World Wide Web: Internet and Web Information Systems, 8, 495–517, 2005
© 2005 Springer Science + Business Media, Inc. Manufactured in The Netherlands.
DOI: 10.1007/s11280-005-0905-x

Searching for Flash Movies on the Web: A Content and Context Based Framework*

JUN YANG yangjun@acm.org
Department of Computer Engineering and Information Technology, City University of Hong Kong, HKSAR, China; Department of Computer Science and Engineering, Zhejiang University, Hangzhou, China, 310027

QING LI itqli@cityu.edu.hk
Department of Computer Engineering and Information Technology, City University of Hong Kong, HKSAR, China

LIU WENYIN csliuwy@cityu.edu.hk
Department of Computer Science, City University of Hong Kong, HKSAR, China

YUETING ZHUANG yzhuang@cs.zju.edu.cn
Department of Computer Science and Engineering, Zhejiang University, Hangzhou, China, 310027

Published online: 1 August 2005

Abstract

The phenomenal growth of online Flash movies in recent years has made Flash one of the most prevalent media formats on the Web. The retrieval and management issues of Flash, vital to the utilization of the enormous Flash resource, are unfortunately overlooked by the research community. This paper presents the first piece of work (to the best of our knowledge) in this domain by suggesting an integrated framework for the retrieval of Flash movies based on their content characteristics as well as contextual information. The proposed approach consists of two major components: (1) a content-based retrieval component, which explores the characteristics of Flash movie content at compositional and semantic levels; and (2) a context-based retrieval component, which explores the contextual information including the texts and hyperlinks surrounding the movies. An experimental Flash search engine system has been implemented to demonstrate the feasibility of the suggested framework.

Keywords: Flash movies, content and context based retrieval, content characteristics, contextual information, Flash search engine

1. Introduction

Flash, a new format of vector-based interactive movie set forth by Macromedia Inc. [16], is gaining popularity at a tremendous rate and has become one of the most prevalent media formats on the Web. Its popularity is manifested by the fact that Macromedia Flash Player, the presentation tool for Flash movies, is distributed among over 98% web browsers, a

*The work described in this paper was supported substantially by a grant (Project No. 7001457), and partially by another grant (Project No. 7001564), both from CityU of Hong Kong.

percentage largely exceeding that for Windows Media Player (69%), Real Player (51%), and QuickTime Player (35%) [17]. Moreover, statistics [17] also show that over 50% of the top 50 websites in the world has adopted Flash in their content. Flash movies[1] are primarily embedded into and delivered with web pages to enhance their interactive and multimedia features, and are also created as cartoons, commercial advertisements, electronic postcards, MTV movies, computer games, or even e-commerce, each of which has large market potentials. The success of Flash can be contributed to its compactness (for fast delivery), rich semantics (due to its vector-based format), ease of composition, and powerful interactivity, which collectively form a great advantage over its competing technologies such as GIF animations and streaming videos.

The attention Flash has aroused in the industrial sector constitutes a sharp contrast with the lack of study in the research community. Little research effort, to the best of our knowledge, has been devoted to the study of Flash movies, whereas extensive research has been conducted on other media formats such as image, video, or audio. On the other hand, the number of on-line Flash movies has grown to a point that makes manual access of user desired movies extremely expensive (if not impossible). Therefore, to utilize the enormous Flash resource, it is imperative to develop effective and efficient tools for searching Flash movies on the Web to satisfy user requirements. It can be expected that such tools will be welcomed by a variety of user groups, ranging from teenagers looking for Flash games, music fans seeking for MTV movies, to Flash developers reviewing the designs of existing movies, and customers searching for Flash advertisements. The call for Flash retrieval tools forms the motivation of this paper, in which we present the first framework (to our knowledge) for searching Flash movies from the Web based on content and context information. Meanwhile, please note that some online Flash galleries (e.g., Flash Kit [10]) have provided keyword-based search functions for manually annotated Flash movies, but their approaches are neither efficient nor adequate in supporting queries specifying the content of Flash movies.

The problem of Flash retrieval in the Web environment can be approached from two perspectives: internally from the *content characteristics* of Flash movies, as well as externally from their *contextual information*. Internally, a typical Flash movie usually contains heterogeneous media ingredients (graphics, images, sounds, etc), dynamic effects, and user interactions, which collectively synthesize and express the semantics of the movie. The characteristics of these expressive movie elements, when properly represented and indexed, are indispensable clues based on which Flash movies can be retrieved to meet user requests. Externally, most Flash movies are embedded in web pages (which are interweaved by hyperlinks into a large network) and thus surrounded by many other media objects (including text). As proved by the success of many commercial image search engines such as Google and Ditto [7], we can expect the performance of Flash retrieval to be largely improved by mining the contextual information (e.g., hyperlinks, surrounding text) of Flash movies as a complement of their content characteristics. In this paper, we investigate a synergy of both approaches by proposing an *integrated* framework for Flash retrieval that is based on both the content characteristics of Flash movies and their contextual information. Due to the intrinsic complexity of Flash and the heterogeneity of its context, this framework requires techniques from a wide range of fields spanning natural language processing, content-based retrieval, link analysis, spatio-temporal database, and data mining. While some of the required techniques are available in the existing work, many need to be developed and/or improved by us and inevitably in a series of follow-up research. Actually, rather than providing a detailed solution, the purpose of this paper is more on defining a generic "skeleton" that allows follow-up works to fill into its components, even though we have actually built a prototype system for Flash retrieval based on the proposed framework.

Our integrated framework for web-based Flash retrieval consists of a content-based retrieval component and a context-based retrieval component. The former deals with the indexing and query processing of Flash movies based on their content characteristics, which include the low-level, primitive features of their media ingredients, dynamic effects, and user interactions, as well as some high-level semantic features summarized from the primitive ones. The latter applies a network model derived from hyperlinks to describe the correlations among Flash movies, based on which relevant movies (of a user query) can be discovered through a proposed link mining algorithm. A combination of both retrieval methods allows a user query, expressed in the form of simple keywords or sophisticated search conditions, to be first processed by the content-based component to generate some initial results, which are then expanded and refined by the context-based component to obtain more and better-quality results.

The rest of the paper is organized as follows. Section 2 presents an overview of web-based Flash retrieval from both content-based and contextual perspectives. In Section 3, we introduce the integrated framework for content and context based Flash retrieval. Two major components of it, one for content-based Flash search and the other for context-based Flash mining, are detailed in Sections 4 and 5, respectively. Section 6 discusses a prototype Flash retrieval system implemented based on the proposed framework. The concluding remark and the future work are given in Section 7.

2. Web-based Flash retrieval: An overview of the research issues

In this section, we present a general discussion of the research issues regarding web-based Flash retrieval through the *content characteristics* of movies as well as their *contextual information*. The previous research efforts related to Flash retrieval are reviewed as well.

