

MUVIS: A Multimedia Browsing, Indexing And Retrieval System

Moncef Gabbouj, Serkan Kiranyaz, Kerem Caglar*, Bogdan Cramariuc, Faouzi Alaya Cheikh,
Olcay Guldogan and Esin Karaoglu

Signal Processing Laboratory, Tampere University of Technology, Tampere-Finland.

*Nokia Mobile Phones, PO Box 88, FIN-33721 Tampere, Finland

E-mail: serkan@cs.tut.fi, kerem.caglar@nokia.com

ABSTRACT

First MUVIS system has been developed three years ago, supporting image indexing and means to retrieve images from large image databases using image visual and semantic features such as color, texture and shape. Recently, MUVIS project has been reformed to become a PC-based system, which supports indexing, browsing and querying on various multimedia types such as audio, video and image. Furthermore the system allows real-time audio and video capturing, encoding by last generation codecs such as MPEG-4 or H.263+ if requested, recording while indexing into a database in such a way that they can be retrieved efficiently. In this paper, we describe the system features with underlying applications and outline the mean philosophy. Query and browsing capabilities of the MUVIS technology will be demonstrated during the conference.

1. INTRODUCTION

Rapidly increasing digital media technologies and compression standards combined with today's significantly cost-effective hardware technologies that are readily available on personal computers, several digital storage peripherals, broadcast systems and Internet lead us to the widespread exchange of multimedia information. This, however, brings the problem of storage, handling and accessibility of such a massive media load. In order to overcome this problem several content-based indexing and retrieval techniques and applications have been developed.

Existing indexing and retrieval systems such as first MUVIS system [1], [2], QBIC, Photobook, VisualSeek, Informedia, Virage, and VideoQ [5]-[11] provide the framework and techniques for indexing and retrieving either image or video information. There are also several initiatives and frameworks such as MPEG-7 [12] for multimedia content description issues.

In 1998, first MUVIS system has been implemented for indexing large image databases and retrieval via semantic and visual features based search and query techniques. Based upon the experience and feedback from this first system, recently a new PC-based MUVIS system, which is further capable of indexing and retrieval of video with audio in addition

to several image types have been developed. The current version of the system supports content-based video (with audio) and image indexing and retrieval combined with several multimedia handling features such as hierarchic video representation, video summarization via scene detection, search and query engine with feature sketching and several enriched image and video browsing capabilities.

The rest of this paper is organized as follows: in section 2 we outline the system philosophy of MUVIS and explain its functionalities. We present the general MUVIS framework with underlying applications in section 3 and in section 4, we demonstrate some experimental results on querying and video summarization. Finally concluding remarks and discussions are in section 5.

2. THE MUVIS SYSTEM

MUVIS aims to achieve a unified and global solution to the media (image, video and audio) capturing, recording, indexing and retrieval combined with browsing and various other visual and semantic capabilities. Figure 1 shows the system block diagram with underlying application.

The proposed MUVIS system is based upon several objectives and ideas. First of all it is intended to support real-time video (with or without audio) indexing. So MUVIS has so called *AVDatabase* application specifically designed and developed for creating Audio/Video Databases by indexing real-time video while capturing it from a peripheral video device. In order to achieve the optimal efficiency in video recording and indexing, last generation compression techniques such as MPEG-4 [3] and H.263+ [4] are used. As mentioned, these compression techniques are also suitable for indexing since they allow built-in shot-detection algorithm inside their encoders for intra encoding so that intra frames can be used as key-frames for feature extraction.

Second, it is further intended to index existing video clips no matter what their formats are. As shown in Figure 1, *DbEditor* is the application for appending such existing video clips into an existing database. Therefore, if any video clip is to be appended to any existing database, it is first converted to our supported video types (i.e. H.263+ or MPEG-4) and then appended to a database whilst the features are extracted accordingly.

