Key Frame Extraction and Indexing for Multimedia Databases

Mohamed Ahmed¶ Ahmed Karmouch¶ Suhayya Abu-Hakima¶¶

¶ School of Information Technology & Engineering (SITE), University of Ottawa, 161 Louis-Pasteur, Ottawa, ON, Canada, K1N 6N5
E-Mail: mahmed@sol.genie.uottawa.ca , karmouch@elg.uottawa.ca

¶¶ AmikaNow! Corporation IPF, Building M-50, Ottawa, ON, Canada, K1A 0R6
E-Mail: suhayya@amikanow.com

Abstract

The need for video processing tools has emerged in recent years. The main problem of video analysis is that it is a notoriously weak-structured problem. There is no fixed video style that we could use to parse directly. Moreover, there are many media formats and standards nowadays. Thus, in this paper, we introduce a new system to analyze and process different media file formats in an efficient and consistent manner. In addition to the different normal media browsing operations, the system implements three different video cut detection mechanisms. These three mechanisms are based upon the color histogram content summarization. One of these algorithms is based on the HSV color space. The other two algorithms are based on the RGB color space. A detailed algorithm will be depicted along with its approach to achieve quick performance although it tries to solve common problems for video cut detection. These problems are the detection of false cuts and missing true cuts. Then, a comparative study will be provided between the different mechanisms to measure their performance and reliability in different configuration environments. This work is part of the Mobile Agents Alliance project among Ottawa university, National Research Council (NRC) and Mitel Corporation in Canada.

1. Introduction

In the past decade, there has been significant work done in the area of image analysis and recognition so that we could partition a video source into separate segments. This process could be used to index the video within the multimedia databases and thus we could query and navigate through the database. Following are some of the algorithms used to measure the differences among consecutive frames. *Pixels-Pair wise comparison* [2] is a simple way to detect a quantitative change between a pair of images by comparing the corresponding pixels in the two frames to determine how many pixels have changed. The total percentage of the pixels changed is evaluated and if this percentage exceeds some preset threshold, we decide that a frame change has been detected. Many color space systems could be used for the comparison such as: RGB, HVC, HSV, YIQ or L*u*v. The advantage of this method is its simplicity. However, its disadvantages exceed the advantage. One disadvantage is that it has a large processing overhead to compare all consecutive frames. Also, it does not conclude if large objects moved within the shot before terminating the continuous shot.

The *Spatial, Temporal Skips* [2] *(Histogram Analysis)* [2, 3, 4] methods benefit from the redundant characteristic of the video frames either in the spatial dimension or the temporal dimension. The use of color histograms has been verified to be more robust against objects and camera movements within the same shot. Moreover, the very near video frames are similar except for the cut frames. We could compare temporally every defined number of frames instead of all the consecutive frames and/or spatially not all the pixels within the frames. Thus, we could save great processing time and resources during the analysis. We could use the color histogram distributions of the frames to make the comparison. The histogram comparison algorithm is less sensitive to object motion than the pixels-pair wise comparison algorithm. Figure 1 shows a histogram distribution function. There are many histogram difference measures that could be used to detect a shot cut.

For example:

1) Difference = $\Sigma \mid H_i(j) - H_{i+1}(j) \mid$
for j = 1 To N

Where, $H_i(j)$ is the histogram for color j in Frame i,
N = Number of used colors

2) Difference = $\Sigma (\mid H_i(j) - H_{i+1}(j) \mid^2 / H_{i+1}(j))$
for j = 1 To N

Where, $H_i(j)$ is the histogram for color j in Frame i,
N = Number of used colors

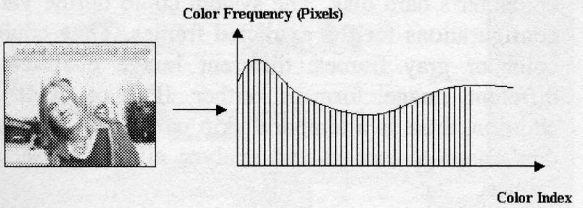

Figure 1. Color Histogram Function

Using a preset threshold, we could discover the existence of a frame change and thus the shot cut operation is detected. N. Hirzalla [1] has described a new detailed design of a key frame detection algorithm using the HVC color space. First, for every two consecutive frames, the system converts the original RBG coloring space into the equivalent HVC Histogram coloring representation because the HVC space mirrors more the human color perception. The system uses the Hue histogram distribution for performing the comparison instead of the intensity distribution to reduce, to some extent, the variations in intensity values due to light changes and flashes.

