CHAPTER 1

INTRODUCTION

With rapid advance in multimedia and network communication technologies, there are more and more multimedia in our daily lives. Through the multimedia media such as Interactive Digital TV, Personal Media Player, 3G Mobile, laptop PC, …, etc., people can watch TV program or video at any time, and the interactive functionality between users and service providers becomes more and more urgent. Since we have countless video sources on the internet, how to find the user desired video efficiently is an interesting issue. In general, a video retrieval system should address two issues: 1) video analysis and representation and 2) the retrieval of video on user queries.

The video type will influence the video analysis performance. Decompressing whole video takes time and a lot of storage and extracting features from a raw video is inefficient. Most of the existing efficient video retrieval systems [2-5] analyze and extract video features from MPEG-1/2 videos without fully decompressing video stream. Due to the development of video compression technology, the state-of-the-art video coding standard, H.264, is introduced. It performs better compression rate than MEPG-1/2 but still preserves the video quality. In the coming day, it will replace the

MPEG-1/2 video compression standard in most video applications.

For a video retrieval, people could make a query by key words, sketch a trajectory, or send a query clip. Some video retrieval systems [1-2] provide a video browsing interface to users, and during the video browsing, users may select any video unit as a query to retrieve from the database, and present the results to users. The drawback of this kind of system is that it needs more feedbacks from users to find the video. Other systems [3-4] provide users to sketch a target object motion trajectory as a query, and these systems will return all video clips which have the similar moving trajectory objects. However, the objects must be extracted from video. Besides, it is not suitable for videos with few moving objects. In practice, a user can only specify a rough trajectory. There will be too many relevant results. Other systems [5] provide users to submit a video clip to retrieve similar video clips or the complete counterpart video. The query clip in [5] is got from the source videos. However, in the real world application, videos with the same content but got from different video capture devices will have different color distributions. This will make their system impractical.

Motivated by the above reasons, a video retrieval system on H.264 video stream is proposed. Our objective is to develop a video retrieval engine to help users find the complete video. By sending the information of query clip to the video retrieval engine, the engine will return the complete video to users. In the compressed domain, we can get the luma information from an I-frame and motion information from a P-frame. Thus, on the video analysis, video is segment into shots first, and two useful features, lumas and motions, are extracted. On the process of luma feature extraction, key frames are selected and a luma calibration is applied. After that similar luma features are efficiently clustered. On the process of motion feature extraction, a statistic scheme is proposed to extract and combine object motion feature and camera motion feature. In the proposed method, no object extraction and camera motion parameter estimation are needed. On the retrieval process, the two features are used to query videos and no feedback requirement. The system architecture is shown in Fig. 1-1.

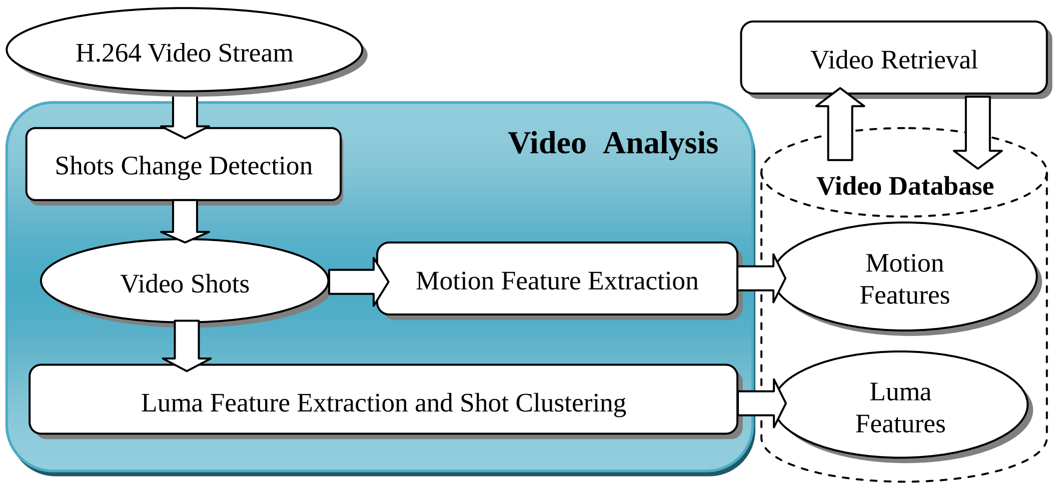


Fig. 1-1. Architecture of proposed video retrieval system.

The rest of this thesis is organized as follows. In Chapter 2, some background knowledge about terminologies and H.264 video compression technology will be introduced. In Chapter 3, we will describe the method of the shot change detection. In Chapter 4, we will introduce the scheme of luma feature extraction and luma clustering. In Chapter 5, the proposed method of motion feature extraction will be introduced. The retrieval process and experiment results are introduced in Chapter 6. Finally, in Chapter 7, we will give the conclusion and future work.