2.1. A content perspective

Most of the existing works on (multimedia) information retrieval rely on the content characteristics of the data to be retrieved: traditional text-based information retrieval (IR) utilizes keywords extracted from textual documents; content-based retrieval (CBR) explores the low-level features calculated from multimedia objects (image, video, sound, etc.). A close examination of Flash movies reveals that Flash has more heterogeneous and complicated intrinsic characteristics than other media formats. Hence, an effective Flash retrieval tool should investigate the rich movie content, which can be characterized from the following three aspects:

- *Heterogeneous media ingredients.* A typical Flash movie consists of ingredient media objects of various types. Texts and graphics of arbitrary complexity can be easily created and inserted as movie ingredients using the Flash authoring tools. Bitmap or JPEG images and QuickTime video clips can be imported into a Flash movie as well. Sounds compressed using ADPCM or MP3 standard are embedded in a movie in one of the two forms: event sound, which is played in response to certain event such as mouse-click, and streaming sound, which is played in synchronization with the advance of the movie. According to the format of Flash [18], all these ingredients are encoded separately in the data file of a Flash movie (SWF file) such that they can be easily recognized and extracted. This differs fundamentally from pixel-level media formats such as image and video. Moreover, a Flash movie can recursively consist of embedded Flash movies, which are defined as a special type of ingredients. An embedded movie can have its own ingredients and support their dynamic effects.
- *Dynamic effect.* A Flash movie is composed of a sequence of frames that are played in a certain order subjected to user interactions. With the progression of frames, movie ingredients can be placed on the current frame, removed from it, or changed with respect to their positions, sizes, shapes, and angles of rotation. The spatio-temporal features of the movie ingredients, as well as the spatio-temporal relationships among them, make up of some high-level dynamic effects, such as morphing, motion, resizing, which also suggest/contribute to the meaning of a movie.
- *User interactivity*. In contrast to a passive media such as streaming video, a Flash movie can be an interactive one in the sense that a user can interfere with the presentation of the movie. As an example, by clicking a button on a movie frame the user can let the movie "jump" to a frame prior to the current frame, while clicking another button may cause the movie jump to a frame behind the current one. Consequently, an interactive Flash movie may have multiple presentations, with each of them being the result of a certain series of user behaviors. That also explains why Flash movies can be made into computer games which involve extensive user interactions.

From a user's point of view, the content characteristics on the three aspects mentioned above are important clues through which users may likely compose queries for Flash movies. For example, a user may search for the movies "*accompanied by the song* '*Yesterday*', movies "*containing the text* '*Lord of rings*", or movies "*describing the scene of a descending sun*". To support such queries, features need to be defined to represent the movie content characteristics, and retrieval models and query specification methods are required for searching movies based on these features. Unfortunately, since the intrinsic complexity of Flash movies surpasses that of any traditional media format (e.g., images are neither dynamic nor interactive; videos do not support interactivity), Flash retrieval cannot be fully addressed by the existing retrieval methods. Rather, it calls for a variety of new techniques to be devised for the indexing and query processing of the content characteristics mentioned above. For example, the modeling of user interaction in multimedia data, which is required for Flash retrieval, remains almost an unexplored area except for the work of Adali et al. [1].

From another perspective, however, since a Flash movie can be viewed as a diverse collection of traditional media objects, many existing retrieval methods can serve as the "component technologies" for Flash retrieval. Specifically, text ingredients in movies can be processed by traditional IR techniques [21], and the indexing of image, video, and audio ingredients can largely benefit from the various content-based retrieval techniques for images [20], videos [24], and audios [11]. Moreover, database queries using declarative query language such as SQL [8] can be designed and applied to the retrieval by structured features, such as the shape and color of graphic ingredients. Finally, the dynamic movie effects can be partially addressed by techniques on modeling spatial/temporal data (e.g., video), although the complexity of the dynamic effects supported by Flash exceeds the capability of these techniques. Examples of such techniques include the 2-D strings [3] and quad-trees [22] representation for spatial data, temporal database techniques surveyed in [19], and the video retrieval model using spatio-temporal attributes in the VideoQ system [4].

As a general rule, human users are inclined to conduct queries using high-level concepts, that is, to search desired data by their *semantic meanings* rather than their *compositional features*. The content characteristics described above do not directly reflect movie semantics and are therefore insufficient and inconvenient for users, especially non-professional users, to conduct effective queries. However, deriving the semantic meaning of a movie from its bits and pieces is probably a problem as hard as image understanding, despite the fact that as a vector-based media Flash conveys semantics at a higher level than that of pixel-based medias such as image or video. For example, it is easier to recognize a moving circle with orange color from a Flash movie than from a MPEG video, but it is extremely difficult to decide whether this circle represents a rising sun or a moving basketball. Nevertheless, it is still feasible to calculate a limited set of *semantic features* from the primitive features, such as category (MTV, game, cartoon, etc.), pace (fast, slow, static, etc.), and so on.

2.2. *A contextual perspective*

Besides their internal content, the external context of Flash movies provides important clues that can be utilized to facilitate the retrieval of Flash. We investigate the contextual information on two aspects: (1) the contextual text in the web pages where Flash movies are embedded, and (2) the hyperlinks among the web pages, as discussed below.

Most Flash movies are contained in and delivered with web pages. A Flash movie is embedded into a web page through a reference (in the HTML file) to the Flash Player as an external object, with the full path of the Flash data file (i.e., SWF file) given as the parameter. Visually, the movie occupies a rectangular area in the web page and is surrounded by the text, images, or other media objects in the web page, among which the text is of particular interest. The contextual text of a movie refers to the textual parts of a web page that are most descriptive to the meaning of the embedded movies. Typical contextual text includes the name of a movie file, the title of the web page, and most importantly, the text spatially adjacent to the movie according to the layout of the web page. A Flash retrieval tool can utilize these contextual texts to support keyword-based search of Flash movies. The effectiveness of this approach has been demonstrated by its extensive usage in both experimental retrieval systems, such as WebSEEK [23], and commercial image search engines, such as Google [12] and Ditto [7], which perform well by using contextual text only.

From a more global perspective, Flash movies are not only associated with web pages, but also interconnected with each other via the hyperlinks among web pages. As a general belief, a hyperlink suggests a certain degree of correlation between the two associated web pages. In the context of Flash retrieval, this observation can be generalized to indicate the relevance between Flash movies that are connected via hyperlinks and containment relationships between movies and web pages. Consequently, link analysis techniques [1,15] can be applied to explore the hyperlink structures to compute the (degree of) relevance between movies, which offers a complementary source of knowledge for Flash retrieval. In fact, link analysis technique has been extensively used in search engines recently. For example, Google [1] has applied this technique to find the most "authoritative" web pages out of the enormous number of relevant web pages returned from a single user query. PicASHOW [15], a prototype image search engine, relies purely on hyperlink analysis to deduce relevant images for user queries with the help of a traditional (textual) search engine. Both the contextual text and hyperlink structure are employed in our integrated Flash retrieval framework to be introduced in the next section.

3. An integrated framework for content- and context-based Flash retrieval

The content and context of Flash movies constitute two complementary sources of knowledge based on which web-based Flash retrieval can be conducted. The content characteristics are automatically extracted from movie files and can be directly matched against user queries. However, these characteristics suffer from the lack of direct associations with movie semantics, and the semantic features derived from them are also limited in descriptive power. As a consequence, it is reasonable to expect that the results obtained from content-based search are insufficient and inaccurate. This explains why all the commercial image search engines are not content-based, and all content-based image retrieval systems are still in laboratory stage. The drawback of content-based search can be partially remedied by the contextual text of movies, from which keywords are extracted to serve as the semantic descriptions of the movies. On the other hand, the hyperlink structure tells how Flash movies are related to each other, though it does not directly tell what each movie is about. This leads to the use of hyperlink structure in a post-processing step to discover more results (i.e., relevant movies) from a small set of known results.

