

AUDIO-BASED MULTIMEDIA INDEXING AND RETRIEVAL SCHEME IN MUVIS FRAMEWORK

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

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

* Nokia Mobile Software, Tampere, Finland

(moncef,serkan)@cs.tut.fi, kerem.caglar@nokia.com

ABSTRACT

MUVIS is a PC-based framework, which supports indexing, browsing and querying of various multimedia types such as audio, video, audio/video interlaced in several formats. It allows real-time audio and video capturing, encoding by last generation codecs such as MPEG-4, H.263+, MP3 and AAC. MUVIS also supports several audio/video file format such as AVI, MP4, MP3 and AAC. Almost all image types in a PC environment including JPEG-2000 can be rendered, indexed and converted within MUVIS framework. Along with such a wide multimedia coverage, MUVIS has been developed to achieve a global and unified solution for content-based indexing and retrieval problem and to provide user-friendly applications and a generic framework especially for third parties to develop their feature extraction modules. In this paper, we present an overview of the MUVIS system and we shall especially focus on the overall audio-based multimedia indexing and retrieval scheme within MUVIS framework.

1. INTRODUCTION

The recent hardware and software improvements in the computer world with the increasing Internet usage have caused a massive usage of digital multimedia in several types, formats and quality. This, however, brings the storage and management problems and especially efficient content-based retrieval of any particular media item become a problematic issue. In order to overcome such problems several content-based indexing and retrieval techniques and applications have been developed such as MUVIS system [1], [2], [14], Photobook [3], VisualSeek [4], Virage [5], VideoQ [6] and VideoAL [18]. The common feature of all such systems is that they all provide some kind of framework and several techniques for indexing and retrieving either still images or audio-video files. MPEG-7 [7] is a recent standard for multimedia content description.

In 1998, first MUVIS system has been implemented for indexing large image databases and retrieval via search and query techniques based on semantic and visual features [14]. Based upon the experience and feedback from this first system, recently a new PC-based MUVIS system, which is further capable of content-based indexing and retrieval of video and audio information in addition to several image types, has been

developed. Table 1 shows the types of multimedia that the new MUVIS system so far supports.

MUVIS Audio			
Codecs	Sampling Freq.	Channel No	File Formats
MP3 [11]	16, 22.050, 24 KHz	Mono	MP3
AAC [12]	32, 44.1 KHz	Stereo	AAC
G721	Any for G721,		AVI
G723	G723 & PCM		MP4
PCM			

MUVIS Video			
Codecs	Frame Rate	Frame Size	File Formats
H263+ [10]	1 - 25 fps	Any	AVI
MPEG-4 [9]			MP4
YUV 4:2:0			
RGB 24			

MUVIS Image Types							
Convertible Formats							
	JPEG	JPEG 2K	BMP	TIFF	PNG		
Inconvertible Formats							
PCX	GIF	PCT	TGA	PCX	EPS	WMF	PGM

Table 1: MUVIS Multimedia Family

The current MUVIS has been adopted by EU ACTION COST 211quat as the COST Framework for CBIR. The current version of the MUVIS framework supports the following multimedia processing capabilities and features:

- Real-time audio and video capturing, encoding and recording,
- Hierarchic video handling and representation,
- Video summarization via scene detection [8],
- An effective framework structure, which provides an application independent basis in order to develop audio

and visual feature extraction techniques that are used dynamically by MUVIS applications for indexing and retrieval.

- The retrieval based on distinct visual and aural queries initiated from any MUVIS database that includes audio/video clips and still images.
- In order to append unsupported multimedia files to a MUVIS database, format/type conversions such as MPEG-1 video (with possible MPEG-1 Layer 2 audio) to any of the supported format is so far provided in MUVIS system.
- All MUVIS applications and available visual and audio feature extraction (*FeX* and *AFeX*) modules within current MUVIS framework are developed to

- Provide such an indexing and retrieval scheme that is robust to (independent from) the multimedia formats, types, encoding/capturing parameters, etc.

The rest of this paper is organized as follows: in section 2, we outline the system philosophy of MUVIS and the general MUVIS framework with underlying applications. Section 3 presents the overall audio-based indexing and retrieval scheme in the MUVIS framework. In section 4, we demonstrate some experimental results on audio-based multimedia retrieval via query and we outline our conclusions and remarks.

