350 models)	2.9 %	1.63 %	1.57 %
Original feature (350 models)	23.7 %	20.7 %	5.8 %
Subspace feature (1000 models)	7.38 %	4.27 %	5.37 %
Original feature (1000 models)	26.25 %	21 %	8 %

The projection matrix computed from 10s clips is applied on 5s and 15s clips to calculate subspace features, and quite low error rates are also achieved. When the number of models increase from 350 to 1000, error rates increase correspondingly, but are still very low, the error rate for 5s clips is below 8%.

By comparison, error rate is also tested using original video feature. As is shown in Table I, longer clips show better robustness to significant distortion by frame dropping and downsizing, since more feature points join clustering process, so the effect of frame distortion and dropping can be better counteracted. However, by subspace feature transformation shorter clips (e.g. below 10s) can result in the same performance with that of longer clips (e.g. 15s).

For testing, those 1000 prototype videos are transcoded by frame downsizing to CIF or frame rate reduction to 15fps alone, and these distorted videos plus other 1000 non-prototype videos are used to test the trained subspace classifier under 1000 prototypes with length set to 10s. False negative error rate is zero, and total error rate is 2.8%. This result shows that since composite distortions by downsizing and frame dropping are maximally compressed in subspace projections, slighter distortion by downsizing or frame dropping alone can also be maximally compressed.

We also tested that when PCA is applied to 10s videos using 64 eigenvector projections corresponding to the largest eigenvalues, minimum error rate is 19.8% in case of 350 models, nearly the same performance with original feature. This result explains that PCA is built for reconstruction and compression, while OPCA is good for classification.

3 Unknown Video Repeat Mining

In this section we propose a novel approach for unknown repeat repeats mining. Two detectors in a cascade structure are employed to achieve fast and accurate detection, and a reinforcement learning approach is adopted to efficiently maximize detection accuracy. In this approach very short video repeats (< 1s) and long ones can be detected by a single process, while overall accuracy remains high. Since video segmentation is essential for repeat detection, performance analysis is also conducted for several segmentation methods. Results of video structure analysis by video repeat mining are also presented.

3.1 Framework

The proposed framework is shown in Fig. 4. We employ two cascade detectors to identify repeated clips, with the first detector discovering potential repeated clips, and the second one improving accuracy.

The first detector includes three temporal level video representations, namely video units (VU), video segments (VS) and video clips (VC), as well as corresponding video similarity measures. The first step is content based video segmentation. Video stream is partitioned into basic video units (VU). The second step is self-similarity analysis. Video units are grouped by a window size W, e.g. two units as one group, to form bigger size video segments (VS), then they are compared with each other to produce similarity matrix S. By similarity measure f_1, two segments will be judged as either identical or non-identical, so S is a binary matrix

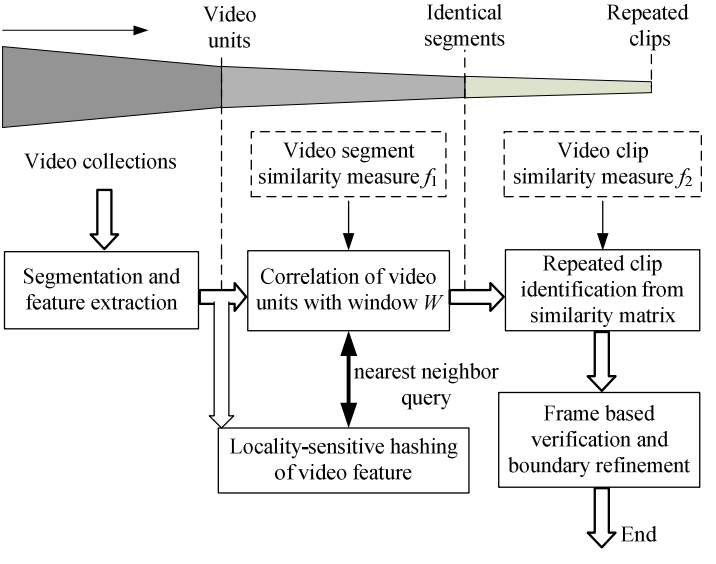


Fig. 4 Framework for repeat video clip identification

which is generally a sparse one that can be compactly represented to save storage. Here *locality sensitive hashing* (*LSH*) is adopted to reduce correlation complexity. The third step is to identify repeat clips from similarity matrix S. Basically repeated clips can be identified from diagonals, which is controlled by similarity measure f_2.

The second detector adopts frame based matching to verify candidate repeated clips for accuracy improvement. After that a boundary refinement step is employed to extend repeated clips' boundaries close to their maximum ones as possible. The last step is repeated clip labeling. Repeated instances will be extracted from repeated clip pairs and grouped into multiple categories. Each category represents a unique repeat pattern.

3.2 Video Representation and Feature Extraction

3.2.1 Video Segmentation and Three Level Representation

In our method video stream is segmented by content based keyframes, and interval between two consecutive keyframes is treated as the basic video unit (VU). Keyframe selection is based on color histogram difference. Suppose H_1 and H_o are color histograms of current frame and the last keyframe respectively, then current frame is selected as new keyframe if the following condition is satisfied,

$$\left|1-\text{inter}(H_1, H_0)\right| > \eta \tag{10}$$

where $\text{inter}(H_1, H_0)$ is intersection of two color histograms, η is threshold.

This representation is a seamless video segmentation without temporal data loss, which is similar to shot segmentation, but its granularity is smaller than shot. Its advantages lie in: First it is robust to boundary shift of repeat clips. Generally shift error can be corrected after a shot cut. Secondly it can reduce correlation between adjacent video units, so diagonal pattern will be sharper and easier be identified. The third advantage is that temporal length of video unit can be added to increase feature discrimination.

The second level video representation (VS) is formed by grouping two neighbor units (W=2). Compared to the first level, the second level has almost the same number of samples, but the discrimination ability will improve a lot, thus providing a less noisy output to build a higher level of video repeat clips.

3.2.2 Video Features

Two types of video features are extracted. The first one is video unit (VU) feature used in the first detector, and the other one is frame feature used in the second detector.

1) *Video unit feature*

Video unit feature includes interval length and color fingerprint proposed by Yang et. al ([12]). A video unit is partitioned into K sub-intervals, and represented by K blending images formed by averaging frames within each sub-interval along

time direction. Each blending image is then divided into $M \times N$ equal size blocks each of which is represented by the major and minor color components among RGB, as illustrated in Fig. 5. Color fingerprint is the ordered catenation of these block features. If $\overline{R}$, $\overline{G}$, $\overline{B}$ are the average color values of a block, and their descending order is (V_1, V_2, V_3), then the major color and minor color are determined by the following rules:

Rule 1: if $V_1 > V_3$,

$$\text{Major Color} = \begin{cases} \arg\max(\overline{R}, \overline{G}, \overline{B}) & \text{if } (V_1 - V_3) > \tau \\ \text{Uncertain} & \text{if } (V_1 - V_3) \le \tau \end{cases}$$

$$\text{Minor Color} = \begin{cases} \arg\min(\overline{R}, \overline{G}, \overline{B}) & \text{if } (V_2 - V_3) > \tau \\ \text{Uncertain} & \text{if } (V_2 - V_3) \le \tau \end{cases}$$

Where τ is the parameter that controls the robustness to color distortion and discriminative ability of this feature.

Rule 2: if $V_1 = V_3$ (gray image),

$$\text{Major Color} = \text{Minor Color} = \begin{cases} \text{bright} & \text{if } V_1 > \tau_1 \\ \text{dark} & \text{if } V_1 \le \tau_1 \end{cases}$$

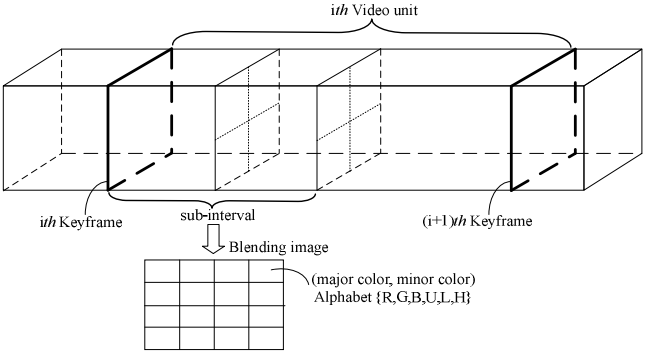


Fig. 5 Illustration of video segmentation and feature extraction

Major and minor color patterns have six possible symbol values from alphabet {R, G, B, U, L, H}, where U, L and H stand for uncertain, dark and bright respectively. In this work one blending image (K=1) is used for each unit, and divided into 8x8 blocks (M=N=8), thus the color feature is a 128 dimensional symbol vector. We also apply LSH indexing on this color fingerprint, and its string representation can be easily transformed to a bit string required by LSH algorithm ([21]) without incurring extra errors. By LSH and unit length filtering, complexity of searching identical video units can be reduced by hundreds of times.

2) *Frame feature*

Each frame is divided into 4 sub-frames, and RGB color histogram(8x8x8 bins) of each sub-frame is quantized to a symbol by VQ, so each frame is represented by 4 symbols.

3.3 Video Similarity Measures

Video similarity measures are conducted at several levels to ensure efficient and robust video repeat discovering: Video Unit, Video Segment, and Video Clip.

3.3.1 Video Segment Similarity Measure

Given two video units vu_i and vu_j, their distance $D(vu_i, vu_j)$ is defined as:

$$D(vu_i, vu_j) = \sqrt{d^2(F_i, F_j) + \left[len(vu_i) - len(vu_j)\right]^2} \tag{11}$$

where F_i, F_j are color fingerprint vectors of vu_i and vu_j, $d(F_i, F_j)$ is color fingerprint distance function ([12]), len(·) is length feature. If VS consists of *W* video units, similarity measure f_1 between the *i*th segment and *j*th segment $VS_j : \{vu_j, vu_{j+1}\}$ is defined as:

$$f_1(VS_i, VS_j) = \begin{cases} 1 & if\ D(vu_i, vu_j) < \varepsilon_1, \cdots, D(vu_{i+W-1}, vu_{j+W-1}) < \varepsilon_1 \\ 0 & otherwise \end{cases} \tag{12}$$

where ε_1 is distance threshold.

3.3.2 Clip Level Aggregation

Repeat clips will appear as diagonals in similarity matrix. However, due to segmentation errors, the line will not be the integrated one. Moreover those line

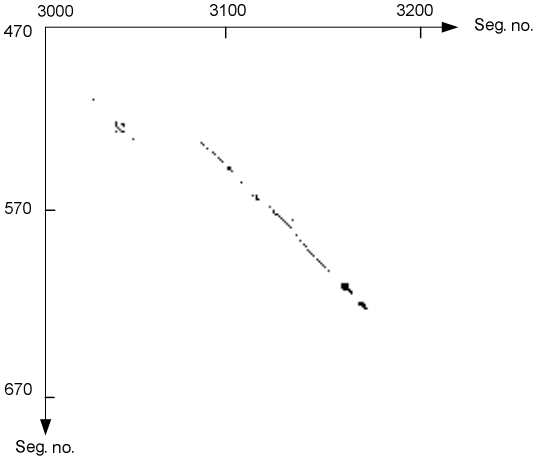


Fig. 6 Example of diagonal tracks for repeat sequences

fragments will not be collinear if non-uniform partition is used. Fig. 6 shows part of a similarity matrix computed in our experiment. As we can see, diagonal tracks are fragmented and contaminated by noises. To get the whole repeat clip correctly we design a hierarchical aggregation algorithm purely based on temporal boundaries of repeat segments.

This algorithm is described as follows:

Step 1: First link strong diagonal tracks whose length exceeds one. The start and end time of two pairs of repeat sequences (I,I') and (II,II') corresponding to two diagonal lines are represented by ($T1_{start}$,$T1_{end}$), ($T1'_{start}$,$T1'_{end}$) and ($T2_{start}$,$T2_{end}$), ($T2'_{start}$,$T2'_{end}$) respectively, which is illustrated in Fig. 7.

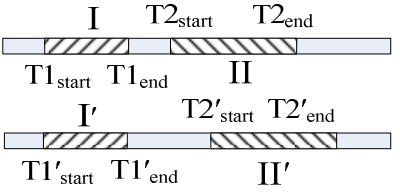


Fig. 7 Illustration of two pairs of adjacent repeat segments

If one of the two conditions in (13) is satisfied, (I,I') and (II,II') will be merged into one repeat pair.

a. Overlap: $T1_{start} \le T2_{start} \le T1_{end}$, $T1'_{start} \le T2'_{start} \le T1'_{end}$

b. Adjacency: $|T2_{start} - T1_{end}| < \mu_1$, $|T2'_{start} - T1'_{end}| < \mu_1$, $|(T2_{start} - T1_{end}) - (T2'_{start} - T1'_{end})| < \varepsilon_2$ (13)

where μ_1 defines neighborhood distance, ε_2 is displacement allowed for neighbor repeat segments, thus controls temporal variations of the whole repeat clip.

Boundaries of merged repeat pair are computed as:

$T_{start}=\min(T1_{start},T2_{start})$, $T_{end}=\max(T1_{end},T2_{end})$;

$T'_{start}=\min(T1'_{start},T2'_{start})$, $T'_{end}=\max(T1'_{end}, T2'_{end})$.

This new repeat pair will be put into the repeats list to replace originals, and the above process is iterated till no change of the list.

Step 2: Connecting single dots based on results of step 1 with the same merging criterion as step 1.

Step 3: The connected sequences after above two steps are further connected and merged until there is no change.

By the above aggregation algorithm the whole image of repeat clips can be well constructed from their local repeat segments, thus providing good foundation for further similarity analysis and boundary refinement. Moreover, this algorithm only needs to store boundaries of repeat segments but not similarity matrix, which can have efficient implementation for even large video data mining.

3.3.3 Second Stage Matching

The second detector adopts frame by frame matching. The total number of identical frames is normalized by the average sequence length to get the similarity score. A repeat pair is judged as true one if the following condition is satisfied,

$$score > (1 + e^{-L})\varepsilon_3 \tag{14}$$

where *score* is the similarity value, *L* is the minimum length of the two clips in seconds, and ε_3 is threshold. This decision rule uses soft thresholds for different length sequences. Since shorter sequences are assumed less reliable ones, they should satisfy more stringent condition to pass through verification. Once a repeat pair is verified, their boundaries are extended frame by frame until dissimilar frames are encountered.

3.4 Reinforcement Learning of Detectors

The two cascade detectors contain several parameters, like distance thresholds, LSH parameters etc., but the intrinsic and crucial ones that affect detection accuracy are ε_1, ε_2 and ε_3 in (12)(13)(14) respectively. Tuning these three parameters can significantly change detection results. The three parameters have clear physical meanings. ε_1 reflects feature distortion of identical video units for certain video data and feature extraction; ε_2 that defines maximum temporal displacement between neighbor repeat segment pairs in clip aggregation function is related to video unit granularity and temporal variation allowed for the whole repeat clips. ε_3 in the second detector balances recall and precision. Parameter μ_1 in (13) defining neighborhood of repeat segments is not crucial for final results as long as it is

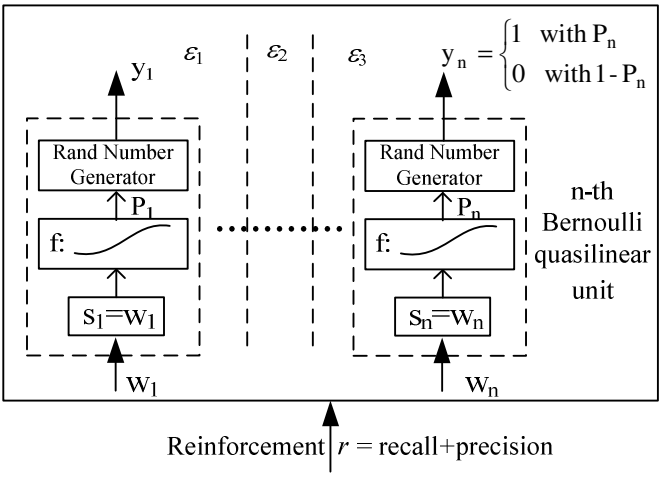


Fig. 8 Connectionist reinforcement learning network

in a range, e.g. 10s ~ 20s. Segmentation related parameter is important for final results, but it is not intrinsic to the detector. Different segmentation methods may have different types of parameters or no parameter at all.

In the following we will propose a method to learn appropriate values of ε_1, ε_2 and ε_3 in order to achieve optimal performance on selected video data. Given certain segmentation and feature extraction, the three parameters in the two detectors are trained together by reinforcement learning in a non-associative paradigm. Given an input, the learning network produces the three parameters, then a scalar indicating "goodness" of detection results under these parameters is immediately used as a reinforcement for the learning network. In our approach the sum of recall and precision is taken as the reinforcement factor. We also adopt the connectionist REINFORCE algorithm ([22]) in which the units of network are Bernoulli quasilinear units whose output is 0 or 1, statistically determined by Bernoulli distribution with parameter $p = f(s) = 1/(1+\exp(-s))$, which is shown in Fig.8. Each Bernoulli quasilinear unit has one input weight, and the three parameters are encoded by gray codes corresponding to the outputs of n Bernoulli quasilinear units. After receiving a reinforcement r, the weights of Bernoulli quasilinear units are updated by (15).

$$\Delta w_i = \alpha(r-b)(y_i - p_i) \tag{15}$$

where α is a positive learning rate, b serves as a reinforcement baseline, y_i is the output of the ith Bernoulli quasilinear unit, and p_i is the Bernoulli distribution parameter. It has been shown by Williams ([22]) that this learning algorithm statistically climbs the gradient of expected reinforcement in weight space, which means that the detector parameters will change in the direction along which the sum of recall and precision increases.

3.5 Results

For news video we chose half-hour CNN and ABC news videos from TRECVID data to form two video collections, each of which contains 12 day programs with 6 hours around. By manually searching short repeat clips including program logos and commercials, but neglecting other repeat scenes, i.e. anchor persons, 34 kinds of repeat clips with totally 186 instances are found from CNN collection, while 35 kinds with totally 116 instances found from ABC collection. In addition broadcast videos of Channel News Asia (CNA) are also used for structure analysis.

3.5.1 Detector Training

Parameters of the two detectors are learned by the approach presented in section 3.4. Three hour CNN news videos are randomly chosen for training. Videos are segmented by content based keyframes.

The reinforcement learning rate α in (15) is set to 0.01 and reinforcement baseline b set to 0.7. Parameters ε_1, ε_2 and ε_3 are each encoded by 5 bit gray code, so

there are totally 15 Bernoulli units in this network. Parameter value range is set to [0,1]. Initial parameters are set to empirical values, and initial weights are all zeros. During each learning round we manually check the detection results to compute recall and precision, then feed their sum as reinforcement of the learning network. Recall and precision are calculated as (16).

$$\text{recall} = \frac{\text{number of correct repeat instances}}{\text{number of all true repeat instances}}$$
$$\text{precision} = \frac{\text{number of correct repeat instances}}{\text{number of all detected instances}} \tag{16}$$

In experiment recall and precision in the first round learning are 74% and 100%, but after ten rounds of learning, recall and precision already climb to 94.2% and 96% respectively. Since the next several rounds of learning do not lead to reinforcement increase, we then stop the learning.

3.5.2 Testing Accuracy

The trained detectors are tested on the rest 3 hour CNN videos and 6 hour ABC videos. Recall and precision on CNN videos are 92.3% and 96%, while 90.1% and 90% those for ABC videos. This accuracy is obtained without setting a minimum sequence length to filter errors, so most of the errors come from those very short clips. The shortest correct repeat detected is just 0.26s (partial of "play of the day" logo in CNN video), while the longest one is 75 seconds long.

Boundary accuracy of repeat pairs is also measured. We selected 300 repeated pairs that cover almost all repeat patterns and checked their boundary shift before boundary refinement. The smallest shift is 0 s, while the largest one is 16.4s, and the average shift is 0.47s. Around 80% of the shifts are within 0.2 seconds. After frame by frame boundary refinement those large shifts can be effectively reduced to 0~1 second.

3.5.3 Performance Analysis of Segmentation Methods

Video segmentation is essential for this approach, so experiments are conducted to compare performances by proposed keyframe based segmentation, uniform segmentation and shot segmentation. The video data are 3 hour CNN videos used in Section 3.5.1. Two keyframe based segmentations are implemented with η =0.15, 0.30 respectively. Uniform segmentation utilizes I frames (every 12 frames). Shot detection includes cuts, fades in-outs and dissolves. Video unit features for all segmentations are color fingerprint and length. The video segment (VS) size W for shot segmentation is set to 1, and the minimum number of diagonal points in repeat aggregation is also set to 1. Thus this method can detect not only single repeat shots, but also repeat clips beyond shots. Detectors are separately trained for each segmentation strategy to achieve their nearly optimal performance, and training results are shown in Table 2.

Table 2 Performance comparison of video segmentation methods

	Uniform sampling	Keyframe (η =0.15)	Keyframe (η =0.30)	Shot based
recall	87.8%	94.2%	90.7%	66.7%
precision	95.9%	96.0%	86.0%	84.7%
Video units	26344	14872	6316	1911

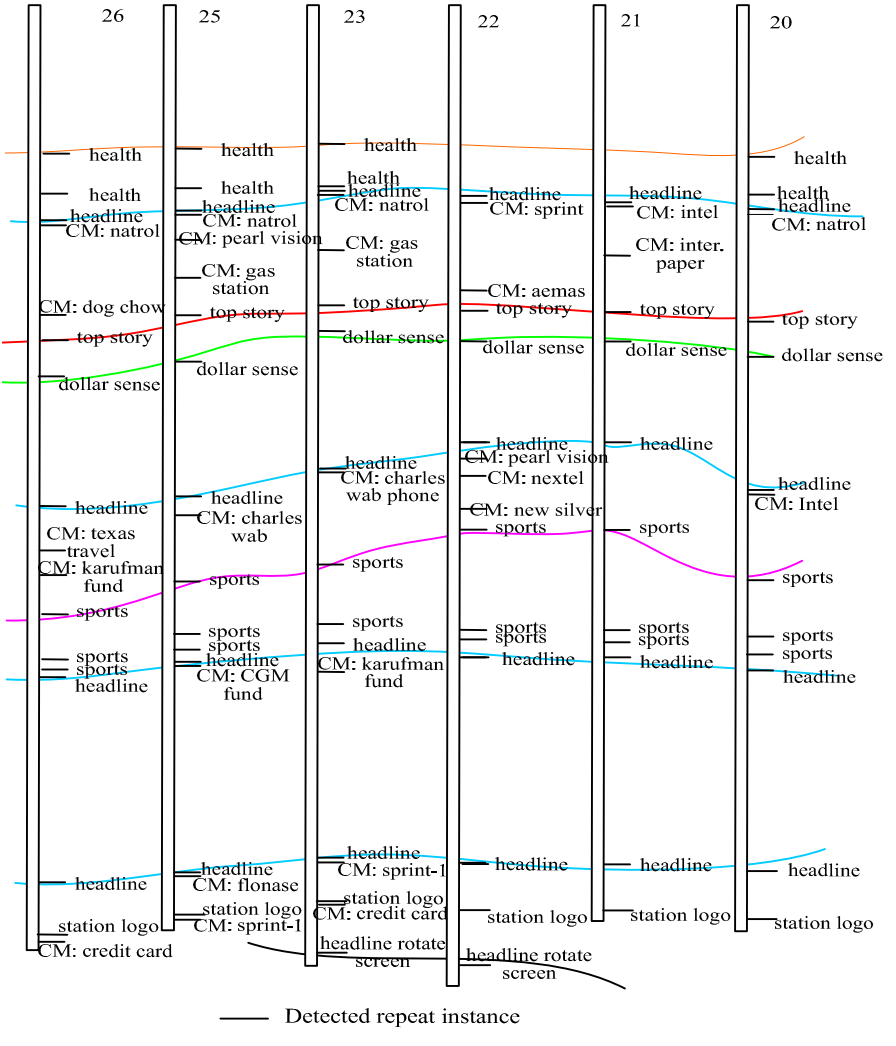


Fig. 9 CNN news video structure analysis by video repeats

From Table 2 we know that keyframe based segmentation achieves best performance. The uniform segmentation results in several times more video units than keyframes, but still gets lower recall on program logos and commercials. Uniform segmentation also detects quite many stationary scenes, such as anchor shots and black frames which occupy nearly 74% of the whole detected repeat clips pool thus overwhelm other interesting repeat patterns like program logos and commercials. Under keyframe based segmentation these still scenes are all filtered, program logos and commercials are main body of detected repeats. Shot based segmentation results in much fewer video units, but its total accuracy is much lower and many fast changing program logos are missed. When granularity of keyframe based segmentation becomes bigger, its performance will also drop because of heavier data loss.