G. Pass et al. [5] proposed a histogram refinement algorithm using color coherence vectors based on local spatial coherence. W. Wolf [6] provides an algorithm to detect the key frames in MPEG video file format using optical flow motion analysis. I. Sethi et al. in [7] use a similarity metric to measure the similarity through both the hue and saturation components of the HSI color space between two images. G. Pass et al. [8] define a notion of Joint Histograms. They use many local features in comparison including color, edge density, texture, gradient magnitude and the rank of the pixels.

2. MediABS System description

A prototype system called "*MediABS*", was implemented addressing multi video format browsing and processing. The system tries to discover the different cut changes within the video file. The system runs on Microsoft-Windows NT and Windows95 environments. The current functions and capabilities of the system are as follows:

- Media Segment:
  - -Play, Pause, Resume, Stop, Repeat and Audio Volume Control
- Media Speed Adjustment
- Sound on/off
- Video Frames Random Access:
  - -GoTo, Next, Previous, Begin, Middle, End, Skip and Scroll
- Verifying the Input Media Format
- Specified Frames regions Extraction and Analysis
- Temporal, Spatial skip - Defaults: 5 frames, 5 pixels respectively
- Frames save Formats: JPG, BMP : Color, Grey-scale
  - -Each compressed frame size is about only 4KB
- Frames Analysis:
  - Hue Histogram Difference ratio
  - 6 Most significant RGB bits Intensity Difference
  - 6 Most significant RGB bits with the use of Blocks Intensity Difference
- Exporting Video analysis configurations and results to a SpreadSheet file

The main window of the system is shown in figure 2. Three cut detection algorithms were implemented in the system. They are the HSV-Hue Histogram Difference, the 6 Most Significant RGB Bits Intensity Difference and the 6 Most significant RGB bits with the use of Blocks Intensity Difference. We could define a certain video segment to be analyzed. Using the temporal and spatial information redundancy within the video, we could improve the performance without sacrificing the accuracy of the algorithm results.

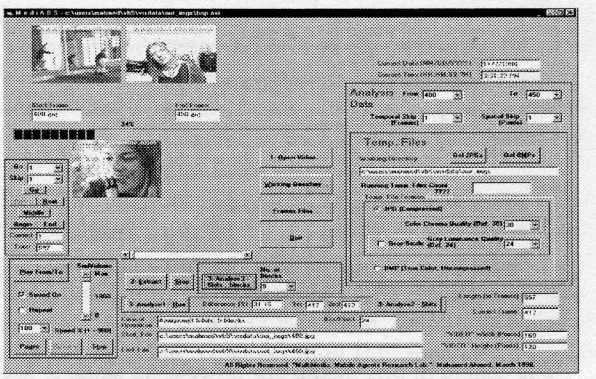

Figure 2. MediABS Main Window

3. Media Unification Pre-Processing Stage

The system uses a media unification phase for different media standards before analyzing the video file. These standards are the Audio Video Interleaved (AVI),

Apple Quick Time for Windows (QTW) and Motion Picture Experts Group (MPEG). The use of this process allows us to simplify the media analysis management afterwards because we don't need to worry about the different characteristics of each format for processing. Irrespective of the information coding in these formats, we use the same processing modules for them. Thus, we could state that this simplifies managing and analyzing the video in all cases. The different components of this pre-processing stage are illustrated in figure 3.