CHAPTER 2

BACKGROUND

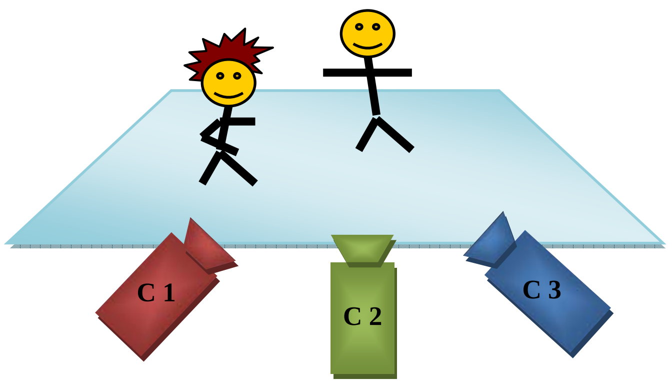
2.1 Terminology

Before talking about each block of our system, we first introduce some important terms used in the digital video retrieval field defined in [7].

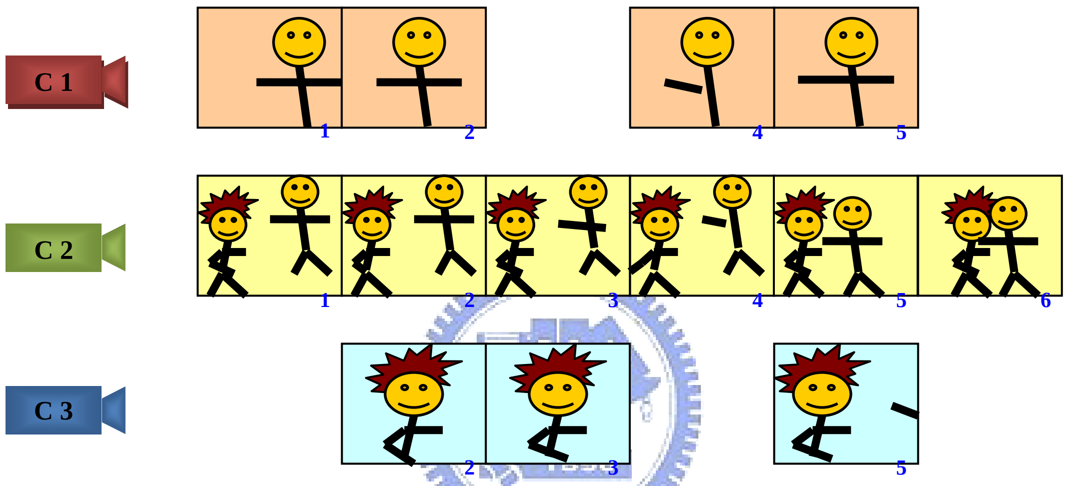
Video shot: A consecutive sequence of frames recorded from a single camera. It is the building block of video streams. Shots originate with the invention of motion cameras and are defined as the longest continuous sequence that originates from a single camera take, which is what the camera images in an uninterrupted run (see Fig. 2-1).

Key frame: The frame that represents the salient visual content of a shot. Depending on the complexity of the content of the shot, one or more key frames can be selected.

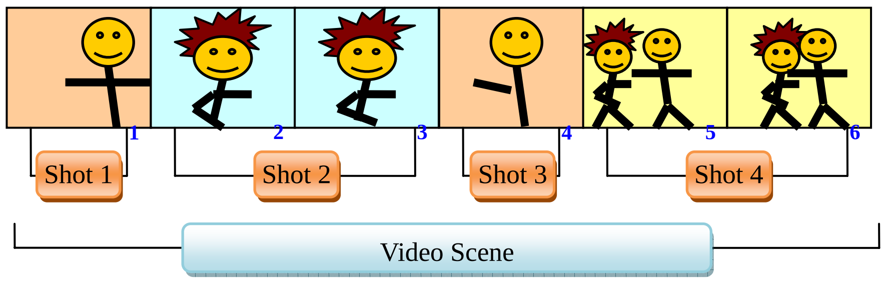
Video scene: A collection of semantically related and temporally adjacent shots, depicting and conveying a high-level concept or story. While shots are marked by physical boundaries, scenes are marked by semantic boundaries.



(a)



(b)



(c)

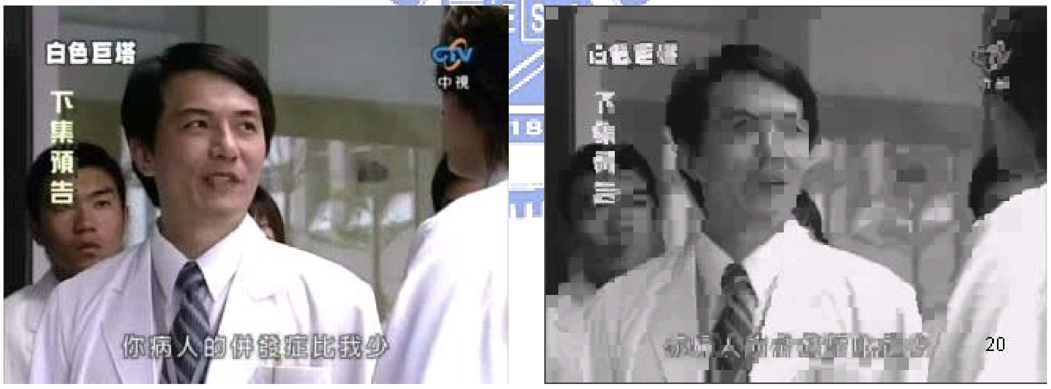
Fig. 2-1. Video production process. (a) Actual objects that are imaged by three cameras C1, C2, and C3. (b) Camera's takes of C1, C2, and C3. (c) Creating a video clip from 3 different cameras' takes.