3.5.4 Searching Efficiency Evaluation

By LSH indexing on color fingerprint, the average number of retrieved units for a query unit of CNN collection (totally 629,380frames and 31496 units) is 320, and the number of color feature comparisons is further reduced to 20 by pre-filtering one dimension length feature at trained distance threshold ε_1 =0.1, thus speedup factor is about 1575 compared to pair-wise searching. For ABC collections (totally 616,780 frames and 29838 units), the average number of retrieved units for a query unit is 1026, and further reduced to 56 by length filtering, thus speedup factor is 533. On PC with Pentium-4 2.5GHz processor the two stage detections on 6 hour CNN videos can be finished in 22 seconds, while 40 seconds for ABC videos.

3.5.5 Video Structure Discovery Results

Fig. 9 shows temporal distribution of short video repeats identified from CNN news videos of six days. Those repeat instances linked by curves are chosen as structural video elements (SVE). From this map we can clearly see that the whole program is segmented by SVEs into several layers each of which contains certain

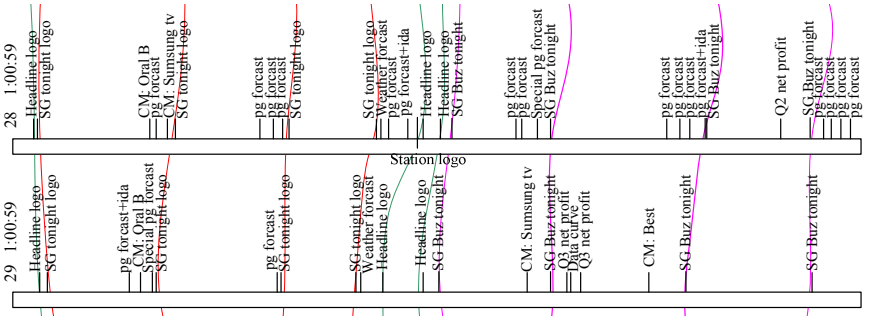


Fig. 10 CNA news video structure analysis by video repeats

topics, such as health program, top stories, financial news, sports news, commercials *et al*. Similar results are also achieved for CNA video structure discovery. Fig. 10 shows distribution of short video repeats identified from CNA one hour news videos in two days.

4 Concluding Remarks

In this chapter we frame known video repeat recognition as a standard pattern recognition problem, and take advantage of the techniques successfully applied to other classical pattern recognition problems such as face recognition, speech recognition and OCR. For example, subspace discriminative analysis is used to optimize video feature representation. Video feature model adopts sampling and clustering strategy to capture typical color and texture features of a video clip, and it shows good robustness to frame distortion and dropping. Video feature's subspace representation computed by OPCA leads to a significant improvement of recognition performance especially for very short video clips (e.g. below 10s). Compared with other robust image descriptors that require high computational complexity, e.g. SIFT ([10]), RGB color histogram and texture feature adopted in this approach can also achieve robustness to common distortions through appropriate coordinate transformation, while computation is much simpler. The proposed statistical analysis method reflects the distribution of video database in feature space by three distance distributions from which nearest neighbor classifier's error probability can be exactly estimated, and optimal classification threshold can be theoretically computed. Classification accuracy is evaluated under 1000 video models, which is a reasonable database size for some real applications, e.g. TV commercial monitoring, and very low error rate is obtained.

In unknown video repeat mining approach, we do not make feature optimization according to specific video database, but rely on self-similarity rule to discover all possible video repeats in an unsupervised way. However, in order to achieve high mining accuracy for given video collections, we adopt a supervised approach to tune the mining parameters, so this approach can be regarded as a mixture of unsupervised discovery and supervised learning. This method achieves robust detection of arbitrary length video repeats by cascaded detectors that employ different features and similarity measures. Quite short repeats (e.g. those less than 1 second) along with long ones can be detected with high accuracy, which is the strength of our approach compared to previous work ([3][9]). Similarity searching complexity of the first detector is reduced hundreds of times through LSH indexing and length filtering. Here color fingerprint is used as video unit feature, for its discrete values are naturally suitable for LSH indexing, and its combination with unit length can give discriminative representation of video units. As a comparison, known video repeat recognition approach adopts color histogram and texture as feature, for they have continuous values that are suitable for statistical analysis and subspace feature transformation. By analyzing detection performance under several video segmentation strategies, we know that video segmentation

utilizing content-based keyframes achieves best balance between detection accuracy and efficiency on short video repeats mining compared to uniform and shot based segmentation. Parameters of detectors can be efficiently optimized in a few rounds of reinforcement learning without knowing statistics of large volume of video data, which makes our approach easily adapt to different video sources. Results also show that short video repeats mining is an effective way to discover syntactic structure of news videos.

References

[1] Yang, X., Xue, P., Tian, Q.: A repeated video clip identification system. In: Proc. ACM Multimedia, Singapore (2005)

[2] Yang, X., Tian, Q., Xue, P.: Efficient Short Video Repeat Identification With Application to News Video Structure Analysis. IEEE Trans. Multimedia 9, 600–609 (2007)

[3] Cheung, S.-C., Ngueyen, T.P.: Mining Arbitrary-length Repeated Patterns in Television Broadcast. In: Proc. IEEE Int. Conf. on Image Processing (2005)

[4] Lienhart, R., Kuhmunch, C., Effelsberg, W.: On the detection and Recognition of Television Commercials. In: Proc. IEEE Int. Conf. Multimedia Computing and Systems (1997)

[5] Snchez, J.M., Binefa, X., Vitri, J.: Shot partitioning based recognition of TV commercials. Multimedia Tools and Applications 18, 233–247 (2002)

[6] Kashino, K., Kurozumi, T., Murase, H.: A quick search method for audio and video signals based on histogram pruning. IEEE Trans. Multimedia 5(3), 348–357 (2003)

[7] Yuan, J., Duan, L.-Y., Tian, Q., Xu, C.: Fast and robust short video clip search using an index structure. In: Proc. ACM Multimedia's Multimedia Information Retrieval Workshop (2004)

[8] Cheung, S.-C., Zakhor, A.: Estimation of web video multiplicity. In: Proc. SPIE, vol. 3964, pp. 34–46 (2000)

[9] Pua, K.M., Gauch, J.M.: Real time repeated video sequence identification. Computer Vision and Image Understanding 93(3), 310–327 (2004)

[10] Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–120 (2004)

[11] Oostveen, J.C., Kalker, A.A.C., Haitsma, J.A.: Visual hashing of digital Video: applications and techniques. In: SPIE applications of digital image processing XXIV, San Diego, pp. 121–131 (2001)

[12] Yang, X., Tian, Q., Chang, E.C.: A Color Fingerprint of Video Shot for Content Identification. In: Proc. ACM Multimedia, NY, USA (2004)

[13] Cevikalp, H., Neamtu, M., Wilkes, M., Barkana, A.: Discriminant Common Vectors for Face Recognition. IEEE Trans. Pattern Analysis and Machine Intelligence 27(1), 4–13 (2005)

[14] Chakrabarti, S., Roy, S., Soundalgekar, M.: Fast and Accurate Text Classification via Multiple Linear Discriminant Projections. In: Proc. Int'l. Conf. Very Large Data Bases, pp. 658–669 (2002)

[15] Belhumeur, P., Hespanha, J., Kriegman, D.: Eigenfaces versus Fisherfaces: Recognition Using Class Specific Linear Projection. IEEE Trans. Pattern Analysis and Machine Intelligence 19(7), 711–720 (1997)

[16] Yang, X., Yuan, M.: Video Copy Recognition By Oriented PCA and Statistical Analysis. In: Proc. IEEE Int. Conf. on Image Processing, Cario, Egypt (2009)
[17] Burges, C.J.C., Platt, J.C., Jana, S.: Distortion Discriminant Analysis for Audio Fingerprinting. IEEE Trans. Speech and Audio Processing 11(3), 165–174 (2003)
[18] Herley, C.: ARGOS: Automatically Extracting Repeating Objects From Multimedia Streams. IEEE Trans. Multimedia 8(1), 113–129 (2006)
[19] Haralick, R.M., Shanmugam, K., Dinstein, I.: Textural features for Image Classification. IEEE Transactions on Systems, Man, and Cybernetics 3(6), 610–621 (1973)
[20] Diamantaras, K., Kung, S.: Principal Component Neural Networks. John Wiley, Chichester (1996)
[21] Gionis, A., Indyky, P., Motwaniz, R.: Similarity Search in High Dimensions via Hashing. In: Proc. Int. Conf. Very Large Data Bases, pp. 518–529 (1999)
[22] Williams, R.J.: Simple Statistical Gradient-Following Algorithms for Connectionist Reinforcement Learning. Machine Learning 8, 229–256 (1992)

Mining TV Broadcasts 24/7 for Recurring Video Sequences

Ina Döhring and Rainer Lienhart

Abstract. Monitoring and analyzing TV broadcasts is an important task in the media as well as the advertising business. An important subtask is the frame-accurate detection of recurring video sequences. Examples of recurring video sequences are commercials, channel advertisements, channel intros, and newscast intros. Most of these different kinds of repeating video clips can automatically be classified by further analyzing their temporal and visual properties. In this work we introduce an algorithm and a real-time system for recognizing recurring video sequences frame-accurately in a highly effective and efficient manner. The algorithm does not require any temporal pre-segmentation by shot detection and can thus, in principle, be applied to any kind of temporal signal. It is frame-accurate, meaning that it exactly identifies with which frame a repeating sequence starts and ends. Thus, the temporal accuracy is 40 milliseconds for PAL and 33 milliseconds for NTSC videos. On a standard PC desktop a 24-hour live-stream can be processed in about 4 hours including the computational expensive video decoding. To achieve this efficiency the algorithm exploits an inverted index for identifying similar frames rapidly. Gradient-based image features are mapped to the index by means of a hash function. The search algorithm consists of two steps: firstly searching for recurring short segments of 1 second duration and secondly assembling these small segments into the set of repeated video clips. In our experiments we investigate the sensitivity of the algorithm concerning all system parameters and apply it to the detection of unknown commercials within 24 and 48 hours of various TV channels. It is shown that the method is an excellent technique for searching for unknown commercials. Currently the system is used 24 hours a day, 7 days a week in various countries to log all commercials broadcast without manual intervention.

Ina Döhring
Multimedia Computing Lab, University of Augsburg, Germany
e-mail: doehring@informatik.uni-augsburg.de

Rainer Lienhart
Multimedia Computing Lab, University of Augsburg, Germany
e-mail: lienhart@informatik.uni-augsburg.de

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 327–356.
springerlink.com

1 Introduction

The influence of digital media on our lives is steadily growing. Lots of data, which were formerly presented by audio, photography, or books, is now shared as digital videos. Information from numberless broadcast stations are available 24 hours a day, 7 days a week. These large amounts of data require effective strategies for keeping it accessible. Thus there is an urgent need for effective search and mining algorithms for a variety of applications. Common tasks in video retrieval range from browsing through video collections over copy detection in the World Wide Web to finding special types of sequences in broadcasts such as commercials or news themes.

In this chapter we will focus on the detection of recurring sequences in TV broadcasts such as commercials, music clips, jingles, news stories, and similar content. For performance analysis and evaluation, however, we will only concentrate on the detection of recurring commercials as they have the advantage that we can easily and fast create frame-accurate ground truth annotations with a video wheel; something that is normally not easily possible for other kinds of recurring video sequences.

Regarding commercials as recurring sequences is a highly effective method to search for them. It only needs to rely as little as possible on vague, and/or easily alterable audio-visual characteristics such as increased audio level or cut-frequency. Usually simple duration constraints are enough to distinguish them from other recurring sequences. As shown in Sec. 4 the percentage of repeated commercials within 24 hours in our test videos is about 90%. In other words, if we can detect all recurring commercial clips, we cover 90% of all broadcast advertisements per day. For longer mining periods such as days or weeks, this fraction will increase, because some commercials that are shown only once a day will be repeated within the next days. The detection of unknown commercials can be used, for instance, to create an ad database as the basis for reliable recognition of broadcast advertisements [1].

This chapter is structured as follows: After some comments on the characteristics of the videos we deal with, we will introduce and discuss two types of image features for fingerprinting our sequences in Sec. 2. In Sec. 3 we explain our search algorithm in more detail. Experimental results are discussed in Sec. 4. We investigate the sensitivity of the algorithm concerning all system parameters and apply it to the detection of unknown commercials within 24 and 48 hours of various TV channels. We give survey on related work in Sec. 5 and finally a summary in Sec. 6.

2 Fingerprinting Video Streams

Video streams are temporal sequences of individual images. With proper image features the essence of these video frames can be captured. Hence a video stream can be considered as a temporal sequence of image features. There already exists a very large variety of image features. For instance, they can be derived from the colors and/or the structures in the picture. Additionally, it may be worthwhile to take temporal information into consideration. After explaining in Sec. 2.1 the context in which we operate with our search algorithm, we introduce in Sec. 2.2 two types of

image features – a color-based and a edge-based feature. Both image features are suitable for real-time fingerprinting [10].

2.1 Video Model

There exist different intentions for searching for repetitions in videos. Depending on the actual goal the search methods may vary, for instance, in the requirements on the image features or the quality and precision of the results that have to be achieved.

We will focus on the mining of repeated sequences from a single sensor such as a specific TV capture card for building a comprehensive database from TV live-streams. Here, we have the constraint to operate in real-time, which means that we need image features that are fast to compute and a very fast algorithm for detecting recurring sequences. Storage needs are assumed to be moderate, because the recurring sequences in live TV streams are expected to be of moderate length.

We also want to point out that our algorithm on purpose is designed for a data stream coming from a fixed, but arbitrary sensor such a TV capture card from a specific vendor. As every human observes the world through the same two eyes and has to learn everything it can see for these two eyes, our system observes everything through the same kind of video capture cards for the same kind of signals. Any kind of artifact created by the sensor is consistent throughout time in the input data stream. Thus our work does not address the copy detection problem on the Internet or across different video formats and video fidelities. This is a completely different problem domain. Nevertheless our task is also challenging since the focus lies on temporal precision.

As we are interested in assembling a database of all commercials broadcast during our 24/7 monitoring, we have to detect repeated sequences in real-time not only within the stream but also in the database that is built on-the-fly during mining. In this setting it can be an advantage to choose image features which are less robust, but posses a ability to reliably distinguish between different images, since it may lead to a better overall performance as precision is more important than recall.

2.2 Image Features

In this Section we introduce two image features: the *Color Patches Feature* (CPF) and the *Gradient Histogram* (GH). A color patches feature is derived by computing color average values for small parts of the image. Gradient Histograms are based on the edges in an image. Advantages and disadvantages of both types of image features are discussed in length in [10]. In the following we will evaluate quantitative and qualitative properties of these image features. For visualization we take two sample video frames from two different TV channel recordings (see Fig. 1).

Color Patches Features. We choose the color patches feature (CPF) as our color-related image feature. It has been shown that it is robust and outperforms, for

(a) (b)

Fig. 1 Sample images from (a) a UK broadcast in PAL norm and (b) a US broadcast in NTSC norm.

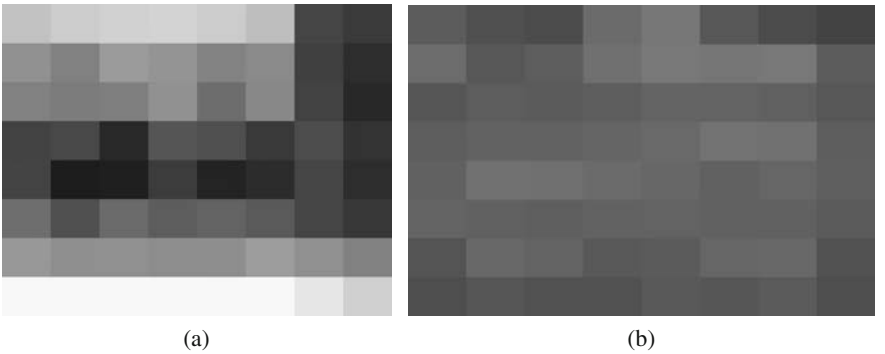
instance, Color Coherence Vectors for the task of image matching [10]. Many variants of CPFs are widely used and amongst others described in [17].

CPFs are derived from mean intensities within small subareas of each color channel. The whole image is divided into $N \times M$ rectangular blocks. For each block the intensity of each color component red, green, and blue is averaged:

$$P_{nm}^{C} = \frac{1}{\sum_{(x_1,x_2)\in I_{nm}} 1} \sum_{(x_1,x_2)\in I_{nm}} C(x_1,x_2) \tag{1}$$

with colors $C \in (R,G,B)$ and subareas I_{nm}, $n = 1,\ldots,N$ and $m = 1,\ldots,M$.

Fig. 2 illustrates the effect of the CPF calculation. Each single-colored rectangle represents three mean RGB intensities in the CPF vector. Thus, the size of the CPF vector is $3 \times N \times M$.



(a) (b)

Fig. 2 The color patches features (CPFs) for the two sample images of Fig. 1: (a) for the UK broadcast image and (b) for the US broadcast image.

Gradient Histograms. An alternative to color-based features are edge-based features. They evaluate color intensity gradients instead of absolute intensities. Edge-based features similar to our gradient histograms are for instance investigated in [17, 19, 26].

In our work we operate on grayscale images only and use intensity differences to approximate the true gradients:

$$\frac{\partial}{\partial x_1} I(x_1,x_2) \sim [I(x_1+1,x_2) - I(x_1-1,x_2)] \tag{2}$$

$$\frac{\partial}{\partial x_2} I(x_1,x_2) \sim [I(x_1,x_2+1) - I(x_1,x_2-1)] \tag{3}$$

We again divide an image into $N \times M$ subareas I_{nm}. For each subarea we generate a gradient direction histogram of K bins by summing sum up in each bin the gradient magnitudes of all gradients, whose directions fall into the interval of the bin:

$$H_{nm}^k = \frac{1}{\sum_{(x_1,x_2)\in I_{nm}} m_g(x_1,x_2)} \sum_{(x_1,x_2)\in I_{nm}} \mathscr{M}_k(x_1,x_2), \tag{4}$$

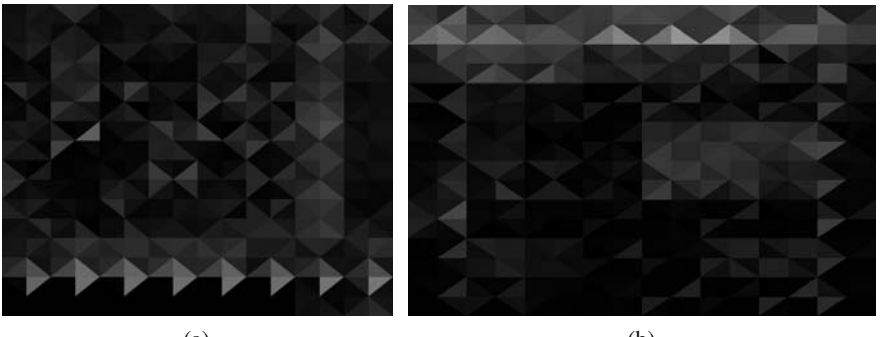
with the gradient magnitude m_g, orientation θ_g, and the auxiliary variables $\mathscr{M}_k$ and θ_k defined by

$$m_g = \sqrt{(I(x_1+1,x_2) - I(x_1-1,x_2))^2 + (I(x_1,x_2+1) - I(x_1,x_2-1))^2}, \tag{5}$$

$$\theta_g = \arctan\left(\frac{I(x_1,x_2+1) - I(x_1,x_2-1)}{I(x_1+1,x_2) - I(x_1-1,x_2)}\right). \tag{6}$$

$$\mathscr{M}_k = \begin{cases} m_g(x_1,x_2) & \text{if } \theta_k \le \theta_g(x_1,x_2) < \theta_{k+1}, \\ 0 & \text{else,} \end{cases} \tag{7}$$

$$\theta_k = (k-1)\,360^\circ/K \tag{8}$$



(a) (b)

Fig. 3 The gradient histograms (GHs) of the two sample images in Fig. 1: (a) for the UK broadcast image and (b) for the US broadcast image.

Figure 3 illustrates GHs. Each sample image has the same partitioning as for the CPFs in Fig. 2. A rectangular subarea is divided into $K = 8$ directional bins to visualize the histogram. The histogram values are encoded by the grayscale intensity: the lighter the gray, the larger the value of that histogram bin. Thus, darker parts in the image mark areas with little structure, whereas rectangles with only a few nearly white segments are typical for strong straight edges (compare to Fig. 1).

Thus a gradient histogram fingerprint consists of $N \times M \times K$ values H^k_{nm}. In our further work we use a compressed feature vector

$$\mathscr{H}^k_{nm} = \min\left(256/L \cdot H^k_{nm}, 255\right), \tag{9}$$

which maps all floating point values to 1-byte integers. For $N = M = K = 8$ $L = 0.02$ is a good choice [10].

