This paper is published in the 3rd International Conference on Web Information Systems Engineering, Singapore, 2002.

Search for Flash™ Movies on the Web

Jun Yang[1,3] Qing Li [1] Liu Wenyin [2] Yueting Zhuang [3]

[1] *Dept. of Computer Engineering and Information Technology*
[2] *Dept. of Computer Science*
City University of Hong Kong, HKSAR, China
yangjun@acm.org {itqli, csliuwy}@cityu.edu.hk

[3] *Dept. of Computer Science*
Zhejiang University
Hangzhou, China, 310027
yzhuang@cs.zju.edu.cn

Abstract

*Flash™ is experiencing a breathtaking growth and has become one of the prevailing media formats on the Web. Unfortunately, no research effort has been dedicated to automatic retrieval of Flash movies by content, which is critical to the utilization of the enormous Flash resource. A close examination reveals that the intrinsic complexity of a Flash movie, including its heterogeneous components, its dynamic nature, and user interactivity, makes Flash retrieval a host of research issues. As the first endeavor in this area, we propose a generic framework termed as FLAME (*FLash Access and Management Environment*) embodying a 3-tier architecture that addresses the representation, indexing, and retrieval of Flash movies by mining and understanding of movie content. In particular, FLAME features a unique* multi-level indexing and retrieval *approach that supports characterization and retrieval of Flash at (1) object level, which describes the heterogeneous components embedded in a movie, (2) event level, which depicts the movie's dynamic effects constituted by the spatio-temporal features of objects, and (3) interaction level, which models the relationships between user behaviors and the consequential events. An experimental prototype for Web-based Flash retrieval is implemented to verify the feasibility and effectiveness of FLAME.*

1. Introduction

Flash™ is a new format of vector-based interactive movie proposed by Macromedia Inc. [13], which can be embedded in web pages and delivered with them over the Web. Since its birth in 1997, Flash has experienced an explosive growth and become one of the most prevalent media formats on the Web. According to the statistics [14], by April, 2002 there are over 440 million Internet users worldwide that can view Flash movies using Macromedia Flash Player, the presentation tool of Flash. The percentage of Web browsers in which Flash Player has been installed is 98%, compared with 86% for Java, 69% for Media Player, and 51% for Real Player. Among the top 50 websites in United States, 58% has adopted Flash movies, including AOL Time Warner Network, Yahoo!, and MSN Sites. This percentage in the global scale is 50%. Flash can be used for a variety of purposes, among which the most prominent use is to enhance the interactive and multimedia feature of static, textual websites. Besides that, Flash movies are being created as cartoons, commercial advertisements, electronic postcards, MTV movies, computer games, or even electronic commerce and interactive media, each of which has huge market potentials. The huge success of Flash can be contributed to several unique features, including small size (for fast delivery), easy composition, rich semantics (due to vector-based format), powerful interactivity, which constitute a great advantage over its competing technologies on the Web, such as streaming video and java script. Thus, the popularity of Flash is likely to persist in the future even if the global deployment of broadband offsets its advantage of compactness.

Given the popularity of Flash and its promising future, it becomes an imperative task of the multimedia community to develop indexing and retrieval technology for Flash movies in order to fully utilize the enormous Flash resource on the Web. Such technology will become an integral part of the Web search engine, not only because Flash movies are prevalent in numbers on the Web, but also because very often they convey the major (or entire) semantics of web pages. (In fact, sometimes web pages are merely the containers of Flash movies). It can be foreseen that Flash retrieval tools will be needed by a variety of user communities, ranging from teenagers looking for Flash games, music fans looking for MTVs, to Flash developers reviewing the designs of existing movies, and customers searching for Flash advertisements. Unfortunately, although previous research has addressed the retrieval of various media types on the Web (e.g., text, image, video), some of which may not even be as popular as Flash, there is no work on the indexing and retrieval of Flash movies in the research community (to the best of knowledge). Note that this claim does not conflict with the fact that some online Flash repositories (e.g., Flash Kit [7]) do provide users with simple search functions based on manual classification and annotation. This approach does not scale to large data collection (as it requires manual efforts), nor does it investigate the rich content of a movie, which is indispensable for understanding movie semantics and evaluating movies towards user queries. In comparison, our motivation is to present the first piece of work (again, to our knowledge) on fully automatic Flash retrieval technology based on the mining and understanding of movie content.