2.2 H.264 Video Compression Technology

Since we do the video analysis on the H.264 bit stream, in this section, we will introduce two main compression technologies used in H.264 standards [13] briefly.

The first technology is the intra frame prediction. In the intra mode, a prediction block P is predicted from neighboring, previously encoded and reconstructed blocks and is subtracted from the current block prior to encoding. For the luma samples, P is formed for each 4 × 4, 8 × 8 blocks or 16 × 16 macroblock. There are total of nine optional prediction modes for each 4 × 4 luma block, four modes for 8 × 8 luma block and 16 × 16 luma macroblock, and four prediction modes for the 8 × 8 chroma components. The encoder typically selects the prediction mode for each block that minimizes the difference between P and the block to be encoded. According to the improvement technology of intra frame encoding used in H.264, there is no DC image information defined in MPEG-1/2 we can get from parsing the H.264 video stream. Fig. 2-2 shows an example of luma part of an intra prediction frame formed by choosing the 'best' 4 × 4 to 16 × 16 prediction mode.

The second technology is the inter frame prediction. MPEG-1/2 used 16 × 16 macroblock as the unit of motion estimation, while H.264 supports a range of block sizes (from 16 × 16 down to 4 × 4) and quarter-sample resolution in the luma component. The luma component of each macroblock may be split up in four ways and motion compensated either as one 16 × 16 macroblock partition, two 16 × 8 partitions, two 8 × 16 partitions or four 8 × 8 partitions. If the 8 × 8 mode is chosen, each of the four 8 × 8 sub blocks within the macroblock may be split in a further 4 *ways, either as one 8 × 8 sub-macroblock partition, two 8 × 4 sub-macroblock* partitions, two 4 × 8 sub-macroblock partitions or four 4 × 4 sub-macroblock partitions. These partitions and sub-macroblock give rise to a large number of possible combinations within each macroblock. Trying all combinations is very expensive, for efficiency, only seven modes are allowed (see Fig. 2-3).



(a) (b)

Fig. 2-2. Intra frame prediction. (a) Original frame.(b**)** Predicted Luma part of (a).

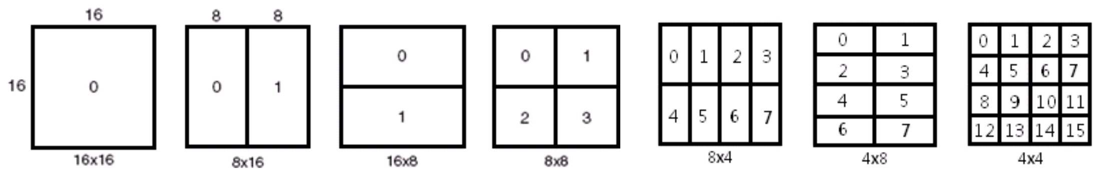


Fig. 2-3. Allowed macroblock partitions.

CHAPTER 3

VIDOE SHOT CHANGE DETECTION

Video shot change detection is usually the first step of the video analysis in most video retrieval systems. There are two types of shot transitions [7-10]: abrupt and gradual transition (dissolve, fade, wipe). The most common transitions appearing in TV programs are abrupt transitions, thus we focus on the issue of abrupt transition detection.

3.1 Abrupt Transition Detection

If a frame has a higher correlation to its reference frame, a shot change is not likely occurs. The method proposed in [10] detects a shot change according to the correlation between a frame and its reference frame. The correlation can be represented by the ratio of intra-predicted macroblocks, R_a, within a P-frame or a B-frame.

$$R_a = \frac{N_a}{N}, \qquad (3\text{-}1)$$

where N_a is the number of intra-predicted macroblocks and N is the number of total macroblocks in a P-/B-frame.

In order to parse the video stream only once, a static threshold is used. According to our experimental testing, the static threshold, T_{cut}, is set to 0.8 to judge if a shot change occurs in the current P-frame or B-frame. The decision rule is formulated as follows:

$$\begin{cases} R_a \geq T_{cut}, & \text{shot\_change} \\ R_a < T_{cut}, & \text{nonshot\_change} \end{cases} \tag{3-2}$$

3.2 Results

Eight video clips captured from TV dramas are used as testing data. To evaluate the performance of the simpfied method, recall rate, R, and precision rate, P, are defined as follows:

$$R = \frac{D}{D + MD} , \tag{3-3}$$

$$P = \frac{D}{D + FD} , \tag{3-4}$$

where D is the number of shot boundary correctly detected, MD is the number of misdetections, and FD is the number of false detections. Based on the above evaluation metric, the shot boundary detection results are reported in TABLE 3-1.