Distance Measure. We measure the distance between two images I_1 and I_2 with the L_1-Norm of the particular feature vector

$$D^{FV}(I_1, I_2) = \begin{cases} \frac{1}{3NM} \sum_{C\in(R,G,B)} \sum_{n=1}^{N} \sum_{m=1}^{M} \left| P^C_{nm}(I_1) - P^C_{nm}(I_2) \right|, & FV = CPF, \\ \frac{1}{NMK} \sum_{n=1}^{N} \sum_{m=1}^{M} \sum_{k=1}^{K} \left| \mathscr{H}^k_{nm}(I_1) - \mathscr{H}^k_{nm}(I_2) \right|, & FV = GH, \end{cases} \tag{10}$$

where FV is the place holder for the feature vector name. The distance between two sequences S_1 and S_2 of length L is given by

$$D^{FV}_L(S_1, S_2) = \frac{1}{L} \sum_{l=1}^{L} D^{FV}(S_1(l), S_2(l)). \tag{11}$$

The distance measures in Eqs. 10 and 11 provide the possibility for defining the equality of two images or image sequences, respectively. We call two images to be equal to each other, if the feature vector distance is less than a threshold Δ^{FV}_I

$$I_1 = I_2 \Leftrightarrow D^{FV}(I_1, I_2) < \Delta^{FV}_I. \tag{12}$$

Accordingly, we define two sequences S_1 and S_2 of length L_1 and L_2 to be equal, if $D^{FV}_L(S_1, S_2)$ is less than a threshold Δ^{FV}_S and their duration difference is less than Δ_L:

$$S_1 = S_2 \Leftrightarrow D^{FV}_L(S_1, S_2) < \Delta^{FV}_S, L = \min(L_1, L_2), |L_1 - L_2| < \Delta_L. \tag{13}$$

3 Searching Frame-Accurately

In this section we introduce our algorithm for frame-accurate identification of duplicate sequences in the fingerprints of live video streams.

The whole algorithm is composed of nearly independent parts, which are sequentially processed and which in principal can be replaced by other choices. After having fingerprinted each frame of a video we need a fast and efficient image search

method. We use an inverted index to identify similar images rapidly and a hash function to map an image feature vector to an one-dimensional table index. Details of this step are explained in Sec. 3.1.

On the basis of the inverted image index we search next for short repeating sequences of about one second and call them *clips*. Candidate duplicate clip pairs are built by looking for short sequence pairs with a required minimal percentage of hash-value-identical frames. Details are explained in Sec. 3.2.

Clip pairs are grouped to longer target sequences according to their temporal coherence and are aligned for proper estimation of start and end frames. This step is explained in Sec. 3.3.

The application of filters dedicated to a search of special interest like commercial mining is discussed in Sec. 3.4.

3.1 Inverted Index and Locality Sensitive Hashing

An inverted index is an efficient method for fast search through large databases. In the text domain, for instance, an inverted index contains a list of all words occurring in a text corpus. For each word, a list of all its positions in the text is provided. Thus, a single look-up in the index is sufficient to retrieve all occurrences. No expensive sequential or tree-based search through the text corpus is required. The only expensive step is the creation of the index. This, however, must only be done once.

A difference between image and text retrieval arises from the high-dimensional and often continuous feature space in image representations. There are several approaches to construct inverted indices for such feature vectors. Hampapur and Bolle (2001) used an inverted index for each component of the feature vector [18]. Shivadas and Gauch (2007) mapped the complete feature vector by means of a hash function to a single scalar value [30]. Hash functions are designed for deterministic but non-injective mapping a sparse representation of a large feature space to an index space whose size is in the order of the data's dense representation. The mapping of different values to the same hash value is called a collision and can be minimized by good mixing properties of the hash function. The modulo function is often a proper choice [24].

The disadvantage of hash functions is that due to the mixing characteristics, no distance measure can be directly deduced from the index values. Neighbored indices do not necessarily belong to features which are close together. One possible answer to this problem is *Locality Sensitive Hashing (LSH)*. Here hash functions are used for which the probability of mapping similar features to the same index is much higher than the probability for mapping more distant features to the same index [15, 23].

Color Patches Features. For the CPFs we evaluate the inverted index on the basis of average image intensities C_I, $C \in (R,G,B)$, which can be easily obtained:

$$C_I = \frac{1}{NM} \sum_{n=1}^{N} \sum_{m=1}^{M} P_{nm}^{C} \qquad \text{with} \qquad C \in (R,G,B). \tag{14}$$

Taking the image averages provides locality sensitivity. The distance in the reduced feature space is always less than or equal to the original distance (see Appendix for a prove). Small variations are handled in the second step: we take the first b bits of each of the averaged values to generate a scalar $3b$-bits integer index value:

$$h^C(I) = C_I \div 2^{(8-b)}, \quad C \in (R,G,B), \quad b \in \{0,\ldots,8\}, \tag{15}$$

$$h^{CPF}(I) = 2^{2b} h^R(I) + 2^b h^G(I) + h^B(I). \tag{16}$$

In this way images with close color average intensities are grouped into the same bins [6]. The bin size $S_{h^{CPF}}$ and the index range $R_{h^{CPF}}$ depend on the choice of b:

$$S_{h^{CPF}} = (2^{8-b})^3, \quad R_{h^{CPF}} = (2^b)^3, \quad b \in \{0,\ldots,8\}. \tag{17}$$

Because of the limited index range $R_{h^{CPF}}$ we do not need any further sample space reduction for our index evaluation.

Gradient Histograms. For GHs we use a three step hashing algorithm to achieve locality sensitivity in our index. We first reduce the feature vector size:

$$\mathscr{H}^k = \sum_{n=1}^{N} \sum_{m=1}^{M} \mathscr{H}^k_{nm}. \tag{18}$$

The image related gradient distribution $\mathscr{H}^k$ is robust to small changes as all feature vectors are mapped to vectors with equal or less distance (see Appendix). The size of the $\mathscr{H}^k$ is $8+P$ bits, if P is the smallest integer with $NM \leq 2^P$.

In the second step we evaluate a first intermediate hash value for every component $\mathscr{H}^k$ by taking the first b bits of every single value $\mathscr{H}^k$:

$$h^1(I) = \sum_{k=0}^{K-1} \left(2^b\right)^k h^1_k(I), \quad \text{with} \quad h^1_k(I) = \mathscr{H}^k \div 2^{(8+P-b)}. \tag{19}$$

The number of possible values $R_{h^1}(b,K)$ gives the range of this first hash value

$$R_{h^1}(b,K) = \left(2^b\right)^K, \quad b \in \{0,\ldots,8+P\}. \tag{20}$$

In dependence on the values of K and b the index range R_{h^1} may be quite large, because we deal with typical sizes of $K = N = M = 8$ [10]. Therefore we apply a modulo function with index range R_{h^2} to the first hash value $h^1(I)$

$$h^2(I) = h^1(I) \mod R_{h^2}. \tag{21}$$

We evaluate Eq. 21 in an iterative way with the Horner scheme:

$$h^2_1(I) = h^1_1(I) \mod R_{h^2}, \tag{22}$$

$$h^2_k(I) = \left(2^b \cdot h^2_{k-1}(I) + h^1_k(I)\right) \mod R_{h^2} \quad \text{for} \quad k = 2,\ldots,K,$$

$$h^{GH}(I) = h^2_K(I). \tag{23}$$

For our calculations we use an index range $R_{h^2} = 100,003$, which meets the criteria for choosing a good table size [24].

The first two steps preserve the locality sensitivity, whereas in the third step, the evaluation of h^2, we use a classical hash function with good mixing properties.

Distance Measure. The hash value representation is the basis for the similarity definition. We call two Images I_1 and I_2 to be *similar*, if their hash values are equal

$$I_1 \sim I_2 \Leftrightarrow h^{FV}(I_1) = h^{FV}(I_2), \quad FV \in (CPF, GH), \tag{24}$$

and we call two image sequences S and T of length L_S and L_T, $L_S \leq L_T$ without loss of generality, to be similar, if a certain amount $\alpha \in (0,1)$ of images is similar

$$S \sim T \Leftrightarrow \sum_{i=1}^{L_S} \theta\left(S(i)\right) > \alpha L_S, \tag{25}$$

with

$$\theta\left(S(i)\right) = \begin{cases} 1, & \text{if} \quad \exists j \in (1, L_T), \quad S(i) \sim T(j), \\ 0, & \text{else.} \end{cases} \tag{26}$$

Thus we distinguish between *identical* (Eq. 13) and *similar* (Eq. 24) images. A small image feature distance (Eq. 10) describes the visual identity of two images. For similar images there is only a certain probability that they are actually identical.

In our search algorithm we first search for similar images and clips, which can be done very fast with the index table, and then refining our results by evaluating the more time consuming image feature distance to identify actually duplicate clips.

Implementation. There are several approaches to implement an inverted index for live-stream applications. One possibility to handle the index search is to deal with a dynamic index table, which is updated after processing each frame. In this case you may execute a table search just in time. Beside the – for our application – unnecessary additional computational overhead, an even more serious issue is that all hashing functions are designed for a certain frame number range of video frames. For instance, the index range $R_{h^2} = 100,003$ for the GH was chosen with a 2 hour mining period in mind, i.e., for 180,000 hash values for PAl videos. Without causing to many collisions the search period cannot just be increased to maybe 24 hours or longer.

As we want to process the live-stream in real-time, while it is acceptable to obtain results with a certain delay, we split the video stream into suitable pieces of about two hours. The resulting fingerprint size of such segments fits into the system memory in a comfortable way. For each slice we build a separate inverted index and store it on the hard drive. Every two hours we invoke a sequence search that takes the latest two hours of fingerprints as the input sequence, searches through it, and

then continues through the previous two hours of fingerprints one after the other until the period we want to search through has been processed (e.g., 24 hours).

3.2 Clip Search

A live video stream appears as an endless sequence of images without any direct hints about embedded repeating sequences. Some information about the underlying structure of the video stream could be gained by shot segmentation. However, in the digital age we can find beside classical hard cuts many hardly to detect complex transitions such as dissolves, wipes, and morphings. The correct detection of all transitions would require separate reliable algorithms for each of them. To avoid errors from cut detection and save the detection time we operate on the raw video stream: we pick random pieces, called clips, out of the stream and search for repetitions of these clips. The duration of a clip may be arbitrarily chosen, but it should be much shorter than the lengths of our target sequences to ensure that there exist clips that are completely contained within each repeated sequence. The smallest unit to search for is a single image. However, single images are often not sufficiently distinctive, and therefore we will settle on one-second clips.

The clip search approach is adopted from DNA sequencing. Complete DNA strands are far too long to be sequenced in one single run. Therefore, they are broken up into much smaller pieces. This can be done in a deterministic or random way (shotgun method). In the latter case all fragments have to be aligned after analysis. There are a variety of sequence alignment methods known in bioinformatics [20].

We can use some pieces of information from the stream to make a proper choice for the processed clips and only start a new clip search at an image, if we find similar images in our inverted index. We scan a video frame by frame, calculate the hash index of each frame and retrieve all similar frames by a look-up in the inverted index. We reject matches, which are temporally very close to our query frame. A minimal gap of 2000 frames is required to avoid detection within the same scene. Furthermore we neglect frames belonging to hash indices with a very large number of entries, i. e. we ignore non-distinctive frames such as black frames or parts of long self-similar sequences.

Figure 4 shows two clips of 25 frames (1 second in PAL). Similar frames with identical hash values are marked. It is possible, that there is no one-to-one mapping between similar images, as visually similar images can be mapped to the same index.

If we find more than 20% similar frames within clips s_1 and s_2, we consider $s_1 \sim s_2$ with $\alpha = 0.2$ (Eq. 25). For all duplicate clip candidates $\mathscr{C}(s_1, s_2)$ we calculate the distance $D_L^{FV}(s_1, s_2)$ (Eq. 11) between them. All unequal candidates (Eq. 13) are discarded, all equal ones are further processed.

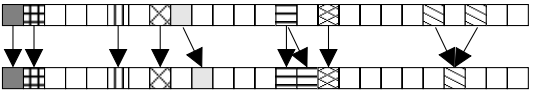


Fig. 4 Two clips with frames matched by hash values.

3.3 Frame-Accurate Repeated Sequences Search

The next task in our algorithm is to build the repeated sequences from the identified clip pairs. At first we group the pairs, which are temporally coherent together. After aligning these groups of clips we can estimate the start and end of all found repeated sequences.

We identify clip pairs $\mathscr{C}$ belonging to the same recurring sequence by looking at their times of occurrence. Two clip pairs are assumed to belong to the same recurring video sequence, if both clips of a pair are closer than 150 frames to the corresponding clips in the other clip pair. This step results in a number of sequences, which are pair-wise identical or similar in parts. Each pair of such sequences is called a duplicate $\mathscr{D}$. Each duplicate is described by two lists S_1 and S_2 representing the corresponding sequences:

$$\mathscr{D} = (S_1, S_2)\,. \tag{27}$$

Each list S_i contains the start frame numbers of the clips building the duplicate:

$$S_i = \left(\text{frame}_1^i, \ldots, \text{frame}_n^i\right), \tag{28}$$

with n denoting the number of clips forming the duplicate. Note that frame_i^1 and frame_i^2 are the start frame numbers of the two clips of a clip pair i. According to our search strategy the frames in S_1 and S_2 are not necessarily in the same temporal order, especially for long still scenes.

Figure 5 illustrates the state in our search algorithm at this point. We have identified a number of small pieces of our target sequence. However, we still miss precise informations about the boundaries.

To reduce the input for the next steps it is possible to do a coarse filtering at this stage. For instance we discard sequences, which do not meet the required minimum number of clips. Content related filtering is discussed in Sec. 3.4.

As it can be seen in Fig. 5 both sequences S_1 and S_2 due to their construction may be shifted to each other, i. e., the offsets o_i with

$$o_i = \text{frame}_i^2 - \text{frame}_i^1 \qquad i = 1, \ldots, n, \tag{29}$$

between corresponding clips are not the same for all n. Hence, we need a method for estimating the real offset between the repetitions of the target sequence.

One simple technique is the iterative alignment during the detection of start and end frames. We find the start and end frame of a found sequence by going backward at the start and forward at the end frame by frame as long as the feature vector distance between the two associated frames is smaller than a threshold. If the image distance signals different frames, we have recognized the end or the start,

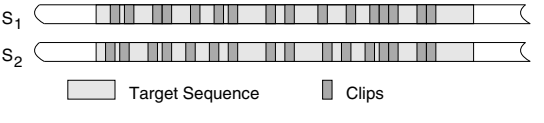


Fig. 5 Recurring video sequence with the clip pairs that were found.

respectively. For sequence alignment we do not stop this procedure, once we have found different frames, but continue the search with a modified offset derived from the original offset by adding a relative offset. We set initially this relative offset to +1 frame. If the distance between two frames under this modified offset is smaller than our threshold, we test the next two frames with the same modified offset until we find different frames again. In the case that the current relative offset has not resulted in frame equality, we try the same relative offset in the opposite direction (i.e., we us the negative offset), and after that we increment the relative offset by one. When a maximal allowed relative offset is reached, the procedure stops and the last found identical frames are rated as start and end, respectively.

We can accelerate this operation by estimating at first the most likely offset o_0 from all offsets o_i. If $F(o_i)$ is the distribution function of the start frame differences of all clip pairs of a sequence candidate, we take the most frequent offset as the first guess for the alignment of both sequences, i.e.,

$$F(o_0) \geq F(o_i) \quad \forall i \in (1,n). \tag{30}$$

After aligning the duplicate sequences with this offset o_0, we can determine the start and end frame with the method described above. At this time, we have frame-accurately determined two occurrences of the same sequence; we know their beginnig and temporal length. To validate results, we can at this stage evaluate the feature-based distance function for the complete sequence and discard eventually false matches.

In a last step we compare the pairwise matched sequences against those from the other duplicates to group frequently occurring sequences together. All detected repeated sequences are compared with the database and added if novel.

3.4 Content-Related Filtering

As already mentioned in Sec. 3.3 we can control the result by filtering the clips found in dependence on the content we search for. Typically not all kinds of recurring sequences are of interest all times. Special kinds of sequences such as commercials can be retrieved by taking their characteristic properties into account and eliminating all non-complying sequences. A useful property is sequence length. For instance, you may distinguish between channel logos (just a few seconds long), commercials (10 to 60 seconds long), and music videos (several minutes long). At this step you may include all features you know are characteristic for the kind of repeating sequences you are looking for. For ads this can be a higher cut rate, black frames, increased audio volume, and others attributes as discussed by Lienhart et al. in [25].

As we want as much as possible to be independent of properties that can be changed by the advertising industry, we restrict our commercial filter to sequence length.

4 Experimental Results

In our experiments we investigate the sensitivity of the algorithm concerning the most relevant system parameters. We test the algorithm by detecting commercials in 24 and 48 hours, respectively, of two manually labeled TV channels. Furthermore we explore the choice of our image features and compare the performance of CPFs against GHs. At last we apply the system to a variety of TV channels with quite different characteristics of advertisement representation.

4.1 Parameters and Numbers of Relevance

For our quantitative experiments we use two 48-hour long video sequences recorded from two different British television channels: Chart TV (a music channel) and Sky Sports News (a sports channel). Both videos are downscaled to half PAL resolution (360×288 pixels) at 25 frames per second. For these two test videos we determined the locations of all occurring commercials manually as our reference truth.

Although our proposed search algorithm mines video streams for all kinds of recurring video sequences, of which commercials are just one example, we focus in our experimental evaluation on the detection of repeating commercials for the following practical reason: It is quite easy and fast to determine manually and unambiguously all recurring commercials in a test video.

Commonly performance of search algorithms is measured by recall and precision: recall R measures the percentage of detected relevant sequences, while precision P reports the percentage of relevant sequences in the search result:

$$R = \frac{\text{number of found relevant sequences}}{\text{number of all relevant sequences}}, \tag{31}$$

$$P = \frac{\text{number of found relevant sequences}}{\text{number of all found sequences}}. \tag{32}$$

Depending on the particular motive of the search, it makes sense to specify slightly modified performance values. Recall and precision from Eqs. 31 and 32 describe the algorithm's performance concerning all repeated sequences in the video stream. As mentioned above we will concentrate on the detection of commercials. We will use the superscript C to indicate this fact. In particular, we will consider the following values, which are either of more theoretical or more practical interest. The most important values for theoretical discussion are R_M^C and P_M^C, which stand for the detection of **M**ultiple occurring commercials, because these values describe the quality of the algorithm. Of more practical interest are the values R_{MD}^C and P_{MD}^C. This pair specifies the detection of **M**ultiple **D**ifferent occurring commercials. Here, we count, if a recurring commercial is detected. It is unimportant whether all repetitions are found. These values are of interest, if we want to build a database of commercials. Here it is sufficient to detect one repetition. Last, but not least, R^C and P^C stands for the detection of all commercials, and R_D^C and P_D^C for all **D**ifferent commercials. The last four values include commercials which are not repeated and

thus cannot detected by the algorithm. They give the possibility to estimate, how successful a repeated sequence search is regarding commercial detection.

It is important to note that our precision values do not correctly reflect the performance of the algorithm. The result list of the search for repeating video sequences will (absolutely correctly) contain plenty of repeated sequences, which are not commercials. Therefore, the reported precision values concerning repeating commercials are quite low, since every recurring sequence that is not a commercial will count as a false alarm. We will try to mitigate this issue by applying a pre-filter to the raw result list that discards all repeating video sequences whose durations divert from the characteristic durations of commercials. In practice, it is our observation that the true precision values of our system are very close to 1 for repeating video sequences. We hardly remember having ever seen a false alarm for repeating video sequences.

A commercial spot is *found*, if the start and end frame differ no more than 5 frames from the exact position. This tolerance is introduced since a commercial spot is sometimes slightly shortened at the boundaries resulting in repetitions of the same spot with slightly different durations. Additionally, commercials are sometimes separated by monochromatic, mostly black frames, which are, if present at the boundaries of all repetitions, strongly spoken part of the repeated sequence, too, but not of the commercial manually marked in the basic truth.

For both 48-hours sequences we manually labeled all commercials occurring within the first 24 hours. In addition, we also labeled the second half of the Chart TV video sequence. This gives us the possibility to estimate the benefit of a 48-hours over a 24-hours search.

Table 1 Ground truth of our test videos: N^C - number of all occurring commercials, N_M^C - number of all repeatedly occurring commercials, N_S^C - number of all singly occurring commercials, N_D^C - number of different commercials, N_{MD}^C - number of repeatedly occurring different commercials, t^C - time covered by all occurring commercials, t_M^C - time covered by all repeatedly occurring commercials, and t_S^C - time covered by all singly occurring commercials.

	Chart TV 24h	Chart TV 48h	Sky Sports News 24h
N^C	486	997	737
N_M^C	428	928	650
N_S^C	58	69	87
N_D^C	164	212	245
N_{MD}^C	106	143	158
N_M^C/N^C	88.1%	93.1%	88.2%
N_{MD}^C/N_D^C	64.6%	67.5%	64.5%
t^C / video length	13.2%	13.5%	20.0%
t_M^C / video length	11.6%	12.5%	17.4%
t_S^C / video length	1.6%	1.0%	2.6%

Table 1 reports key numbers about our test video sequences using the convention that superscript C indicates that all numbers refer to commercials only: N^C specifies the overall number of occurring commercials in the test videos. For each test video N^C can be split into the number of ads N_M^C which are repeated within the overall video sequence and N_S^C which are not repeating (subscript S stands for *single* occurrence). In other words: $N^C = N_M^C + N_S^C$. The overall number of different commercials is denoted by N_D^C, of which only N_{MD}^C are repeated in the overall video sequence. Thus, the following relation holds: $N_D^C = N_{MD}^C + N_S^D$. Thus, the subscript D signifies that a recurring video sequence is counted only once.

As we can see from Table 1, around two thirds of all broadcast commercials appear more than once a day, covering around 88% airtime of all occurring spots. Additionally, the time fraction, which is allocated to TV ads, is shown. In Chart TV about 13% of airtime is devoted to commercials, whereas it is 20% in Sky Sports News. Only between 1% and 3% of the overall airtime is devoted to non-repeating spots. These are the sequences that cannot be found by even a perfect search algorithm for repeating commercials.

Figure 6 depicts the duration distribution of all occurring spots. In Chart TV all commercials – with a few exceptions – are multiples of 10 seconds, whereas in Sky Sports News we find a greater variance in spot durations with a tendency to shorter spots. This distribution encourages the idea of duration filtering to improve the results concerning commercials.

Most of our test cases concerning system parameters use GHs as image features with $N = M = K = 8$. We build-up the inverted index based on hashing as explained in Sec. 3.1.

Content Related Filtering. Our first experiment concerns the last step in our search algorithm - the content related filtering. As discussed above, our proposed algorithm mines videos for all kinds of recurring video sequences, of which commercials are just one example. While recall is not affected by focusing on commercials, it renders the precision value useless. We apply a "commercial filter" to the raw result list in order to give the precision values a meaning: With what precision can TV commercials be found with a repeating video clip search algorithm.