In this section, we sketch an integrated framework for web-based Flash retrieval, which explores the combination of both types of knowledge and allows them to benefit each other for sufficient and accurate retrieval results. As illustrated in Figure 1, a set of *primitive, compositional features* is extracted from each movie to describe its media ingredients, dynamic effects, and the forms of user interactions in that movie. Moreover, *semantic features*, which include a keyword description and descriptors such as thematic category and pace, are derived from the low-level compositional features. Therefore, the compositional

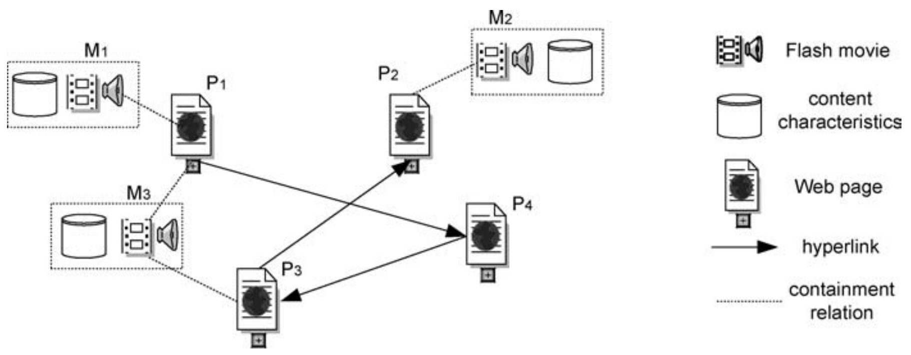


Figure 1: The integrated framework for web-based Flash retrieval.

features tell "what a movie has in it", while the semantic features answer "what a movie is about". The former is about "stuff", while the latter is about "meaning". Note that the keyword description of a movie is extracted not only internally from the text objects embedded in the movie, but also externally from the contextual text in the web page(s) containing that movie. Section 4.2 provides a detailed description on how these semantic features are extracted. The primitive features and the semantic features together form the *content characteristics* of a movie. At the global level, all the movies are interconnected by hyperlinks and containment relationships (between web pages and embedded movies) into a large network graph.

In this framework, a user query is constituted by one or more search conditions, each of them specifying a certain type of content characteristics (both primitive and semantic ones) of the desired movies. For example, a typical query may specify that the desired movies belong to the category of "Cartoon" and contain the keyword "Mother's Day". Internally, a user query is processed in two steps. In the first step, the search conditions of the query are matched against the content characteristics of each candidate movie in the database. The movies matched with a high confidence are regarded as "seed results" in the second step, where the hyperlinks among movies are explored to discover more and better results from the seed results. We elaborate on the technical details of each step in the following two sections.

4. Searching Flash by content

The content-based retrieval component of our approach is illustrated in Figure 2, which has a 3-tier architecture responsible for the representation, indexing, and query processing of Flash movies based on their content characteristics. The details of each layer are presented below and some sample queries are given to demonstrate the usefulness of the whole retrieval component. Interested readers can refer to our previous paper [26] for a more detailed description of the content-based retrieval component for Flash movies.

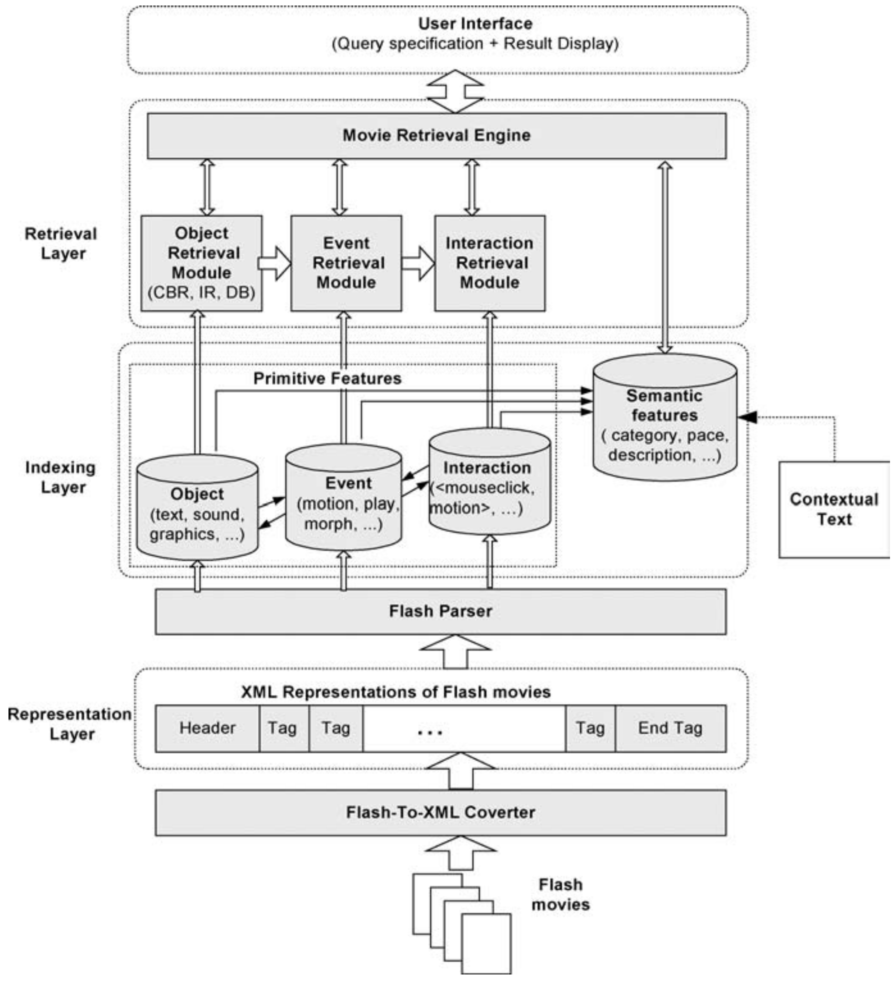

User Interface
(Query specification + Result Display)
Movie Retrieval Engine
Retrieval Layer
Object Retrieval Module (CBR, IR, DB)
Event Retrieval Module
Interaction Retrieval Module
Primitive Features
Semantic features (category, pace, description, ...)
Indexing Layer
Object (text, sound, graphics, ...)
Event (motion, play, morph, ...)
Interaction (<mouseclick, motion>, ...)
Contextual Text
Flash Parser
XML Representations of Flash movies
Representation Layer
Header
Tag
Tag
...
Tag
End Tag
Flash-To-XML Coverter
Flash movies

Figure 2: The content-based Flash retrieval component

4.1. XML-based representation (Representation layer)

Flash movies are delivered over the Internet in the form of Macromedia Flash (SWF) file. Each Flash file is composed of a series of tagged data blocks, which belong to different types with each type having its own structure. In essence, a Flash file can be regarded as an encoded XML [9] file (a Flash file is binary while an XML file is ASCII), and it can be converted into an XML file using tools like JavaSWF [14]. Each tagged data block in a Flash file is mapped to an XML tag, which usually has attributes and embedded tags representing the structured data inside the block. Data blocks of the same type are

