

DYNAMIC INTEGRATION OF EXPLICIT FEATURE EXTRACTION ALGORITHMS INTO MUVIS FRAMEWORK

*Olcay Guldogan**, *Esin Guldogan*, *Serkan Kiranyaz*, *Kerem Caglar** and *Moncef Gabbouj*

Institute of Signal Processing, Tampere University of Technology, Tampere, Finland

* Nokia Mobile Software, Tampere, Finland

{karaoglu | serkan}@cs.tut.fi, {olcay.guldogan | kerem.caglar}@nokia.com, moncef.gabbouj@tut.fi

ABSTRACT

In this paper, we present an integration scheme designed in a content-based multimedia indexing and retrieval framework (MUVIS) for independent feature extraction algorithms. We introduce the general structure of the framework, and describe the interface, the instructions and the conditions for integrating a feature extraction algorithm into MUVIS. Finally, the limitations and the advantages of the integration scheme are discussed by giving examples from the implemented algorithms.

1. INTRODUCTION

First MUVIS system [1] was developed in late 90s to support indexing and retrieval in large image databases using visual and semantic features such as color, texture and shape. Recently MUVIS has been reformed to become a PC-based framework [2], which supports content-based indexing, browsing and retrieval of various multimedia types such as audio, video and image. Additionally, the developed MUVIS system has the following advanced multimedia processing capabilities and features:

- Multimedia collection and multimedia database creation,
- Real-time audio and video capturing, recording and encoding,
- Hierarchic video representation,
- Video summarization via scene detection [3],
- Enriched graphical user interface.

As shown in Figure 2, MUVIS framework is based upon 4 applications, each of which has different responsibilities and facilities. *AVDatabase* and *IDatabase* applications are mainly responsible for video and image database creation respectively. *DbEditor* performs the indexing of the multimedia databases, and therefore the offline feature extraction processing is its main task. *MBrowser* is the primary media browser and retrieval application.

MUVIS manages feature extraction in terms of explicit modules in both indexing and retrieval stages. Feature extraction algorithms can be implemented independently as modules, and then integrated into MUVIS framework via a specific interface called *Feature Extraction Interface* (FeX API).

Features are represented by normalized feature vectors and extracted for indexing purposes. Such feature vectors are extracted by explicit *Feature Extraction Modules* (FeX modules) and are essentially managed and used by *MBrowser* and *DbEditor* applications. Since the features are normalized vectors, *MBrowser* can merge multiple features for querying. Generally speaking, the FeX modules are parameter dependent, and by changing those parameters it is possible to extract multiple features (sub-features) using a single FeX module. In order to query a multimedia item within a MUVIS database, associated features of that item are extracted by *MBrowser* using the same FeX module. Therefore, both *DbEditor* and *MBrowser* applications share all the FeX modules.

There are mainly two visual multimedia items in MUVIS framework: video clips and images. Features of video clips are extracted from the key-frames, which are the INTRA frames in MPEG-4 or H.263 bit-stream. In most circumstances, a shot detection algorithm is used to select the INTRA frames during encoding but occasionally a forced-intra scheme might further be applied to prevent a possible visual degradation. Image features on the other hand are extracted directly from 24-bit RGB frame buffers obtained by decoding images.

Following FeX modules are implemented so far and integrated into MUVIS framework via FeX API:

- Feature extraction modules based on HSV, RGB and YUV color histograms [4],
- Gray Level Co-occurrence Matrix [5] method for texture features,
- Gabor Wavelet Transform [6] method for texture features.

2. INTEGRATION OF FEATURE EXTRACTION ALGORITHMS VIA FeX API

MUVIS provides a feature extraction interface (FeX API) for the integration of visual feature extraction algorithms dynamically without dealing with the implementation details of the MUVIS applications. Hence, any FeX module can be designed and tested using MUVIS framework. Furthermore, retrieval performance of a FeX module can be compared with the other similar FeX modules.

FeX API basically defines specific data structures and API functions, which allow the proper interaction between the MUVIS applications and FeX modules. Figure 1 illustrates such interaction.