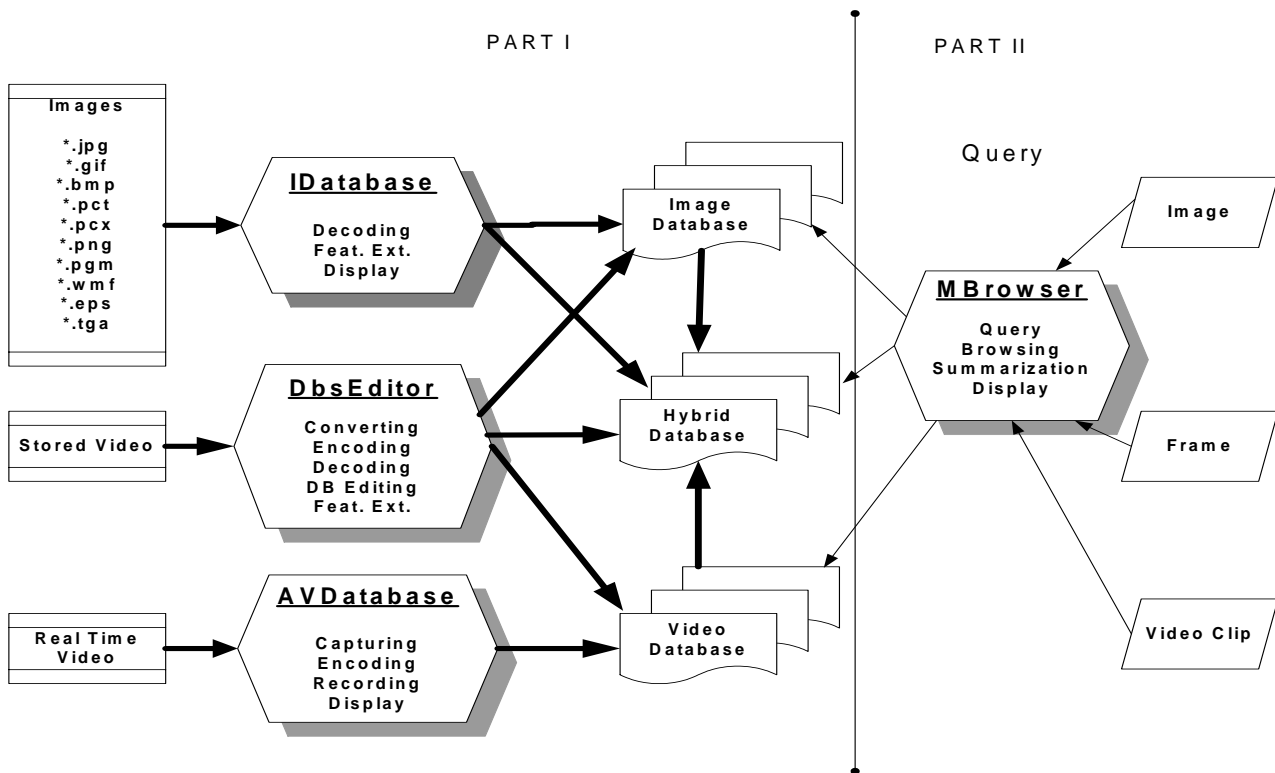


Figure 1 Structure of MUVIS System

Third, images in almost any format (JPEG, GIF, BMP, TIFF, etc.) are also indexed to any existing database with or without the presence of Audio/Video clips. Image databases containing only images can be created or images can also be appended to an existing video database and thus that video database is converted to a hybrid database, which contains both video and images. As shown in Figure 1, the application so called *IDatabase* is developed to handle all image indexing capabilities.

Finally, it is intended to have the multimedia (image and audio/video) retrieval terminal, which is also capable of browsing any media type, obtains hierarchic video representation and summarization combined with several semantic and visual features demonstration. As shown in Figure 1, this terminal is called *MBrowser*, which also has improved user-interface (UI) design that further helps easy browsing and retrieval.

3. APPLICATIONS OF MUVIS FRAMEWORK

As briefly mentioned in the previous section, MUVIS backbone framework consists of 4 main applications. In the following subsections we review the basic features of each application while emphasizing their role in the system.

3.1. AVDatabase

AVDatabase is a real-time video (with or without audio) indexing application and the main video database creator. It has basically two modes: *Create* a database or *Append* into an existing database. In *Create* mode, a number of video clips are captured

periodically (within pre-set interval of time), encoded (if requested, otherwise not encoded at all) and recorded in an indexed way. In *Append* mode one video clip is similarly recorded till the user terminates the recording. The recorded video clip will be indexed as the next clip in the database. Until recently, the features (for retrieval) were also extracted in real time but since it consumes significant CPU time and therefore, degrades the quality of recording process, feature extraction will from now on be handled off-line by *DbsEditor*.

Video information is captured in YUV 4:2:0 format from the peripheral video source that can be a PC camera or a TV card. Capturing is done for 30 f/s and in QCIF (176x144) or CIF (352x288) size for coding efficiency. Encoding is also optional. If requested, MPEG-4 simple profile video encoder or H.263+ video encoder can be used. Otherwise frames are saved in YUV 4:2:0 format directly into the file. Uncompressed scheme is rarely used since it is not efficient to store such massive information. The video encoding parameters such as bit-rate, frame-rate, forced-intra rate (if enabled), etc. can be defined during the recording time. The supported file formats handle the synchronization of video with respect to the encoding timing of each frame.