TABLE 3-1

RESULTS OF SHOT CHANGE DETECTION

Sequence Name	*Frames*	*Actual*	*D*	*MD*	*FD*	*R*	*P*
Test 0	2275	32	32	0	0	100%	100%
Test 1	833	6	6	0	0	100%	100%
Test 2	864	5	4	1	0	80%	100%
Test 3	457	3	2	1	0	66.7%	100%
Test 4	836	17	16	1	0	94%	100%
Test 5	1472	8	8	0	0	100%	100%
Test 6	510	5	5	0	0	100%	100%
Test 7	715	10	10	0	2	100%	83%

The missed detection in test2, test3, and test4 is due to that the shot change frame is encoded as an I-frame. If a shot transition occurs at an I-frame, the method does not detect I-frame and the shot boundary will be missed. And the false positives in test7 are due to the flash and the fast motion of a large object respectively.

CHAPTER 4

LUMA FEATURE EXTRATION

While segmenting video into shots, two useful features, lumas and motions, could be extracted. On the process of luma feature extraction, key frames are selected and the luma calibration is applied. After that similar luma features are efficiently clustered. Compared with lumas, motions offer better temporal information in video descriptions and will be introduced in Chapter 5. In this chapter, the luma feature extraction and the shot clustering will be introduced.

4.1 Luma Feature Extraction

On the process of luma feature extraction, for providing a suitable abstraction of a video shot, at least one key frame is selected depending on the complexity of the content. Besides, to treat this problem that videos with the same content but having different color distributions, the luma calibration is applied for each key frame. Finally, luma feature is extracted from the calibrated key frames. Fig. 4-1 shows the flow chart of luma feature extraction.

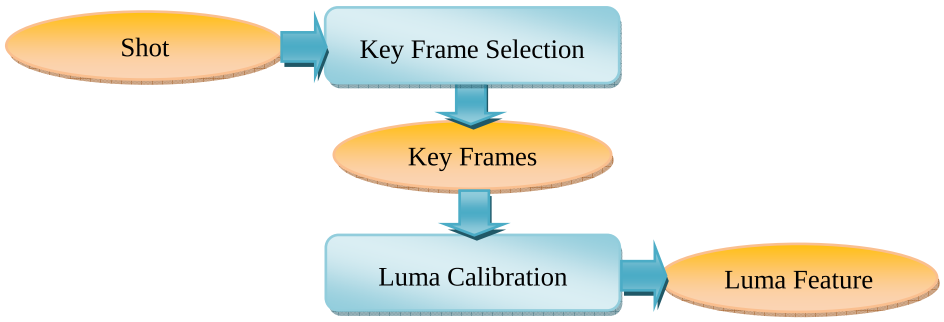


Fig. 4-1. Flowchart of luma feature extraction.

4.1.1 Key Frame Selection

For providing a suitable abstraction of a video shot, at least one key frame is selected depending on the complexity of the content. Frames during a camera motion or an object motion may be blurred, and thus not suitable for the key frame. In [11-12], they measured that a motion pattern is usually composed of motion acceleration followed by deceleration process. Therefore, motion is a salient feature in presenting actions or events in video to determine key frames and key frames should be selected from frames where the metric as a function has its local minima. Based on the idea mentioned above, a motion based scheme [11-12] is simplified for key frame selection.

Key frames are selected from I-frames in a shot. However, bit stream of I-frame does not contain any motion information. The motion information of previous P-frames is used as an approximation. Motion magnitudes and accumulate motion magnitudes (AMM) of successive P-frames before I-frames are calculated. AMM is used to determine whether the following I-frame could be selected as a key frame or not. We consider the I-frame whose AMM values are local minima as candidate key frames.

For finding the motion magnitude of a P-frame, *p*, which is firstly divided into non-overlapping 4 × 4 blocks. The motion magnitude is calculated as

$$M_p = \sum_{b \in B} \|MV_b\|, \tag{4-1}$$

where *B* is the set of 4 × 4 blocks in *p* and MV_b is the motion vector of a block *b*. Then the AMM of an I-frame, *i*, is defined as

$$AMM_i = \sum_{k \in P} M_k, \tag{4-2}$$

where *P* is the set of P-frames in the previous GOP of *i*. An I-frame is selected as a candidate key frame if its AMM is smaller than the AMM of its previous I-frame and the next I-frame. An example of the relation between camera movement and the AMM distribution in a shot is given in see Fig. 4-2. The frames marked in squares are the selected candidate key frames. We can see that the frame 252 is the time that camera changing from panning left to zoom in, and the camera is nearly stopped in that frame.