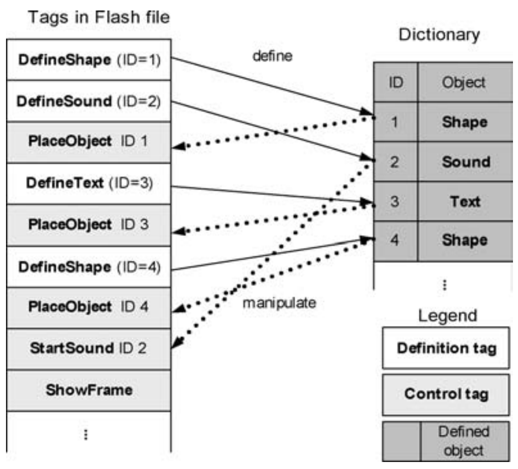


Figure 3: Structure of Macromedia Flash (SWF) file.

mapped to XML tags with the same name. In the representation layer, we convert Flash files into XML formats using Flash-To-XML Converter for two reasons: (a) XML files are readable and thus convenient for us to understand the internal structure of Flash; (b) XML format is a global standard facilitating interoperability with other Web-based applications.

There are two categories of tags in Flash files: *definition tags*, which are used to define various ingredients in a movie, and *control tags*, which are used to manipulate these movie ingredients to create the dynamic and interactive effect of the movie. For example, *DefineShape* and *DefineText* are typical definition tags, while *PlaceObject* (placing a movie ingredient on the frame) and *ShowFrame* (showing the current frame) are typical control tags. All the movie ingredients defined by definition tags are maintained in a repository called *dictionary*, from which control tags can access these ingredients for manipulation. The diagram shown in Figure 3 illustrates the interaction between definition tags, control tags, and the dictionary.

4.2. Content characteristics (Indexing layer)

4.2.1. Compositional features As discussed in Section 2.1, the content of a Flash movie can be characterized from three perspectives, namely, its heterogeneous media ingredients, dynamic effects, and the forms of user interactions. In the indexing layer of the content-based retrieval component, the characteristics of Flash on the three facets are modeled using the concepts of object, event, and interaction, respectively. Specifically, *object* represents movie ingredients as texts, videos, images, graphics, and sounds; *event* describes the dynamic effect of an object or multiple objects with certain spatio-temporal features; *interaction* models the relationships between user behaviors and events resulted from the behaviors. Naturally, these three concepts are at different levels: an event involves object(s) as the "role(s)" playing the event, and an interaction includes event(s) as the consequence of user behaviors. The features describing the objects, events, and interactions in a Flash movie, collectively regarded as the *primitive* part of the content characteristics, are extracted by the Flash Parser from the XML representation of the movie (cf. Figure 1). Despite the importance of temporal properties to a movie, we have not explicitly defined temporal features in our framework, though the definitions of events, interactions, and "pace" feature to be discussed later do involve temporal information. This is due to two reasons: (1) there are no very effective representations and retrieval methods for temporal features, and (2) because of (1), the chance that users issue queries based on sophisticated temporal features (instead of rough ones like "pace") is small. The formal descriptions of each object, event, and interaction are presented immediately below:

- *Object*. An object as a movie ingredient is represented by a tuple, given as:

$$object = < oid, o\text{-}type, o\text{-}feature >$$

where *oid* is a unique identifier of the object, *o-type* ∈{*Text, Graphic, Image, Video, Sound*} denotes the type of the object, and *o-feature* represents its features. Obviously, the particular types of features used to describe an object depend on the type of the object.

Table 1 summarizes the most commonly used features for each type of object, which are extracted from the corresponding definition tags in Flash files either directly or through some calculations. For instance, keywords and font sizes (indicative of the importance of text) can be directly obtained from a text object, whereas the shape of a graphic object has to be deduced from the coordinates of the lines and curves constituting the contour of the graphic object, provided that it is a simple shape such as rectangle or circle.

Table 1: Example features for various types of objects.

Object	Features
Text	Keywords, font size
Graphic	Shape, color, number of edges
Image	Size, color (histogram, coherence, etc.), texture (Tamura texture, wavelet, etc.)
Sound	MFCCs (mel-frequency cepstral coefficients)
Video	Features of a set of key-frames, motion vectors

Table 2: Example features and applicable objects of actions.

Action	Applicable objects	Features
Show	All but sound	Position, start/end frame
Motion	All but sound	Trail, start/end frame
Rotate	All but sound	Angle of rotation, location, start/end frame
Resize	All but sound	Start/end size, location, start/end frame
Morph	Graphic	Start/end shape, number of frame
Play	Sound, video	Current frame
Trace	All but sound	Closeness to mouse
Navigate	N/A	Target URL

The feature extraction techniques for each media type are widely available (see, e.g., [11,20]).

- *Event*. An event is a high-level summarization of the spatio-temporal features of object(s), denoted as:

$$event = < eid, \{action\}n > (n = 1, \dots, N)$$
$$action = < object, a\text{-}type, a\text{-}feature >$$

where *eid* is a unique identifier of the event, followed by a series of actions. Each action is a tuple consisting of an *object* involved as the “role” of the action, *a-type* as the type of the action, and *a-feature* as the attributes of the action. Each type of action is described by a particular set of features and can be applied to certain type(s) of objects (e.g., only graphic objects can be morphed). The relationships among an action type, its applicable objects, and its features (which are derived from control tags) are summarized in Table 2. Two action types require more explanation: (1) “trace” is the action of an object following the movement of the mouse cursor in the movie frame; (2) “navigation” is an action of a web browser being launched and directed to a specified URL, and therefore it does not involve any object.

Compared with the existing models [4,13,26], the concept of event provides a compact, semantics-flavored representation of spatio-temporal features, since the predefined action types directly address the human perception on the dynamic effects of a movie. On the other hand, it is also powerful in terms of expressiveness, mostly because an event can have multiple actions. For example, a graphic object that is moving and resizing simultaneously over frames can be modeled by an event consisting of two actions describing the motion and resizing of the object respectively. Such a multi-action event can be also used to model the recursively embedded movies in a main Flash movie (cf. Section 2). Although an embedded movie is defined by a definition tag *DefineSprite*, we model it as an event whose actions

Table 3: Example features for different interactions

Interaction	Features
Button	Event (press, release, mouse-over, mouse-click, mouse-up), position
Keyboard	Key code
Mouse	Action (drag, move, click, up), position

describe the dynamic features of its ingredients (which are modeled as objects). Another definition tag that is modeled as event is the *DefineMorph* tag, which is decomposed into a graphic object and an event describing the morph of this object.

- *Interaction*. The concept of interaction describes the relationship between user behavior and the event caused by the behavior. Its formal definition is given as follows:

$$interaction =< iid, i\text{-}type, \{event\}n, i\text{-}feature > (n = 1, \ldots, N)$$

where *iid*, *i-type*, and *i-feature* represent the identifier, type, and features of the interaction respectively, and $\{event\}n$ is a set of events triggered by the interaction. The type of interaction indicates the device through which a user action is conducted, including button, mouse, and keyboard. Button is a special movie ingredient for the purpose of interaction, and it responses to mouse and keyword operation as a normal button control does. Interactions involving buttons are classified as "button" interactions, even though they may also involve keyboard and mouse operations.