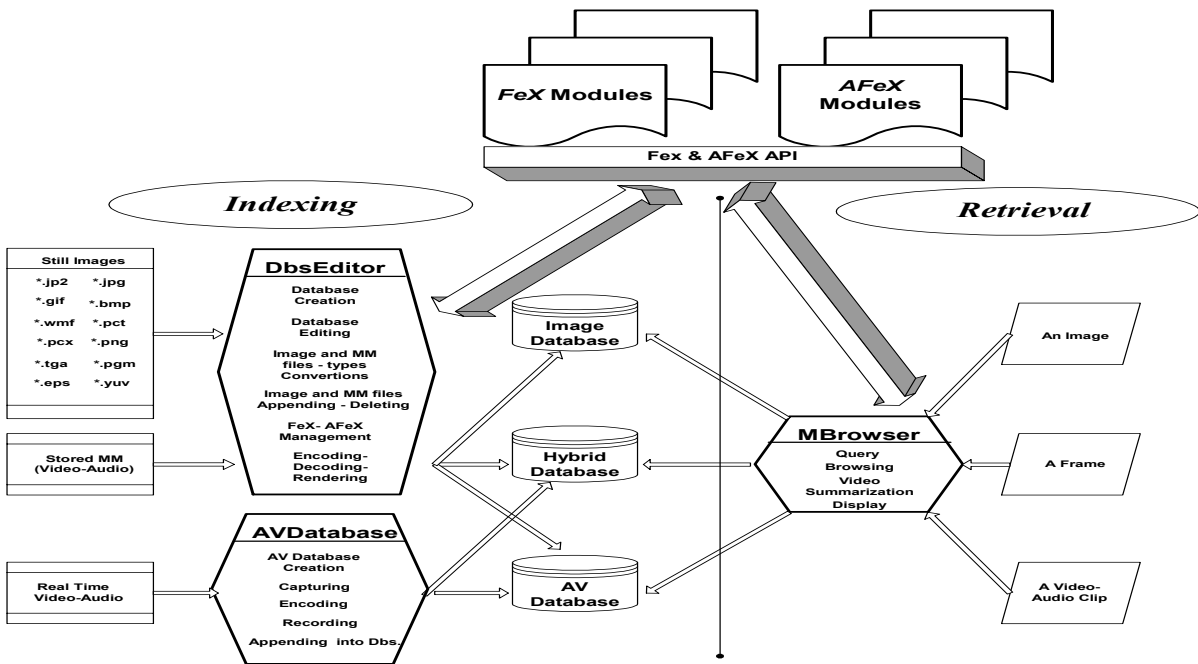


Figure 1: General structure of MUVIS framework

2. APPLICATIONS OF MUVIS FRAMEWORK

As shown in Figure 1, MUVIS framework is based upon three main applications, each of which has different responsibilities and functionalities. *AVDatabase* is mainly responsible for video and/or audio database creation. *DbEditor* performs the general tasks of indexing of multimedia databases, and therefore, offline feature extraction processing is its main task. *MBrowser* is the primary media browser and retrieval application. In the following subsections we review the basic features of each application while emphasizing their role in the overall system.

2.1. AVDatabase

AVDatabase application is specifically designed and developed for creating audio/video databases by collecting real-time audio/video files via capturing from any peripheral video/audio device. An audio/video clip may include only video information, only audio information or both video and audio information interlaced, if supported by the underlying file format. Several video and audio encoding techniques can be used with any encoding parameters specified in Table 1. More detailed information about *AVDatabase* application can be found in [2].

2.2. *DbEditor*

Once the media databases are initially created by the database creator applications, *DbEditor* application is designed to handle indexing and any editor task for the existing databases. Feature extraction is the primary task of *DbEditor* application. This is basically needed for proper indexing of the multimedia databases. Hence *DbEditor* can append/remove visual and audio features to/from any suitable type of MUVIS database. The main functionalities of *DbEditor* can be listed as follows:

- Dynamic integration and management of feature extraction modules.
- Extracting new features or removing existing features of a database using available *FeX* and *AFeX* modules,
- Appending/Removing any audio/video clips and still images to/from any existing database,
- Converting one image format to any convertible image format given in Table 1. If it is a lossy compression type such as JPEG 2000 or JPEG a quality factor can be set to achieve a certain quality level or a bit rate,
- Appending alien audio/video files into any MUVIS database by first converting them into a supported file format with the user specified parameters. It is further possible to make conversions between MUVIS supported file formats in order to change their file types, encoder types, encoder parameters, etc.
- Preview of any audio/video clip or image in a database,
- Providing statistical information of a database and/or items in a database.

2.3. *MBrowser*

MBrowser is the main media browser and retrieval terminal, which works with any kind of MUVIS database. In the most basic form, it has the capabilities of an advanced multimedia player (or viewer) and a simple database browser. Furthermore, it allows users to reach any kind of multimedia easily, efficiently and, for the case of video clips, in any of the available hierarchic levels. *MBrowser* supports a 4-level video display hierarchy: single frame, shot frames (key-frames), scene frames and the entire video clip.

MBrowser has a built-in search and query engine, which is capable of finding multimedia primitives in a database and for any multimedia type that is similar to a queried media item (a video clip, a frame or an image). In order to query an audio/video clip, it should first be appended to a MUVIS database upon which the query will be performed. There is no such necessity for images; any image (inclusive or exclusive) can be queried in a database. Retrieval is based on comparing the similarity distances between the queried media item's feature vector(s) with the feature vectors of multimedia primitives available in the database. The similarity distances are calculated by the

corresponding functions implemented in the corresponding feature extraction module.

Furthermore, *MBrowser* provides the following additional functionalities:

- Video summarization via scene detection,
- Key-frame browsing during video playback,
- Random access support for video playback,
- Displaying information related to the extracted features of the active database,
- Visualizations of feature vectors of the multimedia items.

3. AUDIO INDEXING AND RETRIEVAL FRAMEWORK IN MUVIS

Audio is an important source of information for content-based multimedia indexing and retrieval. Furthermore, it can sometimes be even more important than the visual part since it is mostly unique and significantly stable within the content. However, when dealing with digital audio there are several requirements to be fulfilled and the most important of them is the fact that the content is totally independent from the digital audio capture parameters (i.e. sound volume, sampling frequency, etc.), audio file type (i.e. AVI, MP3, etc.), encoder type (MP3, AAC, etc.) or encoding parameters (i.e. bit-rate, etc.). So the overall structure of the audio-based indexing and retrieval framework has been developed to provide a pre-emptive robustness (independency) to such variations.

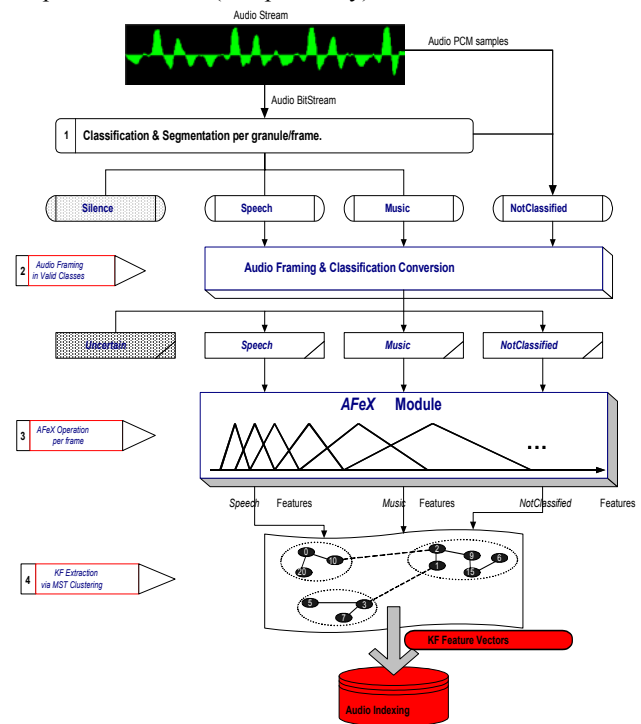


Figure 2: MUVIS Audio Indexing Operation Flowchart

As shown in Figure 2, audio indexing is accomplished in several stages: classification and segmentation, audio framing within segments with certain types, audio feature extraction via

