Chapter 5
Distributed Mobile Computer Vision: Advances, Challenges and Applications

Niki Martinel, Andrea Prati and Christian Micheloni

Abstract The role of mobile devices has shifted from purely passively transmitting text messages and voice calls to proactively providing any kind of information that is also accessible to a PC. The recent advances in the field of micro technology have also made possible to include a camera sensor in any mobile device. This innovation is now attracting both the research community and the industries that aim to develop mobile applications that exploit recent computer vision algorithms. In this chapter we provide an analysis of the recent advances of mobile computer vision, then we discuss the current challenges that the community is currently dealing with. Next, an analysis of two recent case studies where mobile vision is used for augmented reality and surveillance applications is discussed. Finally, we introduce the next challenges in mobile vision where the mobile devices are part of a visual sensor network.

5.1 Introduction

Mobile devices are defined to be Web-enabled devices that are used not in a fixed location but they have been conceived and designed to be portable and usable in mobility. Typical mobile devices include Web-enabled mobile phones and Web-enabled pocket-sized Personal Digital Assistants (PDAs) [44]. The first mobile phone appeared on Detroit police cars in 1921. These devices were communicating together

N. Martinel (✉) · C. Micheloni
Univeristy of Udine, Via Delle Scienze, 206, Udine, Italy
e-mail: niki.martinel@uniud.it

C. Micheloni
e-mail: christian.micheloni@uniud.it

A. Prati
IUAV University of Venice, Santa Croce, 2957, Venice, Italy
e-mail: aprati@iuav.it

with a single high-power antenna installed on a skyscraper that allowed a transmission range of about a hundred kilometers. Twenty five years later the technology had reached the commercial service level.

Two main technological innovations allowed such rapid growing and they are still pushing the design of new, faster and more highly-featured devices. Over the years the advent of transistors boosted mobile device technology: the possibility of building tiny and sophisticated electronic components enabled phone companies to design a large number of models with many advanced features. The other important aspect—strictly connected with the diffusion of mobile devices—was the innovation of the communication infrastructure. Nowadays the community is talking about the fourth generation of mobile networks that should achieve a 1 GB/s transmission data speed. The new, more reliable and faster network infrastructures pushed more and more the mobile device companies to design and develop new, more sophisticated and innovative, mobile devices that take advantage of these resources.

Mobile devices were initially designed with a single physical keyboard, now they are equipped with touch-screens technologies borrowed by Personal Digital Assistans (PDAs) and other handheld devices: the keyboard has been replaced by a screen that allows the user to interact with a graphical interface using fingers or other pointing devices. These, together with faster microprocessors, allow an excellent user experience and make the tasks of multimedia processing and data logistic possible on the fly.

Historically speaking, the PDAs can be seen as the first generation of mobile devices. A PDA is a handheld computing device that combines multiple functions and features including telephone, fax, Internet and networking or other form of different connectivity capabilities. These devices are mainly used by users that need to compute operations while moving, simply called "Mobile Computing". Before that such features were provided only by laptops or desktop computers. The PDA is the first step to a future trend that would be later defined as "Technological Convergence" [17]. The next generation of mobile computing, mainly represented by smartphones, will foster the convergence of communication, computing and consumer electronics, three traditionally distinct industries with quite low interoperability. While mobile phones were previously equipped with a simple address book and agenda, now a smartphone has several features such as a camera, a voice control system and so on. In other words, a smartphone can be seen as a mobile phone with computer capabilities that allows it to interact with computerized system, send email and access to the web.

We are now in an era where the communication and computing environment is moving to interact with the physical environment or even become part of it. Mark Weiser, chief scientist at Xerox PARC and considered the father of Ubiquitous Computing, claimed that in the 21st century the technology revolution will move into "the every, the small and the invisible". That is what is really happening now: the information processing moves to the background so as humans concentrate on the tasks, not on the tools. The technology is viewed as a tool to serve the needs of people, not something to depend on. Weiser suggested that the most deep technologies are those that disappear ("Disappearing Technology"). They weave themselves into the fabric of everyday life until they are indistinguishable from it. The concepts of access anywhere, anytime, from any device are intertwined with the progress of the network infrastructures and with the aspects of convergence.

5.2 Chapter Contributions

Mobile devices are an inseparable part of our society now. The role of such devices has shifted from purely passively transmitting text messages and voice calls to proactively providing any kind of information that is also accessible to a PC. In particular, due to the advances in smart and micro technology, camera sensors are now a standard component in all mobile devices. This innovation is now attracting both the research community and the industries that aim to design and develop applications that exploit the powerful features of mobile devices and combine them together with the more advanced computer vision and image processing techniques.

In this chapter, we contribute to the research in distributed smart cameras introducing the most relevant advancements in mobile computer vision. Then we discuss the main challenges faced by the mobile vision community such as: (1) the limited energy, (2) the limited storage, (3) the limited computational capabilities, (4) the wireless network communication, (5) the scalability of applications. Next, we discuss two main research studies where computer vision techniques and distributed frameworks are used for mobile applications of augmented reality and security purposes. Finally we introduce the concept of smartphone networks, where the mobile device is part of a visual sensor network.

5.3 Mobile Computer Vision

A few years ago, it was very difficult to imagine that in the near future digital cameras would become a standard component of mobile devices. In these days, such devices have achieved a good level of maturity and they are now equipped not only with camera sensors, but with various other sensors such as accelerometers, gyroscopes, and GPS receivers. The exponential evolution of image and video processing devices with ever increasing computational capability equipped with high-resolution cameras and hardware-accelerated graphics has opened to a broad and new emerging research area that exploits the mobile device camera for applications of computer vision technologies. The broadband wireless network connection also enables mobility applications to use the acquired video data to initiate queries and exchanges of information with other mobile devices or higher computational power infrastructures. Though it is still in its infancy, it has attracted much attention from both industry and academia. This is not surprising, since we are indeed in an era of transition from a focus on PC-based computing to a greater emphasis on smart devices and cloud-based computing.

Among the multitude of mobile applications that have been designed to exploit the images coming from device cameras for computer vision applications, we can generally group them into three main clusters: location-based services, mixed/augmented reality, and car safety applications. These topics have become very popular in recent years, largely in the context of consumer applications where user-centered visual computing is essential.

Regarding the location-based services, industry has put a lot of effort to achieve real-time visual image recognition for these applications. Deployments of such systems include Google Goggles [13], Nokia Point and Find [34], Kooaba [20] and Snaptell [3]. A specialized case is represented by Leafsnap [21] which is an electronic field guide that uses visual recognition software to help identify tree species from photographs of their leaves. In essence, all these applications start from a snapshot image taken by a user on a mobile device. Then, the photo is matched against a pre-annotated image database to extract useful information that is provided to the user. In the recent years, the problems of image retrieval [12], landmark [24] and location [39] recognition using appearance-based features have also been deeply investigated by the community. In order to achieve their objectives, these methods match appearance features against a large database of location-tagged images [42]. In [48] authors propose a system for determining a user's location from a mobile device via image matching. The authors first build a "bootstrap database" of images of landmarks and train a CBIR algorithm on it. Since the images in the bootstrap database are tagged with keywords, when a query image is matched against the bootstrap database, the associated keywords can be used to find more textually related images through a web search. Finally, the algorithm is applied to the images returned from the web search to produce only those images that are visually relevant.

Mixed/augmented reality mobile video gaming is another area which has caught the attention of many. With the increasing quality and spatial resolution of mobile device cameras we have now high level of augmented reality where real-time interaction is possible. Another motivation of the spreading of AR solutions is the diffusion of available Software Development Kit (SDK) for efficiently and timely construct your own mobile AR application. Two examples above all: Qualcomm Vuforia [35] which is generic SDK for optimized augmented reality and object recognition and has been used by more than 3,500 mobile apps world wide; and Sentisight [33] which is a SDK for object recognition, computer vision and augmented reality used in a number of mobile vision applications.

Although, it is a matter of fact that, for some applications, we may need superior devices as they typically have much more computational capacity and additional sensors, enabling computationally expansive mobile applications on the fly. A recent demonstration of an outdoor mobile augmented reality application running on a cell phone is Nokia's MARA project [14]. The system does not perform any image analysis, instead it uses an external GPS for localization and an inertial sensor to provide orientation. PhoneGuide [9] is one of the first object recognition systems performing the computation on a mobile phone, instead of sending the images to a remote server. The system employs a neural network trained to recognize normalized color features and is used as a museum guide. Similarly, in [28] authors propose a novel framework to support AR for painting on mobile devices. In [18, 22, 46] recent techniques for tracking and occlusion handling in an AR framework were discussed. In [40] authors introduce a mobile system based on a hand-held device, GPS sensor, and a camera for roadside sign detection and inventory. Their algorithm was efficient enough to ensure good quality results in mobile settings. In the context of augmented reality, in [11] authors use a modified version of the SIFT algorithm for object detection and recognition in a relatively small database of mobile phone imagery of urban environments. The system uses a client-server architecture, where a mobile phone client captures an image of an urban environment and sends it to the server for analysis. The SURF algorithm has been used successfully in a variety of applications, including an interactive museum guide [5]. Local descriptors have also been used for tracking. In [43] authors track SURF features using video coder motion vectors for mobile augmented reality applications. The challenges of real-time recognition and camera pose estimation system for planar shapes were addressed in [16]. The proposed system performs shape recognition by analyzing contour structures and generating projection-invariant signatures. Similarly, in [41] SIFT features are used for recognition, tracking, and virtual object placement. Camera tracking is done by extracting SIFT features from a video frame, matching them against features in a database, and using the correspondences to compute the camera pose. In [10], the monoSLAM system estimates the hand-held camera's motion from the live image stream to achieve high AR performance. In [15], an AR rendering pipeline that supports global illumination techniques was proposed.