The features for each type of interaction are summarized in Table 3. For a button interaction, for example, an important attribute is the button event, such as mouse-over, mouse-click. Similar to even features, the features of interactions and the triggered events are extracted from the control tags of Flash files.

4.2.2. Semantic features The objects, events, and interactions describe the low-level characteristics of a Flash movie's content, i.e., "what the movie has". In comparison, semantic features describe or imply the meaning of a movie, i.e., "what the movie is about". Instead of having people to manually annotate the movies' semantic features, we derive them from the low-level compositional features. Given the difficulty of understanding movie semantics, however, the semantic features that are computable from the compositional features cannot reach the level at which human users would describe movie semantics. For example, it is very hard, if not impossible, to determine a scene of a person walking in the street merely from the numerous objects constituting the scene. However, these semantic features are still more amenable to users to compose effective queries than their low-level counterparts. For example, a query like "find movies containing a red circle and a green triangle" is of limited practical use, while a query like "find a *MTV* Flash movie with keywords ‘Heal the World’ ” can be much more meaningful and useful. Particularly, we consider the derivation of the following semantic features:

- *Keyword description*. Keyword is the most desirable mean through which average users can express their queries, as indicated by the extensive use of the keyword-based search method in nearly all commercial search engines. The keywords describing a Flash movie are collected from two sources. First, many movies contain text objects, from which keywords can be extracted after filtering out stop words and stemming. Here, the stop words include the common stop words in the area of information retrieval as well as those unique to Flash movies, such as “repeat”, “click”, and “play”. This source is not adequate since many movies do not have text objects, and even for those having text objects, keywords are not always extractable because of their graphics-based representation. As discussed in Section 2.2, a complementary source of keywords is from the contextual text in the web pages containing a movie. Specifically, within the web page containing a movie, we draw keywords from the title, anchor text, and the text within a certain distance from the movie. Keywords from the contextual text are added into the keyword description of the movie after having the HTML tags and stop words stripped. The keyword description is represented as a term vector where the keywords are weighted using TF $*$ IDF scheme.
- *Pace*. This feature refers to rate of the performance of a movie. It is a reflection of both the speed and the time span of the events being portrayed in a movie and has a direct influence on the user perception about the atmosphere, or mood, of the movie. The pace of a movie can be calculated as a function of the number and sizes of moving objects, the speed of their motion, the length (in terms of frame count) of the motion, etc., as detailed in [5]. For example, a Flash movie as a shooting game is likely to be full of objects moving violently throughout all the frames, which create a tense and excited feeling among users, while a MTV movie containing slow actions and nearly static scenes results in a calm and peaceful atmosphere. The pace feature calculated from these primitive features is converted into discrete values such as “violent”, “action”, “slow”, “static”, etc.
- *Category*. Flash movies can be classified into different categories according to their purposes, such as game, cartoon, MTV, commercial advertisements, etc. Since the movie category is a very useful clue for users to compose effective queries, automatic classification of Flash movies is highly desired. A close examination reveals that the typical movies of each category possess some distinct primitive features, through which the movies can be classified with reasonable accuracy. We conclude the features representative of each movie category through a statistical study of a large number of typical movies belonging to each category. For example, an MTV movie must contain a streaming sound throughout the movie but normally no event sound, and it usually has little interaction and a relatively large file size (over 500 KB). In contrast, an advertisement movie often contains the hyperlink to a company’s website, and its size is normally small. Due to the variety of the features, a rule-based classification method is insufficient. Therefore, we train a Naive Bayes classifier from pre-classified training movies and use it to classify new movies. The details of this work can be found at [6].

In conclusion, content characteristics of a Flash movie can be represented as the primitive compositional features of objects, events, and interactions, plus the semantic features including keyword description, pace, and category, i.e.,

$$movie =< object_m, \{event\}_n, \{interaction\}_t, description, pace, category >$$

Other semantic features can be defined as long as we know how to derive them from the primitive movie features. The retrieval of Flash movies based on these content characteristics is described in the next subsection.

4.3. Query processing (Retrieval layer)

As shown in Figure 2, the retrieval layer consists of three individual retrieval modules, which are used for matching objects, events, and interactions based on their respective features. Since user queries may involve movie features on multiple aspects, a movie retrieval engine is designed to decompose a user query into a series of sub-queries for objects, events, and interactions that can be processed by underlying retrieval modules, and then integrate and translate the results returned from these modules into a list of relevant movies. In addition, the movie retrieval engine also deals with user queries involving the semantic features of movies. In the following, we describe each retrieval module and the movie retrieval engine in detail. In particular, we define some high-level functions to summarize their functionalities.

- *Object retrieval module*. This module accepts an object type and object features as input, and returns a list of objects of the specified type that are ranked by their similarity to the given features. The retrieval process is summarized by the following function:

object-list : **SearchObject**(*o-type*, *o-feature*)

where *object-list* is a list of <oid, score> pairs with the score indicating the similarity of each object to the feature specified by parameter *o-feature*. If *o-feature* is not specified, all objects of the given type *o-type* are returned. Note that the "search space" of this operator covers all objects of every movie in the database. Thus, the returned objects may belong to different movies.

The type of features specified as search condition varies from one type of object to another. Even for the same type of object, diverse types of features can be used. For example, we can query for image objects either by submitting a sample image or by designating the dominant color as "red". Therefore, various retrieval techniques are needed to cope with different object features: cosine-similarity is used as the similarity metric for matching textual objects represented as term vectors; color histogram based similarity metric is used for image and video objects; DB-style retrieval is used for structured discrete features such as the shape of graphic objects. Due to the similarity-based matching for some objects, the returned objects are ranked in descending order of their similarities. If the matching is not similarity based, the results are ordered randomly.

- *Event retrieval module*. To search events, we need to specify search conditions for not only actions but also objects as the "roles" of the actions:

  event-list : **SearchEvent**(*a-type*, *a-feature*, *object-list*)

  This function returns a list of events having at least one action that satisfies all the following three conditions: (1) the type of the action is equal to *a-type*, (2) the feature of the action is similar to *a-feature*, and (3) the object involved in the action is within *object-list*. If either *a-feature* or *object-list* or both of them are not given, the returned events are those with at least one action satisfying conditions (1) and (3), (1) and (2), or only condition (1). The *event-list* is of the same structure with *object-list*, except that the elements in it are events. One point worth particular attention is that the ranking of events in *event-list* is fully determined by the similarity of their action features with *a-feature*, and is not subject to the ranking of objects in *object-list*. Moreover, since only one action can be specified in **SearchEvent**, the query for multi-action events is handled by firstly performing **SearchEvent** based on each desired action, and then finding the events containing all the desired actions by intersecting multiple *event-list* returned from **SearchEvent**.

- *Interaction retrieval module*. The retrieval of interactions is conducted by the following function:

  interaction-list : **SearchInteraction**(*i-type*, *i-feature*, *event-list*)

  The semantics of this function, its parameters, and its return value are similar to those of **SearchEvent**. The *event-list* specifies the scope of events at least one of which must be triggered in every interaction returned by this function. Similarly, to search for an interaction that causes multiple events, we need to perform this function for each desired event and integrating the results to find the interactions causing all the desired events.