available *AFex* modules, key-framing via minimum spanning tree (MST) clustering [16] and finally the indexing over the extracted key-frames (KFs).

3.1. Audio Classification and Segmentation

In order to achieve a better content analysis, the first step is to perform accurate classification and segmentation over the entire audio clip. In MUVIS system, this operation is performed over AAC and MP3 bit-streams directly from the bit-stream information.

Audio segmentation and classification are closely related and internally dependent problems. Achieving a good segmentation requires good classification and vice versa. Therefore, without any prior knowledge or supervising mechanism, the algorithm proceeds in an iterative way, starting from frame based classification and initial segmentation, to ensure a global segmentation outcome and thus a successful classification per segment at the end. First, a generic MDCT sub-band template is created and then the proposed algorithm is applied to both types of bit-streams (*MP3/AAC*). Such a template makes the algorithm independent from the underlying encoding scheme.

3.1.1. Forming MDCT Template from MP3/AAC Bit-stream

The classification algorithm relies on the MDCT coefficients extracted directly from the *MP3/AAC* bit-stream. The number of MDCT coefficients is not the same for *MP3* (576) and *AAC* (1024). For each *MP3* granule, the MDCT coefficients are directly in the form of a matrix with 32 rows, representing the frequency sub-bands, and with 18 columns each of which contains a MDCT coefficient. In case of a short window, there are 3 windows within a granule containing 6 coefficients. In order to process the same algorithm for both encoding schemes, a similar template structure is applied to *AAC* frames. Thus, in case of a long window *AAC* frame, a 1024 MDCT coefficient array is divided into 32 rows and 32 columns of MDCT coefficients and the template matrix for *AAC* is formed by taking into account that the number of MDCT coefficients for a sub-band is not 18 (as in *MP3*) but now 32. In case of short window *AAC* frame, 1024 coefficients are divided into 8 windows of 128 coefficients each. These 128 coefficients are then divided in 32 sub-bands and a matrix is formed with 4 coefficients in every sub-band in order to have the same template as the *MP3* short window case. Finally, once the MDCT template formation is achieved the classification and segmentation tasks proceed as detailed in the following section.

3.1.2. The Classification and Segmentation Algorithm

There are 4 hierarchical steps for *MP3/AAC* classification and segmentation:

1) Feature Extraction per Granule/Frame: Each granule/frame is classified in one of three categories: speech, music and silence. Silence detection is performed per

granule/frame by applying a threshold to the total energy over all sub-bands calculated by the following equation:

$$En_{MP3} = \sum_i^{32} \sum_j^{18} MDCT_{i,j}^2 \quad En_{AAC} = \sum_i^{32} \sum_j^{32} MDCT_{i,j}^2 \quad (1)$$

If the granule/frame is not classified as *silent*, the band energy ratio (**BER**) is calculated for each granule/frame. **BER** is the ratio between the total energy within two frequency intervals divided by a certain frequency threshold value (i.e. 500Hz.). If this ratio is over a certain value the granule/frame is then classified as *music*, otherwise *speech*. Figure 3 summarizes the operation performed in Step 1.

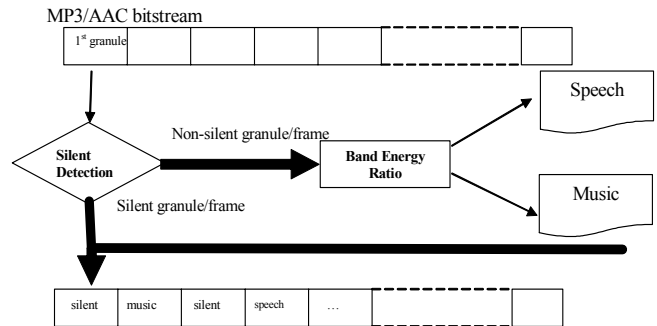


Figure 3: Feature Extraction per granule in Step 1

2) Segmentation and Feature extraction per Segment: In this step, silent and non-silent segmentations are performed. In the previous step, all the silent granules/frames have already been found. So the silent granules/frames are merged to form silent segments. A preset threshold value (i.e. 0.2s) is used to assign a segment as silent segment if a sufficient number of silent granules/frames merges to a silent duration larger than this threshold. All parts left between silent segments are then considered as non-silent segments. Once all non-silent segments are formed, then the classification of these segments is performed using the following features:

- **Dominant BER Classification:** In Step 1, all granules/frames are already classified with respect to **BER**. In this step, for each non-silent segment, the dominant classifier type (the largest number of granule/frame type) will determine the segment type.
- **Pause Rate Classification:** Pause Rate (**PR**) is the ratio between the number of *silent* granules/frames to the total number of granules/frames in a non-silent segment. If this ratio is over a threshold, then the segment is classified as a *speech* segment, otherwise *music*.

3) Segment Merging and Global Classification: After Step 2 is performed; some of the *silent* segments might still be quite small and negligible for the sake of segmentation. Such small *silent* segments reduce the duration of the non-silent segments and thus lead to erroneous calculations for some features such as **Pause Rate** or **Sub-band Centroid**. Therefore, they need to be eliminated to yield a better (global) segmentation that would indeed result in a better classification. There are two conditions in order to eliminate a silent segment:

- i) Silent segment duration is below a threshold value,

ii) Neighbor non-silent segment types are matching.