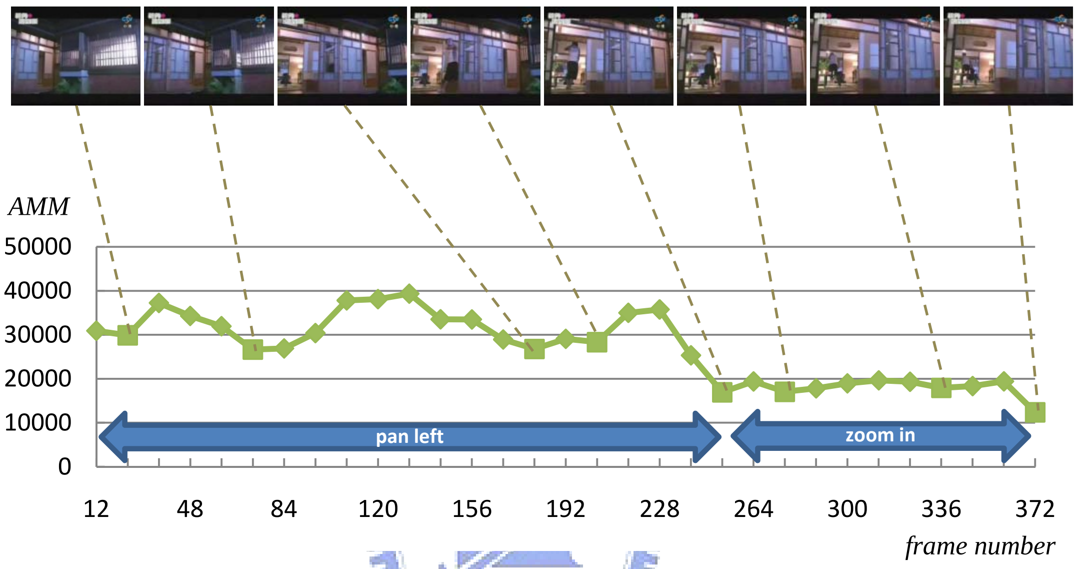
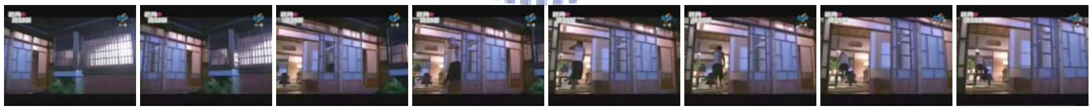


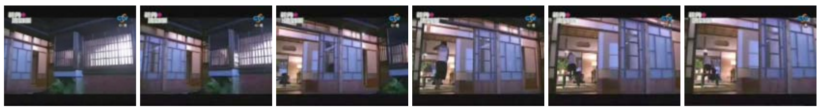
Fig. 4-2. Camera movement and AMM distribution.

After the previous step, too many I-frames are selected as key frames. We hope to remove similar key frames and only preserve some representatives for further video abstraction. Thus, the key frame redundancy removal is applied. The process of key frame removal set the first element of the candidate key frames as the reference key frame and as one representative key frame. Then, for each of the subsequent candidate key frames, the similarity between the candidate key frame and the reference key frame is evaluated. Once they are dissimilar, the candidate key frame is considered as one representative key frame and as the reference key frame for the following candidate key frames.

The key frame representation and the similarity evaluation between two key frames will be introduced in Section 4.1.3 and in Section 4.3. Fig. 4-3 to 4-5 show the results of key frame selection. Fig. 4-3 shows the preserved key frames of Fig. 4-2. Since there are camera movements in that shot, more key frames are preserved. In Fig. 4-4, a video clip is given by diving into six shots. Fewer key frames are preserved of each shot, since there are fewer motions in the video clip. In Fig. 4-5, another video clip is given by dividing into five shots. In Fig. 4-5(c)(d)(e), two key frames are preserved of these shots, since there are more motions in them.



(a)



(b)

Fig. 4-3. Preserved key frames of Fig. 4-2. (a) Candidates. (b) Representatives.