- *Movie retrieval engine*. The results returned by individual retrieval modules are objects, events, and interactions, while the real target of the user queries is typically Flash movies. A primary function of the movie retrieval engine is to translate the retrieved objects (and events, interactions) into a list of relevant movies, as defined by the following function:

  movie-list : ***Rank***(*object-list* | *event-list* | *interaction-list*)

The movies in *movie-list* are those containing the objects in *object-list*, and their similarity scores (and therefore ranks) are identical to their corresponding objects in *object-list*. As an exception, if more than one object in *object-list* belongs to the same movie, the rank and similarity score of the movie are decided by the object with the highest rank. The **Rank** function taking *event-list* or *interaction-list* as parameters has similar syntax and semantics as the one taking *object-list* as the parameter.

Besides searching movies by their objects, events, and interactions, ordinary users are more likely to conduct their queries by specifying the semantic features of the desired movies. The movie retrieval engine also deals with such semantic queries using the function given below:

movie-list : **SearchMovie**(*keyword-set*, *category*, *pace*, *interactivity*)

where *keyword-set* is a set of keywords describing the desired movies, and *category*, *pace*, and *interactivity* are semantic descriptors specifying the desired movies. Not all parameters are required to be specified. A list of movies is returned with each one associated with a similarity score indicating the degree at which the movie satisfies the specified semantic features.

Moreover, it is common that a user query may specify multiple search conditions. To deal with such multi-condition queries, we need to merge multiple lists of movies retrieved based on each search condition into a single list giving the final ranking of similar movies. The **Merge** function is proposed for this purpose:

movie-list : **Merge**($\{$*movie-list*$\}_n$, $\{$*weight*$\}_n$)

where $\{$*movie-list*$\}n$ denotes n movie lists that are obtained based on different search conditions, and $\{$*weight*$\}n$ contains the weight indicating the relative importance of each condition, which is preferably specified by users. If not specified, all the weights are assumed to be 1. Each movie in the returned movie list must appear in at least one input list, and similarity score of the movie (and thus its rank) is determined by the weighted sum of its similarity score in each input list (if it is not in a particular list, its similarity there is assumed to be zero).[2] It does not enforce any temporal or other constraints. As an example, consider the movies from two input lists. Let the first list contain movies with event A, while the second list contain movies with event B, and the output of the merge function be movies with both A and B. Currently there is no mechanism in our framework to specify and enforce the temporal relations between A and B, such as “A proceeds A” or “A and B happens at the same time”.

4.4. Sample query processing

The functions defined above, when used in combination, can support rather sophisticated queries of Flash movies. In this subsection, we describe the processing of some sample queries to demonstrate the usage and expressive power of these functions. Note that all these queries target at realistic sample Flash movies we have collected from the Web.

Example 1 (Search by object). A user trying to find Flash movies about the film "Lion King" through a poster (as an image file 'lion-king.jpg') can issue his query as: *Find all Flash movies that contain images similar to a poster of the film "Lion King"*. This query can be processed as:

Rank(**SearchObject**(image, 'lion-king.jpg'))

Example 2 (Search by semantic features) A user who wants to search for MTV movies of a specific song, say, Joan Lennon's "Imagine", may express his query as: *Find all Flash movies that have keyword "John Lennon" and "Imagine" and belong to the "MTV" category.* This query can be handled by combining the results of a search based on the keyword and another search based on the song:

SearchMoive({"John Lennon", "Imagine"}, 'MTV')

Example 3 (Search by event) A query for movies containing the scene of "sunset" can be expressed as: *Find all Flash movies that contain a red circle descending from the top of the frame to the bottom.* The processing of this query requires specifying both the desired object and its dynamic effect, as formulated below

Rank(**SearchEvent**(motion, 'descending', **SearchObject**(graphic, 'red circle')))

Example 4 (Search by interaction, event, and keywords) Since Flash movies used as commercial advertisements sometimes contain the links to the companies, a query for Flash advertisements of, say, BMW cars, can be expressed as: *Find all movies that have keyword 'BMW' and a button by clicking which the BMW website will be opened in a web browser.* This query can be formulated and processed by a combination of several functions as follows:

Merge ({**SearchMovie**('BMW'),

Rank(**SearchInteraction**(button,'mouse-click',
SearchEvent(navigate, 'www.bmw.com')))})

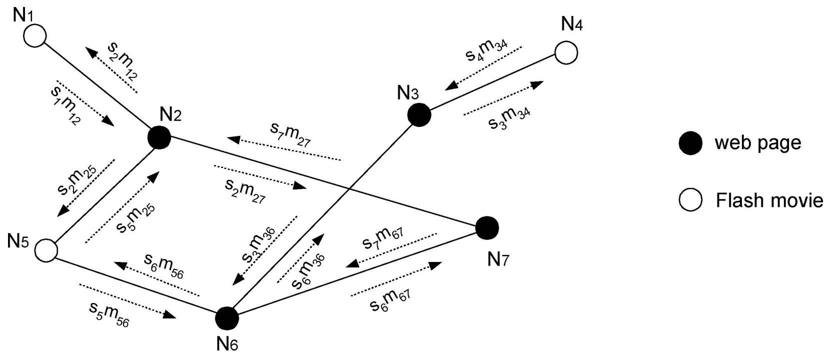


Figure 4: The illustration of the link mining algorithm.

5. Mining Flash by context

For a given user query, the results obtained from the content-based Flash search described in Section 4 can be insufficient and inaccurate, partially due to the fact that the content characteristics of movies are not adequately descriptive of movie semantics. As discussed in Section 2.2, a remedy to this problem is to explore the hyperlinks among web pages containing embedded Flash movies, which indicate semantic relevance among the movies. For this purpose, a *link mining* algorithm is devised to expand an initial set of results (which are returned by content-based search) to a more complete and accurate set of final results.

To illustrate the link mining algorithm, Figure 1 is converted and redrawn in the form of a network model as shown in Figure 4, where each node represents either a web page or a Flash movie, and each link denotes either a hyperlink between two web pages or the containment relation between a movie and a web page. The link mining algorithm is based on the intuitive notion of *similarity propagation*: the similarity (to the query) can be propagated from one node to another through the links between them. To expand and refine the results of a content-based search, we set the similarity of the node denoting each initial result to its similarity score obtained in the content-based search, and set the similarity of all other nodes to zero. Then, we let the similarity score to flow among the nodes through the links in the network. When the propagation finishes, the similarity flowed to each node defines its final similarity to the query.