A detailed anatomy of Flash movie reveals its intrinsic complexity on three major aspects: (1) a typical Flash movie usually contains *heterogeneous components*, including texts, graphics, images, QuickTime™ videos, sounds, and even recursively embedded Flash movies; (2) it features *dynamic nature* that is constituted by the spatio-temporal features of its components; and (3) it enables *interactivity* which allows users to interfere with the playout of the movie. Given this intrinsic complexity, Flash retrieval is likely to be more complicated than and thus cannot be addressed by the retrieval technology for any existing media (e.g., text, image, video); rather, Flash retrieval may employ a synergy of various existing technologies to provide the "ingredients" of the whole retrieval framework, such as content-based retrieval technology (for media components), information retrieval technology (for text component), spatio-temporal indexing techniques, Web mining techniques [4], etc,

In this paper, we propose a generic framework termed as FLAME (*FLash Access and Management Environment*) for users to search Flash movies based on various content characteristics

and using diverse retrieval approaches. FLAME has a 3-tier architecture that addresses the representation, indexing, and retrieval of the Flash content. A unique *multi-level indexing and retrieval* approach is developed in FLAME to support query of Flash movies at different level of details, including (1) heterogeneous components embedded in movies, (2) dynamic effects constituted by the spatio-temporal features of the components, and (3) the relationships between user interactions and the consequential effects. Regarding the sophistication of FLAME (which is due to the complexity of Flash), it is unrealistic to address all the components with satisfactory solutions within the scope of this paper. In fact, the objective of this paper is to come up with a comprehensive "skeleton" such that many follow-up works will be devoted to "fill in" the components of this skeleton. Nevertheless, to validate the feasibility and effectiveness of FLAME, we have also implemented a prototypical Flash retrieval system as an "instance" of the framework.

The rest of the paper is organized as follows. In Section 2, we present an overview of Flash retrieval, addressing the characteristics of Flash, the research issues, and its connection with previous works. We elaborate on the proposed FLAME framework in Section 3. The prototypical system for Web-based Flash retrieval is described in Section 4. The conclusion is given and promising future directions are suggested in Section 5.

2. Flash retrieval: an overview

In this section, we describe the unique properties of Flash and the research issues of Flash retrieval aroused by these properties. We also discuss how Flash retrieval is related to previous works on multimedia retrieval and how it can benefit from them.

2.1. Characteristics of Flash movies

Through a detailed anatomy of the content of several sample movies, we discover that the semantics of a Flash movie is mainly synthesized and conveyed by the following three types of devices:

- *Heterogeneous components*. A typical Flash movie usually consists of component media objects of various types. Texts and graphics (i.e., drawings) of arbitrary complexity can be easily created as components using authoring tools of Flash. Bitmap or JPEG images and QuickTime video clips can be imported into the movie as well. Sounds compressed using ADPCM or MP3 standard are embedded in the 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 [15], all these components are encoded separately such that they can be easily extracted from Flash data files. This differs fundamentally from pixel-level media formats such as image and video. Furthermore, a Flash movie can consist of recursively embedded Flash movies, which are defined as a special type of components. An embedded movie component can also consist of its own components and support dynamic effects of them.
- Dynamic effect. A Flash movie is composed of a sequence of frames that are played in an order subject to user interactions. With the progression of frames, components 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 components, as well as the spatio-temporal relationships among them, make up of some high-level dynamic effects (such as morphing, motion) that suggest the semantic meaning of a movie.
- *User interactivity*. Rather than a passive media such as streaming video, Flash is an interactive movie in the sense that a user can interfere with the presentation of the movie. As an example, by clicking a button in a movie 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 usually has multiple presentations, and each of them is the result of a certain series of user behaviors.