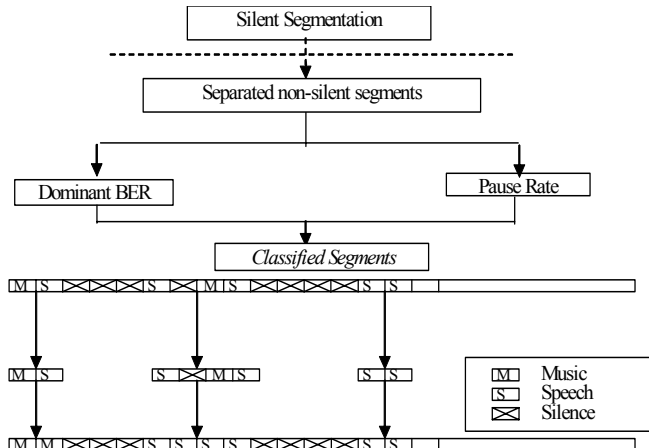


Figure 4: Segmentation and Classification per Segment in Step 2

After merging some of non-silent segments, the overall segmentation scheme is changed and the features have to be re-extracted over the new (emerged) non-silent segments. For all the non-silent segments, **PR** and Dominant **BER** are recalculated and all non-silent segments are re-classified. This new classification of non-silent segments may result into such classification types that allow us to eliminate further silent segments (In the first step they may not be eliminated because the neighbor classification types did not match). So a loop is needed to eliminate all possible small *silent* segments. The iteration is carried out till all small *silent* segments are eliminated and non-silent segments are merged to have global segments, which have a unique classification type.

Once the merging loop is terminated, there may still remain some short non-silent segments that are not merged to any neighbor global segments. Such short non-silent segments are naturally false detections and therefore, should also be eliminated. After the elimination of the short non-silent segments, the remaining non-silent segments are global enough to contain one single classification type. This is further important to have global segments in order to run other feature extraction techniques such as **Sub-band Centroid (SC)** and **Fundamental Frequency (FF)** estimation. **SC** extraction is the detection of the sub-band values which is the nearest from the center of mass over the whole frequency range. It is performed over all non-silent segments while excluding the silent granules/frames. **FF** estimation is performed per granule/frame. If no fundamental frequency is found over a granule/frame, typically that granule/frame is assumed to contain non-harmonic (unvoiced) components. The multiplication of mean **FF** with percentage of harmonic granules/frames within a segment is used as a feature for classification. It is expected to be a smaller value over speech segments (where there are more unvoiced components) compared to music segments. There is a threshold value for this decision so that if it is lower than this threshold the segment is classified as *speech*, otherwise

music. The feature used to perform classification via **SC** is the mean and standard deviation of **SC** in a segment. These features are extracted by smoothly sliding a short window through the entire duration of the non-silent segment. The standard deviation is calculated using local windowed mean and global variance of the segment. If the standard deviation of the **SC** within the segment is below a low threshold value, that segment is classified as *music*. Similarly, if it is above a high threshold value, the segment is classified as *speech*. Otherwise, if the standard deviation of **SC** is in the so-called fuzzy region (above the low threshold value and below the high threshold value) it is classified as *NotClassified*. The final classification type is classification by **Pause Rate** as explained in the previous section.

The aforementioned features are also capable of classifying a segment forcefully into one of the classes. This is forced classification and is applicable for a segment only when a feature is reliable enough to produce an individual classification result that cannot be overruled by any other feature's normal classification. If the result given by **FF** feature is over a maximum threshold, the segment will have a forced-class of *music*. Similarly if **PR** within a segment is above a certain threshold, the segment will have a forced-class of *speech*. Likewise, if the overall segment **SC** standard deviation is above a maximum threshold, it has a forced-class of *speech* and if the **SC** standard deviation is less than a minimum threshold or **SC** mean is larger than a maximum threshold, it has a forced-class of *music*. If the standard deviation of **SC** within a segment is in the close neighborhood of the center of the fuzzy region and the **SC** mean value is within a defined threshold region, the forced-class type for this segment will be *NotClassified*.

The final classification decision over a segment is made when the classification results from each technique (**PR**, **FF**, **SC**) are fed into a sequential three-step decision making process:

- i. **Initial Decision:** The initial decision is based on **PR** classification and **FF** classification. If both of these classifications agree upon the classification type of a segment as *speech* or *music*, it is awarded that class. Otherwise, the class of this segment is kept as *NotClassified*.
- ii. **Intermediate Decision:** The intermediate classification is based on the **SC** classification and the initial decision result. The initial decision for a segment is kept as intermediate decision unless **SC** has classified the segment as *NotClassified*. Otherwise the intermediate decision depends upon the initial decision and the absolute distance of the **SC** standard deviation value from the center of the so-called fuzzy region. In such case, if the initial decision is one of the pure class types (i.e. *speech* or *music*), the intermediate decision is a *NotClassified* class for a segment whose **SC** standard deviation value is strictly closer to the center of the fuzzy region. Otherwise, the initial decision is kept as intermediate decision. Similarly, if both initial decision and **SC** classification agree upon the classification of a segment to be *NotClassified*, it is kept as a *NotClassified* segment in the intermediate

classification, unless the **SC** standard deviation value is strictly closer to the pure class boundaries of the fuzzy region. In this case, the segment is awarded one of the pure class types whose boundary is closer to the **SC** standard deviation value.

- iii. **Final Decision:** The final classification results are derived by superimposing the prioritized forced classification on the intermediate classification if there exists a forced class for that segment. Otherwise, the intermediate classification is kept as the final classification. The prioritized forced classification model handles any contradictions between the features at the forced classification level. This priority model has *music*, *speech* and *NotClassified* in descending order of priority. Figure 5 summarizes all operations in Step 3.