The components of this stage are:

- *Media Format Handler Extension:* This component could be regarded as the media driver of each media format. It handles the special format coding of the corresponding format representation. This layer could be extended to handle other media devices (e.g. CD-ROM, CD-I, the new DVD standard, ...etc). Streaming media could be processed in the same manner as well without re-doing the already implemented processing mechanisms of the next stages.
- *Media Formats Unification Module:* This module is used to unify the environment and the operations that will be used for the media processing in the next stage. For example, some of the functions of this module are controlling the speed of video display, enabling or disabling the audio component of the video, adjusting the audio volume level, the ability to specify certain video segments to analyze and browse, ...etc.

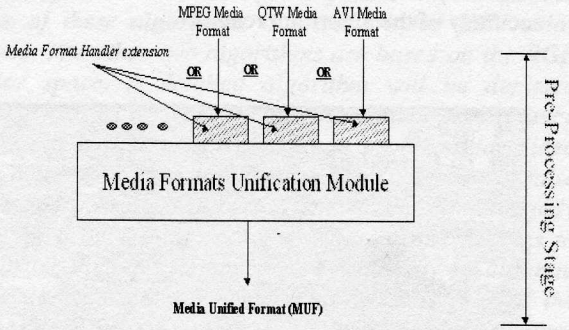

Figure 3. Media Unification Pre-Processing Stage

The output of this media pre-processing stage is seen as a *Media Unified Format (MUF)* content that will be used in further media handling operations. The use of this format will reduce the complexity of handling different input standards since we don't need the special characteristics of these formats in the current implemented media processing algorithms. Thus, for example, we don't have to worry about the format to access certain frames within the video with certain temporal skip though we know that the coding of the information is very much different between MPEG and AVI video formats for instance.

4. Video Change Cut Detection Stage

For this process, the system extracts the consecutive frames needed to detect the camera cut events. Thus, the representing frames are grabbed and stored in the computer's hard disk. The system could define various configurations for the extracted frames. They could be color or gray frames, different image qualities and different image formats (either JPG or BMP). In addition, there is a temporal skip parameter, so that we don't need to extract and analyze all the consecutive frames.
For cut detection, the system implements a spatial skip parameter as well to improve the performance without sacrificing the accuracy of the detection to benefit from the redundant information within the frames. The system, as said before, implements three cut detection algorithms. In the following sections, the description of each algorithm will be depicted.

4.1 The Hue Histogram Difference Algorithm

To realize this algorithm, the system first converts each compared pixel of the frames, taking the defined spatial skip parameter into consideration, from RGB color space into HSV Color space. The Hue [1] component histogram for each frame is evaluated. Then, the system uses this 2 Hue histogram difference to discriminate between every two consecutive frames, taking the defined temporal skip parameter into consideration. The following formula is used:

$$\text{Hue Difference} = \tfrac{1}{2} * \Sigma \, | H_2(i) - H_1(i) | \; / \; \Sigma \, H_1(i) \quad \text{for } i = 1 \text{ To } N$$

Where, $H_1(i)$ is the Hue Histogram distribution for frame M,
$H_2(i)$ is the Hue Histogram distribution for frame (M + Temporal Skip),
N = the possible Hue values

Then, if this Hue Difference exceeds some defined threshold, the 2 frames are said to represent a camera cut detected.

4.2 The 6 Most Significant RGB bits Intensity Difference Algorithm

In this algorithm, the system makes use of the 24 bit RGB color space components of each compared pixels (each component has 8 bits representation) directly.

However, to speed the performance considerably, the system exploits only the 2 most Significant bits [2] of each color (using a masking operation) which means actually that we define only 64 ranges of color degrees for the entire RGB color space. Then, similar steps as in section 4.1 are used to detect the camera cut operation. A snapshot of a system result using this algorithm is shown in figure 4.