2.2. Research issues

The intrinsic complexity of Flash leads to many issues concerning Flash retrieval that have not been addressed in existing research successfully. Some key issues are discussed as follows:

- *Indexing of heterogeneous components, dynamic effects, and user interactions*. The heterogeneous components, dynamic effects, and interactive feature of Flash are all important clues based on which users are likely to query for movies. For example, a user may search for those movies "accompanied by the song 'Yesterday'", movies "containing the text 'Lord of rings'", or movies "describing the scene of sunset". To support such queries, the properties of Flash on each of the three aspects should be indexed with features that facilitate effective retrieval of Flash movies, especially, high-level and semantic-flavored features Obviously, different features are required to describe component objects of different types (i.e., text, graphic, image, video, sound). Although the feature extraction of component objects and their dynamic effects can largely rely on the existing techniques, modeling user interactions poses a brand-new problem. Moreover, the indexing method should take into account the situation that a movie component is recursively embedded in another Flash movie.
- *Retrieval models for diverse features*. Since the features of a Flash movie are diverse in semantics and representation, multiple retrieval methods are needed for Flash retrieval based on various features. For example, text-based information retrieval (IR) methods can be used for Flash retrieval by the text components embedded in movies; content-based retrieval (CBR) techniques are suitable for multimedia components such as images, videos, and sounds; database-style queries can facilitate retrieval by predefined features such as the shape of graphic components. Ad hoc retrieval methods are needed for the features of dynamic effects and user interactions, depending on their respective representations.
- *Query specification and user interface*. Given the diversity of retrieval methods, user queries also need to be specified in a variety of ways, e.g., keyword query for IR approach, query-by-example for CBR approach, query language for database-style search. All the means of query specification should be provided in an integrated interface that allows users to compose various queries conveniently and efficiently. Furthermore, as the query interface directly addresses user experiences, it should be designed friendly and easy-to-use for average users. In particular, for complex query methods such as using a query language, iconic/graphic query specification techniques need to be investigated.

2.3. Connection with previous work

There have been a great number of publications devoted to the retrieval of multimedia data (including text), and for each type of data, some specific retrieval approaches have been proposed.

Among them, text-based information retrieval (IR) technique [17] is mainly used for searching large text collections using query expressed as keywords. Content-based retrieval (CBR) technique is invented by the Computer Vision community to retrieve multimedia objects based on *low-level features* that can be automatically extracted from the objects. CBR techniques have been widely used for image retrieval [16], video retrieval [3], and audio retrieval [8]. The low-level features used in retrieval vary from one type of multimedia to another, ranging from keywords for texts, color and texture for images, and pitch and melody for audios. In addition, database query using declarative query language (such as SQL) [5] is a method widely used by the Database community to retrieve structured data based on predefined attributes, such as captions of images, titles of documents. This approach is basically applicable to any types of data, as long as the data can be represented in a way conforming to certain structures or constraints (i.e., schemas). Despite these extensive works, since none of the media formats addressed in these works has all the three characteristics of Flash (e.g., none of them is interactive), their retrieval methods cannot be applied on Flash retrieval without significant modification. Nevertheless, as a Flash movie usually contains various types of multimedia objects in it, these existing methods can serve as the "enabling techniques" for Flash retrieval based on component objects.

Spatial/temporal features have been addressed by the research on multimedia with dynamic nature, usually, video streams. For instance, in the VideoQ system proposed by Chang et al. [3], videos are retrieved based on the joint spatio-temporal attributes of video objects represented as motion trajectories. In the work of Hjelsvold et al. [10] and Chan et al. [2], specialized query languages augmented with spatial and temporal operators are proposed for video retrieval. However, the complexity of the dynamic effects supported by Flash goes beyond the capability of the current techniques on modeling spatial/temporal features. For example, Flash supports the morphing of a graphic component from a square to a trapezium, which is insufficiently described by any current techniques. On the other hand, the usefulness of some detailed spatio-temporal attributes such as motion trajectory is arguable, since users are unlikely to query movies by specifying the precise movement of a component, say, "find Flash movies with a circle moving from coordinate (0,0) at frame 1 to (100, 50) at frame 10". In contrast, high-level, semantics-flavored queries are usually more preferred by users.

In addition, some research work has been devoted to the modeling and retrieval of generic multimedia presentation, which is defined as a synchronized and possibly interactive delivery of multimedia as a combination of video, audio, text, graphics, still images, and animations [12]. Hence, Flash movie is a typical type of multimedia presentation. Lee et al. [12] adopt an acyclic-graph representation of multimedia presentation and propose a graphical query language as an effective tool to query and manipulate the presentations based on their content. Adali et al. [1] suggest an algebra for creating and querying interactive multimedia presentation databases based on a tree representation. Both approaches are not directly applicable to Flash retrieval since (1) they do not tailor to the specific characteristics of Flash presentation, and (2) as database-flavored approaches they are not automatic.