Another interesting and diffused field of application of mobile vision is the applications for car safety. In this case, the smartphone is positioned in front of the car and used for both analyzing the scene outside (ahead situation) or inside the car. Two existing products deserve special mention. CarSafe [49] uses rear and rear-facing front cameras for in-vehicle applications. The rear camera is used for monitoring distances from other vehicles and for tracking lane changes, whereas, rear-facing front camera tracks the driver's head position and direction as well as eyes and blinking rate as indicators of microsleep, drowsiness, and distraction. iOnRoad [19] uses Qualcomm's FastCV mobile-optimized computer vision library for frontal collision warning and lane departure warning. It also monitors headway and can be used for identifying and locating other cars in the field-of-view.
All of these applications pose a unique set of challenges.

5.4 Challenges

The proliferation of mobile and hand-held devices, along with advances in multimodal and multimedia technologies, are producing a new wave of applications that enable users to quickly and more naturally perform many tasks. These include: finding music, videos, and business listings; surfing the Web; sending a short text message; interacting with social media Web sites; just to mention a few. Mobility is central to this growing number of applications. It is a matter of fact that, as the number of smartphones and emerging devices continue to grow, user demand for new multimodal and multimedia interfaces that allow them to interact with the device while in mobility. Speech recognition and text-to-speech synthesis, are recent examples of this. Over the next few years, we expect to see multiple variants of speech and image processing technologies on smartphones to be features as standard as a keyboard. This is partially driven by regulatory requirements prohibiting texting while driving, and the need to provide a natural and a more compelling user interface with hands-free, eyes-free operation.

Another driver to the mobile revolution is cloud computing. It is significantly reducing the cost for deploying and maintaining large-scale mobile media services and enabling location-aware technologies for video, music, speech, and language to be more readily available as Web services. Indeed, developers can easily access such technologies through Web services application programming interfaces (APIs) that take advantage of standards such as HTML-5 and W3C EMMA [45].

The convergence of media and mobility is not only creating new opportunities, but also opening to a new set of challenges. In addition to the challenge posed by the limited battery life of mobile devices, we can find three prominent challenges in mobile vision when compared with traditional computer vision applications.

First of all, although the computational capacity of mobile devices is constantly increasing, it is still not sufficient to handle large-scale visual computing tasks. From this perspective, migrating much of the computation to powerful devices or to the cloud is essential. But, what part of the processing should be performed on the mobile client, and what part is better carried out by the server? On one hand, processing images is now possible on mobile devices in seconds or less. On the other hand, transmitting an image could take tens of milliseconds over a high speed wireless link. There are several possible architectures we can think about

- The mobile client transmits a query image to the server. The algorithms run entirely on the server, including an analysis of the query image. The final result is then sent back to the device as in a standard client-server architecture.
- The mobile client processes the query image and transmits (abstract) data. The algorithms run on the server using the data as query and returns the result to the device.
- The mobile client downloads data from the server and all the processing is performed on the device. This solution has the advantage to limit the required bandwidth (data can be downloaded only when the application starts or whenever they change), but can be slow due to the limited computational resources of the device.

It is worth emphasizing that this type of applications usually call for a fast response to address the user's requirement of a fluid interaction with the device. Therefore, a way to ensure real-time responses to user interaction will be a major issue to be tackled. Moreover, the bandwidth requirement is an important issue since connectivity can be not always available while on move or be in general expensive for the user.

Secondly, most mobile vision applications are driven by large amounts of annotated visual data. For example, to enable vision-based location recognition, a large collection of street-view images is a prerequisite. How to acquire and process such data is a challenge. One way to acquire the data is to harness the massive user-generated visual databases on the Internet, such as online image/video sharing systems. But then, a lot of information must be either locally stored by the mobile device or exchanged through the network.

Thirdly, not all mobile devices and smartphones that run the same application are equal. Some of them have very high computational resources and memory capabilities, while others only have reduced memories and limited computing cycles. What is more, they may have a reduced network connection speed so as only fewer round trips between the device and the server can be performed per second. This introduces the problem of scaling application requirements to each specific device.
Generally, scaling refers to growing computing resources to support an application. Vertical scaling is increasing single device capabilities, with powerful and faster CPUs, more memory, or faster disk I/O. While, horizontal scaling involves adding additional computational devices to support the overall applications computing requirements, and load balancing across those resources. So, scalability is generally viewed from a hardware designer perspective. We want to change this point of view and refer to the scalability as to adapt an application to support mobile devices that have different features. This aspect will be more clear in the second case study introduced below.

5.5 Case Studies

In this section two recent case studies that exploit advanced computer vision techniques are described. The former introduces an efficient marker-less approach to support augmented reality for paintings. While, the latter introduces a cooperative method to address the challenges of the person re-identification problem using both mobile devices and remote processing units.

5.5.1 Augmented Reality for Musems

Recognizing paintings and computing the transformation to align the acquired image of a painting and its image in a database to support Augmented Reality (AR) applications is nontrivial tasks. Common static computer vision issues (e.g.illumination, view, scale changes, etc.) are more severe in context of moving cameras as different effects (e.g.blur, noise, motion, etc.) arise. In addition, reflection of spotlights, image saturation and image exposure add up for the recognition of the paintings present in exhibitions.

While specifically designed markers have been the dominant choice by state-of-the-art AR methods, here a marker-less approach is introduced. Using the principle of Hough line detection, a method to detect and extract the relevant painting region (RPR) from a given input image is first applied, then local features are extracted from

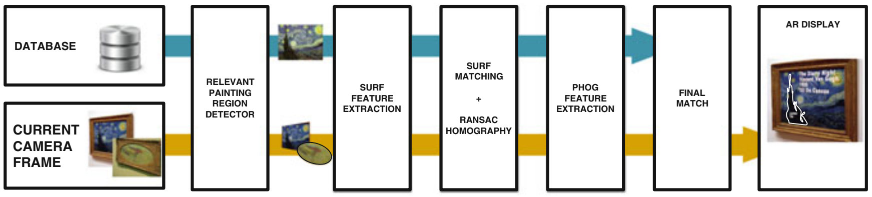


Fig. 5.1 Architecture of the proposed system. The current camera frame is processed by the three main modules of the system. The first module extracts the RPR. The second module matches every database image with the current camera frame. Given the best match, the third module uses the RANSAC homography transformation to overlay the additional content to the current camera frame

the RPR and matched with a candidate target in the database. RANSAC is used to detect feature outliers. As the current image may not be aligned with the candidate target image, extracting global feature from it considerably reduces robustness, so the homography transformation output by RANSAC is used to sidestep this issue and align the current RPR to the candidate target RPR. This allows to extract global feature from the aligned RPR only, thus noticeably improving performance. Then, a weighted similarity measure is used to compute the final match between image signatures composed of local and global features. Once a match is found, the same RANSAC homography is exploited in an AR framework to properly overlay the additional content to the current frame.

5.5.1.1 System Overview

The architecture of the proposed approach is shown in Fig. 5.1. Three main modules are used to achieve the proposed goal: (1) the RPR detector module, (2) the matching module and (3) the AR display module. Given the current camera frame, the RPR detector module extracts the RPR so as the painting frame and the background are not considered anymore. Once the RPR is detected, the matching module extracts the local features from such region and matches them with each candidate target local features. RANSAC is used to reject matching outliers. Then, the homography matrix output by RANSAC is used to align the current RPR with the candidate target RPR from where the global features are extracted. The extracted local and global features are finally matched with the candidate signature using an affine combination of similarity measures. Given the best candidate target match, the same homography transformation output by RANSAC is used to properly overlay the additional content to the current camera frame.

5.5.1.2 Relevant Painting Region Detector

Assuming that the frames of paintings have elliptical or rectangular shape, the RPR detector goal is to remove the background and the painting frame to keep only

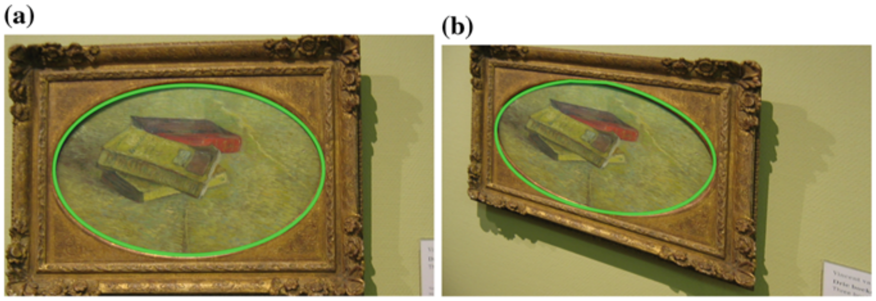


Fig. 5.2 RPR detection examples. In **a** an ideal example where the painting plane is almost aligned with the given image plane is shown. In **b** a harder example with severe perspective distortion is depicted

the portion of the image that contains the painting, i.e. the RPR. To achieve such objective saving computational cycles, the Randomized Hough Transform (RHT) method [47] is used. Such technique is based on the standard Hough Transform (HT) but it avoids the computationally expensive voting procedure. As shown in [47], the RHT can achieve a computational complexity lower than an upper bound of order $O((n_t N^n)/n^n_{min})$, that is considerably smaller than the order $O(NN_a^{n-1})$ of the standard HT. N and N_a are the total number of pixels in the image and the size of the accumulation array respectively. n is the number of curve parameters, n_{min} is the length of the shortest curve in the image, and n_t is a small number.