Specifically, suppose the network can be represented by its adjacency matrix M, where each off-diagonal element mij is set to one if there is a link between node N_i and N_j, or to zero if there is no link between them. All the diagonal elements are set to zero. The propagation process is modeled over discrete steps $t = 0, 1, \ldots, T$. We define $S(t)$ as the similarity vector at step t, whose element $si(t)$ denotes the similarity of node Ni (a web page or a Flash movie) to the query in this step. In the initial vector $S(0)$, the elements corresponding to the movies as initial results are set to their similarity scores calculated in the content-based retrieval process, while other elements are set to zero. A single step of similarity propagation can be formulated as:

$$S(t+1) = [\alpha M + (1-\alpha) I]\, S(t) \quad t = 0, 1, \ldots, T$$

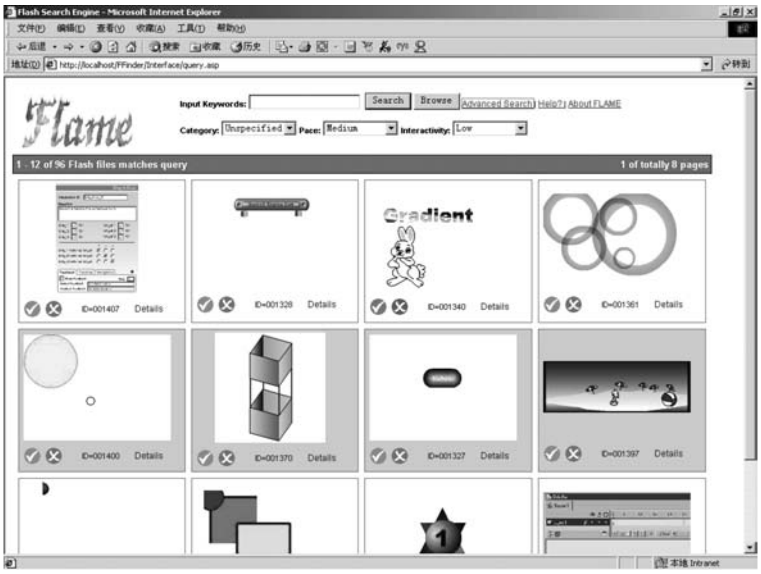
where I is an identity matrix, and α is a parameter in the range of [0,1], which determines the amount of similarity that flows through the links, with larger α corresponding to a larger flow of similarity. In our implementation, α is set to 0.4, and T is set to 10 in order to gather sufficient results. A single round of similarity propagation is illustrated in Figure 4, where the flows of similarity are denoted by dashed arrows. The number of propagation iterations strikes the balance between the computational complexity of the algorithm and the scope and quality of the retrieval result. More iterations of propagation usually generate a larger number of results highly relevant to the query, but also entail greater computational cost. The similarity vector $S(N)$ after the propagation process gives the final similarity of all the nodes (including the nodes denoting Flash movies) to the user query.

The link mining algorithm expands a set of "seed results" matched from content-based search to discover additional results, which are not matched based on their content characteristics but potentially relevant to user queries. Although it undoubtedly improves the coverage of the retrieval results, the quality of the additional results depends on two factors, namely, the accuracy of the seed results as well as the quality of the links being explored. Naturally, seed results of high relevance to a query and hyperlinks between relevant web pages usually lead to high-quality results. In practice, we only use the results matched with high confidence in the content-based retrieval phase as the input of the link mining algorithm, so as to improve the quality of the results. However, so far there is no mechanism to evaluate the quality of hyperlinks in terms of the degree of the semantic relevance between the associated web pages.

6. An experimental Flash search engine

To demonstrate the feasibility of our proposed framework, an experimental Flash search engine named FLAME has been implemented, which is introduced in this section.

According to Section 4.3, our approach supports retrieval of Flash movies based on their semantic features including keyword description, category, and pace, or their primitive features of media ingredients, dynamic effects, or user interactions, or a combination of them. This great variety poses a challenge on user interface design. A good user interface should allow users to compose various types of queries conveniently, as well as to display the retrieved Flash movies in an appropriate layout. The interface of our experimental system is displayed in standard web browsers and accessed remotely over the Internet. In order to achieve good visual experience, we allocate various query functions into two separate interfaces. As shown in Figure 5, the main interface supports Flash retrieval based on semantic features only, where a user can input query keywords and specify the category, and pace of the desired movies. The specified search conditions are matched with the corresponding semantic features of each movie, and the matched movies are used as


Flash Search Engine - Microsoft Internet Explorer

http://localhost/FFinder/Interface/query.asp

Input Keywords:

Category: Unspecified Pace: Medium Interactivity: Low

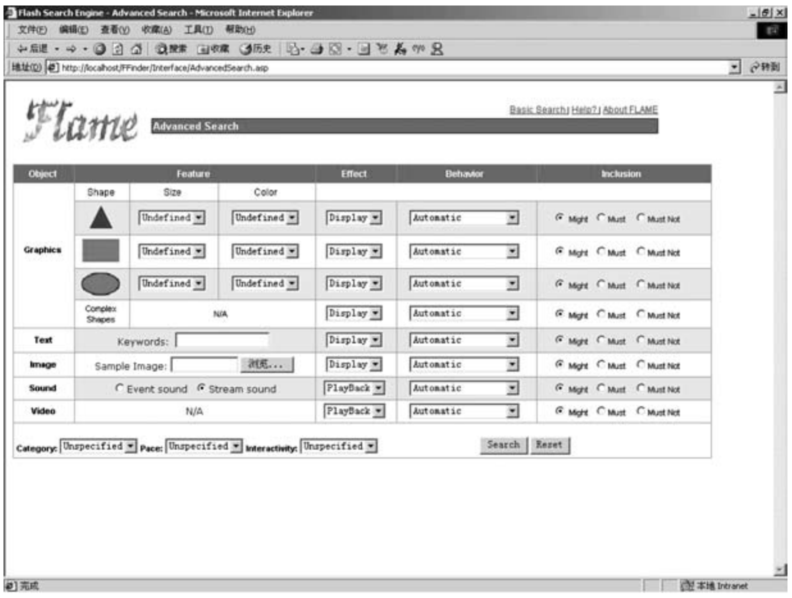
1 - 12 of 96 Flash files matches query 1 of totally 8 pages

Figure 5: The main interface of FLAME.

"seeds" to discover more results using the link mining algorithm presented in Section 5. The "thumbnails" of the retrieved movies (i.e., movies with condensed sizes) ranked in descending order of their similarity to the query are displayed in the lower pane of the interface. Below each "thumbnail" is a hyperlink labeled with "Details", by clicking which the corresponding movie is shown with its original size in a separate window.

The main interface contains a hyperlink labeled as "Advanced Search" pointing to the second interface (see Figure 6), which allows users to compose more sophisticated queries by specifying both the semantic features and the primitive features of the objects, events, and interactions in the desired movies. Since primitive features are more visual oriented than their semantic counterpart, the visualization of query specification methods is essential to the convenience and productivity of users. In the "Advanced Search" interface, users can specify various movie ingredients (including text, images, graphics, videos, sounds) appearing in desired movies with respect to their features, dynamic effects, and user behaviors triggering the effects. Different features are specified in different manners. For example, graphic objects are specified by their shape (chosen from a list of shape icons), size (chosen from a drag-down box), and dominant color (chosen from a color palette), while the features of images are specified by submitting a sample image. The desired dynamic effects of each ingredient in the movie can be designated using the drag-down box in the column labeled as "Effect", and the user behavior triggering the event can be specified using the drag-down box in the column labeled as "Behavior".