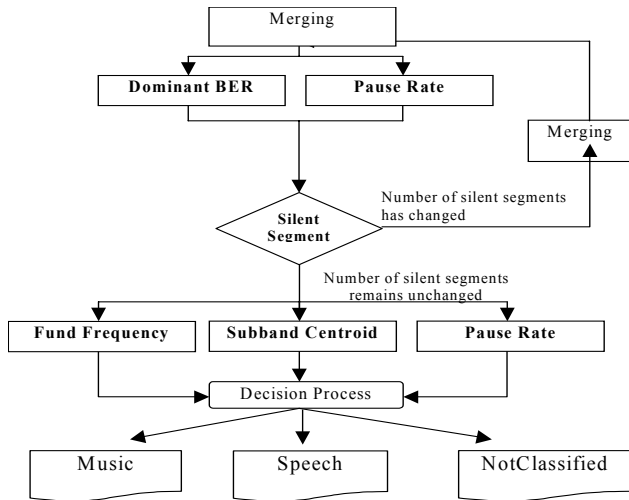


Figure 5: Step 3.

4) Further Segmentation (Sub-Segment Analysis): Once final classification and segmentation is finished in Step 3, a further segmentation is performed in order to separate sub-segments, which are not separated by silent parts. For example within a segment there might be sub-segments that include music and speech without a silent part in between. The first part in this step tests if the non-silent segment's length is significantly larger than a given threshold. In this case *inner-breakpoints detection algorithm* is used to detect the new breakpoints within such a segment. The breakpoint is the real limit between speech and music without a silent part between them. Through inner-breakpoints detection algorithm we find the music granules by detecting the granules that have windowed **SC** standard deviation (as explained in Step 3) less than a threshold. First, we look for a music granule, while the former granule is speech. Once we have found this first granule, we then keep on looking for the next limit granule (granule whose class is different from one of its neighbors). If we find one, we then check the length of the non-music sub-segment following this limit granule. If this sub-segment is shorter than a noise-threshold, we do not

consider the limit granule and keep looking for a valid limit granule. In the opposite case, if this segment is longer than the noise-length threshold, we call it a sub-segment and start the roll-down algorithm. The roll-down algorithm is in charge of finding the first lowest point on the windowed **SC** standard deviation curve inside a short window around the detected breakpoint. This lowest point, so-called breakpoint, is the real limit between speech and music without a silent part between them. The breakpoint is validated only if the resulting segment's length is larger than a given music-length threshold. Otherwise, we keep the large segment unchanged and stop. Once the inner breakpoints are saved a *re-segmentation* is performed. The *re-segmentation* method creates a new segment out of the new breakpoints, only if the classification (based on Step 3) of the new segment is different from the previous one. Otherwise the classification of this segment is not changed. After the re-segmentation, a so-called *collateral decision* is applied to segments of small length. This method finds a short segment and checks if it has at least one close enough (distant less than a threshold) non-silent neighbor segment. If this is the case, the classification of the short segment is changed depending upon the class of the neighbor segment. Finally a *merging* operation is performed on each of those silent segments whose length is less than a threshold if its neighboring segments are matching.

3.2. Audio Framing within Valid Audio Segments

As mentioned in the previous section, there are three valid audio segments: *speech*, *music* and *NotClassified*. Since segmentation and classification are performed per granule/frame basis, such as per MP3 granule or AAC frame, a conversion is needed to achieve a generic audio framing for indexing purposes. The entire audio clip is first divided into a user or model-defined audio frames, each of which will have a classification type as a result of the previous step. In order to assign a class type to an audio frame, all the granules/frames within or neighbor of that frame should have a unique class type to which it is assigned. Otherwise it will be assigned as *uncertain* and will be excluded from any *AFeX* operation and hence audio indexing process. Figure 6 shows an audio framing example. If the audio is in PCM or ADPCM format, all the audio frames in the entire clip will be assigned as *NotClassified* as mentioned before.

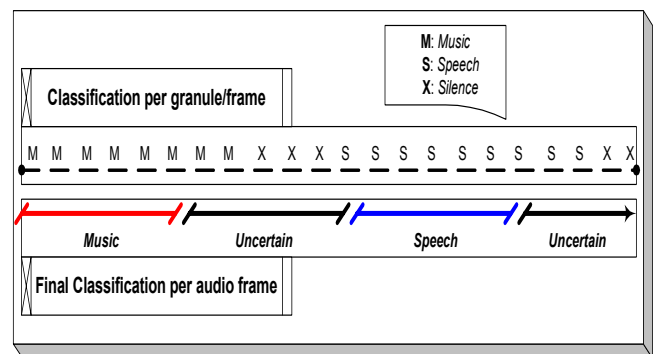


Figure 6: A sample audio classification conversion

The removal of audio frames in *Uncertain* and *Silence* types from *AFeX* operation has two advantages: first there is no need for *AFeX* operation on silent frames since no content information can be retrieved from them. Similarly the *Uncertain* frames are mixed and hence most of the times they are transition frames (i.e. music to speech, speech to silence, etc.). Therefore, the feature extraction will result in an unclear feature vector, which does not contain clean content characteristics. The second advantage is the significant reduction of the number of audio frames from the indexing scheme. This will eventually reduce the computational time for some indexing operations such as audio feature extraction and Key-Framing. Most important of all, this will further reduce the retrieval time of any clip query within a database.