To detect the RPR, we first apply the Canny operator to the grayscale representation of the current camera frame $Q \in \mathbb{R}^{M \times N}$. Then, as shown in Fig. 5.2, the RHT is used to fit ellipses and rectangles. As a painting may contain more than a single rectangular or ellipse shaped object, the RPR boundary is selected as the detected rectangle or ellipse which area is at least r times the input image size. The detected RPR is denoted as $R_Q \in \mathbb{R}^{M' \times N'}$ where $M'N' \geq rMN$ and $r \in [0, 1]$.

5.5.1.3 Matching

Let R_Q and R_I be the RPR of the current camera frame Q and the RPR of a database candidate target image I, respectively. To match R_Q and R_I two local and global features have been considered. In particular to match two RPRs with lower computational costs, the Speeded-Up Robust Features (SURF) [4] and the Pyramid of Histogram of Oriented Gradients (PHOG) [7] features have been used. Both a local and a global feature have been used as the relevant region detector module may fail, i.e. only a small area of the painting or the painting with the frame can be extracted. In such cases, if only global descriptor is extracted non-interesting information is considered.

Local features: As illumination invariance is intrinsic to SURF [4], such features are extracted from the grayscale representation of R_Q by exploiting the standard integral image. The SURF feature detector is based on an approximation of the Hessian matrix, while the feature descriptor $\psi_F^{R_Q} \in \mathbb{R}^{64}$ describes the distribution of Haar-wavelet responses within the neighborhood of the detected interest points $\psi_K^{R_Q} = [x, y]$. The computed SURF feature vector is denoted as $\psi^{R_Q} = \langle \psi_F^{R_Q}, \psi_K^{R_Q} \rangle$.

Given two SURF feature descriptors $\psi_F^{R_Q}(q)$ and $\psi_F^{R_I}(i)$, q, i is defined to be a match if the similarity

$$S_F(\psi_F^{R_Q}(q), \psi_F^{R_I}(i)) = \frac{1}{1 + \|\psi_F^{R_Q}(q) - \psi_F^{R_I}(i)\|_2} \tag{5.1}$$

is higher than a fixed threshold Th_s. If that is satisfied, the two matching SURF feature vectors are analyzed to detect outliers using a RANSAC-based approach similar to the one proposed in [8]. In particular, given 4 feature correspondences, the homography $H_{Q,I}$ is computed using the Direct Linear Transformation method. The process is repeated with t trials, and the solution that has the maximum number of inliers is selected. A SURF feature keypoint $\psi_K^{R_Q}(q)$ is considered to be an inlier if the corresponding keypoint projection $\widehat{\psi}_K^{R_Q}(q)$ is consistent with $H_{Q,I}$ within a tolerance of σ pixels.

Global features: Given $H_{Q,I}$, that is used to to align the two RPR regions, namely R_Q and R_I. The resulting transformed RPR is denoted as $\hat{R}_Q$. By applying such transformation to global feature can be extracted in a more reliable fashion as the edges used to compute the global features are aligned and have similar orientations to the edges of the candidate target.

PHOG features are extracted from $\hat{R}_Q$ to capture information about the shape and the whole appearance of the painting. Before extracting PHOG features, the transformed RPR region $\hat{R}_Q$ is projected into the HSV color space to achieve illumination invariance. Then, edges and orientation gradients are used to compute the PHOG feature matrix $\rho^{\hat{R}_Q} \in \mathbb{R}^{m \times 3}$ by concatenating the PHOG histograms extracted from the three image channels at the different levels of the spatial pyramid. m is the total number of histogram bins for each image channel.

Candidate target matching: Once local and global features have been extracted, the two given images Q and I are matched using an affine combination of feature similarities.

SURF features similarity is computed as

$$\Phi_\psi(\psi^{R_Q}, \psi^{R_I}) = \frac{\sum_{q,i \in match} S_F(\psi_F^{R_Q}(q), \psi_F^{R_I}(i))}{\varepsilon + match} \tag{5.2}$$

where $match$ is the total number of matched SURF features and ε is a small constant used to prevent division by zero.

PHOG features are matched using the same similarity function suggested in [27]. Let $\rho^{\hat{R}_Q}$ and ρ^{R_I} be the PHOG feature matrices of $\hat{R}_Q$ and R_I respectively. The PHOG similarity is computed as

$$\Phi_\rho(\rho^{\hat{R}_Q}, \rho^{R_I}) = 1 - \sum_c \lambda_c \chi^2(\rho_c^{\hat{R}_Q}, \rho_c^{R_I}) \tag{5.3}$$

where $\rho_c^{\hat{R}_Q}$ and $\rho_c^{R_I}$ are the PHOG features computed for the c-th color channel. λ_c is the normalization weight.

As the final objective is to find the database image that gives the best match with the query image, a minimization task needs to be performed. Towards this end, let $\mathbb{I}$ be the set of all database images, then, the final objective is to find $\arg\max_{I \in \mathbb{I}} \Phi(Q, I)$ where

$$\begin{aligned}\Phi(Q, I) = \alpha\, \Phi_\psi(\psi^{R_Q}, \psi^{R_I}) \\ + \beta\, \Phi_\rho(\rho^{\hat{R}_Q}, \rho^{R_I})\end{aligned} \tag{5.4}$$

α and $\beta = 1 - \alpha$ are the affine coefficients.

5.5.1.4 AR Display

The last module of the proposed system is in charge to overlay the additional content to the current camera frame Q. As both paintings and the additional content are planar surfaces, the transformation that needs to compute can be described as an homography. In this work a feature-based homography computation method is used. Towards this goal, the same feature-based homography transformation $H_{Q,I}$ computed in Sect. 5.5.1.3 is exploited. As shown in Fig. 5.3, using the inverse homography transformation matrix $H_{Q,I}^{-1}$ it is possible to overlay the additional content (given in the original region I coordinate system) to the current camera frame Q.

5.5.1.5 Experimental Results

To show the performance of the proposed method, experiments have been carried out on a dataset built using 607 publicly available pictures of 70 Vang Gogh paintings. The dataset has pictures taken from different viewing angles and with severe illumination changes. Some pictures come with light reflections and occlusions as well.

Implementation details: The values of the algorithm parameters given in the following have been selected using 4-fold cross validation:

- RPR boundary: r has been set to 0.55;
- SURF features: features have been extracted using 5 octaves and 4 scale levels;

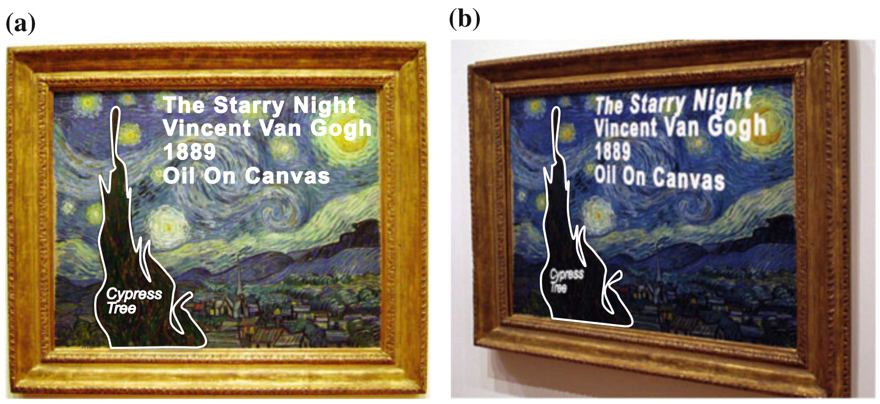


Fig. 5.3 Example of AR. **a** Reference frame with the additional information to display. **b** Current frame with the additional information transformed using the feature-based homography transformation

Table 5.1 Average nAUC values for different values of alpha (from 0 to 1 with steps of 0.01)

α	0	0.25	0.28	0.5	0.75	1
RPR detection	0.7927	0.8110	**0.8320**	0.7485	0.6917	0.6281
No RPR detection	0.6368	0.6572	**0.6635**	0.6470	0.6341	0.6290

Best results are in boldface font

- PHOG features: edges have beeen extracted using the Canny operator, while orientation gradients have been computed using a 3×3 Sobel mask. The extracted HOG features extracted from the 4 levels of the spatial pyramid have been quantized in 9 bins;
- Matching: The normalization weight vector λ has been set to [0.5, 0.3, 0.2]. The tolerance σ has been set to 4 pixels and $t = 500$ trials are performed to compute H. The matching threshold Th_s has been set to 0.85.

To show the performance of the method results have been reported in terms of ROC curves and normalized Area Under Curve (nAUC) values.