3. FLAME: a generic framework for Flash retrieval

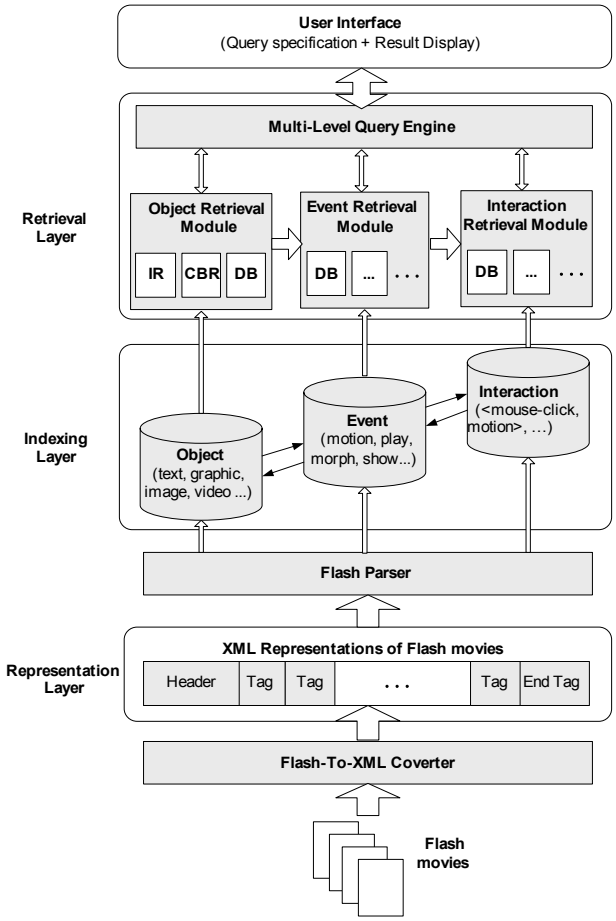

Figure 1: The 3-tier architecture of FLAME

FLAME is proposed as a generic framework for indexing and retrieval of Flash movies by mining and understanding of movie content. As depicted in Figure 1, it has a 3-tier architecture constituted by representation layer, indexing layer, and retrieval layer. The detail of each layer is described in this section, and sample queries are given to demonstrate its usefulness.

3.1. XML representation of Flash movies (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 [11] file (a Flash file is binary while a XML file is ASCII text file), and it can be converted into a XML file using tools such as JavaSWF [11]. Each tagged data block in a Flash file is mapped to a XML tag, which usually has attributes and embedded tags representing the structured data inside the block. Data blocks of the same type are 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) being a global standard XML format facilitates interoperability with other applications.

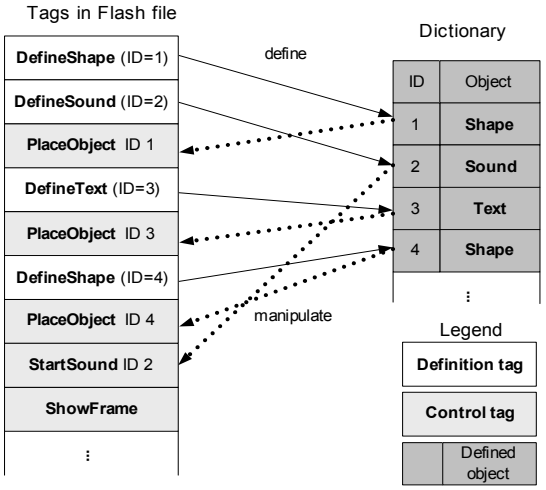

Figure 2: Structure of Macromedia Flash (SWF) file

There are two categories of tags in Flash files: *definition tags*, which are used to define various components in a movie, and *control tags*, which are used to manipulate these components to create the dynamic and interactive effect of the movie. For example, *DefineShape* and *DefineText* are typical definition tags, while *PlaceObject* (placing a component on the frame) and *ShowFrame* (showing the current frame) are typical control tags. All the components defined by definition tags are maintained in a repository called *dictionary*, from which control tags can access these components for manipulation. The diagram shown in Figure 2 illustrates the interaction between definition tags, control tags, and the dictionary.