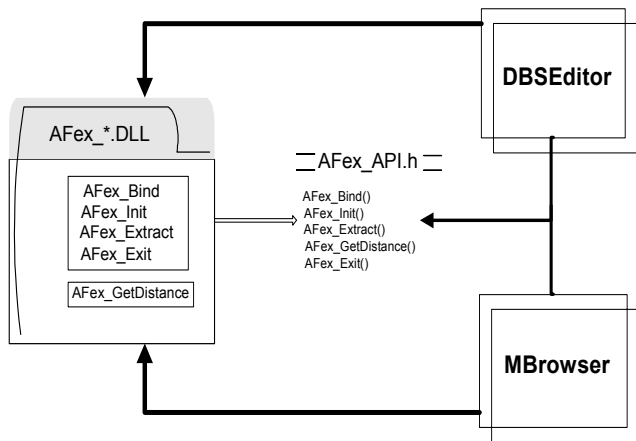


Figure 7: Basic *AFeX* Module interface with MUVIS applications

3.3. Audio Feature Extraction (*AFeX*) Framework

Once audio framing is completed, feature extraction is applied to those frames with a valid class types (*music*, *speech* or *NotClassified*) for indexing. *FeX* framework for visual feature extraction has been presented in [13]. A similar framework (so called *AFeX*) has been developed to accomplish audio feature extraction operations. Therefore, *AFeX* framework mainly supports dynamic audio feature extraction module integration for audio clips. This framework forms a common basis to compare, merge and experiment among several exclusive *AFeX* modules to develop new algorithms and to improve efficiency. Figure 7 shows the API functions and linkage between MUVIS applications and a sample *AFeX* module. All audio feature extraction algorithms should be implemented as a *DLL* with respect to *AFeX* API, and stored in an appropriate folder. *AFeX* API provides the necessary handshaking and information flow between a MUVIS application and an *AFeX* module.

Currently MFCC *AFeX* module is successfully implemented in MUVIS. MFCC stands for Mel-Frequency Cepstrum Coefficients [17] which are widely used in several speech and speaker recognition systems due to the fact that they provide a decorrelated, perceptually-oriented

observation vector in the cepstral domain. Therefore, they are suitable for the human audio perception system. This is the main reason that we use them for audio based multimedia indexing and retrieval in order to achieve a similarity measure close to ordinary human audio perception criteria such as ‘sounds like’ with additional higher level content discrimination via classification (i.e. *speech*, *music*, etc.).

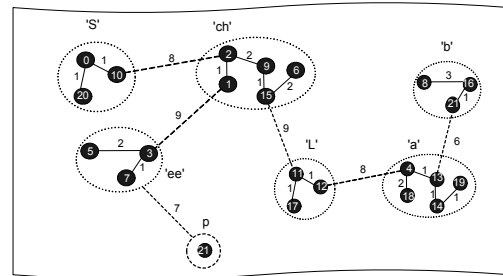
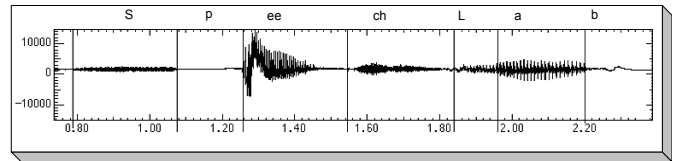


Figure 8: An illustrative clustering scheme

3.4. Key-Framing via MST Clustering

The number of audio frames is proportional to the duration of the audio clip. Once *AFeX* operation is performed, it may result in a massive number of feature vectors, many of which are probably redundant due to the fact that the sounds within an audio clip are immensely repetitive and most of the time entirely alike. In order to achieve an efficient audio-based retrieval within an acceptable time, only the feature vectors of frames from different sounds should be stored for indexing purposes. This is indeed a similar situation with the visual feature extraction scheme where only the visual feature vectors of the Key-Frames (KFs) are stored for indexing. There is however one difference: in the visual case KFs are known before feature extraction. However, in aural case there is no such a physical ‘frame’ structure and hence the audio is framed uniformly with certain duration. Thus, one needs to know features of each frame beforehand in order to perform Key-Frame analysis. This is why *AFeX* operation is performed (over valid frames) first and then KFs are extracted.

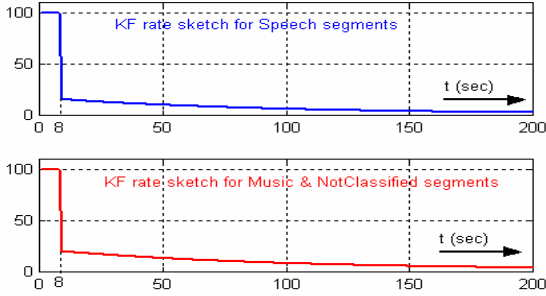


Figure 9: KF rate (%) model sketches

In order to achieve an efficient KF extraction, the audio frames, which have similar sounds (and therefore, similar feature vectors) should first be clustered and one or more frame from each cluster should be chosen as a KF. An ideal case example is shown in Figure 8. Here the problem is to determine the number of clusters that should be extracted over an entire clip. This number will in fact vary with the audio content. For instance a monolog speech will have less number of KFs than an action movie. For this we define *KF rate* that is the ratio between KF numbers over the total number of valid frames. Once a practical *KF rate* is set, the number of clusters can be easily calculated and eventually this number will be proportional to the duration of the clip. However, longer clips will increase the chance of bearing similar sounds. Especially if the content is mostly based on speech, similar sounds (vowels and unvoiced parts) will be repeated over time. Therefore, *KF rate* can be dynamically set via a Key-Framing model that is shown in Figure 9.

Once the number of KFs (*KFno*) is set, the audio frames are then clustered using *Minimum Spanning Tree* (MST) clustering technique [16]. Every node in MST is a feature vector of a unique audio frame and the distance between the nodes is calculated using the *AFEX* module *AFEX_GetDistance()* function. Once the MST is formed, then the longest *KFno-1* branch is broken and as a result *KFno* clusters are obtained. By taking one (i.e. the first) frame as a KF, the feature vectors of the KFs are then used for indexing. There is however one practical problem during MST formation: the number of comparisons and distance calculations is proportional to the square of the number of nodes (valid frames) present. So this brings a practical maximum number (N_c^{MST}) of nodes for MST formation. Therefore, if the number of nodes is exceeding this practical node limit, then the audio frames are first split into chunks, containing N_c^{MST} nodes within chunk c . For each chunk, MST clustering and associated Key-Framing process are performed and the feature vectors of the KFs are then stored for audio indexing.