The algorithm has been tested on a standard PC with P4 CPU 2.0GHz, 1GB RAM, Windows XP and on a Tablet with ARMv7 processor 1GB RAM, Android 4.2.2. In the first test with non-optimized MATLAB code the average recognition and registration time for a single frame was 0.591s, while in the latter with optimized code the same activities took 0.632s.

Experiments: In the following we show the performance of the proposed method as a function of both the scale of the images and the rotation of them with respect to the target image orientations.

In Table 5.1, nAUC values computed as a function of the similarity normalization weight α are reported. Each value is computed averaging all the results computed for images scaled to $1/2$, $3/4$ and $1/1$ of the original image size and for different image

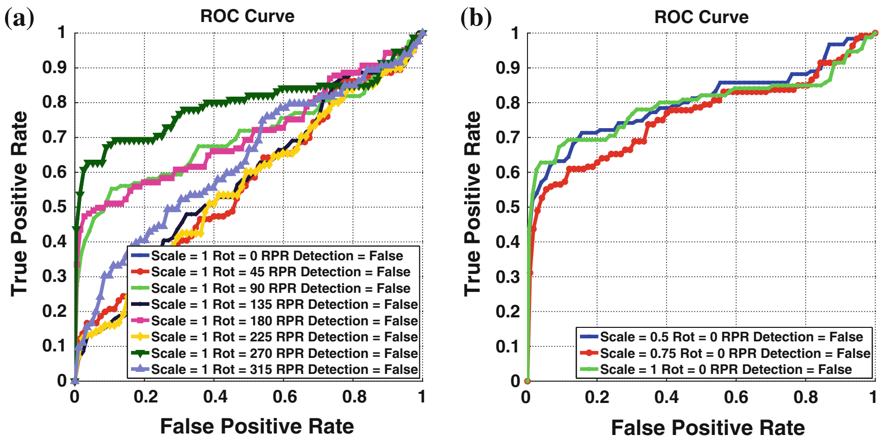


Fig. 5.4 Recognition performance computed without using the relevant region detector module. In **a** test images are rotated by multiples of 45°. In **b** test images are not rotated but their scaling factor has been changed

rotations from 45° to 315°, with intermediate rotations of 45°. The best results are achieved for $\alpha = 0.28$. In light of this, such value has been used in the following experiments.

In Fig. 5.4 the performance of the proposed method without using the RPR detector are shown. In Fig. 5.4a results are shown as a function of the rotation of the test images. The original scale has been used. The method achieves reasonable results for rotations multiple of 90°. The performance decreases in the other cases. This is probably due to the changes occurring in the oriented gradients used to compute the PHOG features. On average, a true positive rate of 49 % is achieved for a false positive rate of 20 %. In Fig. 5.4b results for different image scales are shown. Thanks to SURF invariance properties and the pyramidal approach used to compute PHOG, the performance are not much affected by the scaling issues. In such scenario, an average true positive rate of 67 % is achieved for a false positive rate of 20 %.

In Fig. 5.5 the performance of the proposed method using the RPR detector are depicted. In Fig. 5.5a results have been computed for different rotations to the test images as in Fig. 5.4a. On average, a 71 % true positive rate is reached for a false positive rate of 20 %. Though, the worst results are reached for rotations of 135° and 225°, where a true positive rate of about 59 % is reached for the same false positive rate of 20 %. If compared to the results shown in Fig. 5.4a, a significant improvement has been achieved. Most importantly, the performance has increased of about 33 % for a false positive rate of 0 %. In Fig. 5.5b the results are computed varying the scale of test images. If compared to Fig. 5.4b, an average improvement of 37 % is achieved for a false positive rate of 0 %.

In Table 5.2 results of the proposed method are shown as nAUC values. Images scaled to 1/2, 3/4 and 1/1 of the original image size and rotations from 45° to

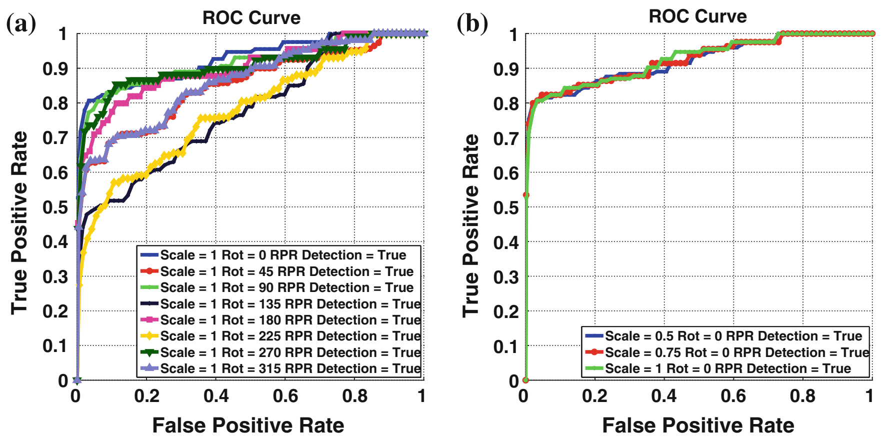


Fig. 5.5 Recognition performance computed using the relevant region detector module. In **a** test images are rotated by multiples of 45°. In **b** test images are scaled down using multiple reduction factors

Table 5.2 nAUC values computed for test images scaled to 1/2, 3/4 and 1/1 of the original size and rotated from 0° to 315°(steps of 45°)

	Rotation	0°	45°	90°	135°	180°	225°	270°	315°
RPR detection	Scale = 0.5	0.9223	0.8025	0.8918	0.7178	0.9020	0.6708	0.8949	0.7518
	Scale = 0.75	0.9243	0.8049	0.9032	0.7158	0.9128	0.7010	0.8986	0.8209
	Scale = 1	0.9258	0.8233	0.9138	0.7188	0.9180	0.7044	0.9053	0.8226
No RPR detection	Scale = 0.5	0.8130	0.5502	0.7015	0.5492	0.6810	0.5476	0.7963	0.6133
	Scale = 0.75	0.8104	0.5529	0.7023	0.5510	0.6966	0.5498	0.8082	0.6214
	Scale = 1	0.8244	0.5655	0.7026	0.5738	0.7082	0.5585	0.8144	0.6329

315° (with steps of 45°) have been considered. The first three rows show the results of the proposed method using the proposed RPR detector, while in the last three rows results have been computed without using the RPR detector. For both such cases the best results are achieved when the original image size is kept and no rotation is applied. However, using the relevant region detector, performance increases of more than 17 % on average. In particular, for rotation of 90°and 180°, an average increment of 20 % is achieved.

5.5.1.6 Conclusion and Discussion

In this case study, a marker-less method for painting recognition and registration to support mobile AR applications has been introduced. A RPR detector is used to extract only the relevant painting region, that is next considered to extract local features that are matched with a candidate target using RANSAC. The homography transformation output from RANSAC is also applied to align the detected RPR to the RPR of the candidate target. This allows to robustly extract global features that are finally used, together with local features, to compute the match between the current frame and the candidate target. Once a valid match is detected, the same homography transformation output by RANSAC is used to overlay the additional content to the current frame. The method has been evaluated using a dataset of publicly available images showing significant achievements.

5.5.2 Cooperative Person Re-Identification

As discussed before, mobile devices with exponentially increasing capabilities have been recently introduced into the market, thus boosting the development of computer vision algorithms for mobile devices. Though standard computer vision algorithms can be ported to mobile devices, the computational costs required and the limited resources have reduced their applicability. Many problems have been investigated by the computer vision community especially for security purposes [31]. Despite this, there is limited work that exploits the cooperation between mobile devices and a camera network for surveillance purposes.

To monitor a wide area traditional surveillance CCTV systems are still the most common choice. Despite the benefits of surveillance systems and the advantages of recent and performing surveillance applications [29], static camera infrastructures still show several disadvantages. Among of them we find the restricted field-of-views, the low resolution of most static cameras, and the limited communication bandwidth, just to mention a few. In this section a client-server system is introduced to tackle these challenges with particular focus to the task of person re-identification, that is, the task of assigning the same label to the same person viewed by different cameras at different time instants. The proposed system brings several advantages as it allows to monitor a large portion of the environment using moving cameras (i.e. mobile cameras). It also introduces a better use of resources and reduces the computational cost of the re-identification. In particular, in this case study the objective is to find the configuration of the mobile client device that should be used to achieve a real-time processing while keeping high re-identification performance. To achieve such objective and save network resources the system limits the exchanged information -between the device and the server- to data vectors of small dimensions.