This interface arrangement is based on the belief that keyword and other semantic descriptors are the most natural and preferred ways for users to express their requests,


Flash Search Engine - Advanced Search - Microsoft Internet Explorer

http://localhost/FFinder/Interface/AdvancedSearch.asp

Basic Search | Help? | About FLAME

Advanced Search

Object	Feature			Effect	Behavior	Inclusion
	Shape	Size	Color			
Graphics	▲	Undefined	Undefined	Display	Automatic	[✓ Might] Must Must Not
	■	Undefined	Undefined	Display	Automatic	[✓ Might] Must Must Not
	⬬	Undefined	Undefined	Display	Automatic	[✓ Might] Must Must Not
	Complex Shapes	N/A		Display	Automatic	[✓ Might] Must Must Not
Text	Keywords:			Display	Automatic	[✓ Might] Must Must Not
Image	Sample Image:			Display	Automatic	[✓ Might] Must Must Not
Sound	Event sound [✓ Stream sound]			PlayBack	Automatic	[✓ Might] Must Must Not
Video	N/A			PlayBack	Automatic	[✓ Might] Must Must Not

Category: Unspecified Pace: Unspecified Interactivity: Unspecified Search Reset

Figure 6: The advanced search interface of FLAME.

a "rule of thumb" proved by many commercial search engines. Meanwhile, providing sophisticated query possibilities in a separate interface allows professional users to conduct complicated and powerful queries without confusing non-professional users.

In our current work, we have not conducted a quantitative evaluation of the retrieval performance (e.g., in terms of precision and recall) of the experimental system for two reasons: (1) there is no Flash collection yet serving as standard dataset for testing retrieval systems (indeed to the best of our knowledge, FLAME is the first retrieval system developed specifically for Flash); (2) due to the intrinsic complexity of Flash, there are likely a variety of subjective criteria regarding the relevance of Flash movies to a given user query. Therefore, it is very difficult to define an objective "ground truth" for the test dataset. Nevertheless, we have conducted some preliminary experiments to evaluate certain aspects of the system, such as the effectiveness of semantic features, the processing time, the details of these experiments can be found in [5,27].

7. Conclusions and future work

This paper has investigated the problem of web-based Flash retrieval, which is critical to the better utilization of the proliferating online Flash resource but has been unfortunately overlooked by the research community. In this paper, we have presented an integrated framework for Flash retrieval based on both the content and the context of Flash movies. The presented framework consists of two components: a content-based retrieval component, which relies on the content characteristics of movies at compositional and semantic levels, as well as a context-based retrieval component, which explores the contextual information such as the texts and hyperlinks surrounding the movies. An experimental Flash search engine system has been implemented based on the suggested framework.

Although this paper covers a broad range of research issues, there remains much room for future research regarding Flash retrieval and management. In particular, the feature sets used to describe Flash movies can be expanded to include more sophisticated ones such as temporal and spatial features. We also plan to investigate the role of human-computer interaction for better management and retrieval of Flash. A foreseeable work is to adopt relevance feedback technique on Flash retrieval to enhance the retrieval performance based on user evaluations. On the other hand, the research on Flash retrieval can also be generalized to the retrieval of other types of multimedia representations largely existing on the Web, such as PowerPoint, SMIL [25], etc.

Notes

1. If not indicated explicitly, we use "movie" to refer to "Flash movie" in this paper.
2. Note that this function implements only the simplest method of merging multiple similarities, further discussion of which is itself a separate research topic and is beyond the scope of this paper.

References

[1] S. Adali, M. L. Sapino, and V. S. Subrahmanian, "An algebra for creating and querying multimedia presentations," *ACM Multimedia Systems* 8(3), 2000, 212–230.

[2] S. Brin and L. Page, "The anatomy of a large-scale hypertextual web search engine." in *Proc. of the 7th Int. World Wide Web Conf.*, 1998, pp. 107–117.

[3] S. K. Chang, Q. Y. Shi, and C. Y. Yan, "Iconic indexing by 2-D strings," *IEEE Tran. Pattern Anal. Machine Intell.* 9(3), 1987, 413–428.

[4] S. F. Chang, W. Chen, H. J. Meng, H. Sundaram, and D. Zhong, "VideoQ: An automated content based video search system using visual cues," in *Proc. ACM Int. Multimedia Conf.*, 1997, pp. 313–324.

[5] D. W. Ding, J. Yang, Q. Li, L. P. Wang, and W. Y. Liu, "Towards a flash search engine based on expressive semantics," in *Proc. of 13th Int'l Conf. on World Wide Web*, 2004, pp. 472–473.

[6] D. W. Ding, J. Yang, Q. Li, L. P. Wang, and W. Y. Liu, "Automatic detection of flash movie genre using bayesian approach" in *Proc. of IEEE Int'l Conference on Multimedia and Expo (ICME)*, Taipei, Tawan, 2004 (CDROM).

[7] Ditto image search engine http://www.ditto.com

[8] R. Elmasri and B. Navathe, Fundamentals of Database Systems, 2nd Edition. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994.

[9] Extensible Markup Language (XML) http://www.w3.org/XML/

[10] Flash Kit http://www.flashkit.com/index.shtml

[11] J. Foote, "An overview of audio information retrieval," *ACM Multimedia Systems* 7, 1999, 2–10.

[12] Google http://www.google.com

[13] V. N. Gudivada and Raghavan, "Design and evaluation of algorithms for image retrieval by spatial similarity," *ACM Trans. Information Systems* 13(2), 1995.

[14] JavaSWF. http://www.anotherbigidea.com/javaswf/

[15] R. Lempel and A. Soffer, "PicASHOW: Pictorial authority search by hyperlinks on the web," in *Proc. 10th Int. World Wide Web Conf.*, 2001, pp. 438–448.

[16] Macromedia, Inc., www.macromedia.com

[17] Macromedia Flash Player adoption statistics www.macromedia.com/software/player_census/flashplayer

[18] Macromedia Flash File Format (SDK). http://www.macromedia.com/software/flash/open/licensing/fileformat/

[19] G. Ozsoyoglu and R. Snodgrass, "Temporal and real-time databases: A survey," *IEEE Trans. on Knowledge and Data Engineering* 7(4), 1995, 513–532.

[20] Y. Rui, T. Huang and S. Chang, "Image retrieval: current techniques, promising directions and open issues," *Journal of Visual Communication and Image Representation* 10, 1999, 1–23

[21] G. Salton, and M. J. McGill, *Introduction to Modern Information Retrieval*. McGraw-Hill Book Company, 1983.

[22] H. Samet, "The quadtree and related hierarchical data structures," *ACM Computing Surveys* 16(2), 1984, 187–260.

[23] J. R. Smith, and S. F. Chang, "Visually searching the web for content," *IEEE Multimedia Magazine* 4(3), 1997, 12–20.

[24] S. W. Smoliar and H. J. Zhang, "Content based video indexing and retrieval," *IEEE Multimedia* 1, 1994, 62–72.

[25] Synchronized Multimedia Integration Language (SMIL) http://www.w3.org/AudioVideo/

[26] J. Yang, Q. Li, W. Y. Liu, and Y. T. Zhuang, "Search for flash movies on the web," Prof. of the 3rd Int'l Conf. on Web Information Systems Engineering, *Workshop on Mining for Enhanced Web Search*, Singapore, 2002.

[27] J. Zhai, J. Yang, Q. Li, Liu W. Y., and B. Feng. "Rich media retrieval on the web—A multi-level indexing approach," in *Proc. of 12th Int'l World Wide Web Conf.*, 20–24 May 2003, Budapest, Hungary.