Audio information storage is totally optional. If enabled it can be captured, encoded and recorded in real-time similar to the video. For audio encoding we also use last generation audio encoders giving significantly high quality even in very low bit-rates such as *MP3* (MPEG1, 2 Layer 3) [5] and *AAC* (Advanced Audio Codec) [5]. *ADPCM* can also be used for low complexity or even audio can be recorded in raw (*PCM*) format.

Each video database has a corresponding file (**vdbs** extension is used for video databases) storing the overall

database parameters such as number of clips, features extracted (if already done), last create/append date, etc.

3.2. IDatabase

This is the main image-indexing program. So far it can handle the following image types: *BMP* (Windows Bitmap), *TGA*, *TIFF*, *JPEG*, *PNG*, *PCX*, *PGM*, *GIF*, *WMF*, *PCT*, and *EPS*. Creating a “dummy” image database is the way to start forming image databases. After creating an initial image database the user can append as many images to turn it to a real image database. There are two *Append* modes: *Append* via Manual Selection and *Append* via *AFS* (Automatic File Search). In the first mode, user can select one or more images and append them into an existing database. This database does not have to be necessarily an image database; it can be a video database or a hybrid database (containing both image and audio/video information). In the second mode user just choose the types of the images to be appended and an initial directory in the host machine to initiate the search & append operation. Once *AFS* started, it will find all the images in that directory and all the sub-directories in a recursive manner and append them into the current database. In both methods, the images that are found or chosen are copied into the database directory, renamed as in an indexed style. The original images are left untouched in their original locations.

While images are appended into any database, image feature vectors are extracted and stored in the root of the database. If the database is a dummy database initially, the user will also feed the feature extraction parameters. If database has already contained one or more images (which means the feature parameters are already fixed), then parameters present in the database are used to extract features. Currently, similar to the video databases, HSV bins are extracted for each image. Those bins are then stored in a distinct (with **BINS** extension) file for all of the images in the database.

IDatabase creates image and hybrid databases. Nevertheless, *IDatabase* can append images to any of the following database types: Image (idbs), Video (vdbs) or Hybrid (hdbs) databases. The idea is the following: even if only one image is appended to an existing video database, it is then automatically changed to a hybrid database since it now contains both of the media types (video and image).

3.3. DbsEditor

Once the media databases are created by the database creator applications, the application so called *DbsEditor* is designed to handle all the off-line issues for the existing databases. Editing (adding or removing) the features of a database, appending external media sources such as images or video clips in supported (H.263+ or MPEG-4) or other types and formats.

Feature editing is the primary task of *DbsEditor* application, especially for video databases because even though feature extraction is handled by *IDatabase*

while appending the images, so far *AVDatabase* does not extract features for the video clips in real-time because of the reasons explained earlier. So *DbsEditor* creates the features off-line for the video databases. Moreover *DbsEditor* can also add and remove features to/from any type of a database since MUVIS system is now being improved to support querying based upon multiple features.

Since *AVDatabase* is specifically designed to index real-time video into the video databases, the need has been aroused to index the existing video clips that might be in any type and format. This off-line task is also carried out by *DbsEditor*, which includes a built-in multimedia converter for the conversion of the video file type and format to the one of our supported codec (MPEG-4 and H.263+) types and file formats. Finally *DbsEditor* is designed to handle other database manipulations, such as merging two databases. In this case the new set of feature parameters can be specified for the final (merged) database.

3.4. MBrowser

This application is the main media retrieval and browser terminal, which supports any kind of database, image and video types so far discussed. In the most basic form, it has capabilities of a powerful media player (or viewer) and a database browser that allows user to reach any kind of media easily, efficiently and in any of the hierarchic level that is designed for the video clips. Currently the application supports 4 levels of video display hierarchy: single frame (1st frame), shot frames (key-frames), scene frames and the entire video. This kind of hierarchic representation is especially useful to get multiple levels of representation for the video clips.

MBrowser has a built-in search and query engine, which is capable of finding media primitives in any database and for any media type (video or image) that is similar with the queried media source. This source can be external that does not belong to the database or it can be internal that is from the database currently active in the application. Retrieving the queried source is performed by comparing its feature vector(s) with the feature vectors of media primitives available in the database. If the database contains images, for each image its normalized HSV bins are used as the feature vectors. Normalization is done with respect to QCIF (176x144) size. Therefore, the computed histogram bins are multiplied by the ratio between QCIF size and the size of the image. For video clips, the normalized histogram bins are computed for all of the key-frames and stored along with the video clip. Therefore, it is possible to query a video clip, any key-frame of the video clip or a still image within any database as shown in Figure 1, part II. If a frame from a video clip or an image is queried, its feature vector is compared with the feature vectors of the images and each video clip's key-frames in that database. Euclidean distance is used as a similarity measure between two feature vectors and obviously minimum Euclidean distance yields the best similarity. Accordingly all media primitives are sorted and afterwards the query results are shown in the user-interface of *MBrowser*. On the other hand if a video clip

is queried, all features of the key-frames of the video clip is compared with the feature vectors of all the media primitives. Since this time there may exist more than one feature vector, the best similarity (minimum Euclidean distance) measure for each key-frame is computed and all media primitives in the database are sorted according to their sum of best similarity measures with respect to the key-frames of the queried video clip.