5.5.2.1 System Architecture

An overview of the proposed system architecture is shown in Fig. 5.6. The system flow goes as follow. Given the image of a suspicious person acquired by a camera in the network, the server computes its signature and sends it to all the mobile devices (i.e. clients). Once the signature is received, each client starts capturing the scene. Then, the first acquired frame of a person is sent to the server together with mobile

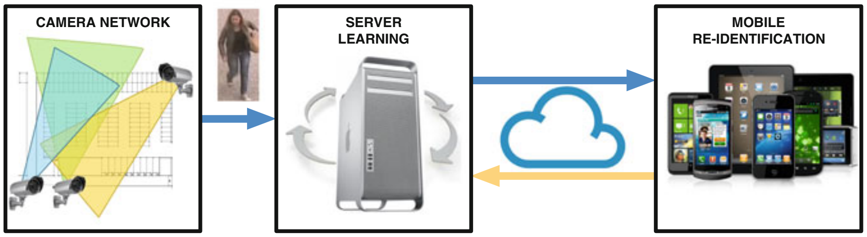


Fig. 5.6 Client-server communication overview. The image of a suspicious person acquired by a camera in the network is sent to the server that routes it to each of the connected mobile devices. Then, the first image acquired by each mobile device is sent back to the server, that, exploiting a learnt model, instructs the mobile device about which features have to be extracted to perform the re-identification

device information regarding its computational and networking capabilities. The server analyzes the received data and instructs the client with which features should be extracted to achieve the best re-identification. This process is mainly guided by a model learned during a separate off-line training phase (details in the following). Then, each client starts to process the frames to detect the persons blob and to extract the features as instructed by the server. The computed features are finally matched with the previously received signature to perform the re-identification. However, the re-identification performance may not stable over a certain period due to the changes in illuminations, pose, etc.. To sidestep such issue a server re-configuration is performed. This is done by forcing the client to send another frame and restarting the whole procedure.

More in detail, as shown in Fig. 5.7, the proposed system has been designed to exploit three main phases: (1) an offline training phase, in which the server collects images and device characteristics (resolution, CPU performance, etc.) to train a classifier; (2) an online learning phase used to instruct the querying device about which feature algorithm should be used; (3) a re-identification phase where the selected feature algorithm is used to extract and match the person acquired by the querying device with the suspect image sent by the server.

5.5.2.2 Image Complexity and Feature Extraction

The training phase and the re-identification phase share a common step, that is, the computation of the image complexity and the extraction of image feature.

Images are classified with respect to their complexity by extracting the edges and corners using Canny and Harris corner detector methods respectively. Let n_e be the number of edges and n_c be the number of detected corners for a given image. The image complexity denoted as $C \in [0, 1)$ is computed as

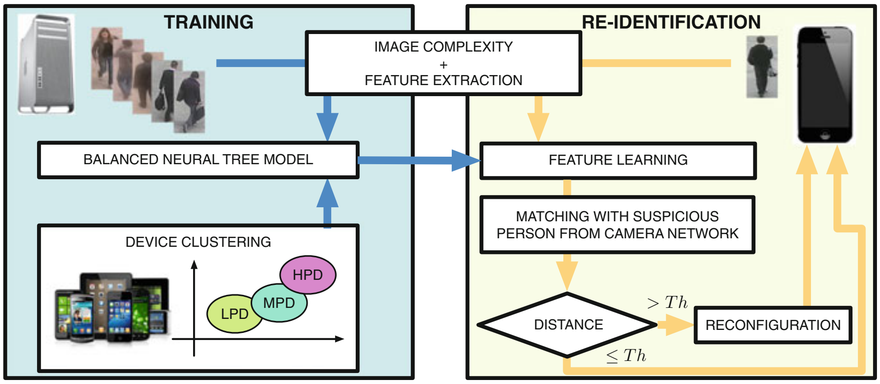

TRAINING
IMAGE COMPLEXITY + FEATURE EXTRACTION
BALANCED NEURAL TREE MODEL
DEVICE CLUSTERING
HPD
MPD
LPD
RE-IDENTIFICATION
FEATURE LEARNING
MATCHING WITH SUSPICIOUS PERSON FROM CAMERA NETWORK
DISTANCE
> Th
≤ Th
RECONFIGURATION

Fig. 5.7 Main system training and re-identification steps. A set of training images and the device complexity clustering model are used to train a balanced neural tree. During the re-identification phase, frames acquired by the mobile devices are sent to the server together with device details. The server analyses the input data to select the most discriminative features for re-identification according to model learnt in the training phase. Then the matching is performed with such features. If the threshold on feature distances is reached, the system gets re-configured and the querying devices gets instructed again about what features should be used in the next steps. This allows to adapt the re-identification performance to the context

$$C = 1 - \left(\alpha \frac{1}{n_e + 1} + \beta \frac{1}{n_c + 1}\right) \tag{5.5}$$

where α and $\beta = 1 - \alpha$ are the normalization weights.

The community has developed a large number of approaches to detect and describe discriminative patterns of a given image. Scale Invariant Feature Transform (SIFT) [26] and Speed-up Robust Feature (SURF) [4] are two well-known feature algorithms that achieve great performance under a variety of image transformations. More recently other feature detector algorithms as BRISK [23], FAST [37], STAR [1] and ORB [38] have been proposed and designed to be runtime efficient. A more detailed description of these can be found in the computational performance evaluation review given in [32]. In our approach we used the aforementioned feature detectors implemented in the OpenCV libraries. As the FAST and ORB do not incorporate a descriptor, these features are completed with the Fast Retina Keypoint (FREAK) descriptor [2].

5.5.2.3 Training Phase

As described in Sect. 5.5.2.2, several feature algorithms can be used but, due to their different implementations and configurations, they differ in performance and in the number of features extracted. Differences arise depending on the kind of image and on the device capabilities as well. An example of this is shown in Fig. 5.8. The key

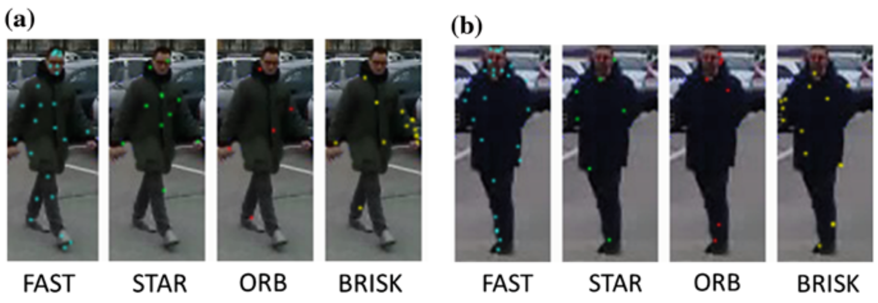
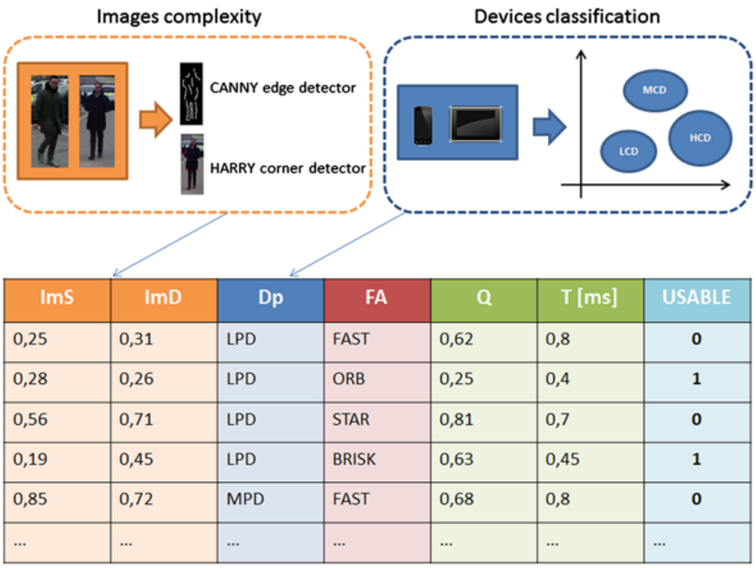


Fig. 5.8 Different state-of-the-art features are extracted from two different persons using the same mobile device with same configuration



ImS	ImD	Dp	FA	Q	T [ms]	USABLE
0,25	0,31	LPD	FAST	0,62	0,8	**0**
0,28	0,26	LPD	ORB	0,25	0,4	**1**
0,56	0,71	LPD	STAR	0,81	0,7	**0**
0,19	0,45	LPD	BRISK	0,63	0,45	**1**
0,85	0,72	MPD	FAST	0,68	0,8	**0**
...	...	...	...	...	...	...

Fig. 5.9 Training set example. The first two columns represent the complexity of images captured from static cameras (*ImS*) and from mobile device cameras (*ImD*). *Dp* represents the performance index of each device. *FA* represents the feature algorithm used to compute the match. Finally *Q* and *T* are two performance parameters and refer to the quality index and to the computational times respectively

idea is to determin -for a given image complexity/structure and for a given device- the most discriminating features for re-identification.

To achieve such an objective we first partitioned the space consisting of all mobile devices by clustering it into three different groups on the basis of their capabilities. The amount of RAM, the frequency and the CPU model (e.g.dual core, quad core etc.) have been considered to manually cluster devices in: (1) low-performance devices (LPD), (2) medium-performance devices (MPD), and (3) high-performance devices

(HPD). This has been accomplished by first manually labeling a set of devices, then a multiclass SVM has been trained using the one-vs-all method. SVM parameters have been computed using 4-fold cross-validation.

Then, given a set of training images acquired by mobile devices, the image complexity and the device capabilities are used to compute the performance of each considered feature algorithm (see Sect. 5.5.2.2). In particular, we measure the feature extraction time, and a quality parameter (a distance value better discussed in Sect. 5.5.2.4) to decide whether a feature algorithm can be used or not. If the extraction times is too high or the quality is too low, the feature algorithm is considered as not usable on the device that is currently under inspection. It is usable vice versa. The so computed images complexity, the device type and feature type, form the input pattern in the training set. The label *usable (1) / not usable (0)*, represents the label that the classifier should return for such an input pattern. Once the training set is built, it is used to train a balanced neural tree [30] that is next exploited by the re-identification phase.