3.2. Multi-level movie indexing (Indexing layer)

According to the research issues discussed in Section 2.2, a Flash movie can be characterized from three perspectives as its heterogeneous components, dynamic effects, and user interactions. In the indexing layer of FLAME, the characteristics of Flash on the three facets are modeled using the concepts of object, event, and interaction respectively. Specifically, *object* represents movie components 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 are extracted by the Flash Parser from the XML representation of the movie (see Figure 1). The formal description of each concept is presented as follows:

- **Objects**. A component object in Flash 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* $\in$ {*Text, Graphic, Image, Video, Sound*} denotes the type of the object, and *o-feature* represents its features. Obviously, the particular types of feature 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 size (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, as long as it is a simple shape such as rectangle or circle. The feature extraction techniques for each media type are widely available [8, 16, 19].

Table 1: Features for various types of objects

Object	Features
text	keywords, font size
graphic	shape, color, number of borders
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

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

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

  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 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.

Table 2: 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

Compared with the existing models [2, 3, 10], 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 describe the dynamic features of its components (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.

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

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

  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 in the interaction. The type of interaction indicates the device through which user behavior is conducted, including button, mouse, and keyboard. Button is a special component in Flash movies 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" interaction, even though they may also involve keyboard and mouse operations. The feature for each type of interaction is 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.

Table 3: 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

So far we have described the concept of object, event, and interaction. The index of a Flash movie can be represented as the collections of objects, events, and interactions that are embodied in it, given as:

$$movie = <\{object\}_m, \{event\}_n, \{interaction\}_t>$$

The retrieval of Flash movies is conducted based on such multi-level features, as described in the next subsection.

3.3. Multi-level query processing (Retrieval layer)

As shown in Figure 1, the retrieval layer of FLAME consists of three individual retrieval modules for matching objects, events, and interactions based on their respective features. These three modules do not work independently; rather, since interactions involve events, which in turn involve objects, the retrieval modules for higher-level concepts need to "call" the modules for lower-level concepts. For example, to find movies containing, say, a rotating rectangle graphic object, the event retrieval module, which matches the "rotate" action, needs to cooperate with the object retrieval module, which matches the "rectangle" graphic object. On the other hand, as user queries usually target at movies and may involve features at multiple levels, a multi-level query engine is designed to decompose user queries into a series of sub-queries for objects, events, and interactions processed by underlying retrieval modules, and then integrate and translate the results returned from these modules into a list of relevant movies. In the following, we describe each retrieval module and the multi-level query engine in detail. In particular, we define some high-level functions as the summarizations of their functionalities.

- **Object retrieval module**. This module accepts the type and features of object as inputs, 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 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 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. Specifically, IR approach is used for the keyword feature of text objects, CBR approach is used for the low-level features of video, image, and sound objects, and DB-style retrieval is well-suited for structured features such as the shape of graphic objects.

- **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.
- **Multi-level query engine**. The results returned by individual

retrieval modules are objects, events, and interactions, while the real target of the user queries is Flash movies. A primary function of the multi-level query 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 semantics of **Rank** taking *event-list* or *interaction-list* as parameters is similar.

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

where $\{movie\text{-}list\}_n$ denotes *n* movie lists that are obtained based on different search conditions, and $\{weight\}_n$ contains the weight indicting 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). Note that this function implements only the simplest method of merging multiple similarities, which is itself a separate research topic and is beyond the scope of this paper.

3.4. 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.

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 compose 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 multiple objects)

A user who wants to search for MTV movie of a specific song from a known singer can probably express his query as: *Find all Flash movies that have keyword "John Lennon" and are accompanied by the song 'Imagine'. (Suppose the audio file of the song is 'imagine.mp3'.)* This query can be handled by combining the results of a search based on the keyword and another search based on the song:

Merge ({**Rank** (**SearchObject**(text, 'John Lennon')),
Rank (**SearchObject**(sound, 'imagine.mp3'))})

Example 3: (Search by event)

A query for movies containing the scene of "sunset" can be composed 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:

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

Example 4. (Search by interaction and object)

Since the Flash movies as commercial advertisements sometimes contain the link to the company, 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 is processed by a combination of all functions defined in Section 3.3:

Merge ({**Rank** (**SearchObject** (text, 'BMW'),
Rank (**SearchInteraction** (button, 'mouse-click', **SearchEvent**(navigate,'www.bmw.com')))))}

4. An experimental prototype

To demonstrate the feasibility and effectiveness of FLAME, an experimental prototype as a Web-based Flash search engine system has been built based on FLAME. The prototype system has implemented the indexing and retrieval functions for most types of objects, events, and interactions supported by FLAME. Other functions are currently left out either because they are difficult and time-consuming to implement or because they are not very critical compared with other functions.