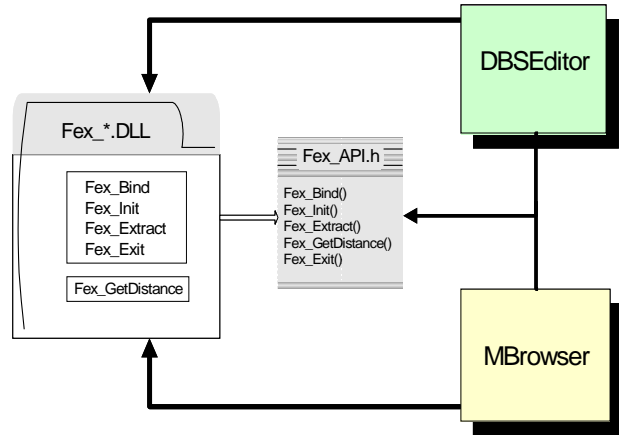


Figure 1: FeX module interaction with MUVIS applications

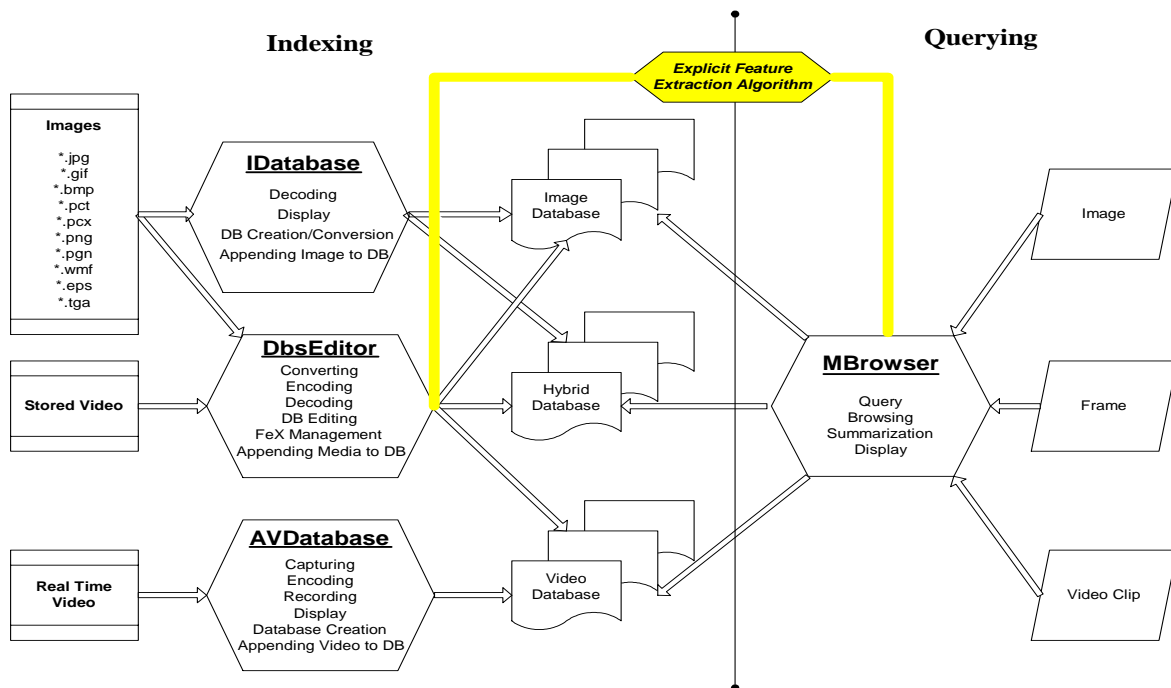


Figure 2: General structure of MUVIS framework

2.1. FeX API

Any feature extraction algorithm should be implemented as a *Dynamic Link Library* (DLL) using FeX API, which is defined in a header file (*FeX_API.h*). Such DLLs are referred to as *FeX Modules* in MUVIS.

Mainly *FeX_API.h* defines five different API functions required to manage all feature extraction operations in a dynamic way. It also specifies the necessary data structures for feature extraction and communication between the modules and the applications. Additionally, there is a naming convention for any FeX module, such as *FeX_[FourCC string].dll*, (i.e *FeX_HSV.dll*). *FourCC string* is a representation consisting of 4 or less number of characters. *FourCC code* is a 4-bytes integer number obtained from the FourCC string using Eq. 1.

FourCC string = 'ABCD',

Let $a = \text{ASCII code ('A')}$, $b = \text{ASCII code ('B')}$,
 $c = \text{ASCII code ('C')}$, $d = \text{ASCII code ('D')}$,

$_FourCC('ABCD') = 2^{24} * d + 2^{16} * c + 2^8 * b + a$,

where $\text{ASCII code}(X)$ is an OS dependent number, which represents 1-byte code (number) of character X.