5.5.2.4 Re-identification Phase

During the re-identification phase, the image acquired by the mobile device camera is used to create the pattern that is input to the trained balanced neural tree such that the best feature algorithm for re-identification is chosen. Once the feature algorithm is selected, it is used to extract image features and match the current image with the signature of the suspicious person given by the camera network.

Feature Learning: The main goal of the learning is to automatically select the best feature algorithm for the specific type of mobile device making the request. This operation is required in two different situations: (1) when the client application starts and sends for the first time a frame to the server and (2) when the re-identification performance is not stable (i.e. accuracy is decreasing) over a given period of time. In a typical scenario, the client device sends to the server the first frame and information related to its performance (e.g.CPU model, RAM usage etc.). Once the server receives such information, the complexity of the image is computed and the client device is classified. A pattern containing the complexity of the images, the device classification and the type of feature algorithm to use is created and input to the neural tree classifier. The results of this operation is a packet containing the type of feature and the parameters that should be used by the client.

Matching metchanism: Once the client receives the configuration packet the selected feature is used to perform re-identification. This process is executed locally on-board of the device. Since the exploitable feature represents a subset of the suspect's signature, only the corresponding subset is used to match the device image with the suspect. To measure the distance between the features descriptors computed by the mobile device $feat^C$ and the feature descriptors of the suspicious person signature $feat^S$ we compute

$$\Phi(feat^C, feat^S) = 1 - \frac{1}{\sum_{c,s \in match} d(feat^C(c), feat^S(s))/(match + \varepsilon)} \tag{5.6}$$

where $d(\cdot, \cdot)$ is the L^2-norm distance between feature descriptors. ε is a small value that avoid dividing by zero, and $match$ is the set of matches such that the L^2-norm distance between feature descriptors is lower than a given threshold Th_{l2}. c and s represent the subset of features matched from $feat^C$ and $feat^S$ respectively. If the distance $\Phi(feat^C, feat^S)$ is higher than a given threshold Th, a system reconfiguration is required.

5.5.2.5 Experimental Results

In this section we report the performance of the proposed method. First, an analysis of the feature algorithm is given, then results of the re-identification are presented.

Experimental scenario: Tests have been performed using an Asus TF101 tablet (device A) and a Samsung Galaxy III smartphone (device B). Both the devices run Android O.S. version 4.2.1. For each of them two different videos (V1 and V2) that have 60 and 70 persons respectively, have been used. The video were acquired with a resolution of 358×255 pixels at 30 fps. To calculate which is the best detector algorithm for the device in use, the computational time required by each feature algorithm has been computed considering only the time required to detect and extract the keypoints from the given frame. Other image processing operations on the captured frame were not considered. The reference frames from the camera network have been acquired by an AXIS 213 PTZ and an AXIS 221 camera. The default configuration defined by the OpenCV libraries has been used for STAR, FAST, ORB and BRISK algorithm parameters.

Training: The first evaluation index taken into consideration to build the training pattern is the features extraction time. As shown in Fig. 5.10 for both video V1 and video V2, ORB (red lines) is the fastest algorithm in terms of computational times. The average frame is processed in 0.62 ms. FAST (green chequered) is the one achieving lowest performance with a processing time of about 0.81 ms.

The second evaluation index taken into consideration is the number of matching features. Table 5.3 presents the average number of features matched by devices A and B with the reference images acquired by the two cameras. Best average values are highlighted in boldface font. FAST algorithm gives the best results with an average of 71 matches. STAR and BRISK match respectively an average of 29 and 21 matches. ORB achieves the lowest performance with an average of 17 matches.

All such results are then used to form the train set for the balanced neural tree.

Re-identification results: During the re-identification phase the threshold Th_{l2} was set to 0.25 for both V1 and V2. Similarly Th was set to 0.29. These values were chosen by using 4-fold cross-validation.

In Fig. 5.11 the matching results of the first 30 frames of one person in video V1 captured using the mobile device A are shown. ORB features have been extracted

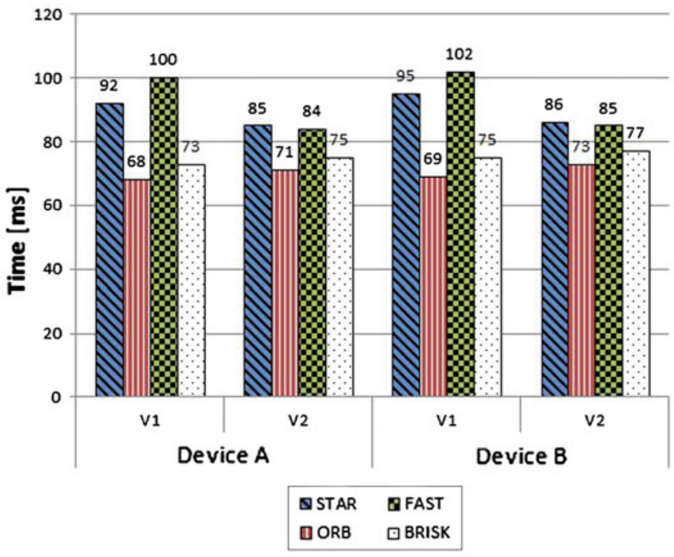


Fig. 5.10 Evaluation results for STAR, ORB, FAST, BRISK feature algorithms on the used devices

Table 5.3 Average number of matching features for both device A and B. The highest average number of matching features is highlighted in boldface font

	Feature type	Video 1	Video 2	Average
Device A	STAR	31.6	27.1	29.3
	ORB	19.1	15.6	17.4
	FAST	70.1	72.1	**71.1**
	BRISK	20.8	22.3	21.6
Device B	STAR	30.1	26.7	24.2
	ORB	18.8	14.6	16.4
	FAST	71.3	73.9	**72.1**
	BRISK	20.9	23.0	22.2

from the first 15 frames as instructed by the feature learning mechanism. Then, at frame 15, the distance value increases to 0.6 due the change of brightness. Such a value is greater than Th, thus a new configuration from the server was required. The device was then instructed to extract the FAST features. That results in a new a distance value of 0.17 between the reference frame and the current image.

Re-identification performance on video V2, where the same device A was used, are shown in Fig. 5.12. In this case two reconfigurations occurred. At frame 13, FAST feature detector has been replaced by ORB as the distance of 0.34 was higher than the threshold. Next, at frame 25 a change of the brightness forced another device reconfiguration. The STAR feature algorithm was then used.

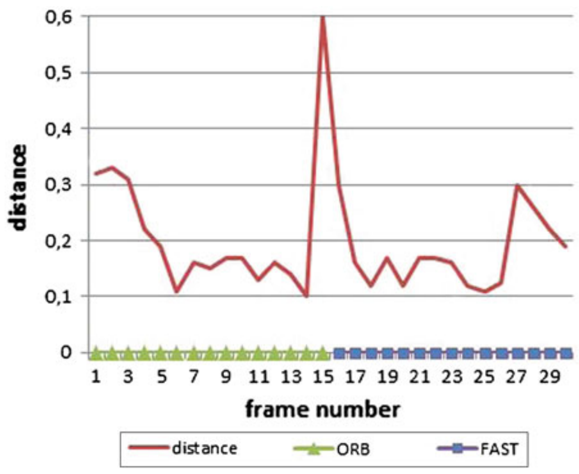


Fig. 5.11 Re-identification performance on video V1 using device A. The first 15 frames were acquired using ORB feature detector. After a change in image brightness, the device got reconfigured and instructed to use the FAST algorithm

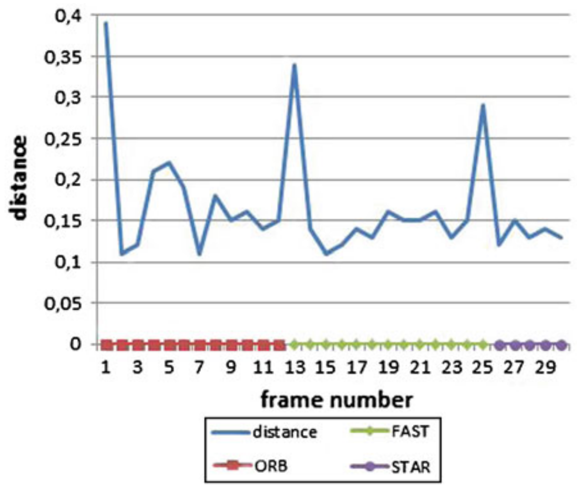


Fig. 5.12 Re-identification performance on video V2 using device A. Three changes of the feature detector was required. The average distance calculated was around the value of 0.15. Two peaks occurred respectively at frame 13 and 26

In Fig. 5.13a results of re-identification performance for device A are shown in terms of ROC curves. The feature algorithms in Sect.5.5.2.2 have been compared to the proposed adaptive feature algorithm. Results shows that the adaptive algorithm outperform all other methods with a true positive rate of about 80 % for a false

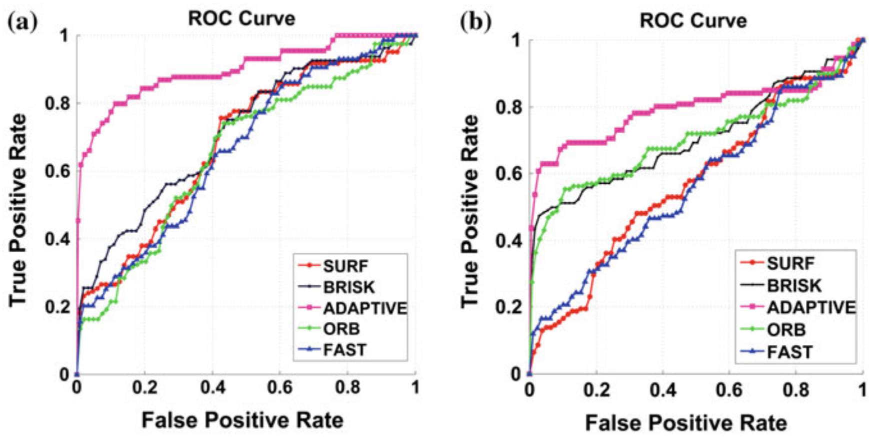


Fig. 5.13 Re-identification results in terms of ROC curves. The ROC curves have been computed using all the frames in V1. In **a** results are computed for device A. In **b** results are computed for device B

positive rate of about 20 %. For the same false positive rate, the other algorithms achieve a true positive rate of about 37 %.