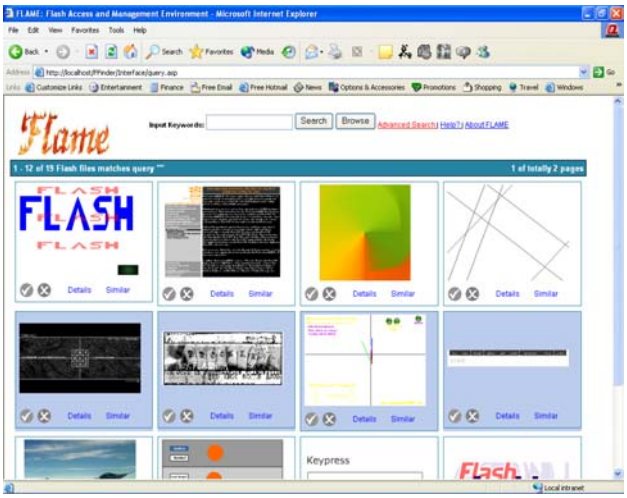

Figure 3: The main interface of the prototype

The great variety of queries supported by FLAME poses a challenge on user interface design. A good user interface should allow users to compose various types of queries conveniently and efficiently, as well as display the retrieved Flash movies in an appropriate layout. The interface of our prototype system can be displayed in standard Web browsers and accessed remotely over the Internet. In order to achieve good visual experience, we divide various query methods into two separate interfaces. The main interface shown in Figure 3 supports only keyword-based query, where a user can input query keywords and receive a list of Flash movies whose text (objects) matches with the query. The "thumbnails" of the retrieved movies ranked in descending order of their similarity (to the query) are displayed in the lower pane of

the interface. The main interface contains a hyperlink labeled as "Advanced Search" pointing to the second interface (shown in Figure 4), which allows users to compose more sophisticated queries by specifying the objects, events, and interactions appeared in the desired movies. This interface arrangement is based on the fact that keyword is the most natural device for users to express their requests, a "rule of thumb" proved by many commercial search engines. Providing advanced query options in a separate interface allows professional users to conduct more complicated queries without confusing non-professional users.

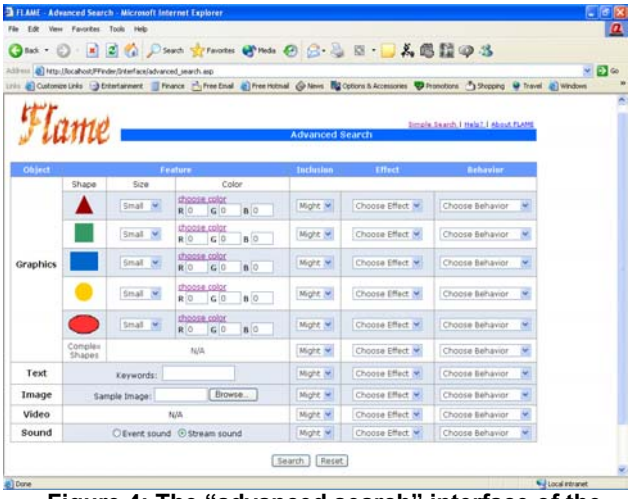

Figure 4: The "advanced search" interface of the prototype

The visualization of query specification methods is essential to the convenience and thus productivity of users. In the "advanced search" interface, we adopt an iconic specification of sophisticated queries. As shown in Figure 4, users can specify various components (including text, images, graphics, videos, sounds) appearing in desired movies with respect to their (the components') features, dynamic effects, and user behaviors triggering the effects. The features of these components 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 through a sample image. The desired dynamic effects of each component 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".