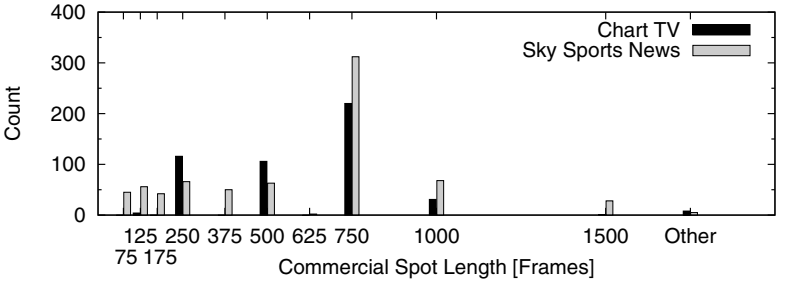


Fig. 6 Duration distribution of TV commercials.

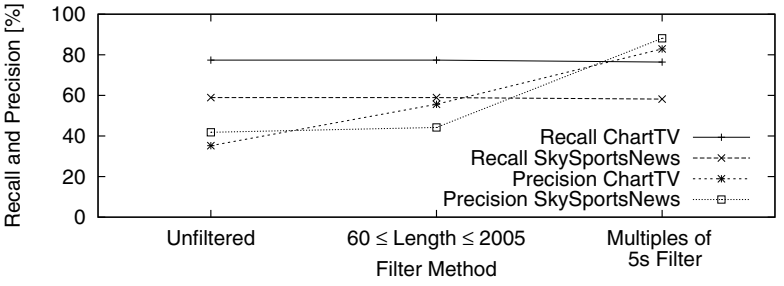


Fig. 7 Content related filter methods.

Figure 7 shows recall and precision in dependence on the filter method. We compare the unfiltered output with a simple length filter, which discards very short and very long sequences, and a more sophisticated filter, which focuses on typical durations of TV commercials. The *simple duration filter* keeps all repeating sequences with a length between 60 and 2005 frames. The more *sophistic filter* keeps all repeating sequences with durations representing multiples of 125 frames or 5 seconds (with a tolerance of $\pm$ 5 frames). Additionally, we include sequences of 75 and 175 frames length. These values are derived from the duration distribution in Fig. 6.

As shown in Fig. 7 the "typical length filter" improves the precision significantly compared to the unfiltered case. There is only a minor decrease in recall due to a small amount of commercials with non-typical durations (Fig. 6). In the following experiments we will always filter the raw result lists with the "typical length filter", i.e., with the "multiples of 5s filter".

Alignment Method. We evaluate the two alignment methods introduced in Sec. 3: (1) The frame-by-frame iteratively estimated offset (SIMPLE) and (2) its variant where the initial offset is the most frequent offset (see Eq. 30) occurring between the clip pairs of a duplicate (OFFSET).

As we can see in Fig. 8 performance values for both methods show only small differences with a slight advantage of the OFFSET method . We tested both methods for several maximum relative offset values, at which the algorithm stops. It can be seen that the performance only degrades for maximum relative offset values of two or less frames. We can reason that our clip pairs in most cases are already aligned up to two frames. However the impact of the maximum allowed offset on execution time is quite small, so we can try larger values to improve the results further.

All subsequent experiments are carried out with the OFFSET alignment method of frame offset and variable start and end frame detection with a maximum relative offset of 20 frames.

Clip Length. The following three test cases concern clip search. At first we investigate the impact of the length of the short segments. Figure 9 shows the recall and precision values for both test videos as well as the execution times for clip search and the whole sequence identification. Execution times are only depicted for the

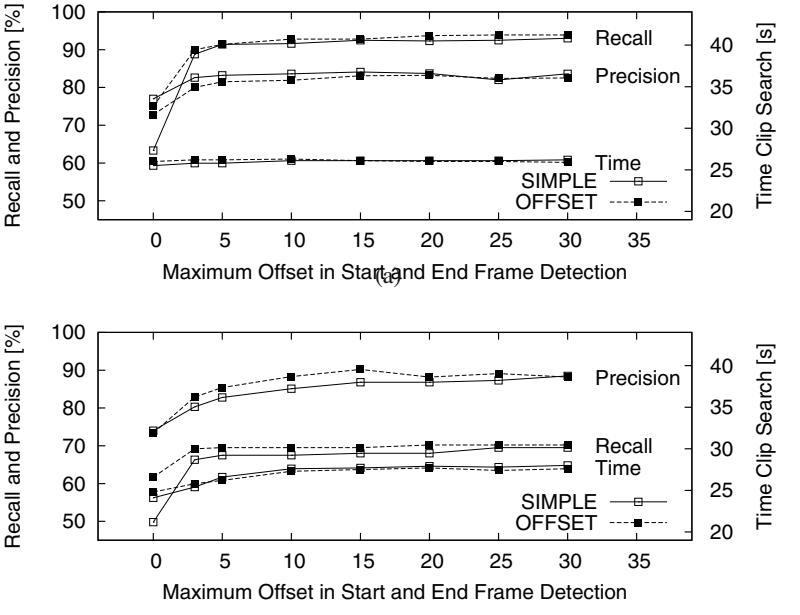


Fig. 8 Performance of the two alignment methods for (a) Chart TV and (b) Sky Sports News: SIMPLE - offset estimation by iteratively varying start and end frames, OFFSET - the offset is initialized with the most frequent start difference between corresponding clips before SIMPLE is applied. Recall, precision, and execution times for repeated sequence search are plotted against the maximum allowed offset.

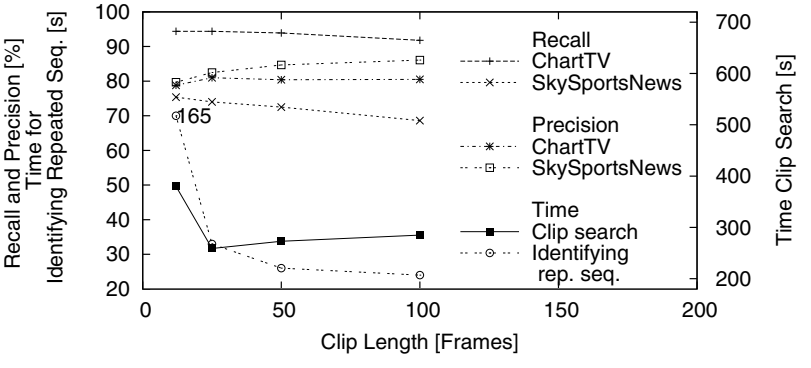


Fig. 9 Performance and execution times in dependence on the duration of the clips.

Chart TV video in order to report the order of magnitude, since it is the same for both videos.

We can recognize that by and large recall decreases with an increase in the length of the clips, whereas the precision increases. In principal, the shorter the duration

of the clips, the better the alignment that can be achieved due to the finer granularity of the samples. The disadvantage of shorter durations is the higher hit rate in non-commercial recurring sequences, which in turn affects precision negatively and leads to an increase in execution times. Especially the time for identifying long repeated sequences is significantly higher for short clips. The increase is mostly caused by the chosen alignment method, with other methods the difference was not that high [11]. This behavior indicates a worse alignment for clips of half a second ($\tilde{1}2$ frames in PAL), the smallest length of clips we have investigated.

All in all a clip length between 25 and 50 frames (corresponds to 1 to 2 second-segments) seems to be an appropriate choice. In our further work we use clips of 25 frames in length.

Minimum Fraction of Matched Frames. In this section we discuss the parameter α which determines when two clips are regarded as similar (Eq. 25).

Fig. 10 plots performance values and execution times against different threshold values α for clip similarity. It is not surprising that recall decreases for higher thresholds, because more segments are missed. Nevertheless, there is a range for smaller threshold values with little impact. The precision values reveal no significant dependence on the test parameter, being stable over a wide range.

Lower threshold values lead to a higher number of falsely detected clips. Most of them are discarded by the image feature based distance measure. Therefore, there is mainly an influence on evaluation time. The more visual distances must be computed, the longer the execution times. This is clearly revealed in Figure 10 by the rapid decline in execution time for clip search for match ratios larger than or equal to 20%.

As we find the range between 20% and 30% quite stable regarding performance as well as execution time, we choose a threshold of 20% (i.e., $\alpha = 0.2$) of matched frames for the further investigations.

Maximum Number of Entries in Hash Table. This test case concerns the search for similar frames in the inverted index. As explained in Sec. 3.1 we find similar

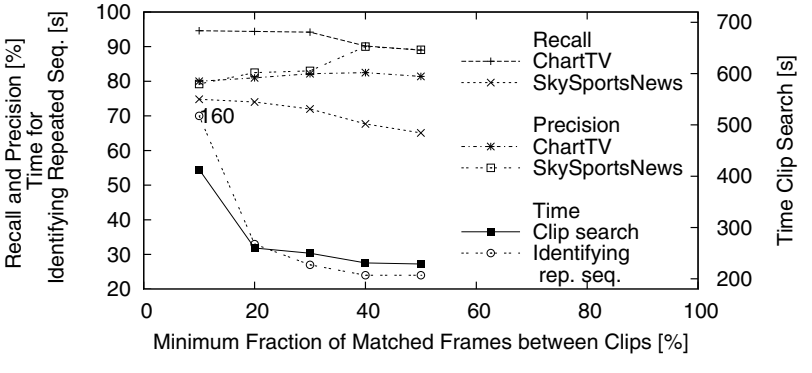


Fig. 10 Performance and execution times in dependence on the minimum required fraction of matched frames.

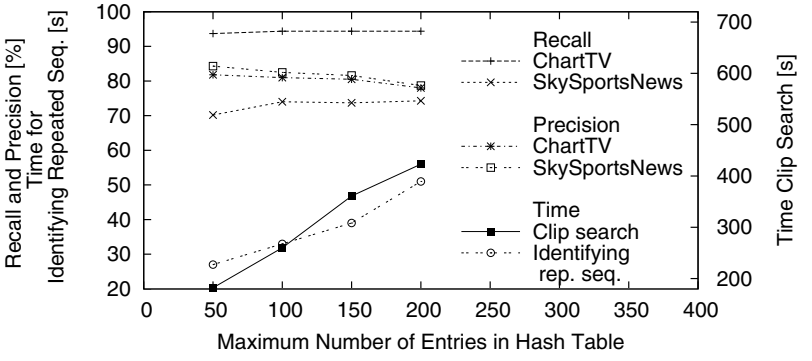


Fig. 11 Performance and execution times in dependence on the upper limit of entries per index in the hash table. All values refer to 2 hour slices.

frames by looking up the list of frame numbers with the same hash value in the inverted index table. In practice, however, all hash values those frame number list exceeds an upper limit of entries must be disregarded. A very large number of entries can either be caused by too many collisions of the hash function or result from unspecific (generic) images such as black frames. Thus, this knockout criterion is introduced to keep evaluation times low for video streams with long self-similar or unspecific image sequences.

As revealed in Figure 11, the influence of this parameter is as expected: Recall can be increased, if we expand our image lists, but simultaneously precision decreases slightly. However, we find this parameter to be most relevant for execution time. In fact, we introduced this limit after watching a tennis match that dramatically slowed down the whole system. The parameter keeps execution time nearly constant, even if there are long periods of very similar images in the stream. Keeping in mind that commercials are short and in general of dynamic content, this parameter is not a restriction.

According to our tests a maximum number of 100 entries per hash value in order to take corresponding frames into account is a good choice for our test configuration of 2 hour slices, i. e. per 180,000 frames. Clearly, this value highly depends on the chosen image features, the applied hash function as well as the duration of the video that is mined.

Minimum Length of Duplicates. This test case applies to the identification of repeated sequences only. Requiring a minimum sequences duration S_i when forming a duplicate $\mathscr{D}$ is one possible mechanism for coarse filtering as described in Sec. 3.4. In fact we require a minimum count n of clips as well as a minimum length $(\text{frame}_n^i - \text{frame}_1^i)$ before creating a duplicate as a possible candidate for a repeated clip. In this way very short and sporadic similar clips can be discarded.

In Figure 12 performance values and execution times for different required sequence lengths are shown. During these experiments we scaled the required minimum number n of clip proportionally.

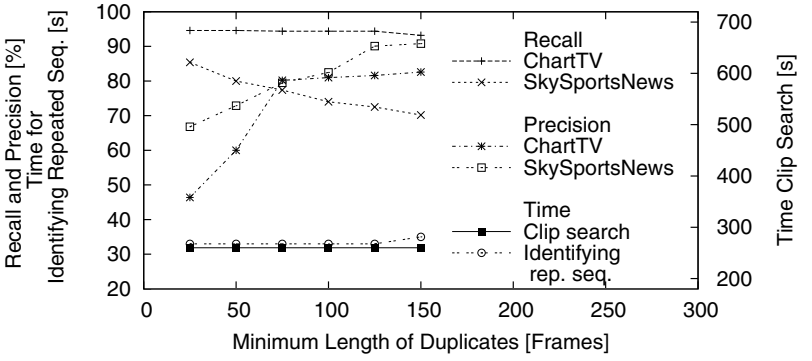


Fig. 12 Performance and execution times in dependence on the minimum length of sequence duplicates.

The recall of Chart TV is nearly constant until the minimum required length exceeds the smallest occurring commercials with a length of 125 frames (see Fig. 6). The precision is lower for small values, but keeps nearly constant for lengths greater than 80 frames. For TV content with such clear commercial characteristics as shown by Chart TV this filter is a good choice for reducing false matches. For Sky Sports News the situation is more complicated. There are many short ads of 75 frames only as well as more recurring non-commercial clips with typical ad durations than in Chart TV. Thus, we find that recall is negatively influenced by increased duration thresholds, whereas the precision can be enhanced significantly. The execution times for clip search is not influenced, because the filter is applied afterwards; the time for identifying repeated sequences is nearly independent, too. For video content like Chart TV a minimum length of 100 frames is an appropriate value, whereas in Sky Sports News this threshold discards the very short commercials.

4.2 Detection Performance

After our sensitivity analysis in the various parameters we finally summarize the performance of our system in detecting unknown commercials by means of their repetitions over one day. Additionally, we discuss the improvements that can be achieved by searching for repeated clips throughout two broadcast days. We use the parameters, which led to the best results during our tests: we set the clip length to 25 frames, require 20% of matched frames between two clips to be considered similar, take only frames into account which belong to hash values with less than 100 entries, and require a minimum length of a 100 frames for sequences assembled from clips. We initialized the frame-by-frame search for the estimation of start and end frames with the most frequent offset from the clip pair set of the duplicate, and filter the result list of repeated clips by the 'multiple of 5s' filter that accounts for the typical ad durations.

Table 2 Recall, precision and execution times for our test videos: R_M^C and P_M^C - recall and precision of all repeatedly occurring commercials, R_{MD}^C and P_{MD}^C - recall and precision of repeatedly occurring different commercials, R^C - recall of all occurring commercials, R_D^C - recall of different commercials.

	Chart TV 24h	Chart TV 48h	Sky Sports News 24h
R_M^C	94.4%	95.8%	74.0%
R_{MD}^C	93.4%	93.7%	79.1%
R^C	83.1%	89.2%	65.3%
R_D^C	63.4%	67.5%	51.4%
P_M^C	81.0%	78.6%	82.5%
P_{MD}^C	82.0%	70.9%	80.6%
Search Time Clips [s]	260	1362	207
Search Time Rep. Seq. [s]	33	491	32

Table 2 lists the different performance values discussed above for our two 24 hours test videos as well as for the 48 hours search through Chart TV.

Concentrating on the performance of finding repeated sequences recall values R_M^C and R_{MD}^C are of interest. R_M^C is the rate for detecting all recurring commercials, while R_{MD}^C concerns all recurring different commercials. The relationship between both values depends on the number of repetitions of each spot. The detection rate for Chart TV is – independent of the video length – better than for Sky Sports News. The main reasons are already discussed above and are mainly due to the occurrence of shorter commercials in Sky Sports News.

With regard to a commercial detection system R^C and R_D^C are of further interest. R^C captures the recall with respect to the detection of all occurring commercials N^C and R_D^C to all occurring different commercials N_D^C. Repetition is not required for these recall values. Due to the high rate of repeated commercials (see Table 1), values are strongly correlated to R_M^C and R_{MD}^C, resulting in a reasonable recall concerning all commercials for Chart TV. Results for Sky Sports News could be enhanced by channel specific parameter settings.

The precision value P_M^C relates to all detected repeating video sequences, while P_{MD}^C focus on repeated different sequences. Again, the relation between both values depends on the number of repetitions of detected commercials and detected non-commercial sequences. If commercials are more often repeated, P_M^C is higher. Note that precision drops for a longer search time due to finding more repeated non-commercial sequences and a higher probability for coalescing two commercials to one if both commercials are repeated in the same temporal order.

Run times in Tab. 2 belong to search runs through 24 and 48 hours videos, respectively. Here, the search time does not scale linearly, because we compare n_T 2 hours slices with each other, resulting in $n_T(n_T+1)/2$ operations. Therefore, in our live detection system the search is carried out each 2 hours, and execution time

scales linearly with the number of past slices we search through, because in this case only the actual slice is compared with those of the past resulting in n_T operations.

4.3 Color Patches Features and Gradient Histograms

All experiments in the previous two sections have been carried out with Gradient Histograms as image feature. In this section we investigate the influence of the chosen image feature on the overall performance. Thus we compare the results with GHs against the results with CPFs.

Figure 13 shows the precision and recall values for both of our test videos for both image features: CPFs and GHs. Performance is plotted against the number of significant bits b used in the locality sensitive hashing step. Both image features perform in a similar way. Searching for clips with GHs is up to 30% faster than using CPFs. This is probably due to the better mixing characteristics of the gradient-based features. Color features are usually more clustered in feature space [10].

4.4 Mining Different TV Channels

This section is dedicated to the practical operation of our system. We apply the system to a variety of broadcast stations from different countries with different content, different ad presentation styles as well as different amounts of advertisements. For every video analyzed we build a database of repeating sequences and label all entries $\tilde{N}$ in the database manually as commercials or non-commercials. According to these values we can evaluate the precision of our automatic mining for commercials. However, it is almost not possible to predict the recall from the maximum expected amount of advertisements, because the real fraction may vary from station to station (see Tab. 1), and is influenced by the time of day, the day of week, or even single

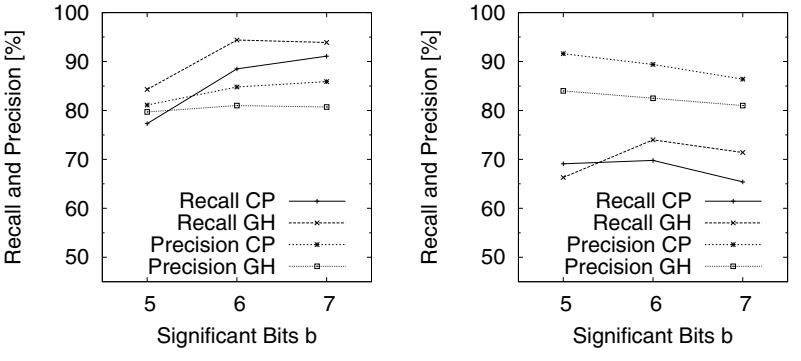


Fig. 13 Performance values for test videos (a) Chart TV and (b) Sky Sports News plotted against the number of significant bits b used for quantization in the computation of the inverted index.

Table 3 Commercial detection performance of our system for different broadcast stations. Given are the durations of the videos we mine, the number $\tilde{N}^C$ of commercials found, the number $\tilde{N}$ of all repeated clips found, the precision $\tilde{P}^C = \tilde{N}^C/\tilde{N}$, the number $\tilde{N}^C_{\pm 0}$ of all frame-accurately found commercials, and the number $\tilde{N}^C_{\pm 5}$ of commercials found if a maximum error of ± 5 frames at the start and/or end position is tolerated.

TV station	Video length [h]	$\tilde{N}^C$	$\tilde{N}$	$\tilde{P}^C$ [%]	$\tilde{N}^C_{\pm 0}$	$\tilde{N}^C_{\pm 5}$
ARD	60	10	49	20.4	2	8
RTL A	13	31	34	91.2	1	30
RTL B	11	12	16	75.0	0	12
RTL C	13	25	35	71.4	1	24
Chart Show TV	51	102	130	78.5	96	6
MTV	48	67	138	48.6	59	8
MTV with mask	48	70	115	60.9	58	12
Sky Sports News A	48	166	221	82.6	156	10
Sky Sports News B	63	178	315	56.5	147	31
Gemini	48	25	79	31.6	20	5
CBS 5 A	4	15	18	83.3	4	11
CBS 5 B	12	32	45	71.1	9	23
ESPN	10	21	27	77.8	1	20

events. Note, that the precision $\tilde{P}^C$ may differ from the true value reported in Tab. 2, because it may happen here that due to a recognition error a sequence is added twice as different sequences to the database.

Table 3 lists the experimental results for various broadcast stations. We tested the algorithm on TV channels, which contain only a small amount of commercials such as the German public broadcaster ARD, as well as on private broadcast channels like Sky Sports News (UK) or RTL (Germany), which devote about 20% of their broadcast time to commercials (see Tab. 1). Note that 20% is the limit in the EU for private broadcast channels due to a directive of the European Union [2], allowing a maximum of 12 minutes of advertisement per hour.

The regulations for public broadcast in Germany are even stricter. Averaged over the year only 20 minutes per day of commercials are allowed to be broadcast, but not more than 25 minutes per day (see [3]). These rules make commercials a rare event in our ARD video stream. Consequently, we only detected 10 distinct commercials within 2.5 days at a low precision of about 20%. Among the falsely detected clips are a number of channel previews with durations typical for commercials. The remaining false detections are mainly caused by repeated news stories. For the three relatively short searches in RTL we achieve a good precision of 70–90%.

MTV (UK) is not only a music channel like Chart Show TV (Chart TV), but also broadcasts shows and TV series, especially for young people, with a large amount of commercials. Here, we can improve the low precision of less than 50% by using a spatial mask, which neglects the first two rows of the 8×8 subareas. By this means we can significantly reduce false detections, which have been caused by overlaying

the music clip title in the upper part of the video frames. Because these overlays may vary across repetitions of a music clip, clips are not detected as a whole. Detected subsequences may unfortunately have the typical durations of commercial.

Sky Sports News A corresponds to our test video. Sky Sports News B is a different recording of the same channel. This time precision is much lower as our algorithm fails probably due to repeatedly broadcast sports events from NTSC footage that is not repeated frame identical. This results in the detection of sequences with arbitrarily beginning and ending. The lack of robustness in this specific case is a disadvantage of our algorithm.

Gemini is an Indian TV channel. Here, we guess a precision of about 30%. However, for non-natives it has been difficult to rate the results: what are commercials and what not? For the same reason we could not analyze the problems of our algorithm.