In Fig. 5.13b, results of re-identification performance using device B are shown. As for device A, the proposed adaptive algorithm outperform all other algorithms. In particular, considering a false positive rate of about 40 %, the adaptive algorithm achieves a true positive rate of 80 %, while other methods have performance of 45 % on average.

5.5.2.6 Conclusion and Discussion

In this case study a client-server system for re-identification using smart devices has been introduced. The system allows to save network and computational resources by exploiting a feature learning mechanism. A training phase is performed to cluster devices on the basis of their capabilities and to train a classifier to select the best feature algorithm for a given device and image complexity. Such classifier is used in the re-identification phase to select which feature should be used to maximize the re-identification performance. Experimental results that the client-server method outperforms all other standard methods.

5.6 Smartphone Networks: What Next?

Camera networks have received increasing attention in recent years, in part due to their many uses in applications ranging from surveillance and security, smart spaces, urban monitoring, traffic management, etc. Another recent development is the diffusion of consumer devices with built-in cameras, e.g., mobile phones, tablets, and Google Glass. Currently, however, camera networks and these consumer devices occupy different use spaces. An obvious question then is whether it is possible to integrate these consumer devices into camera networks, treating these devices as sensor nodes within the larger camera network infrastructure. In fact, it should be technically feasible to integrate such consumer devices into more traditional camera networks and video surveillance infrastructures comprising primarily of stationary and pan/tilt/zoom (PTZ) cameras. These new camera networks will usher a whole new set of applications, from crowd-sourced and people-centric surveillance to participatory event recording.

However, none of the existing mobile vision applications and frameworks considers the possibility of multiple consumer devices working together towards common sensing (or imaging) tasks. Furthermore, these techniques do not explore the possibility of integrating a camera-enabled consumer device into existing (surveillance) camera networks. The work in [6] is noteworthy in that it hints at the possibility of setting up a smart camera network comprising mobile phones. This work also lists the key requirements for setting up such a camera network. Chiefly among these requirements is the cooperative sensing by the mobile phones comprising the network [6], however, fails to provide an in depth analysis of the technical challenges associated with (1) setting up smart camera networks using mobile phones and (2) integrating camera-enabled consumer devices into existing (surveillance) camera networks. The greatest shortcoming of the proposal outlined in [6], perhaps, is the fact that it assumes that mobile devices are stationary. It then essentially treats mobile cameras as traditional cameras, ignoring the most notable feature of mobile phones, i.e. mobility. The fact that mobile devices are not stationary makes setting up camera networks of these phones that much more interesting and challenging at the same time.

The advances in mobile computing, sensing hardware and communication technologies combined with the success of mobile vision have made it possible to stretch the concept of a "‘smart camera"’ to the extreme: *every smart device equipped with a camera (and may be other sensors) can serve as a node in a smart camera network*. Setting up ad hoc networks comprising camera-equipped consumer devices or integrating such devices into existing camera networks poses unique challenges, in addition to those already mentioned in Sect. 5.4, with regard to stand-alone mobile vision applications:

- These devices exhibit *rapid, jittery, fast and unconstrained motions*. Most existing computer vision algorithms cannot deal with imagery collected under such extreme motions.
- Depending on the application, it may be required that the (camera) nodes in a camera network establish a *common coordinate system*. In case of mobile devices, no existing calibration algorithms can be readily applied to estimate the extrinsic parameters of a mobile device, in part because the extreme and unknown motions these devices go through. The method presented in [25] employs four reference points (correspondences) whose locations are encoded by means of some location

sensor (such as, an ultra-sound Cricket receiver). Four low-resolution pictures (taken from different viewpoints) of a reference point are sufficient to estimate the location and the orientation of a visual sensor node. There are, however, several issues with this approach, including the requirement of having this large number of reference points to be present in the scene and the quality or accuracy of the calibration.

- For the reasons of the above point, the nodes within the camera network also *need a common clock* for the purposes of collaborative sensing. A common clock allows information captured at multiple nodes to be fused together to construct a more complete picture of the event in question. Even in the absence of fine-grained time synchronization, it is often desirable to (time) order lower-level events to exploit causal relationship and infer higher-order event detection. An interesting paper on this topic is [36] where the issues related to the time synchronization in ad-hoc wireless sensor networks are studied. An important result reported in [36] is that classical clock synchronization algorithms are unsuitable for wireless sensor networks for two reasons: (1) limited communication range of each node and (2) high mobility of these nodes. [36] addresses these problems as follows: it does away with synchronizing clocks at each node. Rather it generates timestamps using (unsynchronized) local clocks. These timestamps are shared between nodes. The timestamps are transformed to the local time of the receiving node. While this method works well for ad hoc wireless sensor networks, it might not be straightforward to deploy it on camera networks comprising mobile devices. As stated earlier, mobile devices can exhibit a very high degree of mobility, which can make it difficult to ensure that the assumption of connection between two nodes for the time necessary for the complete exchange of sync messages holds under all conditions.
- Mobile devices are unique in how these are used. A typical (camera) node in a camera network is subservient to the network. A mobile device, on the other hand, is there to serve the need of its user. The user might choose to use this device to act as a part of a larger camera network. Similarly, a user might remove this device from the camera network without any notice. A mobile device is not always on. Even if it is turned on, it may not be pointing the right direction, the user may not want to use it to record the events that are of interest to the camera network, the user may be doing some else that is totally unrelated to the sensing or processing requested by the camera network, etc.. This *dynamic availability of the phone camera* is a crucial challenge that must be addressed before we can integrate mobile devices into camera networks. One possibility is to devise ways to inform the user that the mobile device is needed by the camera network to carry out sensing and processing.
- Another challenge of using mobile devices in camera networks has to do with their motion profile. As mentioned elsewhere in this chapter, mobile devices exhibit unpredictable and extreme motions, so it is often not easy to estimate the network topology. Centralized processing, perhaps, represents a good starting point for estimating network topology when mobile devices are integrated into camera networks. Another option might be setup spatially-oriented clusters, where different

mobile devices are clustered together based upon their locations as estimated by GPS- or WIFI-based localization.

5.7 Conclusions

Mobile devices are an inseparable part of our society now. In particular, due to the advances in smart and micro technology, camera sensors are now a standard component in all mobile devices. This innovation has attracted both the research community and the industries that aim to build advanced applications that exploit the powerful features of mobile devices and combine them with the more advanced computer vision and image processing techniques. In this chapter, we contributed to the research in distributed smart cameras introducing the most relevant advancements in mobile computer vision, then we discussed the main challenges that the community for such real-time distributed smart systems is facing at this time. We also introduced two main research studies where computer vision techniques and distributed frameworks are used for mobile applications of augmented reality and security purposes. Finally, we discuss the next challenges in mobile vision where the mobile devices become part of a visual sensor network.