Equation 1

2.1.1. Data Structures Defined in FeX API

One enumeration type and two structures are declared in *FeX_API.h* as follows:

- **FrameType Enumeration:** It defines the frame formats used for feature extraction. Currently RGB 24 bit and YUV 4:2:0 frame buffers are supported.
- **FexParam Structure:** It is created (allocated) and filled by the FeX module for handshaking. A FeX module should set the following structure members to introduce itself to the application:
 - *Feature name:* Textual description of the feature (i.e. "HSV Histogram"). It is simply used by the MUVIS applications as the feature title.
 - *Feature FourCC code:* Unique identification code for the *FeX module*. A macro is defined for converting a given character array to FourCC code (i.e. `_FourCC('HSV')`).
 - *Feature parameter no:* Number of feature parameters (i.e. 3 for HSV: nH, nS and nV that are the number of bins in each color dimension).
 - *Feature parameter FourCC:* Array of parameter FourCC codes (i.e. `[_FourCC('H'), _FourCC('S'), _FourCC('V')]`). It is used for displaying the feature parameter names by the MUVIS applications.
 - *Default feature parameter values:* Array of parameter default values in double precision (i.e. [8,4,4] for HSV).
 - *Frame type:* Frame type that should be specified by the FeX module to the application, so that the application can supply the frame buffer in the specified format (i.e. RGB24 or YUV4:2:0).
- **FrameParam Structure:** The frame buffer is stored in this structure and sent to the FeX module for feature extraction. It includes the following members:
 - *Buffer:* Raw frame data in a single dimension array. Frame pixels are located in left-to-right top-to-down raster scan order. If the frame is in YUV4:2:0 format, then first part of the buffer keeps the Y values for each pixel, followed by U and V array of values. Each value represents average chrominance value for 4 pixels. If the frame is in RGB 24bit format, then R, G and B values are all in raster-scan order (i.e. RGBRGBRGB...).
 - *Width and height* of the frame.

2.1.2. API Functions Defined in FeX API

Five API function properties (name and types) are declared in *Fex_API.h*. The owner of a FeX module should implement all specified API functions. There also exists a macro in order to convert a character array to a *FourCC* code. The API functions are described below:

- **Fex_Bind:**
I/O parameters: FexParam object / success-fail flag (return).

Function description: It is used for handshaking operation between a MUVIS application and a FeX module. FeX module fills the given *FexParam* structure to introduce itself to the application. This function is called only once at the beginning, just after the application links the FeX module in run-time. It should return 0 if a problem occurs, and a nonzero value if successfully completed.

- **Fex_Init:**
I/O parameters: Parameter array / success-fail flag (return).

Function description: It is used to initialize the FeX module. The feature extraction parameters are given in a double array, which has the size defined in the *FexParam* structure. The FeX module performs necessary initialization operations, i.e. memory allocation, table creation etc. This function is called for the initialization of a unique sub-feature extraction operation. It should return 0 if a problem occurs, and a nonzero value if successfully completed.

- **Fex_Extract:**
I/O parameters: FrameParam object / Feature vector, Vector size.

Function description: It is used to extract the features of a frame (buffer) stored inside the *FrameParam* structure. The FeX parameters should be available when *Fex_Init* is called. This function is called for feature extraction of each frame. It returns the feature vectors, which have the double precision values. As mentioned before, these vectors should be normalized in such a way that each value in the feature vector and the total length of the vector should be in between 0.0 and 1.0. This normalization is required for merging multiple features while querying in *MBrowser*. The size of the feature vector is returned via the given reference parameter.

- **Fex_Exit:**
I/O parameters: FexParam object.

Function description: It is used for resetting and terminating the FeX module operation. It frees the memory spaces allocated by *FexParam* object during *Fex_Bind* function call. Additionally, if *Fex_Init* has been called already, this function resets the FeX module to perform further feature extraction operations. This function is called at least once while the MUVIS application is terminated, but it might be called at the end of each FeX operation per sub-feature extraction.

- **Fex_GetDistance:**
I/O parameters: Feature vectors, Vector size / Similarity distance

Function description: This function is only used by *MBrowser* application to obtain the similarity measure via calculating the distance between two feature vectors. Both feature vectors and the common vector size are given as parameters. Therefore, the appro-

appropriate distance measurement algorithm should be implemented in this function. The calculated similarity distance is returned as a double precision number.