CBS 5 and ESPN are US broadcast stations. They differ from the other stations by their TV norm NTSC instead of PAL. Nevertheless, we get comparable precision values. Most false alarms result from previews and channel advertisements, which are presented commercial-like. We include the relatively short recording named CBS 5 A into our experiments, because we know the ground truth of this video. We can determine the recall to be 78.9%.

In Tab. 3 we added information about the accuracy of the detected sequences. Usually, 80% and more of the detected commercials have been detected frame-accurate, except for the German and US stations. Here, the separating black frames are detected as parts of the repeated sequences. we identify dissolves and fade-ins and outs between commercials in the US TV as the source of problems.

5 Related Work

Analyzing TV broadcasts comprises several tasks such as the recognition of copies of known video clips, the identification of certain events, or the detection of video sequences of interest. In most cases all approaches are related to each other. So, you may first detect sequences of interest and then search for copies. Or you first identify all repeating sequences and than extract a particular subset. Commercial detection is one well-studied topic of how all of these approaches can be applied to TV streams. Other topics are, for example, news tracking or monitoring of sports events.

In the following we will give an overview of the approaches for commercial detection and cast the detection of repeated sequences in the context of video retrieval.

If we do not have a database of commercials at hand, a feature-based detection approach is needed. A first step is to identify commercial blocks based on the special characteristics of commercials as well as their presentation by the TV stations. In [25] Lienhart et al. (1997) derived typical ad properties such as their restricted lengths, their high dynamic content, which can be measured by an increased cut-rate, and the occurrence of still images at the end of spots presenting company related information, which are typically accompanied by the appearance of a certain amount of text. Additionally, they extracted characteristic features of commercial

blocks, which include the separation of single commercials by black frames and an increased audio volume. The particular laws of a country may add more properties such as the requirement of intro/outro sequences which clearly separate advertisements from program content and the disappearance of the channel logo during commercial breaks.

This approach of identifying commercial blocks has been refined by other investigators. Dimitrova et al. (2002) [9] and Agnihotri et al. (2003) [4] extracted commercial triggers from low-level MPEG decoding. At this level they could derive information about the occurrence of black and unicolored frames, hard cuts, and changes from and to letterbox formats. In [4] a genetic algorithm is used to exploit the various analyzed features. Glasberg et al. (2006) [16] combined the appearance of black frames, the disappearance of station logo, shot duration, and cut rate in a decision tree to reliable recognize commercials, whereas Albiol et al. (2004) [5] limit characteristic features to logo disappearance and shot duration (cut rate). In [8] Chen et al. (2005) focused on the separation of commercials from news content. Therefore, they combined cut frequency with a caption detector due to fewer captions in commercials than in newscasts. Furthermore, they implemented a speech-music discriminator by combining typical audio-visual characteristics. Duan et al. (2006) implemented a complete system consisting of commercial boundary detection, commercial classification and commercial identification [12]. In the first step they seek for commercials using shot detection and audio scene change together with the detection of black frames, silence and still images at the end of commercial clips, which they mark as frames with product information. Especially from these frames they extract keywords by means of OCR for the classification step and get information about commercial content to identify the product line. For identifying clips they use image features like ordinal representations and histograms of them.

There are several disadvantages of methods recognizing commercial blocks based on their characteristic features. A certain amount of commercials are atypical. They have, for example, only a few cuts or even no cuts at all. They can appear like news stories, movies or cartoons. Also laws vary between countries and in time, which makes e.g. the logo disappearance during ad breaks not universally applicable. Besides the separation by black frames we can find hard cuts or dissolves between consecutive commercials. Advertisements must not be part of a commercial block at all, but can be broadcast as a single spot. This implies the need for a more universal algorithm.

Detecting known commercials avoids the limitations of the feature-based approach. Therefore, a lot of frameworks follows this idea. Lienhart et al. (1997) proposed an algorithm which makes use of approximate substring matching for comparing the video input with sequences from a database [25]. Fingerprints based on color coherence vector [27] of all known commercials are stored in the database. Alternatives for fingerprinting as well as the detection and comparison methods have been proposed. Sánchez and Binefa (1999), for example, reduced the fingerprint storage by only storing image features for key frames [29]. Each key frame represents a single shot. They reduce the dimension of the fingerprint further by applying Principle Component Analysis (PCA) to the color histograms feature vectors.

Not only image features, but also temporal information from video stream can be taken into account. Hoad and Zobel (2003) [21] improved their very compact signature from [22], which only contains the duration of shots of a given video sequence, to make it applicable to video clips containing only a few cuts. They investigate features, which describe the differences between consecutive frames, such as color shift, the distance between color histograms, and centroids, which correspond to the motion vectors of the darkest and the lightest part of the pictures.

As a fast alternative for comparing two sequences, the use of inverted indexes and hash tables has been investigated by Shivadas and Gauch (2007) in [30]. They implemented a real-time commercial recognition system on the basis of color moments with frame-level hashing. The use of hash tables provides constant access to large databases. Inverted indexes have been introduced to image retrieval by Squire et al. (1999) [31].

For building a database for commercial recognition the detection of repeated sequences is an appropriate technique. Pua et al. (2004) introduced a real-time repeated video sequence identification system [28]. They extract color moments features for all frames. Additionally they execute a temporal segmentation to get the shot length to which an image belongs. Based on this temporal segmentation they look for repeated shots by counting similar images, which are fast identified by a hash table look-up. Gauch and Shivadas (2005) extend this approach to a commercial detection system by adding a classifier, which labels repeated sequences as commercials or non-commercials [14]. At this step they investigate the already mentioned typical properties such as black frames or high cut rate.

Duygulu et al. (2004) only operate on the shot level by extracting key frames after temporal segmentation [13]. They carry out a search for similar images concerning a combination of color and edge based features together with a face detector. In their second approach they merge audio and color features. With a proper combination of both strategies they can improve their performance values. Together with Can he developed in 2007 a system for near duplicate sequence detection, which are not real copies, but can vary in illumination or view points [7]. These aspects are more related to news tracking, and require image features, which are robust to such variations, such as SIFT features [26] and HSV statistics. They construct a tree for finding sequences within the lists of similar key frames.

In contrast to the previous approaches Yuan et al. (2007) provide an algorithm without shot detection for finding repetitive clips [32]. They chop the video stream into small overlapping segments and extract visual signatures for these small segments by averaging image color histograms and combining ordinal measures of all frames into a histogram for the video segment. They build continuous paths for finding repeated sequences through the lists of similar video segments.

6 Conclusion

We introduced an algorithm for detecting repeated sequences in TV streams. The algorithm is designed to operate in real-time on live streams. The main intention of

our framework is the creation of databases of recurring sequences. For later use of such databases it is necessary to determine beginning and end of such sequences frame-accurately, in contrast to other applications, which, for instance, only count the number of repetitions. We applied the system to commercial detection as a possible field of application.

Our algorithm works well for broadcast station with only little amount of advertisements as well as for stations heavy loaded with commercials. It is applicable to both PAL and NTSC broadcasts. Our experiments showed that it is sometimes necessary to adjust the system to channel characteristics to improve the performance of repeated sequence detection in general and commercial detection based on simple duration filters in specific. For channels making heavy use of overlay information such as MTV or some sports casts, blocking out the spatial areas, which are often used to show overlay text, from the image feature computation improves the detection performance significantly. In other cases we had to deal with constantly appearing black frames for commercial separation. There was only one situation our algorithm could not handle: the non-frame accurate repetition of sequences. Our algorithm could not robustly detect these repetitions, since no provisions have been taken for this specific scenario.

In our tests we investigated two different image features: color-based Color Patches Features (CPFs) and edge-based Gradient Histograms (GHs). For our test videos both feature types performed in a similar way. Color patches are faster to evaluate and need a smaller amount of storage, but they are less discriminative and may cause problems for ill-conditioned video streams [10].

We can finally conclude that we developed an algorithm for reliable real-time detection of recurring sequences, which is successfully applied to broadcast stations from all over the world.

Acknowledgements. This project was supported by Half Minute Media Ltd.

Appendix

The reduction of the feature vectors' dimensionality by summation over subsets or all components, respectively, preserves the relation of distances in the sense that images, which are close together in the feature space, are also close together in the reduced space. In the following we proof that the distance according to the L_1-norm in the reduced dimension is always equal to or less than the original distance.

Let $\mathbf{a} = (a_1, \ldots, a_n)$ and $\mathbf{b} = (b_1, \ldots, b_n)$ be two n-dimensional feature vectors with the L_1-distance

$$d(\mathbf{a}, \mathbf{b}) = \sum_{i=1}^{n} |a_i - b_i| , \tag{33}$$

and $\mathbf{A} = (A_1, \ldots, A_k)$ and $\mathbf{B} = (B_1, \ldots, B_k)$, $k < n$, the corresponding feature vectors in the k-dimensional reduced space with each of their components being a sum of a subset of the components of the feature vectors $\mathbf{a}$ and $\mathbf{b}$, respectively, i.e.,

$$A_i = \sum_{j=m_i}^{n_i} a_j, \qquad B_i = \sum_{j=m_i}^{n_i} b_j, \tag{34}$$

where without loss of generality

$$m_1 = 1, \qquad m_i \leq n_i \quad \forall i, \qquad m_{i+1} = n_i + 1 \quad \forall i, \qquad n_k = n.$$

Then, the L_1-distance D(**A**, **B**) is always less than or equal to d(**a**,**b**), because

$$\begin{aligned} D(\mathbf{A},\mathbf{B}) &= \sum_{i=1}^{k} |A_i - B_i| && (35)\\ &= \sum_{i=1}^{k} \left| \sum_{j=m_i}^{n_i} a_j - \sum_{j=m_i}^{n_i} b_j \right| && (36)\\ &= \sum_{i=1}^{k} \left| \sum_{j=m_i}^{n_i} (a_j - b_j) \right| && (37)\\ &\leq \sum_{i=1}^{k} \sum_{j=m_i}^{n_i} |a_j - b_j| && (38)\\ &\leq \sum_{i=1}^{n} |a_i - b_i| && (39)\\ &\leq d(\mathbf{a},\mathbf{b}). && (40) \end{aligned}$$

References

1. Half minute media, http://www.halfminute.com
2. Council directive 89/552/eec of 3 October 1989 on the coordination of certain provisions laid down by law, regulation or administrative action in member states concerning the pursuit of television broadcasting activities (1989)
3. Staatsvertag für Rundfunk- und Telemedien (Rundfunkstaatsvertrag – RStV –) (2008)
4. Agnihotri, L., Dimitrova, N., McGee, T., Jeannin, S., Schaffer, D., Nesvadba, J.: Evolvable visual commercial detetctor. In: Proceedings of the 2003 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2003), IEEE Computer Society, Madison (2003)
5. Albiol, A., Fullà, M.J.C., Albiol, A., Torres, L.: Detection of tv commercials. In: Proceedings of the International Conference on Acoustics, Speech and Signal Processing, Montreal, Canada, vol. 3, pp. 541–544 (2004)
6. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Commun. ACM 51(1), 117–122 (2008), http://doi.acm.org/10.1145/1327452.1327494
7. Can, T., Duygulu, P.: Searching for repeated video sequences. In: MIR 2007: Proceedings of the international workshop on multimedia information retrieval, pp. 207–216. ACM, New York (2007), http://doi.acm.org/10.1145/1290082.1290112

8. Chen, J.C., Yeh, J.H., Chu, W.T., Kuo, J.H., Wu, J.: Improvement of commercial boundary detection using audiovisual features. In: Ho, Y.-S., Kim, H.-J. (eds.) PCM 2005. LNCS, vol. 3767, pp. 776–786. Springer, Heidelberg (2005)
9. Dimitrova, N., Jeannin, S., Nesvadba, J., McGee, T., Agnihotri, L., Mekenkamp, G.: Real time commercial detection using mpeg features. In: Proceedings of the 9th International Conference on Information Processing and Management of Uncertainty in Knowlwdge-based Systems (IPMU 2002), Annecey, France, pp. 481–486 (2002)
10. Döhring, I., Lienhart, R.: Fast and effective features for recognizing recurring video clips in very large databases. In: International Workshop on Video and Multimedia Digital Library (VMDL 2007), Modena, Italy, pp. 65–70 (2007)
11. Döhring, I., Lienhart, R.: Fast frame-accurate mining for repeating video clips. Technical Report Institut für Informatik, Universität Augsburg (2008), http://www.opus-bayern.de/uni-augsburg/volltexte/2008/130014
12. Duan, L.Y., Wang, J., Zheng, Y., Jin, J.S., Lu, H., Xu, C.: Segmentation, categorization, and identification of commercial clips from tv streams using multimodal analysis. In: MULTIMEDIA 2006: Proceedings of the 14th annual ACM international conference on Multimedia, pp. 201–210. ACM, New York (2006), http://doi.acm.org/10.1145/1180639.1180697
13. Duygulu, P., yu Chen, M., Hauptmann, A.: Comparison and combination of two novel commercial detection methods. In: Proceedings of the 2004 IEEE International Conference on Multimedia and Expo., Taipei, Taiwan (2004)
14. Gauch, J.M., Shivadas, A.: Identification of new commercials using repeated video sequence detection. In: Proceedings of the 2005 International Conference on Image Processing, Genoa, Italy, pp. 1252–1255 (2005)
15. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: VLDB 1999: Proceedings of the 25th International Conference on Very Large Data Bases, pp. 518–529. Morgan Kaufmann Publishers Inc., San Francisco (1999)
16. Glasberg, R., Tas, C., Sikora, T.: Recognizing commercials in real-time using three visual descriptors and a decision tree. In: Proceedings of the 2006 IEEE International Conference on Multimedia and Expo., pp. 1481–1484 (2006)
17. Hampapur, A., Bolle, R.: Feature based indexing for media tracking. In: Proceedings of International Conference on Multimedia and Expo., New York (2000)
18. Hampapur, A., Bolle, R.: Videogrep: Video copy detection using inverted file indices. Technical report on some unpublished work, IBM Research (2001), http://www.research.ibm.com/ecvg/pubs/arun-vgrep.html
19. Hampapur, A., Bolle, R.M.: Comparision of distance measures for video copy detection. Computer Science RC 22056, IBM Research (2001)
20. Hansen, A.: Bioinformatik: Ein Leitfaden für Naturwissenschaftler, 2nd edn. Birkhäuser, Basel (2006)
21. Hoad, T.C., Zobel, J.: Fast video matching with signature alignment. In: MIR 2003: Proceedings of the 5th ACM SIGMM international workshop on Multimedia information retrieval, pp. 262–269. ACM, New York (2003), http://doi.acm.org/10.1145/973264.973304
22. Hoad, T.C., Zobel, J.: Video similarity detection for digital rights management. In: ACSC 2003: Proceedings of the 26th Australasian computer science conference, pp. 237–245. Australian Computer Society, Inc., Darlinghurst (2003)

23. Indyk, P., Motwani, R.: Approximate nearest neighbors: towards removing the curse of dimensionality. In: STOC 1998: Proceedings of the thirtieth annual ACM symposium on Theory of computing, pp. 604–613. ACM, New York (1998), http://doi.acm.org/10.1145/276698.276876
24. Knuth, D.E.: Sorting and Searching. In: The Art of Computer Programming, 2nd edn., vol. 3. Addison-Wesley, Reading (1998)
25. Lienhart, R., Kuhmünch, C., Effelsberg, W.: On the detection and recognition of television commercials. In: Proc. IEEE Conf. on Multimedia Computing and Systems, Ottawa, Canada, pp. 509–516 (1997)
26. Lowe, D.G.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2) (2004)
27. Pass, G., Zabih, R., Miller, J.: Comparing images using color coherence vectors. In: ACM Conference on Multimedia, Boston, Massachusetts, pp. 65–74 (1996)
28. Pua, K.M., Gauch, J.M., Gauch, S.E., Miadowicz, J.Z.: Real time repeated video sequence identification. Comput. Vis. Image Underst. 93(3), 310–327 (2004), http://dx.doi.org/10.1016/j.cviu.2003.10.005
29. Sánchez, J.M., Binefa, X.: Audicom: A video analysis system for auditing commercial broadcasts. In: International Conference on Multimedia Computing and Systems, vol. 2, p. 272 (1999), http://doi.ieeecomputersociety.org/10.1109/MMCS.1999.778373
30. Shivadas, A., Gauch, J.M.: Real-time commercial recognition using color moments and hashing. In: CRV 2007: Proceedings of the Fourth Canadian Conference on Computer and Robot Vision, pp. 465–472. IEEE Computer Society, Washington (2007), http://dx.doi.org/10.1109/CRV.2007.53
31. Squire, D.M., Müller, H., Müller, W.: Improving response time by search pruning in a content-based image retrieval system, using inverted file techniques. In: CBAIVL 1999: Proceedings of the IEEE Workshop on Content-Based Access of Image and Video Libraries, p. 45. IEEE Computer Society, Washington (1999)
32. Yuan, J., Wang, W., Meng, J., Wu, Y., Li, D.: Mining repetitive clips through finding continuous paths. In: MULTIMEDIA 2007: Proceedings of the 15th international conference on Multimedia, pp. 289–292. ACM, New York (2007), http://doi.acm.org/10.1145/1291233.1291294

YouTube Scale, Large Vocabulary Video Annotation

Nicholas Morsillo, Gideon Mann, and Christopher Pal

Abstract. As video content on the web continues to expand, it is increasingly important to properly annotate videos for effective search and mining. While the idea of annotating static imagery with keywords is relatively well known, the idea of annotating videos with natural language keywords to enhance search is an important emerging problem with great potential to improve the quality of video search. However, leveraging web-scale video datasets for automated annotation also presents new challenges and requires methods specialized for scalability and efficiency.

In this chapter we review specific, state of the art techniques for video analysis, feature extraction and classification suitable for extremely large scale automated video annotation. We also review key algorithms and data structures that make truly large scale video search possible. Drawing from these observations and insights, we present a complete method for automatically augmenting keyword annotations to videos using previous annotations for a large collection of videos. Our approach is designed explicitly to scale to YouTube sized datasets and we present some experiments and analysis for keyword augmentation quality using a corpus of over 1.2 million YouTube videos. We demonstrate how the automated annotation of web-scale video collections is indeed feasible, and that an approach combining visual features with existing textual annotations yields better results than unimodal models.

Nicholas Morsillo
Department of Computer Science, University of Rochester, Rochester, NY 14627
e-mail: morsillo@cs.rochester.edu

Gideon Mann
Google Research, 76 Ninth Avenue, New York, NY 10011
e-mail: gideon.mann@gmail.com

Christopher Pal
Département de génie informatique et génie logiciel, École Polytechnique de Montréal, Montréal, PQ, Canada H3T 1J4
e-mail: christopher.pal@polymtl.ca

D. Schonfeld et al. (Eds.): Video Search and Mining, SCI 287, pp. 357–386.
springerlink.com

1 Introduction

The web has become an indispensable resource for media consumption and social interaction. New web applications, coupled with spreading broadband availability, allow anyone to create and share content on the world wide web. As a result there has been an explosion of online multimedia content, and it is increasingly important to index all forms of content for easy search and retrieval.

Text-based search engines have provided remarkably good access to traditional web media in the online world. However, the web is rapidly evolving into a multimedia format, and video is especially prominent. For example, YouTube receives over twenty hours of new video uploads every minute. Standard search engines cannot index the vast resources of online video unless the videos are carefully annotated by hand. User-provided annotations are often incomplete or incorrect, rendering many online videos invisible to search engine users.

Clearly there is an immediate need for video-based search that can delve into the audio-visual content to automatically index videos lacking good textual annotations. Video indexing and retrieval is an active research discipline that is progressing rapidly, yet much of this research avoids the difficult issue of web-scalability. We need robust techniques for analyzing and indexing videos, and we also need techniques to scale to handle millions of clips. Furthermore, we desire that web-scale approaches benefit from the increase in data by learning improved representations from the expanded datasets.

In this chapter we review a portion of the image and video mining literature with a critical eye for scalability. Our survey covers both low level visual features and higher level semantic concepts. We observe that in contrast to the relatively new discipline of video annotation, image annotation has received considerably more research attention in the past. Much of the image annotation work can be transferred to the video domain, particularly where scalability issues are involved.

Video search and mining is a broad field, and here we choose to focus on the task of automated annotation from web datasets. We propose a novel technique to automatically generate new text annotations for clips within a large collection of online videos. In contrast to existing designs which may involve user feedback, visual queries, or high level semantic categories, our approach simply attempts to enhance the textual annotations of videos. This is beneficial as user feedback and visual query specification can be time consuming and difficult, and our approach is not constrained to a fixed set of categories. Additionally, the enhanced annotations resulting from our approach can be used directly in improving existing text-based search engines.

Our method is depicted in Figure 1, and an overview of the procedure is as follows. Beginning with a query video to annotate, we decompose the video into shots using a shot boundary detector. Visually similar shots are discovered from the pool of shots across all videos in the corpus. Then, a probabilistic graphical model is used to decide which annotation words to transfer from neighboring shots to the query video. Key to this approach is a scalable approximate nearest neighbor algorithm implemented using MapReduce [9], coupled with a compact representation of shot

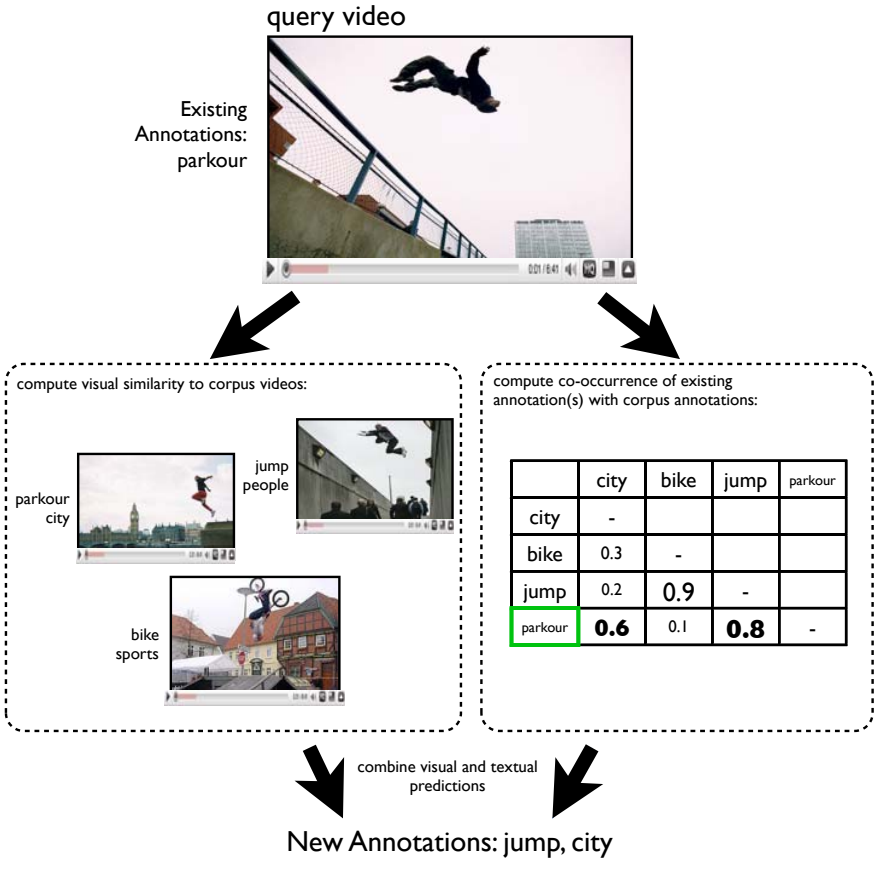


Fig. 1 An example of our approach to generating new text annotations for online videos.

feature vectors. The probabilistic model maintains efficiency by approximating the contributions of the majority of corpus video shots which are not found to be nearest neighbors to a query.