The graphical user interface (GUI) of *MBrowser* is designed to obtain efficient browsing and media accessibility features. By means of such a design an ordinary user can use it easily, search and query any media type desired, access to any exclusive media type or database primitives efficiently and obtain multi-level representation of video clips. Furthermore it allows video summarization via scene detection, key-frame browsing during the video playback, post-processing to the video frames and various other means of media handling tools.

4. EXPERIMENTAL RESULTS

As shown in Figure 2, an image database containing 894 images is loaded and an image in this database is queried on the GUI window of *MBrowser*. First and best 12 query results are shown on the right side and the queried image is shown on the top-left side of the figure. The bottom-left side shows a sketch of the feature vector of the queried image (up) and a sketch of the feature vector of the 3rd best (in the first row, 3rd from the right) image (down). Figure 3 shows GUI window with a video playback on the top-left side with its key-frames in the bottom-left side. Key-frames are displayed simultaneously with the video playback. This video clip is also queried through a hybrid database, which contains 73 video clips and 464 sport images. First and best 12 query results are shown on the right side, 6 video clips are displayed on the top two rows and 6 images on the bottom two rows

5. CONCLUSIONS AND FUTURE WORK

The proposed MUVIS system is designed to bring a unified and global solution to content-based multimedia indexing and retrieval problem. The unified solution is basically achieved by designing the whole system of applications, which are handling the media during its life-time starting from capturing till indexing in such a way that the media can be indexed as efficiently as possible. The solution is also global since the system covers wide range of media types and formats for handling, indexing and retrieving.

Moreover it supports browsing, hierarchic video representation and summarization.

Future work will focus on enhancing the feature extraction and querying parts. The multi-feature support, which possibly uses shape, texture and audio along with the color information, will be integrated.

6. REFERENCES

- [1] M.Trimeche, F.Alaya Cheikh, M.Gabbouj and B.Cramariuc, "Content-based Description of Images for Retrieval in Large Databases:MUVIS", X European Signal Processing Conference, Eusipco-2000, Tampere, Finland, Sep 5-8, 2000
- [2] F.Alaya Cheikh, B.Cramariuc, C.Reynaud, M.Quinghong, B.Dragos-Adrian, B.Hnich, M.Gabbouj, P.Kerminen, T.Mäkinen and H.Jaakkola, "MUVIS: a system for content-based indexing and retrieval in large image databases", Proceedings of the SPIE/EI'99 Conference on Storage and Retrieval for Image and Video Databases VII, Vol.3656, San Jose, California, 26-29 January 1999
- [3] ISO/IEC 14496-2:2001 JTC 1/SC 29/WG 11 N5350, "Information Technology - Coding of Audio-Visual Objects - Part 2: Visual ", International Organization for Standardization, Sydney, July-2001.
- [4] ITU-T Draft, " Recommendation H.263 - Video Coding for low bit rate communication", International Telecommunication Union, November 1995.
- [5] Karl-Heinz Brandenburg, "MP3 and AAC Explained", AES 17th International Conference, Florence, Italy, September 1999.
- [6] Niblack W, Barber R, Equitz W, Flickner M, et al. "The QBIC project: querying images by content using color, texture ad shape", Proc SPIE 1908:173-187, 1993.
- [7] M. A. Smith and M. Cristel, "Automating the Creation of a Digital Video Library", Proc. ACM Multimedia, San Fran. 1995.
- [8] Virage. [URL:www.virage.com](http://www.virage.com)
- [9] S.F. Chang, W.Chen, J.Meng, H.Sundaram and D.Zhong, "VideoQ: An Automated Content Based Video Search System Using Visual Cues", Proc. ACM Multimedia, Seattle, 1997.
- [10] Pentland A, Picard RW, Sclaroff S, "Photobook: tools for content based manipulation of image databases", Proc SPIE (Storage and Retrieval for Image and Video Databases II) 2185:34-37,1994.
- [11] Smith JR, Chang SF, "VisualSEEk: a fully automated content-based image query system", ACM Multimedia, Boston, Nov. 1996.
- [12] F. Nack and A. T. Lindsay, "Everything You wanted to Know About MPEG-7: Part 1.", IEEE Multimedia, 6(3) 1999



Figure 2: MBrowser GUI with an Image Database loaded and one image is queried.



Figure 3: MBrowser GUI with playback of a video clip (with its key-frames) that is queried in a hybrid database.