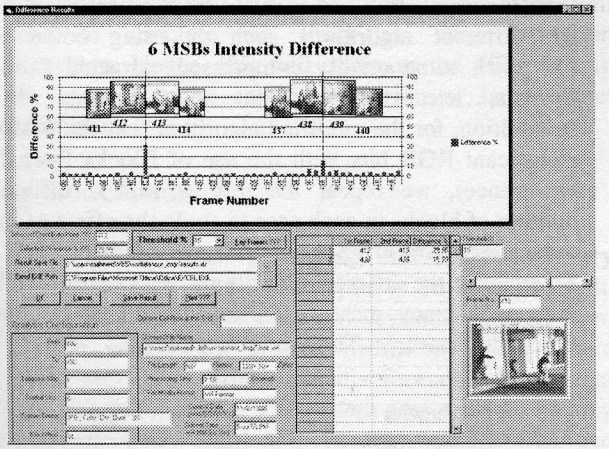

Figure 4. MediABS 6 MSBs Histogram results

4.3 The 6 Most significant RGB bits with the use of Blocks Intensity Difference

Following the results of the previous algorithm (The 6 Most Significant RGB bits Intensity Difference Algorithm), the system gave poor results in few cases. One problem was that the previous algorithm makes use only of the global color distribution information. This results in the system not discovering cut detection even there is a cut detection in the compared frames. The two examples, shown in figure 5, illustrate this. The problem being that the previous algorithm ignores the locality information of the color. That means that the 2 consecutive 6 MSBs color histograms are similar despite the fact that the actual spatial distributions of the colors are not the same. Thus, the algorithm was extended to suit these circumstances.

We make use of partitioning each frame into a number of disjoint blocks. This allows us to make use of the locality information of the histogram distribution to a great extent. The system evaluates every corresponding two block histogram difference between the compared frames. Thus, the solution to the previously described problem is shown in figure 6.

There is another issue that was discovered while experimenting. It is the false detection problem. It occurs here mainly because of the use of the temporal skip for processing. Thus, it was realized that if this temporal skip is significantly high along with a quick change in the same continuous shot due to object movement or camera operation such as tilting, panning

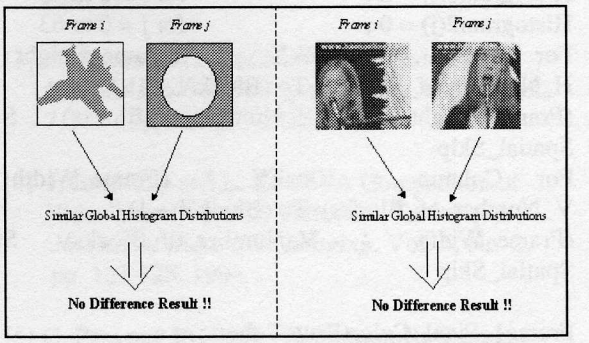

Figure 5. Wrong Frames Unchange Decision !

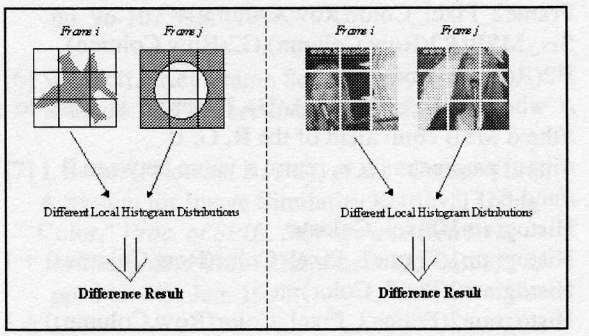

Figure 6. Correct Frames Change Decision

or zooming, the algorithm will recognize the frames as significantly different despite this not being true. Thus, an additional step is advised. After the first cut detection process, we need to analyze these specific changed frames more comprehensively [2] provided that the temporal skip is already greater than one. We do that in order to compensate for the speed of possible object and camera movements taking into consideration that the use of blocks difference magnifies the effect of any object or camera movements. Therefore, we re-analyze these region frames but with temporal skip equal to one. Hence, the system first needs to extract all the frames included in the current analyzed region. The algorithm thus became more accurate in refusing false cuts obtained from the first process while maintaining true camera cut change accuracy.