Video search and mining research has traditionally involved known datasets with fixed sets of keywords and semantic concepts, such as TRECVID [41] and the Kodak benchmark dataset [26]. A key difference in our work is the absence of a constrained set of annotation keywords. We construct an annotation vocabulary directly from annotations provided by users and uploaders of online videos. This permits a larger range of annotations that are tailored for the data and it avoids costly manual construction of vocabularies and concept ontologies. However, it also introduces new challenges for measurement and performance evaluation, since ground truth labels are not fixed or verified.

We conclude the chapter with preliminary results of our approach applied to a large portion of the YouTube corpus.

2 Image and Video Analysis for Large-Scale Search

Successful annotation and search is supported by deep analysis of video content. Here we review methods for analyzing and indexing visual data, and observe how these methods relate to the problem of large-scale web video search. We begin our review with image analysis techniques as many successful video analysis methods are constructed from them, and they offer unique insights into the scalability problem. In some cases image techniques may be applied directly to still video frames. We then turn our attention to techniques specifically for video.

2.1 Image Analysis and Annotation

Image annotation is an active field of research that serves as a precursor to video annotation in numerous ways. Video features are often inspired and sometimes directly borrowed from image techniques and many methods for image indexing are also easily applied to video. Here we survey some of the most relevant static image annotation literature including modern trends in the field and adaptations of techniques for static image annotation to video. We also cover emerging and state of the art feature extraction techniques specifically designed for video. We review image features, indexing techniques, and scalable designs that are particularly useful for working with web-scale video collections.

2.1.1 Image Features for Annotation

The relatively early work on image annotation by Mori et al. [31] used the co-occurrence of words and quantized sub-regions of an image. They divided an image into a number of equally sized rectangular parts, typically 3x3 or 7x7. They then use a 4x4x4 cubic RGB color histogram, and 8-direction and 4 resolution histogram of intensity after Sobel filtering. This procedure gives them 96 features from an image. Duygulu *et al.* [10] cast the object recognition problem as a form of machine translation and sought to find a mapping between region types and annotation keywords. They segmented images using normalized cuts then only used regions larger than a minimum threshold for their visual representation. This procedure typically lead to 5-10 regions for an image. From these regions they used k-means to obtain 500 blobs. They computed 33 features for each image including: region color and standard deviation, region average orientation energy (12 filters), region size, location, convexity, first moment, and the ratio of region area to boundary length squared. Their model was trained using 4500 Corel images where there are 371 words in total in the vocabulary and each image has 4-5 keywords. Jeon *et al.* [22] used the same Corel data, word annotations and features used in [10]. They used this vocabulary of blobs to construct probabilistic models to predict the probability of generating a word given the blobs in an image. This general approach allows one to annotate an image or retrieve images given a word as a query.

More recently Makadia *et al.* [28] have proposed a new, simple set of baseline image features for image annotation problems. They also proposed a simple technique

to combine distance computations to create a nearest neighbor classifier suitable for baseline experiments. Furthermore, they showed that this new baseline outperforms state of the art methods on the Corel standard including extensions of Jeon *et al.* [22] such as [14]. The new baseline was also applied to the IAPR TC-12 [18] collection of 19,805 images of natural scenes with a dictionary of 291 words as well as 21,844 images from the ESP collaborative image labeling game [45]. We discuss this method in some detail because it produces state of the art results and we will use these features in our own experimental work on video annotation. We will refer to these features as JEC (Joint Equal Contribution) features as the authors advocate computing distances using an equal weighting of distances computed for a given feature type when making comparisons. JEC features consist of simple color and texture features. Color features are created from coarse histograms in three different color spaces: RGB, HSV and LAB. For texture, Gabor and Haar Wavelets are used.

Color 16-bin per channel histograms are used for each colorspace. In a comparison of four distance measures (KL-divergence, a χ^2 statistic, L_1-distance, and L_2-distance) on the Corel dataset, [28] found that L_1 performed the best for RGB and HSV while the KL-divergence was better suited for LAB.

Texture Each image is filtered with Gabor wavelets at three scales and four orientations. From each of twelve response images a histogram for the magnitudes is built. The concatenation of these histograms is a refered to as feature vector 'Gabor'. The second component of the Gabor feature vector captures the phase by averaging the phase over 16×16 blocks in each of the twelve response images. The mean phase angles are quantized into eight values or three bits and concatenated into a feature vector referred to as 'GaborQ'.

Haar wavelets are generated by block convolution with horizontally, diagonally and vertically oriented filters. After rescaling an image to 64×64 pixels a 'Haar' feature is generated by concatenating the Haar response magnitudes. Haar responses are quantized into the three values: 0, 1 or -1 for zero, positive and negative responses. A quantized version of the Haar descriptor called 'HaarQ' was also explored. They used L_1 distance for all texture features.

2.1.2 Image Features for Object, Category and Scene Recognition

Object recognition, scene recognition and object category recognition can all be thought of as special cases of the more general image annotation problem. In object recognition there has been an lot of interest in using techniques based on SIFT descriptors [27]. We briefly review SIFT here since they are representative of a general "keypoint plus descriptor" paradigm. SIFT and other similar variants consist of two broad steps, keypoint detection and descriptor construction. First, one detects keypoints or interest points in an image. In the SIFT approach one detects minimal or maximal points in a difference of Gaussians scale space pyramid, but other methods use Harris corners or other interest point detection techniques. The local descriptor centered at the interest point is assigned an orientation based on the image gradient in a small region surrounding the key-point. Finally, given a scale and orientation,

the SIFT descriptor itself is built from histograms of image gradient magnitudes in the region surrounding the scaled and oriented region surrounding the key-point.

SIFT based techniques work well for detecting repeated instances of the same object from images of multiple views. However, other research [13] has suggested that for recognizing more variable visual concepts like natural scene categories such as forest, suburb, office, etc. it is better to build SIFT descriptors based on dense sampling as opposed to centered on an interest point detector. More recently histogram of oriented gradient or HoG features [7] have been receiving increased attention for more general visual recognition problems. Such features are similar to SIFT descriptors but take a dense sampling strategy. More specifically, Dalal and Triggs [7] studied each stage of descriptor computation and the effect on performance for the problem of human detection. They concluded that fine-scale gradients, fine orientation binning, relatively coarse spatial binning, and high-quality local contrast normalization in overlapping descriptor blocks are all important for good results. Since these observations HoG features have been a popular and effective choice for various groups participating in the Pascal Visual Object Classes challenge [11]. The Pascal challenge is a good example of a well organized competition focusing on the task of recognition and detection for a 20 object classes, namely recognizing: *People*, *Animals* - bird, cat, cow, dog, horse, sheep, *Vehicles* - aeroplane, bicycle, boat, bus, car, motorbike, train, and *Indoor items* - bottle, chair, dining table, potted plant, sofa, tv/monitor. Larger object category data sets such as CalTech101 [12] or CalTech256 [17] with 101 and 256 object categories respectively have also received considerable attention. Indeed, many of the recent developments in visual features have been motivated by improving recognition performance for these object category problems.

There has been increasing interest in addressing much larger problems for object, category and scene recognition. A common theme amongst many of these approaches is the use of vocabulary trees and data structures for indexing large visual vocabularies. In fact these vocabularies are typically so large that the indexing step serves to operate much like an approximate nearest neighbor computation. We discuss a few prominent examples.

Recently, Nistér and Stewénius [32] developed a system able to recognize in real-time specific CD covers from a database of 40,000 images of popular CDs. They also presented recognition results for 1619 different everyday objects using images of four different views of each object. For their features they use an interest point detection step obtained from Maximally Stable Extremal Regions (MSERs) [29]. They obtained an elliptical patch from the image centered at the interest point which they warped into a circular patch. From this patch they computed a SIFT descriptor. They quantized descriptors using k-means and to accelerate the matching of these features to a large database they created a hierarchical cluster tree. They used a bag of visual words representation and performed retrieval using term frequency inverse document frequency *tf-idf* commonly used in text retrieval.

In another example, Schindler *et al.* [37] used a similar approach for the problem of location recognition from a city scale database of roadside images. Their imagery continuously covered a 20 kilometer stretch of road through commercial, residential

and industrial areas. Their database consisted of 30,000 images and 100 million SIFT features. They used hierarchical k-means to obtain a vocabulary tree and they experimented with different branching factors and techniques for identifying informative features.

Finally, the work of Torralba *et al.* [42] represents an important shift towards addressing problems related to extremely large data sets. They have used text based queries to image search engines to collect 80 million low resolution images from the web. Natural language annotations are used such that imagery is associated with words; however, language tags are only based on the initial query terms used to fetch imagery and the results are noisy. However, they have been able to demonstrate that a large database of small images is able to solve many different types of problems. Similar to other large scale techniques they use variations of nearest neighbor methods to leverage the information contained in large data sets.

2.2 Analyzing and Searching Videos

In contrast to static images, working with video provides a fresh set of opportunities as well as new challenges. Video carries additional modalities of information including motion cues, trajectories, temporal structure, and audio. These additional data streams are rife with useful, search-relevant information, but they are also very difficult to model. While audio is an important element of video we will focus our discussion and experiments here on visual features.

2.2.1 Adapting Methods for Static Imagery to Video

One way to obtain features for video annotation is to directly adapt techniques developed for static image annotation. For example, [14] extends and adapts the initial static image annotation approach presented in Jeon *et al.* [22] to create what they call multiple bernoulli relevance models for image and video annotation. In this approach, a substantial time savings is realized by using a fixed sized grid for feature computations as opposed to relying on segmentations as in [22] and [10]. The fixed number of regions also simplifies parameter estimation in their underlying model and makes models of spatial context more straightforward. To apply their method to video they simply apply their model for visual features within rectangular regions to the keyframes of a video. They compute 30 feature vectors for each rectangular region consisting of: 18 color features (including region color average, standard deviation and skewness) and 12 texture features consisting of Gabor energy computed over 3 scales and 4 orientations).

The underlying multiple bernoulli relevance model consists of a kernel density estimate for the features in each region conditioned on the identity of the video and a multivariate bernoulli distribution over words, also conditioned on the identity of the video. As we shall see shortly, when we seek to use a kernel density type of approach for extremely large datasets such as those produced by large video collections, we must use some intelligent data structures and potentially some approximations to keep computations tractable. The authors of [14] also argue that their underlying

bernoulli model for annotations is more appropriate for image keyword annotations where words are not repeated compared to the multinomial assumptions used in their earlier work [22]. The experimental analysis of the multiple bernoulli model of [14] used a subset of the NIST Video Trec dataset [34]. Their dataset consisted of 12 MPEG files, each 30 minutes long from CNN or ABC including advertisements. There were 137 keywords in the annotation vocabulary and they found their model produced a mean average precision of .29 for one word queries.

The "Video Google" work of Sivic and Zisserman [40] is representative of a different approach to video retrieval based more on object recognition and SIFT techniques. The approach allows for searches and localizations of all the occurrences of a user outlined object in a video. Sivic and Zisserman compute two different types of regions for video feature descriptors. The first, referred to as a Shape Adapted (SA) region, is constructed by elliptical shape adaptation around an interest point. The ellipse center, scale and shape are determined iteratively. The scale is determined from the local extremum across scale of a Laplacian and the shape is determined by maximizing the intensity gradient isotropy over the region. The second type of region, referred to as a Maximally Stable (MS) region, is determined from an intensity watershed image segmentation. Regions are identified for which the area is stationary as the intensity threshold is varied. SA regions tend to be centered on corner like features and MS regions correspond to blobs of high contrast with respect to surroundings. Both regions are represented as ellipses and for a 720×576 pixel video frame one typically has 1600 such regions. Each type of region is then represented as a 128 dimensional SIFT descriptor. Regions are tracked through frames and a mean vector descriptor is computed for each of the regions. Unstable regions are rejected giving about 1000 regions per frame. A shot selection method is used to cover about 10,000 frames or 10% of the frames in a typical feature length movie resulting in a data set of 200,000 descriptors per movie. A visual vocabulary is then built by K-means based clustering. Using scenes represented in this visual vocabulary they use the standard term frequency-inverse document frequency or *tf-idf* weighting and the standard normalized scalar product for computation for retrieval. This approach produced impressive query by region demonstrations and results and while it was not directly designed for the problem of video annotation, the approach could easily be adapted by transferring labels from matching videos.

2.2.2 More Video Specific Methods

We now turn our attention to techniques much more specifically designed for video. Certainly it is the spatio-temporal aspect of video that gives video feature computations their distinctive character compared to techniques designed for static imagery. The particular task of human activity recognition frequently serves as a motivating problem. Early work on activity recognition analyzed the temporal structure of video and built a table of motion magnitude, frequency, and position within a segmented figure [35] or involved building a table with the presence or recency of motion at each location in an image [4]. Of course, highly specific approaches for activity recognition can use fairly detailed and explicit models of motions for

activities to be recognized. These techniques can be very effective, but by their nature, they cannot offer general models of the information in video in the way that less domain-specific features can.

Recent developments in video feature extraction have continued to be strongly influenced by activity recognition problems and have been largely based on local spatio-temporal features. Many of these features have been inspired by the success of SIFT like techniques and the approach of Sivic and Zisserman [40] described previously is an early example. Similar to the SIFT approach, a common strategy for obtaining spatio-temporal features is to first run an interest point detection step. The interest points found by the detector are taken to be the center of a local spatial or spatio-temporal patch, which is extracted and summarized by some descriptor. Frequently, these features are then clustered and assigned to words in a codebook, allowing the use of bag-of-words models from statistical natural language processing for recognition and indexing tasks. Considerable recent work in activity recognition has focused on these types of bag-of-spatio-temporal-features approaches, often explicitly cast as generalizations of SIFT features. These techniques have been shown to be effective on small, low resolution (160x120 pixels per frame) established datasets such as the KTH database [38] with simple activities such as people running or performing jumping jacks. Recent extensions of the space-time cuboid approach [23] have been applied to learn more complex and realistic human actions from movies using their associate scripts. This work has sought to identify complex actions like answering a phone, getting out of a car or kissing. This work also emphasizes the importance of dealing with noisy or irrelevant information in the text annotation or associated movie script.

Features based on space-time cuboids have certain limits on the amount of space and time that they can describe. Human performance suggests that more global spatial and temporal information could be necessary and sufficient for activity recognition. In some of our own recent research [30] we have proposed and evaluated a new feature and some new models for recognizing complex human activities in higher resolution video based on the long-term dynamics of tracked key-points. This approach is inspired by studies of human performance recognizing complex activities from clouds of moving points alone.

2.3 TRECVID

TRECVID [41] is an ongoing yearly competitive evaluation of methods for video indexing. TRECVID is an important evaluation for the field of video search as it coordinates a rigorous competitive evaluation and allows the community to gauge progress. For these reasons we briefly review some relevant elements of TRECVID here and discuss some recent observations and developments. More details about the 2008 competition are given in [34].

One of the TRECVID tasks is to identify “high level features” in video. These features can be thought of as semantic concepts or annotation words in terms of

our ongoing discussion. The following concepts were used for the 2008 evaluation: classroom, bridge, emergency vehicle, dog, kitchen, airplane flying, two people, bus, driver, cityscape, harbor, telephone, street, demonstration or protest, hand, mountain, nighttime, boat or ship, flower, singing. Given the visual concepts and a common shot boundary reference, for each visual concept evaluators return a list of at most 2000 shots from the test collection, ranked according to the highest possibility of detecting the presence of the visual concept (or feature in the TRECVID language). In 2004, Hauptmann and Christel [19] reviewed successful past approaches to the challenge. Their conclusions were that combined text analysis and computer vision methods work better than either alone; however, computer vision techniques alone perform badly, and feature matching only works for near-duplicates. It is interesting to note that this supports the notion that with a much larger corpus there is a much better chance of finding near duplicates. In contrast to TRECVID, in our own experiments that we present at the end of this chapter we are interested in dramatically scaling up the amount of data used to millions of videos as well as extending the annotation vocabulary to a size closer to the complete and unrestricted vocabulary of words used on the web. The evaluation for this type of scenario poses the additional real-world challenge of working with noisy web labels.

2.4 The Web, Collaborative Annotation and YouTube

The web has opened up new ways to collect, annotate store and retrieve video. Various attempts have been made to solve the annotation problem by allowing users on the web to manually outline objects in imagery and associate a text annotation or word with the region. The LabelMe project [36] represents a prominent and representative example of such an approach. In contrast, Von Ahn *et al.* have developed a number of games for interactively labeling static images [45, 46]. These methods are attractive because the structure of the game leads to higher quality annotations and the annotation process is fun enough to attract users. These projects have been successful in obtaining moderate amounts of labeled data for annotation experiments; however, the rise of YouTube has opened up new opportunities of unprecedented scale.

YouTube is the world's largest collection of video. In 2008 it was estimated that there are over 45,000,000 videos in the YouTube repository and that it is growing at a rate of about seven hours of video every minute [2]; in 2009 that rate was measured at twenty hours of video per minute. Despite this there has been relatively little published research on search for YouTube. Some recent research as examined techniques for propagating preference information through a three-month snapshot of viewing data [2]. Other works have examined the unique properties of web videos [48] and the use of online videos as training data [44]. In contrast, we are interested in addressing the video annotation problem using a combination of visual feature analysis and a text model. YouTube allows an uploader to associate a substantial text annotation with a video. In our own experimental analysis we will use the title of the video and this annotation as the basis of the text labels for the video. While there are also facilities for other users to comment on a video, our initial observations were

that this information was extremely noisy. We thus do not use this information in our own modeling efforts. In the following section we will discuss some of the key technical challenges when creating a solution to the video annotation problem for a video corpus the size of YouTube.

3 Web-Scale Computation of Video Similarity

Any technique for automatic annotation on YouTube must be designed to handle vast amounts of data and a very large output vocabulary. With traditional classification approaches, a new classifier must be trained for each distinct word in the vocabulary (to decide how likely that particular word is to be an appropriate label for that video). Clearly, this approach cannot support a vocabulary size in the hundreds of thousands. In our scenario, retrieval based approaches offer an appealing alternative, since one similarity function can be used to transfer an unbounded vocabulary. These issues motivate the following discussion on nonparametric approaches to computing visual similarity. The methods detailed in this section are designed for scalability and we focus on those that have been applied experimentally in the literature to large scale image and video problems.

We begin this section by considering the fundamental benefits and drawbacks of shifting toward web-scale visual datasets. Next we examine current nonparametric techniques for computing visual similarity. Noting that our problem setting of YouTube analysis requires extensive computational resources, we conclude the section with a discussion of the MapReduce framework as it is an integral component of the implementation of our proposed methods.

3.1 Working with Web-Scale Datasets

It has only recently become possible to attempt video search and mining on YouTube sized datasets. Besides the steep computational resources required, the most obvious impediment has been the collection of such a massive and diverse pool of videos. The data collection problem has been conveniently solved by social media websites coupled with contributions from millions of web users. Now that working with web-scale video data is a real possibility, we must consider whether it is a worthwhile endeavor.

Firstly, there are clear drawbacks to working with massive online video collections. It is difficult and computationally costly to process even moderately sized video datasets. The cost is not always justified when existing datasets including TRECVID remain adequately challenging for state of the art research. Furthermore, web data suffers from quality control problems. Annotations of online videos are notoriously incomplete. Working with TRECVID avoids this issue by providing handcrafted annotations and carefully constructed categories. The goals of the TRECVID challenge are clearly defined making direct comparative performance evaluations easy. In contrast, noisily annotated online media have an unbounded and incomplete annotation vocabulary, making performance evaluation difficult. When measuring

annotation accuracy of online videos and treating user annotations as ground truth, false positives and false negatives may be tallied incorrectly due to mistakes inherent in the ground truth labels.

There are, however, a number of compelling reasons to focus research on web-scale video datasets. The most immediate reason is that there is suddenly a practical need for indexing and searching over these sets. Online video sites wish to have all of their videos properly accessible by search. When videos are uploaded without good metadata, those videos effectively become invisible to users since they cannot be indexed by traditional search engines.

Working with real world web datasets provides new opportunities for studying video annotation using large unbounded vocabularies. Most search terms posed to sites like YouTube do not correspond to the limited category words used in TRECVID; it is of practical importance to be able to annotate videos with a much larger pool of relevant words that approach the natural vocabulary for describing popular videos on the web. As we shall see in our experiments, an automatically derived vocabulary from YouTube tags looks quite different from the TRECVID category list.

Finally, a number of recent works in the image domain have shown promising results by harnessing the potential of web image collections [42, 6, 20]. New avenues of research have become possible simply from the growth and availability of online imagery, and we expect the same to be true of online video.

3.2 Scalable Nearest-Neighbors

Quickly finding neighboring points in high dimensional space is a fundamental problem that is commonly involved in computing visual similarity. The k nearest neighbor problem is defined on a set of points in d-dimensional vector space R^d, where the goal is to find the k nearest points to a query vector under some distance metric. Euclidean distance is the most common and we use it exclusively in the following sections. We examine 3 of the most prominent methods for approximate nearest neighbor search which are well suited to large scale video analysis problems. Here the focus is on search algorithms, keeping in mind that most of the feature representations discussed earlier can be substituted into these procedures.

We begin with vocabulary trees which are an extension of the visual vocabulary method of quantizing and indexing visual features. Vocabulary trees (and visual vocabularies) are usually not branded as nearest neighbor methods; however, we observe that as the number of nodes in a tree increases, feature space is quantized at finer levels until the result resembles a nearest neighbor search.

Locality sensitive hashing (LSH) and spill-trees are also surveyed. These methods are true approximate nearest neighbor search structures with nice theoretical performance guarantees, and spill-trees are particularly useful as they are easy to implement in a computing cluster.

3.2.1 Vocabulary Trees

The vocabulary tree method of Nister and Stewenius [32] is an efficient way of partitioning a vector space and searching over the partitions. The algorithm amounts to a hierarchical k-means clustering and proceeds as follows. In the first step, all of the data points are clustered using k-means. If there are many points a sampling of them may be used in this step. Each point is assigned to its nearest cluster center, then all points belonging to the individual centers are clustered using k-means again. The process is repeated recursively until the number of points assigned to leaf clusters in the tree is sufficiently small. The shape of the tree is controlled by k, the branching factor at each node.