2.2. Step-by-Step Feature Extraction in MUVIS

The following steps are carried in *DbEditor* application for feature extraction of the media items in a database:

- 1- Whenever *DbEditor* is started, it checks for the FeX modules (DLLs) in proper directories, loads them and links all their functions. It proceeds by calling their *Fex_Bind* functions to establish handshaking operation. Associated *FexParam* structures are filled by the modules, so that *DbEditor* has all the necessary information ready for feature extraction.
- 2- Adaptive *DbEditor* user interface lets the user supply the FeX parameters, and once the parameters are set for a single sub-feature, it first calls the *Fex_Init* function with the given parameters to initialize the FeX module.
- 3- For each frame in the database (either all the key-frames or images), *DbEditor* calls the *Fex_Extract* function with the *FrameParam* structure, and the function returns the feature vector along with the vector size.
- 4- Once all the sub-features with certain parameters are extracted, *Fex_Exit* function is called to reset the FeX module with the *FexParam* structure that is already filled via *Fex_Bind* function.
- 5- In order to extract a new feature (if required), steps 2 to 4 are repeated for each sub-feature of the new feature.
- 6- When *DbEditor* application is through termination, *Fex_Exit* function is called for a final clean-up (i.e. deallocation of memory space). This is necessary in case steps 2 to 4 have never been executed during the lifetime of the application.

The following steps are carried in *MBrowser* for feature extraction of a media item (a video clip or an image) during querying:

- 1- Same actions are carried in this step as in the first step of feature extraction in *DbEditor*.
- 2- Whenever user queries a media, *MBrowser* first calls *Fex_Init* function with the chosen sub-feature parameters to initialize the FeX module.
- 3- For each frame of the media (either key-frames of a video clip or image itself), *MBrowser* calls *Fex_Extract* function with the *FrameParam* structure, and the function returns the feature vector along with the vector size.
- 4- Once all the sub-features with certain parameters are extracted, *Fex_Exit* function is called to reset the FeX module with the *FexParam* structure that is already filled via *Fex_Bind* function.
- 5- For all the media primitives in the database, associated feature vectors per frame are read from the appropriate database feature files and the similarity dis-

tances between those vectors and the feature vectors of the queried media are obtained using *Fex_GetDistance* function. The distances are sorted and the query results are displayed accordingly.

- 6- To query a new media, steps 2 to 5 are repeated, if the application is not terminated meanwhile.
- 7- Same actions are carried in this step as in the last step of feature extraction in *DbEditor*.

3. CONCLUSIONS AND FUTURE WORK

MUVIS framework provides an efficient basis to implement independent feature extraction algorithms and integrate them into the MUVIS system via a well-defined interface (FeX API) for verification and testing purposes. MUVIS FeX API specifies some restrictions for feature extraction algorithms to provide a common platform for several implementations. Among others, most important restriction is the restriction of the vector representation for the features. Nevertheless, several algorithms comply with such restrictions and can be implemented with respect to FeX API. There are also ongoing studies for implementing and integrating other feature extraction algorithms based on shape and audio information. Furthermore, studies are carried out to improve FeX API by developing a more flexible interface in order to extend the scope of the feature extraction algorithms that can be integrated into MUVIS.

4. 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] M. Gabbouj, S. Kiranyaz, K. Caglar, B. Cramariuc, F. Alaya Cheikh, O. Guldogan, and E. Karaoglu, "MUVIS: A Multimedia Browsing, Indexing and Retrieval System", *Proc. of IWDC 2002 Conference on Advanced Methods for Multimedia Signal Processing*, Italy, Sep. 2002.
- [3] S. Kiranyaz, K. Caglar, B. Cramariuc and M. Gabbouj, "Unsupervised Scene Change Detection Techniques in Feature Domain via Clustering and Elimination", *Proc. of IWDC 2002 Conference on Advanced Methods for Multimedia Signal Processing*, Italy, Sep. 2002.
- [4] M. J. Swain, D. H. Ballard, "Color Indexing", *International Journal of Computer Vision*, Vol. 7, No. 1, 1991.
- [5] M. Partio, B. Cramariuc, M. Gabbouj, A. Visa, "Rock Texture Retrieval Using Gray Level Co-occurrence Matrix", *Proc. of 5th Nordic Signal Processing Symposium*, Oct. 2002.
- [6] W. Y. Ma, B. Manjunath, "Texture Features for Browsing and Retrieval of Image Data", *IEEE Trans. PAMI*, vol. 18, pp 837-842, Aug.