References

1. Agrawal M, Konolige K, Blas MR (2008) CenSurE: center surround extremas for realtime feature detection and matching. In: Forsyth D, Torr P, Zisserman A (eds) European conference on computer vision. Lecture notes in computer science, vol 5305. Springer, Berlin, pp 102–115. doi:10.1007/978-3-540-88693-8
2. Alahi A, Ortiz R, Vandergheynst P (2012) FREAK: fast retina keypoint. In: International conference on computer vision and pattern recognition, pp 510–517. doi:10.1109/CVPR.2012.6247715, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6247715
3. Amazon: SnapTell (2007) http://www.A9.com
4. Bay H, Ess A, Tuytelaars T, Van Gool L (2008) Speeded-up robust features (SURF). Comput Vis Image Underst 110(3):346–359. doi:10.1016/j.cviu.2007.09.014, http://linkinghub.elsevier.com/retrieve/pii/S1077314207001555
5. Bay H, Fasel B, Gool LV (2006) Interactive museum guide: fast and robust recognition of museum objects. In: International workshop on mobile vision
6. Bolliger P, Köhler M, Römer K (2007) Facet: towards a smart camera network of mobile phones. In: 1st international conference on Autonomic computing and communication systems, p 17
7. Bosch A, Zisserman A, Munoz X (2007) Image classification using random forests and ferns. In: International conference on computer vision, pp 1–8. doi:10.1109/ICCV.2007.4409066
8. Brown M, Lowe DG (2006) Automatic panoramic image stitching using invariant features. Int J Comput Vis 74(1):59–73. doi:10.1007/s11263-006-0002-3
9. Bruns E, Bimber O (2008) Adaptive training of video sets for image recognition on mobile phones. Pers Ubiquitous Comput 13(2):165–178. doi:10.1007/s00779-008-0194-3
10. Davison AJ, Reid ID, Molton ND, Stasse O (2007) MonoSLAM: real-time single camera SLAM. IEEE Trans Pattern Anal Mach Intell 29(6):1052–67. doi:10.1109/TPAMI.2007.1049, http://www.ncbi.nlm.nih.gov/pubmed/17431302

11. Fritz G, Seifert C, Paletta L (2006) A mobile vision system for urban detection with informative local descriptors. In: International conference on computer vision systems, pp 30–30. doi:10.1109/ICVS.2006.5, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1578718
12. Girod B, Chandrasekhar V, Chen D, Cheung NM, Grzeszczuk R, Reznik Y, Takacs G, Tsai S, Vedantham R (2011) Mobile visual search. IEEE Sig Process Magazine 28(4):61–76. doi:10.1109/MSP.2011.940881, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5888642
13. Google: Google Goggles (2009). http://www.google.com/mobile/goggles
14. Greene K (2006) Hyperlinking reality via phones. MIT Technology Review, USA
15. Gruber L, Richter-Trummer T, Schmalstieg D (2012) Real-time photometric registration from arbitrary geometry. In: 2012 IEEE international symposium on mixed and augmented reality (ISMAR), pp 119–128. doi:10.1109/ISMAR.2012.6402548
16. Hagbi N, Bergig O, El-Sana J, Billinghurst M (2011) Shape recognition and pose estimation for mobile Augmented Reality. IEEE Trans Vis Comput Graph 17(10):1369–79. doi:10.1109/TVCG.2010.241, http://www.ncbi.nlm.nih.gov/pubmed/21041876
17. Hall SP, Anderson E (2009) Operating systems for mobile computing. J Comput Sci Coll 25(2):64–71
18. Hoff WA, Nguyen K, Lyon T (1996) Computervision-based registration techniques for augmented reality. In: Casasent DP (ed) Intelligent robots and computer vision XV, 2904, pp 538–548, doi:10.1117/12.256311, http://proceedings.spiedigitallibrary.org/proceeding.aspx?
19. IOnRoad: iOnRoad. http://www.ionroad.com/
20. Kooaba: Kooaba (2007). http://www.kooaba.com
21. LeafSnap: LeafSnap. http://www.leafsnap.com/
22. Lepetit V (2008) On computer vision for augmented reality. In: 2008 international symposium on ubiquitous virtual reality, pp 13–16 (2008). doi:10.1109/ISUVR.2008.10, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4568635
23. Leutenegger S, Chli M, Siegwart RY (2011) BRISK: binary robust invariant scalable keypoints. In: 2011 International conference on computer vision, pp 2548–2555. doi:10.1109/ICCV.2011.6126542, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6126542
24. Li Z, Yap KH (2012) Content and context boosting for mobile landmark recognition. In: IEEE Sig Process Lett 19(8):459–462. doi:10.1109/LSP.2012.2203120, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6213072
25. Liu X, Kulkarni P, Shenoy P, Ganesan D (2006) Snapshot: a self-calibration protocol for camera sensor networks. In: IEEE 3rd international conference on broadband communications, networks and systems, 2006 (BROADNETS 2006), pp 1–10
26. Lowe DG (2004) Distinctive image features from scale-invariant keypoints. Int J Comput Vis 60(2):91–110. doi:10.1023/B:VISI.0000029664.99615.94
27. Martinel N, Micheloni C (2012) Re-identify people in wide area camera network. In: 2012 IEEE computer society conference on computer vision and pattern recognition workshops, pp 31–36. IEEE, Providence, RI. doi:10.1109/CVPRW.2012.6239203, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6583277
28. Martinel N, Micheloni C, Foresti GL (2013) Robust painting recognition and registration for mobile augmented reality. IEEE Sig Process Lett 20(11):1022–1025. doi:10.1109/LSP.2013.2279014, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6604402
29. Martinel N, Micheloni C, Piciarelli C, Foresti GL (2013) Camera selection for adaptiveHuman-computer interface. In: IEEE transactions on systems, man, and cybernetics: systems pp 1–1. doi:10.1109/TSMC.2013.2279661, http://www.sciencedirect.com/science/article/pii/S0893608011002693
30. Micheloni C, Rani A, Kumar S, Foresti GL (2012) A balanced neural tree for pattern classification. Neural Netw 27:81–90. doi:10.1016/j.neunet.2011.10.007
31. Micheloni C, Remagnino P, Eng HL, Geng J (2010) Intelligent monitoring of complex environments. IEEE Intell Syst 25(3):12–14. doi:10.1109/MIS.2010.85

32. Miksik O, Mikolajczyk K (2012) Evaluation of local detectors and descriptors for fast feature matching. In: International conference on pattern recognition. Tsukuba, Japan, pp 2681–2684. ISBN:978-1-4673-2216-4
33. Neurotechnology: Neurotechnology Sentisight. http://www.neurotechnology.com/sentisight.html
34. Nokia: Nokia Point and Find (2006) https://betalabs.nokia.com/trials/nokia-point-and-find
35. Qualcomm: Qualcomm Vuforia. http://www.qualcomm.com/solutions/augmented-reality
36. Römer K (2001) Time synchronization in ad hoc networks. In: Proceedings of the 2nd ACM international symposium on Mobile ad hoc networking and computing, pp 173–182. ACM (2001)
37. Rosten E, Porter R, Drummond T (2010) Faster and better: a machine learning approach to corner detection. IEEE Trans Pattern Anal Mach Intell 32(1):105–19, doi:10.1109/TPAMI.2008.275, http://www.ncbi.nlm.nih.gov/pubmed/19926902
38. Rublee E, Rabaud V, Konolige K, Bradski G (2011) ORB: an efficient alternative to SIFT or SURF. In: IEEE International conference on computer vision. Barcellona, Spain, pp 2564–2571. doi:10.1109/ICCV.2011.6126544, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6126544
39. Schroth G, Huitl R, Chen D, Abu-Alqumsan M, Al-Nuaimi A, Steinbach E (2011) Mobile visual location recognition. IEEE Sig Process Mag 28(4):77–89. doi:10.1109/MSP.2011.940882, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=5888650
40. Seifert C, Paletta L, Jeitler A, Hödl E, Andreu JP, Luley P, Almer A (2004) Visual object detection for mobile road sign inventory. In: International symposium on mobile, human–computer interaction, pp 491–495. doi:10.1007/978-3-540-28637-0_63
41. Skrypnyk I, Lowe D (2004) Scene modelling, recognition and tracking with invariant image features. In: IEEE International symposium on mixed and augmented reality, pp 110–119. doi:10.1109/ISMAR.2004.53, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=1383048
42. Takacs G, Chandrasekhar V, Gelfand N, Xiong Y, Chen WC, Bismpigiannis T, Grzeszczuk R, Pulli K, Girod B (2008) Outdoors augmented reality on mobile phone using loxel-based visual feature organization. In: Proceeding of the 1st ACM international conference on Multimedia information retrieval—MIR'08, p 427. doi:10.1145/1460096.1460165, http://portal.acm.org/citation.cfm?doid=1460096.1460165
43. Takacs G, Chandrasekhar V, Girod B, Grzeszczuk R (2007) Feature tracking for mobile augmented reality using video coder motion vectors. In: IEEE international symposium on mixed and augmented reality, pp 1–4. doi:10.1109/ISMAR.2007.4538838, http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=4538838
44. W3C: Mobile Web Initiative Device Description Working Group Charter (2005) http://www.w3.org/2005/01/DDWGCharter/
45. W3C: W3C Extensible multiModal annotation markup language (2009) http://www.w3.org/TR/emma/
46. Wagner D, Reitmayr G, Mulloni A, Drummond T, Schmalstieg D (2010) Real-time detection and tracking for augmented reality on mobile phones. IEEE Trans Visual Comput Graph 16(3):355–68 (2010). doi:10.1109/TVCG.2009.99, http://www.ncbi.nlm.nih.gov/pubmed/20224132
47. Xu L, Oja E (1993) Randomized hough transform (RHT): basic mechanisms, algorithms, and computational complexities. CVGIP Image Underst 57(2):131–154. doi:10.1006/ciun.1993.1009, http://linkinghub.elsevier.com/retrieve/pii/S1049966083710090
48. Yeh T, Tollmar K, Darrell T (2004) Searching the web with mobile images for location recognition. In: IEEE International conference on computer vision and pattern recognition, vol 2, pp. 76–81. doi:10.1109/CVPR.2004.1315147
49. You B, Lane ND, Chen F, Wang R, Chen Z, Bao TJ, Cheng Y, Lin M, Torresani L, Campbell AT (2013) CarSafe App: alerting drowsy and distracted drivers using dual cameras on smartphones. http://now.dartmouth.edu/2012/09/dartmouth-smartphone-app-targets-driver-safety/