The tree is queried by recursively comparing a query vector to cluster centers using depth first search. At each level of the tree k dot products are calculated to determine the cluster center closest to the query. [32] use an inverted file approach [47] to store images in the tree for efficient lookup. Local image feature vectors are computed on the image database to be searched, and these vectors are pushed down the tree. Only the image identifier and the count of local features reaching each node are stored in the tree; the feature vectors need not be stored resulting in vast space savings.

A downside to this approach is inaccuracy incurred by the depth first search procedure, which does not guarantee that the closest overall leaf node will be found. Schindler et al. [37] call this the Best Bin First (BBF) algorithm and propose Greedy N-Best Paths (GNP) as a solution. GNP simply allows the N closest cluster centers to be traversed at each level of the tree, thus increasing the computation by a factor of N and providing an easy tradeoff between accuracy and efficiency.

Generally, performance on retrieval tasks using vocabulary trees is observed to improve as the size of the vocabulary increases. In [32] and [37] trees with as many as 1 million leaf nodes were used. We can view the vocabulary tree with GNP search as a nearest neighbor method that searches for the N closest vocabulary cluster centers. As the vocabulary size approaches the number of data points, results closely approximate those of traditional nearest neighbor search over the data.

3.2.2 Locality Sensitive Hashing

Relaxing the goal of finding a precise set of nearest neighbors to finding neighbors that are sufficiently close allows for significant computational speedup. The $(1+\varepsilon)-NN$ problem is presented by [16] as follows: for a set of points P and query point q, return a point $p \in P$ such that $d(q,p) \leq (1+\varepsilon)d(q,P)$, where $d(q,P)$ is the distance of q to the closest point in P. Early solutions to this problem fall under the category of locality sensitive hashing (LSH).

LSH is an approach wherein a hash function is designed to map similar data points to the same hash bucket with high probability while mapping dissimilar points to the same bucket with low probability. By choosing an appropriate hash function and hashing the input multiple times, the locality preserving probability can be driven up at the expense of additional computation. LSH has been applied

successfully to visual data in numerous works, cf. [39, 16]. However, we choose to focus on spill trees since they are particularly easy to implement in MapReduce and are known to perform well in similar large scale situations [25]. For further details on LSH we refer to the survey by Andoni et al. [1].

3.2.3 Spill Trees

Liu et al. [24] present the spill tree as a fast, approximate extension to the standard nearest neighbor structures of k-d, metric, and ball trees. The k-d tree [15] is an exact solution where the vector space is split recursively and partition boundaries are constrained along the axes. Search in a k-d tree proceeds by traversing the tree depth-first, backtracking and skipping nodes whose spatial extent is outside the range of the nearest point seen so far during the search. Metric and ball trees [43, 33] follow the same design but allow for less constrained partitioning schemes.

Spill trees build on these traditional nearest neighbor structures with a few critical enhancements. Most importantly, the amount of backtracking during search is reduced by allowing the partitioned regions at each node in the tree to overlap. Using an *overlap buffer* means that points near to the partition boundary of a node will be included in both of its children. Points in the overlap buffer are the ones most frequently missed during depth first search of the tree. When a query point is close to the partition boundary for a node, its actual nearest neighbor may fall arbitrarily on either side of the boundary. Backtracking during search is no longer needed to recover points that fall in the overlapping regions. This partitioning scheme is detailed in Figure 2.

Efficient search in tree-based nearest neighbor structures results from splitting the points roughly evenly at each node, leading to logarithmic depth of the tree. If

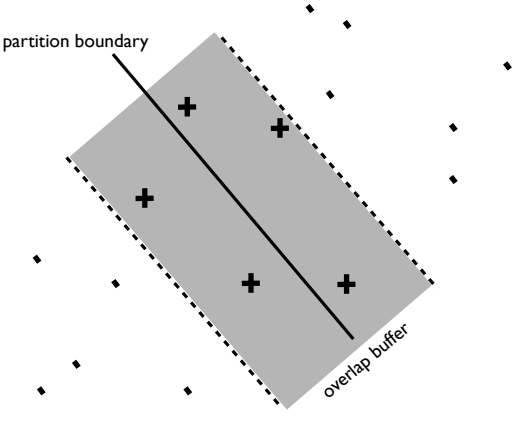


Fig. 2 Partitioning at a node in the spill-tree with an overlap buffer. A buffer area is placed around the partition boundary plane and any points falling within the boundary become members of both children of the node.

the spill tree overlap buffer is too large, each child of a node will inherit all data points from the parent resulting in a tree of infinite depth. This motivates another key enhancement of the spill tree: nodes whose children contain too many points are treated as standard metric tree nodes, having their overlap buffers removed and backtracking enabled. The hybrid spill tree becomes a mix of fast overlap nodes requiring no backtracking and standard nodes where backtracking can be used during search. Experimental comparisons show that spill trees are typically many times faster than LSH at equal error rates for high dimensional data [24].

3.3 MapReduce

The algorithms we've described for finding neighboring vectors must scale to vast amounts of data to be useful in our setting. Since storage for reference points is expected to surpass the total memory of modern computers, we turn our attention to issues of distributed computing. Specifically, we examine the MapReduce framework and explain how it is used to implement a distributed version of spill trees.

MapReduce is a parallel computing framework that abstracts away much of the difficulty of implementing and deploying data-intensive applications over a cluster of machines [9]. A MapReduce program is comprised of a series of 3 distinct operations:

1. **Map.** The map operation takes as input data a set of key-value pairs, distributing these pairs arbitrarily to a number of machines. Data processing happens locally on each machine, and for each data item one or more output key-value pairs may be produced.
2. **Shuffle.** The key-value pairs output by **map** are grouped under a new key by a user-defined **shuffle** operation. This phase is typically used to group data that must be present together on a single machine for further processing.
3. **Reduce.** The grouped key-value pairs from **shuffle** are distributed individually to machines for a final processing step, where one or more key-value pairs may be output.

Each phase of a MapReduce program can be performed across multiple machines. Automatic fault tolerance is built in so that if a machine fails, the work assigned to it is simply shifted to a different machine on the computing cluster.

Liu et al. [25] developed a distributed version of the spill tree algorithm using MapReduce, and applied it to the problem of clustering nearly 1.5 billion images from the web. We sketch the MapReduce algorithm for constructing and querying the distributed spill tree here, as we have used this approach extensively in our experiments in the following sections. The algorithm begins by building a root tree from a small uniform sampling of all data points. This tree is then used to bin all of the points in the dataset by pushing them down the tree to the leaf nodes. Each leaf node can then be constructed as a separate spill-tree running on a separate machine. In this way, the destination machine for a query point is first quickly determined by the root tree, then the bulk of the computation can be done on a separate machine dedicated to the particular subtree nearest to the query point.

In the Map step of a batch query procedure where there are multiple points to be processed, the query points are distributed arbitrarily to a cluster of machines. Each machine uses a copy of the root tree to determine the nearest leaf trees for input query points. Then, the Shuffle operation groups query points by their nearest subtrees. In the Reduce phase, one machine is given to each subtree to process all the points that were assigned to it. This procedure replicates the original spill tree algorithm with the minor restriction that no backtracking can occur between nodes of the subtrees and the root tree.

4 A Probabilistic Model for Label Transfer

In this section we develop a generative probabilistic model for label transfer that is designed for scalability. Rather than constructing classifiers for a fixed set of labels, our approach operates on a much larger pool of words that are drawn from existing annotations on a video corpus. We extend the features used in the JEC method of image annotation [28] for use with video shots, and use a distributed spill-tree to inform the model of neighboring shots within the corpus. Each step of our approach is implemented in MapReduce for scalability and efficiency.

In this model new annotations are computed based on (1) visual similarity with a pool of pre-annotated videos and (2) co-occurrence statistics of annotation words. We begin with an overview of the generative process, then detail an approximate inference algorithm based on nearest neighbor methods. It is important to note that the structure of the probabilistic model has been designed specifically to take advantage of the efficiency of our distributed spill tree implementation on Google's MapReduce infrastructure.

4.1 Generating New Video Annotations

Suppose we have a collection of N videos where each video V has a preexisting set of N_{V_w} text labels $\{w\}$. Given a query video V with an incomplete set of labels our goal is to predict the most likely held-out (unobserved) label, denoted w_h.

We have two sources of information with which we can make our prediction:

1. Occurrence and co-occurrence statistics of labels. We compute $p(w_i|w_j)$ for all words i, j over the entire set of annotated videos. The co-occurrence probabilities suggest new annotation words that are seen frequently in the video corpus with any of the existing annotation words for the query video.
2. Visual similarity between videos. Using nearest-neighbor methods to find similar video shots, we compute the probability of a query shot belonging to each video in the corpus. Under the assumption that visually similar videos are likely to have the same labels, a high probability of a query shot being generated by corpus video V increases the chance of transferring one of V's annotations to the query.

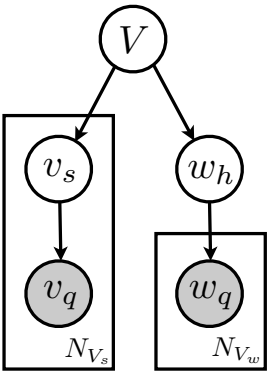


Fig. 3 A generative graphical model for automated annotation, with corpus videos V at the root node.

The graphical model depicted in Figure 3 defines the generative process for videos, shots, and annotations. We begin by considering each variable in the figure:

- V is an index variable for a video in the training set.
- v_s is a single shot from the video V. Thus V is described by an unordered collection of N_{V_s} video shots.
- v_q is an observed video shot feature vector belonging to the query.
- w_h represents a held-out annotation word for the query video, and is the hidden variable we wish to predict.
- w_q is an observed annotation word for the query video.

With these definitions in place we can list the generative process:

1. Pick V, a single video from the corpus.
2. Sample N_s shots from V.
3. Generate an observation feature vector v_q for each shot v_s.
4. Sample an annotation word w_h from V.
5. Generate a set of co-occurring annotation words $\{w_q\}$ conditioned on w_h.

In the model only the variables involving the query video, $\{v_q\}$ and $\{w_q\}$, are observed. In order to perform exact inference on w_h, we would sum over all shots $\{v_s\}$ of all videos V in the corpus. In this setting, the joint probability of an observation consisting of $(\{v_q\}, \{w_q\}, w_h)$ is:

$$p(\{v_q\}, \{w_q\}, w_h) = \sum_V \left\{ p(V) \prod_{v_q} \left[\sum_{v_s} p(v_s|V) p(v_q|v_s) \right] \times p(w_h|V) \prod_{w_q} p(w_q|w_h) \right\}. \quad (1)$$

We choose the most likely assignment to unobserved variable w_h as

$$\begin{aligned} w_h^* &= \arg\max_{w_h} \frac{p(\{v_q\},\{w_q\},w_h)}{p(\{v_q\},\{w_q\})} \\ &\propto \arg\max_{w_h} p(\{v_q\},\{w_q\},w_h) \end{aligned} \tag{2}$$

since $p(\{v_q\},\{w_q\})$ remains constant for a given query video. The model can be called *doubly nonparametric* due to the summation over V and over each shot v_s of V, a trait in common with the image annotation model of [14] where there is a summation over images and image patches.

We consider each factor of Equation 1:

- $P(V) = \frac{1}{N}$ where N is the number of videos in the training set. This amounts to a uniform prior such that each training video is weighted equally.
- $p(v_s|V) = \frac{1}{N_{V_s}}$ if v_s belongs to video V, 0 otherwise.
- $p(v_q|v_s) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(v_q-\mu_{v_s})^2}{2\sigma^2} \right)$, a normal distribution centered on v_s with uniform variance σ. The probability of query shot v_q being generated by corpus shot v_s falls off gradually as the distance between their feature vectors increases.
- $p(w_h|V) = \frac{1}{N_{V_w}}$ for words belonging to the annotation set of V, and is set to 0 for all other words.
- $p(w_q|w_h)$ is determined by the co-occurence statistics of words w_q and w_h computed from the training set. $p(w_q|w_h) = \frac{N(w_q,w_h)}{N(w_h)}$, where $N(w_q,w_h)$ is the number of times w_q and w_h appear together in the video corpus, and $N(w_h)$ is the total occurrence count of w_h.

Substituting these definitions yields

$$p(\{v_q\},\{w_q\},w_h) = \sum_V \left\{ \frac{1}{N} \prod_{v_q} \left[\sum_{v_s \in V} \frac{1}{N_{V_s}} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(v_q-\mu_{v_s})^2}{2\sigma^2} \right) \right] \times \frac{N(w_h \in V)}{N_{V_w}} \prod_{w_q} \frac{N(w_q,w_h)}{N(w_h)} \right\}. \tag{3}$$

We must carefully handle smoothing in this model. The shots of each V are treated as i.i.d. and as such, if any shot has zero probability of matching to the query video, the entire video-video match has probability zero. In most cases, similar videos share many visually close shots, but inevitably there will be some shots that have no match, resulting in $p(v_q|v_s) = 0$ in the product over s in Equation 3. To alleviate this problem we assign a minimal non-zero probability ε to $p(v_q|v_s)$ when the computed probability is close to zero. ε can be tuned using a held-out validation set to achieve a proper degree of smoothing. Equation 3 becomes:

$$P(\{v_q\},\{w_q\},w_h) = \sum_V \left\{ \frac{1}{N} \prod_{v_q} \left[\sum_{v_s \in V \wedge v_s \in NN_{v_q}} \frac{1}{N_{V_s}} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(v_{q,s}-\mu_{v_s})^2}{2\sigma^2} \right) + \sum_{v_s \in V \wedge v_s \notin NN_{v_q}} \frac{1}{N_{V_s}} \varepsilon \right] \times \frac{N(w_h \in V)}{N_{V_w}} \prod_{w_q} \frac{N(w_q, w_h)}{N(w_h)} \right\}. \quad (4)$$

4.2 Nearest Neighbor for Scalability

Performing exact inference by summing over all shots of every corpus video is prohibitively expensive. However, we know that for a given query, most corpus shots are dissimilar and contribute little or nothing to the computation by virtue of our definition of $p(v_q|v_s)$. In fact, the only corpus shots that matter for a query are the ones that are nearest to the query shot vectors.

This scenario is perfect for application of the approximate nearest neighbor methods discussed earlier. We choose the distributed spill-tree structure for finding neighboring shots in our system. Spill trees operate on data points within relatively low dimensional vector space, and this motivates our choice of feature representation for shot matching. We opt to use modified versions of the JEC features [28] which were discussed in Section 2.1. These image features include LAB and HSV global color histograms, and the Haar and Gabor wavelets. The features are modified for video shots by computing them at evenly spaced frames throughout a shot and concatenating the results. This representation retains the simplicity and good performance of JEC features, and it also captures some of the temporal information in each shot.

Our proposed shot feature vectors are very high dimensional, typically one to two orders of magnitude larger than JEC features depending on how many frames are sampled from each shot. In order to generate a single summary vector per shot, the vectors from each feature modality are concatenated, resulting in an even larger vector. Since the spill tree requires a low dimensional representation, the shot vectors are reduced to 100 dimensions by random projection [3]. Random projection is simple, fast, and is known to preserve neighborhood structure [8] making it a good choice for preprocessing nearest neighbor data. The complete procedure for preparing the nearest neighbor component of the model is given by Figure 4.

Working from Equation 1, we distinguish between videos V_{NN} which have at least one nearest neighbor shot discovered by the spill tree, and videos $V_{\overline{NN}}$ which do not have any:

$$p(\{v_q\},\{w_q\},w_h) = \sum_{V_{NN}} \left\{ p(V_{NN}) \prod_{v_q} \left[\sum_{v_s} p(v_s|V) p(v_q|v_s) \right] \times p(w_h|V_{NN}) \prod_{w_q} p(w_q|w_h) \right\} + \sum_{V_{\overline{NN}}} \left\{ p(V_{\overline{NN}}) \prod_{v_q} \left[\sum_{v_s} \frac{\varepsilon}{N_{V_s}} \right] \times p(w_h|V_{\overline{NN}}) \prod_{w_q} p(w_q|w_h) \right\}. \quad (5)$$

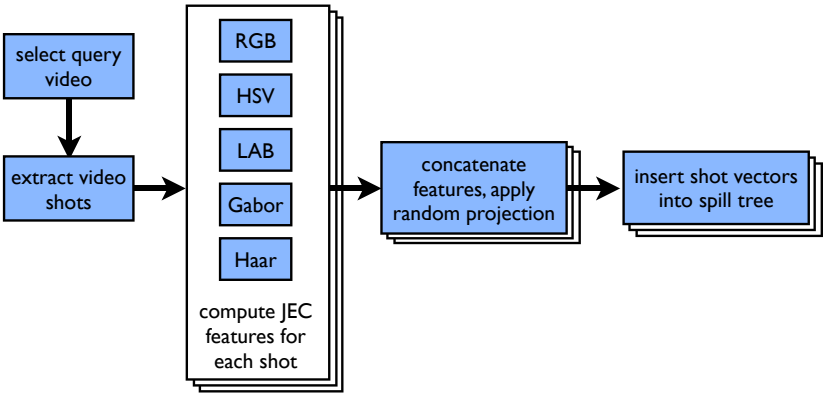


Fig. 4 Procedure for preparing videos for shot-based nearest neighbor lookup.

The distance to each of the N_{V_s} shots of a $V_{\overline{NN}}$ corpus video is approximated by ε. Notice that all terms of Equation 5 involving visual features of corpus videos without nearest-neighbor shots ($\{V_{\overline{NN}}\}$) reduce to a single constant, hereafter denoted λ. At this point, we make a simplifying approximation to completely remove dependence on all videos $\{V_{\overline{NN}}\}$ by substituting the prior word probability $p(w_h)$ for $p(w_h|V_{\overline{NN}})$. With this simplification, the model requires word counts and distances to features for only the small subset of corpus videos that have neighboring shots to the query, as determined by the distributed spill tree. Thusly, the spill tree provides not only the distance information to nearby points but also information about which small subset of corpus videos is relevant in answering the query.

Combining these observations with Equation 4, we arrive at the complete annotation likelihood for our model:

$$
\begin{aligned}
&P(\{v_q\},\{w_q\},w_h) = \\
&\quad \sum_{V_{NN}} \left\{ \frac{1}{N} \prod_{v_q} \left[\sum_{v_s \in V_{NN} \wedge v_s \in NN_{v_q}} \frac{1}{N_{V_s}} \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(v_{q,s}-\mu_{v_s})^2}{2\sigma^2} \right) + \right.\right. \\
&\quad \left.\left. \sum_{v_s \in V_{nn} \wedge v_s \notin NN_{v_q}} \frac{\varepsilon}{N_{V_s}} \right] \times \frac{N(w_h \in V_{NN})}{N_{V_{NN,w}}} \prod_{w_q} \frac{N(w_q,w_h)}{N(w_h)} \right\} + \\
&\quad \sum_{V_{\overline{NN}}} \left\{ \lambda\, p(w_h) \prod_{w_q} \frac{N(w_q,w_h)}{N(w_h)} \right\}.
\end{aligned}
\tag{6}
$$

For a candidate annotation w_h we have a likelihood that is a weighted combination of the prior over w_h and the conditionals $p(w_h|V_{NN})$ based on nearest-neighbor visual similarity. The balance between these elements depends on ε, and we note that the only free parameters of the model (besides ones belonging to low level video features and spill tree computation) are ε and σ, both of which can be tuned by cross validation.

4.3 Alternative Models

The annotation model presented in the previous section is scalable, principled, and easy to implement, but it is only one of many potential solutions to the video annotation problem. Here we consider a few alternative formulations based on the original model that deserve further consideration.

The original model in Figure 3 incorporates information from all videos for which there is a neighboring shot to the query. A simpler approach is to transfer all labels from the single closest video, which is determined by:

$$V^* = \arg\max_V p(V) \prod_{v_q} \left[\sum_{v_s} p(v_s|V) p(v_q|v_s) \right]. \tag{7}$$

This *1-nn* approach, while simplistic, has been shown to provide state of the art results for image annotation [28]. Taking the idea further, the single-best annotation could be selected for each query shot individually, rather than for a complete query video comprised of a collection of shots. Then, the most likely annotation is selected from the query shot with the best match to the corpus:

$$V^* = \arg\max_{V, v_q, v_s} p(V) p(v_s|V) p(v_q|v_s). \tag{8}$$

In a slightly different approach, we can consider a model that computes annotation probabilities directly by assigning words to corpus shots and abandoning the concept of corpus documents. The model presented in the previous section is document-centric in the sense that the root node V of the model is a corpus document. It was shown in [5] that for image classification tasks, category-based nearest-neighbor models outperform instance-based models. In a category-based approach the unobserved root node of the graphical model indexes annotation words rather than instances in the dataset. We can apply this idea to our task of video annotation and arrive at the model depicted in Figure 5. Computing the annotation likelihood becomes:

$$p(\{v_q\}, \{w_q\}, w_h) = p(w_h) \left\{ \prod_{v_q} \left[\sum_{v_s \in V_{w_h}} p(v_s|w_h) p(v_q|v_s) \right] \times \prod_{w_q} p(w_q|w_h) \right\}. \tag{9}$$

Intuitively, this annotation-centric model should improve upon the document-centric one when annotation words correlate strongly with individual shots rather than entire videos. This is a property of the dataset that is likely to vary between different videos and annotation words. Automatic model selection among the alternatives presented here is likely to enhance performance.

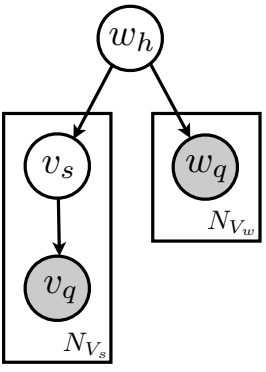


Fig. 5 An alternative graphical model for automated annotation with a hidden annotation word as the root node.

5 Experiments

Here we present preliminary results of testing our system on a large portion of the YouTube corpus. Our goal is to demonstrate the feasibility of methods detailed in the previous section under the emerging real-world scenario of extremely large online video collections.

Approximately 1.2 million of the most popular YouTube videos were selected for the experiments. From these videos an annotation vocabulary of 1000 words was formed by filtering words taken from titles, descriptions, and uploader tags through common stoplists. An initial set of annotations for each video was generated by filtering the metadata through the annotation vocabulary. Then, for each video we computed shot boundaries and JEC features for each shot. The pool of features from all 1.2 million videos was used to construct a distributed spill tree using MapReduce.

We note that the spill tree must be kept in memory to retain its efficiency, and for large video collections this can only be accomplished by distributing the tree across multiple machines. If each 100-dimensional shot vector consumes 400 bytes of data, we can store at most approximately 10 million shots on a machine with 4GB ram. Typical videos in YouTube are found to have 20-40 shots by our shot boundary detector, further increasing memory requirements and necessitating the use of MapReduce to distribute the load. The complete preparation procedure for our experiments is shown in Figure 6.