Now, the modified algorithm could be described as follows for every two consecutive frames, taking the temporal skip into consideration (i.e. separated by (the temporal skip – 1) number of frames):

```
Total_Blocks_Difference = 0 ;
For BlockX = 0 To (H_Number_of_Blocks – 1)
For BlockY = 0 To (V_Number_of_Blocks – 1)
// where,H_Number_of_Blocks is the Horizontal
//Number of Blocks
// and V_Number_of_Blocks is the Vertical
```

```
//Number of Blocks
{
Histogram1(j) = 0 ;                    for j = 0 to 63
Histogram2(j) = 0 ;                    for j = 0 to 63
For Row = BlockX * (Frame_Height / H_Number_of_Blocks) To (BlockX + 1) * (Frame_Height / H_Number_of_Blocks) Step Spatial_Skip
For Column = BlockY * (Frame_Width / V_Number_of_Blocks) To (BlockY + 1) * (Frame_Width / V_Number_of_Blocks) Step Spatial_Skip
{
Frame1_Pixel_Color(Row,Column)=
Six_MSB(R1(Row,Column),G1(Row,Column),
B1(Row,Column)) ;
Frame2_Pixel_Color(Row,Column)=
Six_MSB(R2(Row,Column),G2(Row,Column),
B2(Row,Column)) ;
// where, Six_MSB(R, G, B) Function is used to get
//the 6 MSB equivalent of the R, G, B
//input parameters (i.e. return a value between 0
//and 63)
Histogram1(Pixel_Color)=
Histogram1(Frame1_Pixel_Color(Row,Column)) + 1 ;
Histogram2(Pixel_Color)=
Histogram2(Frame2_Pixel_Color(Row,Column)) + 1 ;
}
Block_Difference = ½ * Σ | Histogram2(i) – Histogram1(i) | / Σ Histogram1(i) ;  for i = 0 to 63
Total_Blocks_Difference=Total_Blocks_Difference + Block_Difference ;
}
Final_Difference_Avg = Total_Block_Difference / (H_Number_of_Blocks * V_Number_of_Blocks) ;
IF Final_Difference_Avg > Threshold AND Temporal_Skip >1 THEN
{
Temporal_Skip = 1 ;
Extract all the current region included frames ;
Repeat the process again using Temporal_Skip =1 ;
// i.e. re-analyze again
}
IF Final_Difference_Avg > Threshold AND Temporal_Skip = 1 THEN
{
Decision of Cut Detection ;
Exit ;
}
Else
{
Decision of NO Cut Detection ;
Exit ;
}
```

5. Testing Evaluation

Various testing experiments have been undertaken. We used different file formats and these files have different testing conditions such as dark shots and camera operations. We tested the three cut detection algorithms to compare their behavior with the different file formats and the different testing conditions. In addition, the testing uses some other constant testing conditions for the different algorithms such as using color, 24 bits/pixel, same quality compressed extracted frames formats, …etc.

In addition, for the modified algorithm (i.e. the 6 Most significant RGB bits with the use of Blocks Intensity Difference), we repeat the testing with a different number of blocks in each case to study the effect of this parameter on the efficiency of this algorithm.

To evaluate the different algorithms, we used two well-known accuracy measures. They are the Recall and Precision measures. They are defined as:

$$\text{Recall} = \frac{\text{Correct Cuts}}{\text{Correct Cuts} + \text{Missed Cuts}} \quad , \quad \text{Precision} = \frac{\text{Correct Cuts}}{\text{Correct Cuts} + \text{False Cuts}}$$

The testing makes use of initial temporal skip and spatial skip values of 5 and 5 respectively. This resulted clearly in very quick performance as expected. The summary of the results, using 7 different video files, is as shown in table 1.

The results show a better accuracy for the 6 Most Significant RGB bits with the use of Blocks Intensity Difference algorithm over the other two algorithms for detecting camera cut changes. The correct cuts detection has increased significantly.