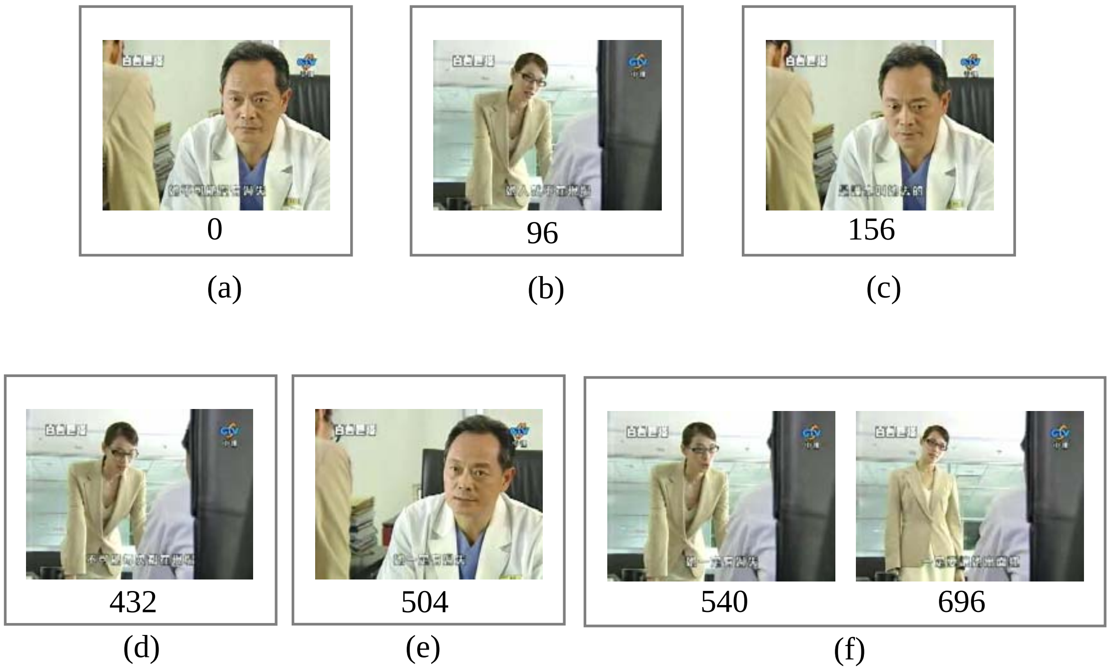


Fig. 4-4. Selected key frames of video clip “test1”. (a) Shot 1: one key frame is selected with frame number 0. (b) Shot 2: one key frame is selected with frame number 96. (c) Shot 3: one key frame is selected with frame number 156. (d) Shot 4: one key frame is selected with frame number 432. (e) Shot 5: one key frame is selected with frame number 504 (f) Shot 6: two key frames are selected with frame number 540 and 696.

4.1.2 Luma Calibration

Before we extract the luma feature from the selected key frames, there is an important issue we need to concern. Since the video trailer or clip may get from different kinds of media. Videos with the same content but got from different video capture devices have different color distributions (see Fig. 4-6). This is caused by video capture devices setting parameter differently and some physical constraint. To fix this issue, we apply a simple but useful calibration method to the predicted luma part of each key frame. The histogram equalization is applied to make every gray value distribute equally (see Fig. 4-7).

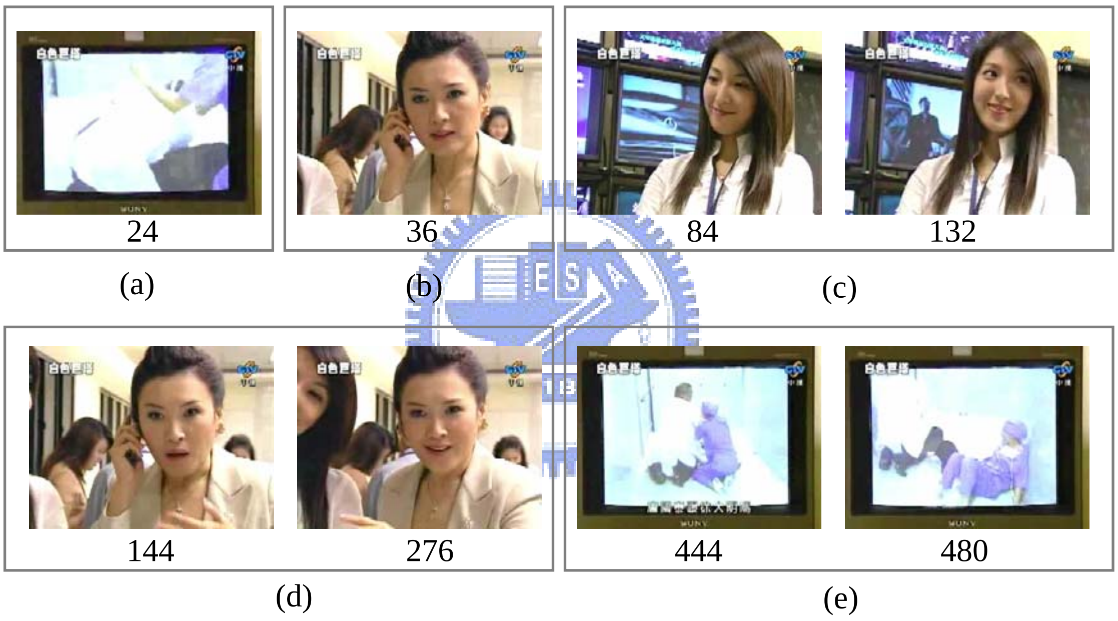
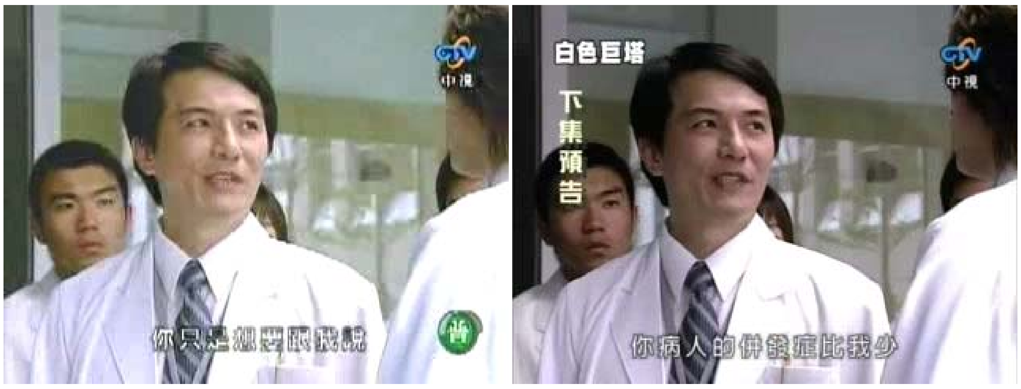


Fig. 4-5. Selected key frames of video clip “test2”. (a) Shot 1: one key frame is selected with frame number 24. (b) Shot 2: one key frame is selected with frame number 36. (c) Shot 3: two key frames are selected with frame number 84 and 132. (d) Shot 4: two key frames are selected with frame number 144 and 276. (e) Shot 5: two key frames are selected with frame number 444 and 480.