3.5. Aural Retrieval-by-Query Scheme

As explained in detail in the previous sections, the audio part of any multimedia item within a MUVIS database is indexed using one or more *AFEX* modules that are dynamically linked

to the MUVIS application. As explained in section 3.1, the indexing scheme uses the audio classification per segment information to improve the effectiveness in such a way that during an audio-based query scheme, the matching (same audio class types) audio frames will be compared against each other via the similarity distance. There is one particular exception to this. As shown in Figure 2, some of the segments of an audio clip might have *NotClassified* type due to some ambiguity in the classification scheme or simply the lack of a reliable classifier. Naturally this should not prevent the retrieval since such parts may include important content information. Therefore, the audio frames with *NotClassified* type are entirely included in the retrieval scheme and since their class types are unknown, they are to be compared with all the frames, which have a valid class type. For instance, the similarity measurement for the audio frames of the queried clip in *speech* type will only be performed with the audio frames of the database clips having *speech* type and *NotClassified* type. This scheme based on classification will prevent any false (mismatched) query retrievals and reduce the computational time significantly.

In order to accomplish an audio based query within MUVIS, an audio clip is chosen from a multimedia database and queried through the database if the database includes at least one audio feature. Let NoS be the number of feature sets existing in a database and let $NoF(s)$ be the number of sub-features per feature set, where $0 \leq s < NoS$. As mentioned earlier, sub-features are obtained by changing the *AFEX* module parameters or the audio frame size during the audio feature extraction process. Let the similarity distance function be $SD(x(s, f), y(s, f))$ where x and y are the associated feature vectors of the feature index s and the sub-feature index f . Let i be the index of the audio frames within class C_q of the queried clip. Due to the aforementioned reasons, the similarity distance is only calculated between a sub-feature vector of this frame (i.e. $QFV_i^{C_q}(s, f)$) and an audio frame (index j) of the same class type from a clip (index c) within the database. For all the frames that have the same class type ($\forall j \Rightarrow j \in C_q$), one audio frame, which gives the minimum distance to the audio frame i in the queried clip is found ($D_i(s, f)$) and used for calculation of total sub-feature similarity distance ($D(s, f)$) between two clips. Figure 10 illustrates the class matching and minimum distance search mechanisms during the similarity distance calculations per sub-feature. Furthermore, two factors should be applied during the calculation of $D(s, f)$ in order to achieve unbiased and robust results:

i) Penalization: If no audio frames with class type C_q can be found in clip c then a penalty is applied during the calculation of $D(s, f)$. Let $N_Q(s, f)$ be the number of valid frames in a query clip and let $N_Q^\emptyset(s, f)$ be the number of frames that are not included for the calculation of the total sub-feature similarity distance due to the mismatches of their class types. Let $N_Q^\circ(s, f)$ be the number of the remaining frames, which will all be used in the calculation of the total sub-feature

similarity distance. Therefore, $N_Q(s, f) = N_Q^{\ominus}(s, f) + N_Q^{\ominus}(s, f)$ and the class mismatch penalty can be formulated as follows:

$$P_Q^c(s, f) = 1 + \frac{N_Q^{\ominus}(s, f)}{N_Q(s, f)} \quad (2)$$

If all class types of the query clip match with the class types of the database clip c , then $N_Q^{\ominus}(s, f) = 0 \Rightarrow P_Q^c(s, f) = 1$ and this case naturally applies no penalty to the calculation of $D(s, f)$.

ii) Normalization: Due to the possibility of the variation of the audio frame duration for a sub-feature, the number of frames having certain class types might change and this results in a biased (depending on the number of frames) similarity sub-feature distance calculation. In order to prevent this, $D(s, f)$ should be normalized by the total number of frames for each sub-feature ($N_Q(s, f)$).

Therefore, this will yield a normalized $D(s, f)$ calculation, which is nothing but the sub-feature similarity distance per audio frame. Since the audio vectors are normalized, the total query similarity distance (QD_c) between the query clip and a clip c in the database is calculated with a weighted sum. The weights can be used for experimentation in order to find an optimum merging scheme for the audio features (and associated sub-features) available in the database. The following expression formalizes the calculation of QD_c .

Calculation of QD_c via Eq. 3 is only valid if there is at least one matching class type between the query clip and the database clip c . If no matching class types exist, then QD_c is assigned as ∞ and hence it will be placed among the least matching clips (to the end of the query retrieval queue). This is an expected case since the two clips having nothing in common with respect to a high-level content analysis such as their mismatching audio class types per segment.

$$D_i(s, f) = \begin{cases} \min [SD(QFV_i^{c_q}(s, f), DFV_j^{c_q}(s, f))]_{j \in C_c} & \text{if } j \in C_q \\ 0 & \text{if } j \notin C_q \end{cases} \quad (3)$$

$$D(s, f) = \frac{P_Q^c(s, f)}{N_Q(s, f)} \cdot \sum_q \sum_i^{i \in C_q} D_i(s, f)$$