Testing proceeded using the model defined by Figure 3 and the procedure shown in Figure 7. Each existing annotation word for each video was withheld in turn, and for each a new list of all possible annotations was generated, ordered according to the likelihoods computed using Equation 2. Annotation words were held out one at a time because the remaining words for the query video provide co-occurrence information to the inference procedure.

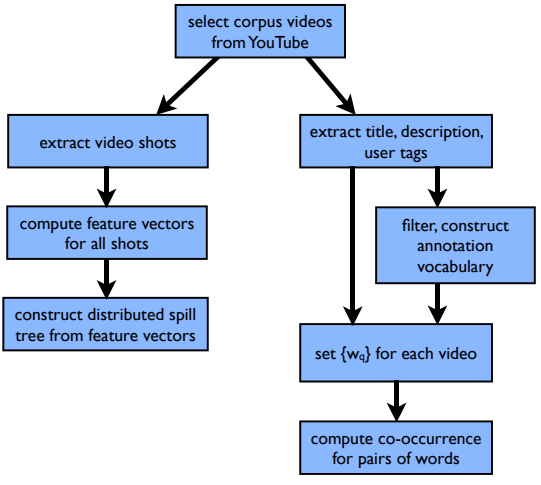


Fig. 6 Preprocessing steps for our YouTube labeling experiments. The bulk of the computation time and resources in this procedure is consumed by the feature computation step. Each step is implemented as a separate distributed program using the MapReduce framework.

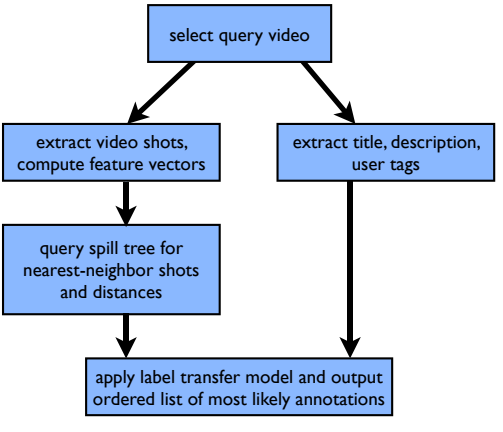


Fig. 7 Steps for performing new annotation inference on a YouTube video.

When developing an evaluation metric for our results, we note that our approach is designed explicitly for generating annotations and is not synonymous with video retrieval. Rather than computing a list of videos for a particular annotation word, we generate an ordered list of most likely annotations for each query video. As such, the standard retrieval measure of precision-recall is not appropriate for our task.

We consider a closely related measure instead: for an annotation list position and a particular annotation word, we measure the probability of that word being correctly assigned *at precisely that list position*. Probability of correctness for word

w at list position n is computed by counting the number of times w is observed in the generated annotation lists at position n for all query videos having w as a held out word, divided by the total number of occurrences of w at position n for all query videos. Formally, our measure is computed as:

$$Prob_w^n = \frac{\sum_{V:w \in V_w} I_V^{w,n}}{\sum_V I_V^{w,n}} \tag{10}$$

where $I_V^{w,n}$ is 1 if w appears on V's new annotations list at position n, and is 0 otherwise.

As an example, consider our measure for the annotation word "music" *at list position 1*. We count all the times "music" is the first annotation word returned for all videos where "music" is the correct word currently being held out, and divide by the total number of times "music" is returned as the first annotation for all videos in the corpus. If this ratio is close to 1, we conclude that the system accurately assigns the word "music" when it is the first annotation word returned for a query. In contrast to more conventional cumulative measures, our approach displays a conditional measurement: we are computing the probability of a returned annotation word equalling the held-out annotation word, *given that it was returned at list position n.* Suppose "music" is recovered at equal rates at positions 1 and 2 when it is the missing held-out word, and it is always recovered at position 1 for videos where it is not a correct annotation. Then, "music" is more likely to be a correct annotation if it is recovered at position 2 rather than position 1.

Figure 8 displays our measure for list positions 1 through 200. Probabilities are computed for each word individually and at each list position using the method described above, then the probabilities at each position are averaged over all annotation words. The resulting curves provide the average word-normalized precision at each position on the list. Better results are indicated by higher average probability *at earlier positions on the list*: this means that held out annotation words are recovered correctly more often and with higher likelihood than other words. If we imagine a usage scenario where just the top 20 new annotations are retained for each query video, then we only care about the probability of these top 20 being correct; the rest of the list is unimportant unless more annotations are desired.

Two variants of the model were tested for comparison. We applied the same test protocol using only the portion of the graphical model involving visual features and excluding word co-occurrence. Similarly, we repeated the procedure using only co-occurrence and excluding vision. We observe in Figure 8 that these model variants exhibit drastically different behaviors at various list positions. Importantly, the combined vision+text model is more than the sum of its parts, showing better average precision than either vision or text alone among the top suggested annotations.

Peak performance reaches 25 percent average precision for the full model, but this does not mean that the majority of annotations generated are necessarily "inappropriate". As we have noted earlier, working with user-annotations rather than

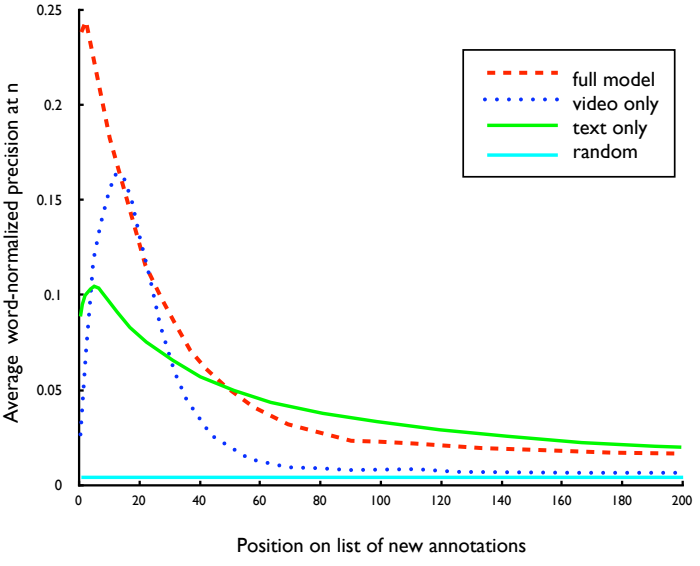


Fig. 8 Average word-normalized precision at each list position using variants of the full model specified in Figure 3.

hand-labeled data makes performance measurement challenging. When user labels are incomplete, a typical characteristic among social media sources including YouTube, our proposed system may generate reasonable annotations that were originally absent. Conversely, incorrect original user annotations that do not relate to the video content may not be predicted by our system, thus penalizing performance for correct behavior.

Table 1 lists the accuracy (recovery rate) of held-out annotations among the top 20 annotations returned for 100 words selected from the vocabulary. In this measurement we have set a cutoff by observing occurrence within the first 20 list positions; a natural alternative is to set a threshold on the returned annotation word probabilities, and we wish to explore this in future work. This data highlights the unique nature of large, automatically generated annotation vocabularies. In contrast to existing methods which use small, carefully constructed category labels, the list of words produced by our method is data-driven and tailored to the dataset. The results in table 1 indicate that our method performs well on an extremely diverse collection of annotation words. Table 2 narrows these results to words that naturally co-occur. One might expect that these words would be predicted entirely by co-occurrence probabilities with their counterparts. We find that the combined model using both text and vision almost always produces the best results, an observation that is shared with other works in image and video annotation [21, 19].

Table 1 Top-20 annotation accuracy for top 100 words of the annotation vocabulary, ordered by the results of the full text+vision model of Figure 3.

#	annotation	vid-only	text-only	**text+vis**	chance	#	annotation	vid-only	text-only	**text+vis**	chance
1	roses	.000	.535	.924	.003	51	birthday	.000	.173	.601	.002
2	daily	.000	.151	.875	.001	52	baseball	.000	.094	.600	.001
3	avril	.000	.712	.810	.002	53	boom	.000	.156	.600	.001
4	totally	.000	.419	.800	.002	54	brawl	.000	.169	.598	.002
5	discussed	.000	.546	.791	.001	55	mac	.000	.265	.597	.002
6	related	.000	.494	.788	.001	56	aguilera	.000	.091	.596	.003
7	montana	.600	.482	.787	.002	57	rangers	.000	.265	.596	.001
8	theft	.000	.433	.783	.001	58	pic	1.000	.514	.595	.002
9	hannah	.100	.445	.781	.003	59	travel	.000	.253	.591	.001
10	rated	.000	.535	.776	.002	60	bull	.429	.180	.591	.001
11	hearts	1.000	.353	.769	.002	61	nba	.560	.313	.583	.004
12	gta	.000	.255	.757	.002	62	pants	.000	.123	.581	.002
13	lavigne	.000	.604	.743	.001	63	william	.083	.145	.575	.002
14	tree	.000	.295	.742	.003	64	eurovision	.000	.130	.571	.001
15	zac	.000	.328	.735	.002	65	pokemon	.789	.356	.570	.004
16	efron	.000	.334	.734	.001	66	hero	.120	.380	.566	.003
17	miley	.000	.332	.730	.001	67	indie	.000	.096	.563	.002
18	viewed	.000	.501	.729	.002	68	ufo	.500	.231	.560	.002
19	knight	.657	.241	.724	.001	69	purely	.118	.130	.558	.002
20	favorites	.000	.461	.723	.002	70	winter	.000	.166	.557	.002
21	advertising	.000	.167	.720	.001	71	expert	.000	.204	.556	.001
22	cnn	.000	.013	.714	.002	72	sunday	.100	.264	.556	.002
23	smash	1.000	.231	.709	.002	73	crap	.000	.086	.556	.002
24	cyrus	.000	.329	.697	.002	74	legendary	.000	.088	.551	.002
25	runescape	.862	.019	.694	.003	75	vanessa	.000	.222	.546	.001
26	chocolate	.000	.183	.693	.002	76	alba	.000	.094	.545	.002
27	potter	.962	.486	.691	.002	77	dutch	1.000	.083	.542	.002
28	lindsay	.000	.188	.688	.001	78	def	.000	.151	.537	.002
29	mexican	1.000	.136	.677	.003	79	trip	.000	.197	.536	.004
30	harry	.684	.552	.668	.003	80	soulja	.000	.250	.531	.001
31	alien	1.000	.232	.667	.002	81	blooper	.000	.143	.530	.002
32	kingdom	1.000	.439	.661	.003	82	manga	.754	.061	.529	.003
33	clinton	.000	.133	.650	.002	83	greece	.000	.103	.529	.002
34	fantastic	.000	.233	.650	.001	84	batman	.000	.244	.526	.001
35	jonas	1.000	.243	.648	.002	85	hill	.000	.390	.524	.003
36	tokio	.000	.285	.647	.002	86	nick	1.000	.207	.524	.004
37	wave	.000	.124	.645	.001	87	student	.000	.189	.523	.001
38	diamond	.000	.367	.637	.002	88	hockey	.500	.139	.520	.003
39	bros	.625	.224	.637	.003	89	fanmade	.077	.116	.517	.002
40	lucky	.667	.279	.636	.001	90	journey	.000	.195	.515	.002
41	sims	.000	.094	.635	.001	91	busty	.000	.195	.513	.001
42	warcraft	.647	.346	.634	.003	92	bang	.512	.113	.510	.002
43	tna	.000	.266	.633	.002	93	store	.000	.167	.508	.001
44	bow	.000	.169	.627	.001	94	foot	.000	.390	.507	.005
45	crank	.000	.291	.622	.001	95	alternative	.000	.113	.496	.002
46	commercials	.000	.103	.622	.002	96	sweden	.000	.068	.494	.002
47	holiday	.000	.134	.615	.000	97	guns	.500	.304	.493	.004
48	lion	.000	.257	.613	.002	98	jamie	1.000	.068	.490	.002
49	obama	.200	.231	.612	.002	99	chapter	.500	.201	.488	.001
50	martial	.000	.230	.607	.002	100	company	.135	.412	.486	.003

Table 2 Top-20 annotation accuracy of selected pairs of words which naturally co-occur together. In most cases the full model combining text and vision achieves the best accuracy.

annotation	vid-only	text-only	**full-model**	chance acc.
guns	**.500**	.304	.493	.004
roses	.000	.535	**.924**	.003
harry	**.684**	.552	.668	.003
potter	**.962**	.486	.691	.002
grand	.308	.381	**.461**	.004
theft	.000	.433	**.783**	.001
gta	.000	.255	**.757**	.002
hannah	.100	.445	**.781**	.003
montana	.600	.482	**.787**	.002
avril	.000	.712	**.810**	.002
lavigne	.000	.604	**.743**	.001

6 Conclusions and Discussion

As online video content continues to grow, automatic tools for annotation are becoming a critical component of web indexing and search. Automated video annotation must explicitly address the issue of scalability, both in terms of the quantity of video and the expansiveness of the annotation vocabulary. Research in video search and mining techniques is progressing rapidly yet most works are limited by small vocabularies and dataset sizes. In this chapter we examined modern, scalable techniques for computing visual similarity, with special emphasis on nearest neighbors and distributed computing.

Based on these observations, we developed a prototype system to enhance web-scale video search with automated text annotation. Our system uses features inspired by recent image annotation work, and it efficiently computes similar video shots using a distributed approximate nearest neighbor technique. A probabilistic model ties visual similarity with annotation co-occurrence statistics to generate new annotation suggestions. Testing the model on a portion of YouTube demonstrates the scalability and efficacy of our approach. Our results indicate that web-scale automated video annotation using large, data-driven vocabularies is feasible and deserves further research exploration.

References

1. Andoni, A., Indyk, P.: Near-optimal hashing algorithms for approximate nearest neighbor in high dimensions. Communications of the ACM 51(1), 117–122 (2008)
2. Baluja, S., Seth, R., Sivakumar, D., Jing, Y., Yagnik, J., Kumar, S., Ravichandran, D., Aly, M.: Video suggestion and discovery for youtube: taking random walks through the view graph. In: WWW 2008: Proceeding of the 17th international conference on World Wide Web, pp. 895–904. ACM, New York (2008)

3. Bingham, E., Mannila, H.: Random projection in dimensionality reduction: applications to image and text data. In: Proceedings of the seventh ACM SIGKDD international conference on Knowledge discovery and data mining, pp. 245–250. ACM, New York (2001)
4. Bobick, A., Davis, J.W.: The recognition of human movement using temporal templates. IEEE PAMI 23, 257–267 (2001)
5. Boiman, O., Shechtman, E., Irani, M.: In defense of nearest-neighbor based image-classification. In: Computer Vision and Pattern Recognition (2008)
6. Crandall, D., Backstrom, L., Huttenlocher, D., Kleinberg, J.: Mapping the world's photos. In: WWW (2009)
7. Dalal, N., Triggs, B.: Histograms of oriented gradients for human detection. Computer Vision and Pattern Recognition 1, 886–893 (2005)
8. Dasgupta, S., Gupta, A.: An elementary proof of the Johnson-Lindenstrauss lemma. Tech. Rep. TR-99-06, Intl. Comput. Sci. Inst. (1999)
9. Dean, J., Ghemawat, S.: MapReduce: Simplified Data Processing on Large Clusters. In: OSDI (2004)
10. Duygulu, P., Barnard, K., de Freitas, J.F.G., Forsyth, D.A.: Object recognition as machine translation: Learning a lexicon for a fixed image vocabulary. In: Heyden, A., Sparr, G., Nielsen, M., Johansen, P. (eds.) ECCV 2002, Part IV. LNCS, vol. 2353, pp. 97–112. Springer, Heidelberg (2002)
11. Everingham, M., Van Gool, L., Williams, C.K.I., Winn, J., Zisserman, A.: The PASCAL Visual Object Classes Challenge (VOC 2008) Results (2008), http://www.pascal-network.org/challenges/VOC/voc2008/workshop/index.html
12. Fei-Fei, L., Fergus, R., Perona, P.: One-shot learning of object categories. IEEE Transactions on Pattern Analysis and Machine Intelligence 28(4), 594–611 (2006)
13. Fei-Fei, L., Perona, P.: A bayesian hierarchical model for learning natural scene categories. Computer Vision and Pattern Recognition 2, 524–531 (2005)
14. Feng, S., Manmatha, R., Lavrenko, V.: Multiple bernoulli relevance models for image and video annotation. In: Computer Vision and Pattern Recognition (2004)
15. Friedman, J., Bentley, J., Finkel, R.: An algorithm for finding best matches in logarithmic expected time. ACM transactions on mathematical software 3(3), 209–226 (1977)
16. Gionis, A., Indyk, P., Motwani, R.: Similarity search in high dimensions via hashing. In: Proc. 25th Internat. Conf. on Very Large Data Bases (1999)
17. Griffin, G., Holub, A., Perona, P.: Caltech-256 object category dataset. Technical Report 7694, California Institute of Technology (2007)
18. Grubinger, M., Clough, P.D., Leung, C.: The iapr tc-12 benchmark for visual information search. IAPR Newsletter 28(2), 10–12 (2006)
19. Hauptmann, A., Christel, M.: Successful approaches in the TREC video retrieval evaluations. In: Proceedings of the 12th annual ACM international conference on Multimedia, pp. 668–675. ACM, New York (2004)
20. Hays, J., Efros, A.: Scene Completion Using Millions of Photographs. ACM Transactions on Graphics (SIGGRAPH 2007) 26(3) (2007)
21. Iyengar, G., Duygulu, P., Feng, S., Ircing, P., Khudanpur, S., Klakow, D., Krause, M., Manmatha, R., Nock, H., Petkova, D., et al.: Joint visual-text modeling for automatic retrieval of multimedia documents. In: Proceedings of the 13th annual ACM international conference on Multimedia, pp. 21–30. ACM, New York (2005)
22. Jeon, J., Lavrenko, V., Manmatha, R.: Automatic image annotation and retrieval using cross-media relevance models. In: SIGIR 2003: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval, pp. 119–126. ACM, New York (2003)

23. Laptev, I., Marszalek, M., Schmid, C., Rozenfeld, B.: Learning realistic human actions from movies. In: IEEE Conference on Computer Vision and Pattern Recognition (2008)
24. Liu, T., Moore, A., Gray, A., Yang, K.: An investigation of practical approximate nearest neighbor algorithms. In: Advances in neural information processing systems (2004)
25. Liu, T., Rosenberg, C., Rowley, H.: Clustering billions of images with large scale nearest neighbor search. In: Proceedings of the Eighth IEEE Workshop on Applications of Computer Vision. IEEE Computer Society, Washington (2007)
26. Loui, A., Luo, J., Chang, S., Ellis, D., Jiang, W., Kennedy, L., Lee, K., Yanagawa, A.: Kodak's consumer video benchmark data set: concept definition and annotation. In: Proceedings of the international workshop on multimedia information retrieval, pp. 245–254. ACM, New York (2007)
27. Lowe, D.: Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision 60(2), 91–110 (2004)
28. Makadia, A., Pavlovic, V., Kumar, S.: A New Baseline for Image Annotation. In: Forsyth, D., Torr, P., Zisserman, A. (eds.) ECCV 2008, Part III. LNCS, vol. 5304, pp. 316–329. Springer, Heidelberg (2008)
29. Matas, J., Chum, O., Martin, U., Pajdla, T.: Robust wide baseline stereo from maximally stable extremal regions. In: BMVC, vol. 1, pp. 384–393 (2002)
30. Messing, R., Pal, C.: Behavior recognition in video with extended models of feature velocity dynamics. Technical report, AAAI Spring Symposium Technical Reports (2009)
31. Mori, Y., Takahashi, H., Oka, R.: Image-to-word transformation based on dividing and vector quantizing images with words. In: International Workshop on Multimedia Intelligent Storage and Retrieval Management (1999)
32. Nistér, D., Stewénius, H.: Scalable recognition with a vocabulary tree. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2006, vol. 2, pp. 2161–2168 (2006)
33. Omohundro, S.: Efficient algorithms with neural network behavior. Complex Systems 1(2), 273–347 (1987)
34. Over, P., Awad, G., Rose, T., Fiscus, J., Kraaij, W., Smeaton-Alan, A.: TRECVID 2008–Goals, Tasks, Data, Evaluation Mechanisms and Metrics (2008)
35. Polana, R., Nelson, R.: Detecting activities. In: Computer Vision and Pattern Recognition, CVPR 2003 (2003)
36. Russell, B., Torralba, A., Murphy, K., Freeman, W.: LabelMe: a database and web-based tool for image annotation. International Journal of Computer Vision 77(1), 157–173 (2008)
37. Schindler, G., Brown, M., Szeliski, R.: City-scale location recognition. In: IEEE Conf. on Computer Vision and Pattern Recognition, CVPR (2007)
38. Schuldt, C., Laptev, I., Caputo, B.: Recognizing human actions: A local svm approach. In: International Conference on Pattern Recognition (2004)
39. Shakhnarovich, G., Viola, P., Darrell, T.: Fast pose estimation with parameter-sensitive hashing. In: Ninth IEEE International Conference on Computer Vision (2003)
40. Sivic, J., Zisserman, A.: Video Google: A text retrieval approach to object matching in videos. In: Ninth IEEE International Conference on Computer Vision (2003)
41. Smeaton, A., Over, P., Kraaij, W.: Evaluation campaigns and TRECVid. In: Proceedings of the 8th ACM international workshop on Multimedia information retrieval, pp. 321–330. ACM, New York (2006)
42. Torralba, A., Fergus, R., Freeman, W.T.: 80 million tiny images: a large dataset for non-parametric object and scene recognition. IEEE PAMI (2008)
43. Uhlmann, J.: Satisfying general proximity/similarity queries with metric trees. Information Processing Letters 40(4), 175–179 (1991)

44. Ulges, A., Schulze, C., Keysers, D., Breuel, T.: A System that Learns to Tag Videos by Watching Youtube. In: Gasteratos, A., Vincze, M., Tsotsos, J.K. (eds.) ICVS 2008. LNCS, vol. 5008, p. 415. Springer, Heidelberg (2008)
45. Von Ahn, L., Dabbish, L.: Labeling images with a computer game. In: Proceedings of the SIGCHI conference on Human factors in computing systems, pp. 319–326. ACM, New York (2004)
46. Von Ahn, L., Liu, R., Blum, M.: Peekaboom: a game for locating objects in images. In: Proceedings of the SIGCHI conference on Human Factors in computing systems, pp. 55–64. ACM, New York (2006)
47. Witten, I., Moffat, A., Bell, T.: Managing Gigabytes: Compressing and Indexing Documents and Images. Morgan Kaufmann, San Francisco (1999)
48. Zanetti, S., Zelnik-Manor, L., Perona, P.: A walk through the webs video clips. In: IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops (2008)

Author Index