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 prototype system for two reasons: (1) there is no Flash collection serving as standard test dataset for retrieval systems (to the best of our knowledge, FLAME is the first Flash retrieval system); (2) due to the intrinsic complexity of Flash, there are probably a variety of subjective criteria regarding the relevance of Flash movies to a given query. Therefore, it is very difficult to define an objective "ground truth" for the test dataset. More likely, the retrieval performance of the system can be only evaluated by a large number of human subjects who conduct random queries and judge the quality of the retrieval results according to their own criterion.

5. Conclusions

This paper has investigated the problem of content-based Flash retrieval, which is critical to better utilization of the proliferating Flash resource on the Web but unfortunately has not been noticed by the research community. In this paper, we have presented an overview of Flash retrieval covering its characteristics, important research issues, and the connection with previous works. As our main contribution, a generic framework called FLAME has been put forward, which has a 3-tier architecture for the representation, indexing, retrieval of Flash movies by mining and understanding movie content. This framework features a unique *multi-level indexing and retrieval* approach that facilitates query of Flash movie based on the characteristics of its heterogeneous components, its dynamic effects, and the means of user interactions supported in it. An experimental prototype for Web-based Flash retrieval has been implemented to verify the feasibility of FLAME.

Although FLAME has covered a broad range of research issues, there remains much room for future research on Flash retrieval and management. One interesting future direction is 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. Other management issues, such as storage, navigation, classification, and clustering of Flash collections, are equally important and promising directions for effective Flash retrieval. On the other hand, the research on Flash retrieval can be generalized to the retrieval of other types of multimedia representations, such as PowerPoint, SMIL[19], etc.

References

1. Adali, S., Sapino, M.L., Subrahmanian, V.S. An algebra for creating and querying multimedia presentations. *ACM Multimedia Systems*, 8(3): 212-230, 2000.
2. Chan, S.S.M. and Li, Q. Developing an object-oriented video database system with spatio-temporal reasoning capabilities. In *Proc. 18th Int. Conf. on Conceptual Modeling (ER'99)*, LNCS 1728, pp. 47-61, 1999.
3. Chang, S.F., Chen, W., Meng, H.J., Sundaram, H., Zhong, D. VideoQ: An automated content based video search system using visual cues. In *Proc. ACM Int. Multimedia Conf.*, pp. 313-324, 1997.
4. Chen, Z., Liu, W.Y., Zhang, F., Li, M.J., Zhang, H.J., "Web Mining for Web Image Retrieval", *J. of the American Society for Information Science and Technology*, 52(10): 831-839, 2001
5. Elmasri, R. and Navathe, B. *Fundamentals of database systems*, 2 Edition. The Benjamin/Cummings Publishing Company, Inc., Redwood City, CA, 1994.
6. Extensible Markup Language (XML). http://www.w3.org/XML/
7. Flash Kit. http://www.flashkit.com/index.shtml
8. Foote, J. An overview of audio information retrieval. *ACM Multimedia Systems*, 7: 2-10, 1999.
9. Hauptman, A.G. and Smith, M.A. Text, speech and vision for video segmentation: the Informedia project. In *Proc. AAAI*

Fall Symposium of Computational Models for Integrating Language and Vision, 1995.

10. Hjelsvold, R. and Midtstraum, R. Modelling and querying video data. In *Proc. 20th Int. Conf. Very Large Database*, pp. 686-694, 1994.
11. JavaSWF. http://www.anotherbigidea.com/javaswf/
12. Lee, T., Sheng, L., Bozkaya, T., Ozsoyoglu, G., Ozsoyoglu, M. Querying multimedia presentations based on content. *IEEE Trans. Knowledge and Data Engineering*, 11(3):361-387, 1999.
13. Macromedia, Inc. www.macromedia.com.
14. Macromedia Flash Player adoption statistics. www.macromedia.com/software/player_census/flashplayer
15. Macromedia Flash File Format (SDK) http://www.macromedia.com/software/flash/open/licensing/fileformat/
16. Rui, Y., Huang, T., and Chang, S. Image retrieval: current techniques, promising directions and open issues. *J. of Visual Communication and Image Representation*, 10: 1-23, 1999.
17. Salton, G. and McGill, M.J. *Introduction to modern information retrieval*. McGraw-Hill Book Company, 1983.
18. Smoliar, S.W. and Zhang, H.J. Content Based Video Indexing and Retrieval. *IEEE Multimedia*, 1: 62-72, 1994.
19. Synchronized Multimedia Integration Language (SMIL). http://www.w3.org/AudioVideo/