Table 1. Results summary of using 7 different formats files

	Correct	False	Missed	*PRECISION*	*RECALL*
Hue Difference Percentage	40	10	26	0.8	0.606061
6 MSBs Intensity Difference	57	4	9	0.93442623	0.863636
6 MSBs, 4 BLOCKS Intensity Difference	61	3	5	0.953125	0.924242
6 MSBs, 9 BLOCKS Intensity Difference	62	4	4	0.939393939	0.939394
6 MSBs, 25 BLOCKS Intensity Difference	64	4	2	0.941176471	0.969697
6 MSBs, 64 BLOCKS Intensity Difference	63	4	3	0.940298507	0.954545

The recall measure is improved because of the significant decrease of the missed true cuts. This occurs because of the use of disjoint blocks as shown previously in figure 6. The precision measure is improved mainly because of the increase in the correct cut detected in addition to reducing the false cuts generated comparing to the Hue Histogram Difference algorithm.

Another issue is the number of the used blocks. The results show the consistent behavior of the algorithm with respect to the number of blocks. Nevertheless, the 25 number of blocks (5 Horizontally * 5 Vertically) has shown slightly better overall results relative to the other numbers. However, we need to mention here that we should not increase the number of blocks very much. That is first because it will result in slow performance of the cut detection and in addition the algorithm will tend to simulate the Pixel-Pair wise histogram algorithm. Hence, this will decrease the efficiency in the case of quick camera operation or large object movement.

6. Conclusion

The work in video processing and analysis is very interesting and important in spite of its complexity. In this paper, we described a new system, MediABS, to browse, process and analyze media files. The system deals with three different media formats in consistent and similar processes. The system uses a pre-processing stage to unify the view of the management and processing of the video files irrespective of the input file format. The browsing and processing mechanisms of the different files do not change from one format to another. This eases the handling of the video files to a great extent. This architecture could be used in future to support other media formats such as streaming media.

The system implements three different video processing algorithms for camera cut detection. We described a better algorithm to detect camera cut in different conditions. The algorithm partitions the video frames into disjoint blocks to diminish the missed true cuts problem. Also, the problem of false cut detection is reduced through the comprehensive analysis of the included frames in the case of cut detection provided that the temporal skip was not equal to one. The algorithm makes use of temporal and spatial skips to speed the performance significantly.

The algorithm has shown promising results in different conditions relative to the other two algorithms. In future, the system could be extended to cover the detection of different camera operations such as panning, tilting and zooming. In addition, the system will be used to provide a key framing service remotely for users over the Internet.

References

[1] N. Hirzalla "Media Processing and Retrieval Model for Multimedia Documents," *PhD Thesis,Ottawa University*, Jan. 1997.

[2] H. Zhang, A. Kankanhalli and S. Smoliar "Automatic Partitioning of Full-Motion Video," *Multimedia Systems*, Vol. 1, No. 1, pp. 10-28, Apr. 1993.

[3] M. Swain and D. Ballard "Color Indexing," *International Journal of Computer Vision*, Vol. 7, No. 1, pp. 11-32, 1991.

[4] J. Boreczky and L. Rowe "A Comparison of Video Shot Boundary Detection Techniques," *Journal of Electronic Imaging*, Vol. 5, No. 2, pp. 122-128, 1996.

[5] G. Pass and R. Zabih "Histogram Refinement for Content-Based Image Retrieval", *In IEEE Workshop on Applications of Computer Vision*, pp. 96-102, Dec. 1996.

[6] W. Wolf, "Key Frame Selection by Motion Analysis," *in Proceedings of ICASSP '96.*

[7] I. Sethi, I. Coman, B. Day, et al., "Color-WISE: A System for Image Similarity Retrieval Using Color," *Proc. of SPIE, Storage and Retrieval for Image and Video Databases VI*, Vol. 3312, pp. 140-149, Jan. 1998.

[8] G. Pass and R. Zabih "Comparing Images Using Joint Histograms", *ACM Journal of Multimedia Systems*, 1998.