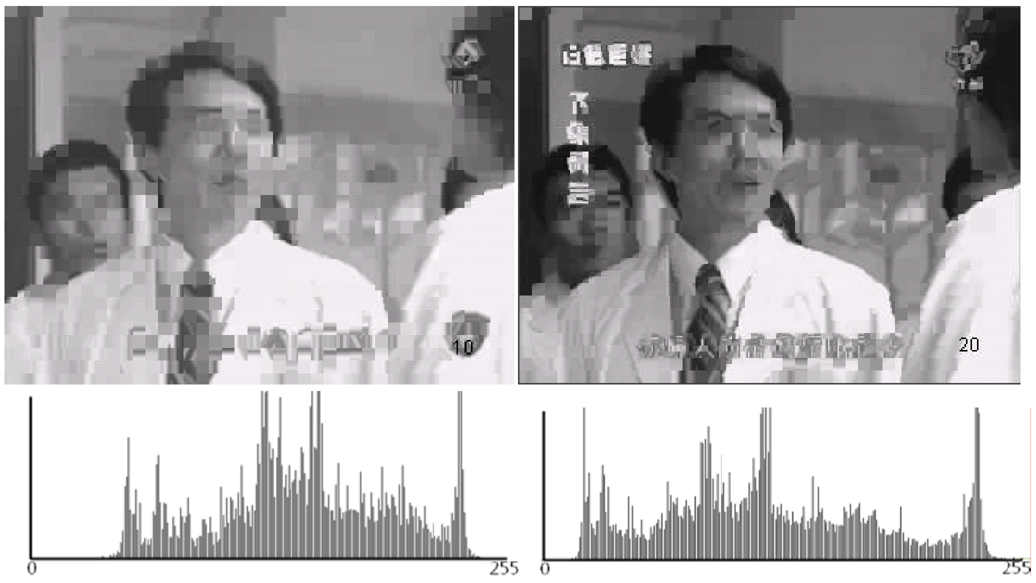


(a) (b)

Fig. 4-6. Two frames with nearly the same content but captured from different video devices. (a) Frame captured from a drama. (b) Frame captured from the drama trailer.

4.1.3 Key Frame Representation

Histogram is used to represent luma information of a key frame. The predicted luma part of each key frame is decoded and applied a uniform quantization to reduce the gray level into 32 levels. However, the histogram gives the global information of a key frame, to compare two key frames based on the information of histogram is not enough. Therefore, to get the local information, we divide a key frame into four vertical parts to provide four local sub histograms. Then these four sub histograms with 32 bins for each part is considered as the luma feature of a key frame.


(a)

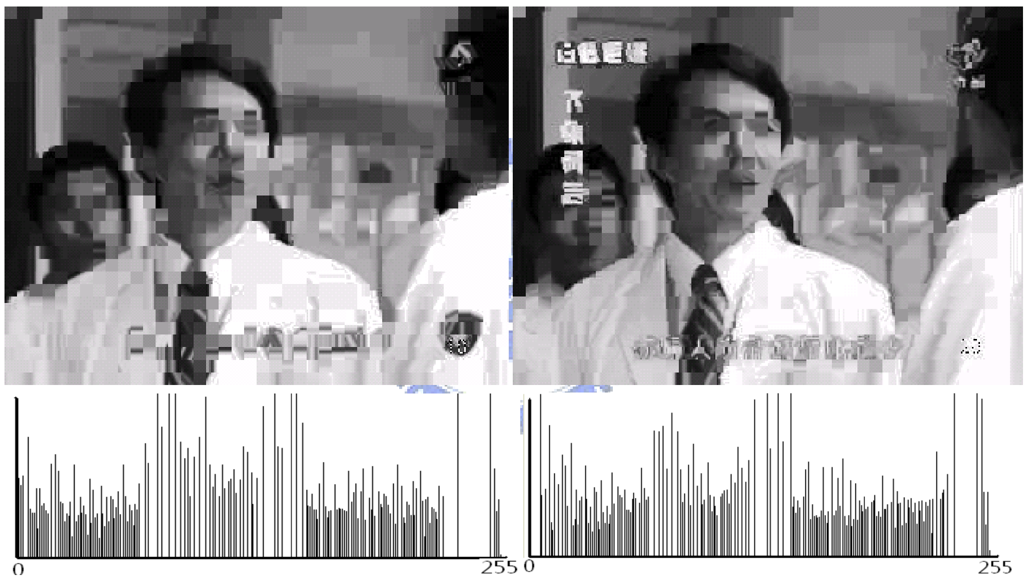

(b)

Fig. 4-7. The luma calibration. (a) Predicted luma part of Fig. 4-6(a) and Fig. 4-6(b) respectively. (b) (a) After histogram equalization.