$$QD_c = \sum_s \sum_f^{NaF(s)} W(s, f) \cdot D(s, f)$$

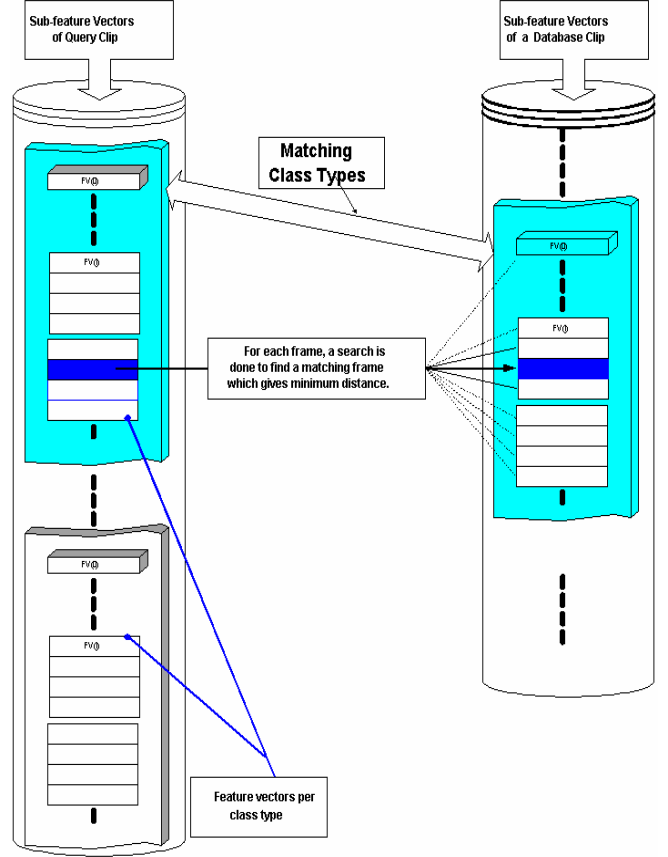


Figure 10: A class-based audio query illustration showing the distance calculation per audio frame

4. REMARKS AND CONCLUSIONS

Audio-based retrieval will be demonstrated in the presentation. Two particular experiments can be mentioned. In the first one, a video clip (with MP3 audio) is queried over a database containing over 800 audio and video (with audio) clips with a total duration around 28 hours. The second experiment involves a database of 181 video (with audio) clips with a total duration of around 12 hours. The capturing/encoding parameters (i.e. sampling frequency, sound volume level, bit-rate, etc.), encoder types (MP3, AAC, G721, etc.), clip duration and file formats of the clips are varying among the values given in Table 1 in both databases. The best matches in both experiments are video (with audio) and audio clips from the database with a significant relevance to the query clip.

Several retrieval experimental results show the effectiveness of the overall $FeX - AFeX$ indexing and retrieval framework and the available $FeX - AFeX$ modules. The audio indexing framework achieves a significant query performance and it provides a robust and generic solution for several multimedia types, capture parameters, coding methods, file formats and several other factors that MUVIS system so far supports.

MUVIS framework 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 framework supports browsing, hierarchic video representation and summarization. Most important of all, MUVIS framework supports integration of the aural and visual feature extraction algorithms explicitly. This brings a significant advantage for third parties to develop and test several feature extraction modules that are independent from the MUVIS applications.

5. REFERENCES

- [1] 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 the IWDC 2002 Conference on Advanced Methods for Multimedia Signal Processing*, Capri, Italy, September 2002.
- [2] S. Kiranyaz, K. Caglar, O. Guldogan, and E. Karaoglu, "MUVIS: A Multimedia Browsing, Indexing and Retrieval Framework", *Proc. Third International Workshop on Content Based Multimedia Indexing, CBMI 2003*, Rennes, France, 22-24 September 2003.
- [3] A. Pentland, R.W. Picard, S. Sclaroff, "Photobook: tools for content based manipulation of image databases", *Proc SPIE (Storage and Retrieval for Image and Video Databases II)* 2185:34-37, 1994.
- [4] J.R. Smith and Chang, "VisualSEEK: a Fully Automated Content-Based Image Query System", *ACM Multimedia*, Boston, Nov. 1996.
- [5] Virage. [URL:www.virage.com](http://www.virage.com)
- [6] 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.
- [7] ISO/IEC JTC1/SC29/WG11, "Overview of the MPEG-7 Standard Version 5.0", March 2001.
- [8] S. Kiranyaz, K. Caglar, B. Cramariuc, and M. Gabbouj, "Unsupervised Scene Change Detection Techniques In Feature Domain Via Clustering and Elimination", *Proc. IWDC 2002 Conference on Advanced Methods for Multimedia Signal Processing*, Capri, Italy, September 2002.
- [9] ISO/IEC JTC1/SC29/WG11, "Coding of Moving Pictures and Audio: Overview of MPEG-4 Standard", V. 21, March 2002.
- [10] ITU-T Recommendation H.263, "Video Coding For Low Bit Rate Communication", February 1998.
- [11] ISO/IEC 13818-3:1997, Information Technology – Generic Coding of Moving Pictures and Associated Audio Information – Part3: Audio, 1997.
- [12] ISO/IEC CD 14496-3 Subpart 4: 1998, Coding of Audiovisual Object Part3: Audio, 1998.
- [13] O. Guldogan, E. Guldogan, S. Kiranyaz, K. Caglar and M. Gabbouj, "Dynamic Integration of Explicit Feature Extraction Algorithms Into MUVIS Framework", *Proc. Finnish Signal Processing Symposium, FINSIG 2003*, Tampere, Finland, 2003.
- [14] 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", *Proc. SPIE/EI'99 Conference on Storage and Retrieval for Image and Video Databases VII, Vol.3656*, San Jose, California, 26-29 January 1999.
- [15] S. Kiranyaz, M. Aubazac, M. Gabbouj, "Unsupervised Segmentation and Classification over MP3 and AAC Audio Bitstreams", *WIAMIS Workshop*, pp. 338-345, London, 2003.
- [16] Graham, R.L., and O. Hell, "On the History of the Minimum Spanning Tree Problem," *Ann. Hist. Comput.* 7, pp. 43-57. 1985.
- [17] L. R. Rabiner and B. H. Juang, "Fundamental of Speech Recognition", Prentice hall, 1993.
- [18] Ching-Yung Lin, Belle L. Tseng, Milind Naphade, Apostol Natsev and John R. Smith, "VideoAL: A Novel End-to-End MPEG-7 Video Automatic Labeling System," *Proc. IEEE ICIP 2003*, Barcelona, Spain, 14-17 September 2003.