4.2 Shot Clustering

In a video, there are many shots captured by the same camera in the same scene.

These shots are often similar to each other. As a result, taking luma features of all the shots in the video is not necessary. Besides, they are often temporal adjacent, and assume that two similar shots are adjacent no more than W shots. According to the above two facts, an efficient shot clustering method with timeout mechanism is proposed. The basic clustering method is described as follows.

(1) Set the first candidate shot into the reference list.

(2) For each of the subsequent candidate shots, the similarity between it and every shot in the reference list is evaluated.

(3) If the candidate shot is dissimilar with all of the reference shots, the candidate shot is put into the reference list and as a new element of the reference list.

(4) Repeat (2) and (3) until the candidate list is empty, and all the elements in the reference list are the representatives.

where the candidate list is the set of all candidate shots, and the reference list is the set of all reference shots.

Since these similar shots are often temporal adjacent. In order to prevent the reference list from being too long and to save comparison time and memory space, we set a timeout mechanism on the reference list. Each element in the reference list has a value called *timeout* to memorize the left time to stay in the reference list. Before the step (2), the *timeout* of each reference shot is deceased by one. After the step (3), if the candidate shot is similar with any one of reference shots, reset the *timeout* of the most similar reference shot to W. After that, those reference shots whose *timeout* become zero will be removed from the reference list. The computational time of the shot clustering is below 0.1 seconds in a 10 minutes video. Fig. 4-8 shows the preserved shots for a given video clips 'test 1'and only two shots are preserved.

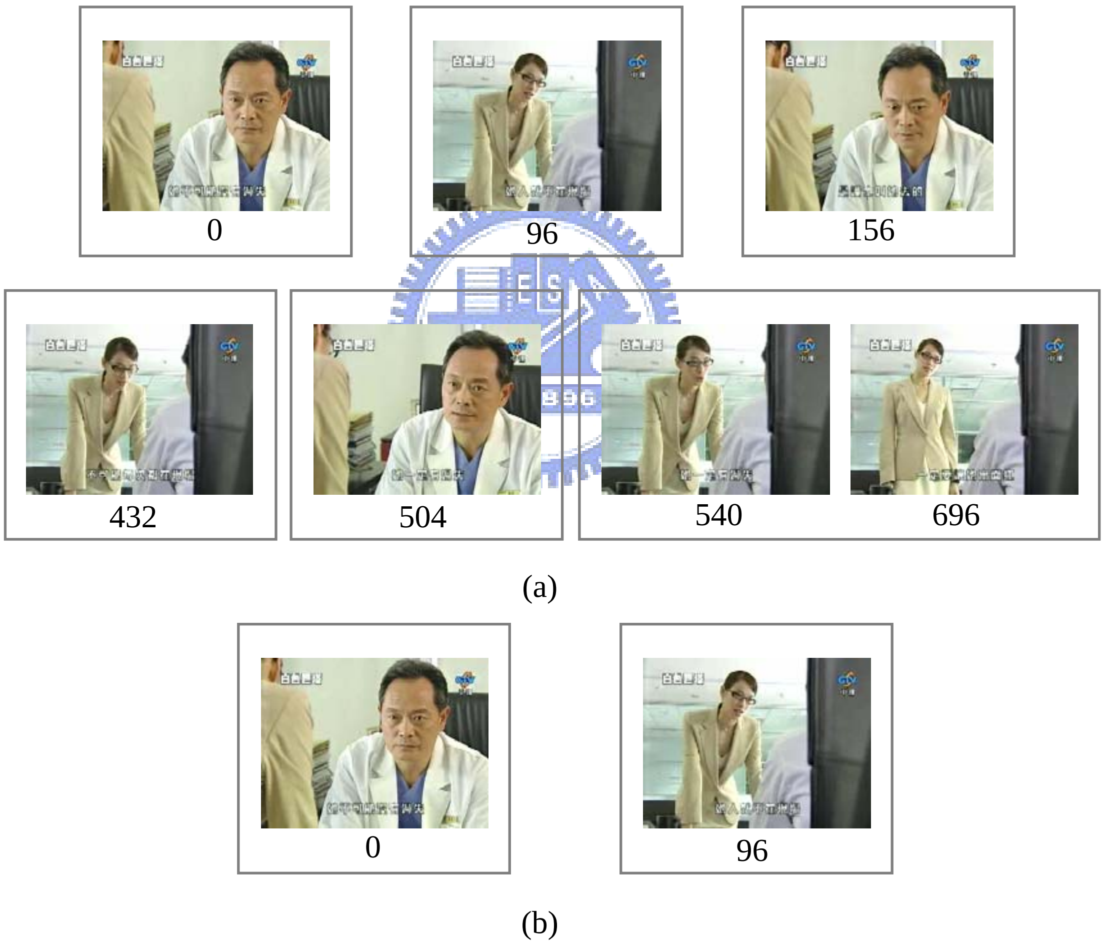
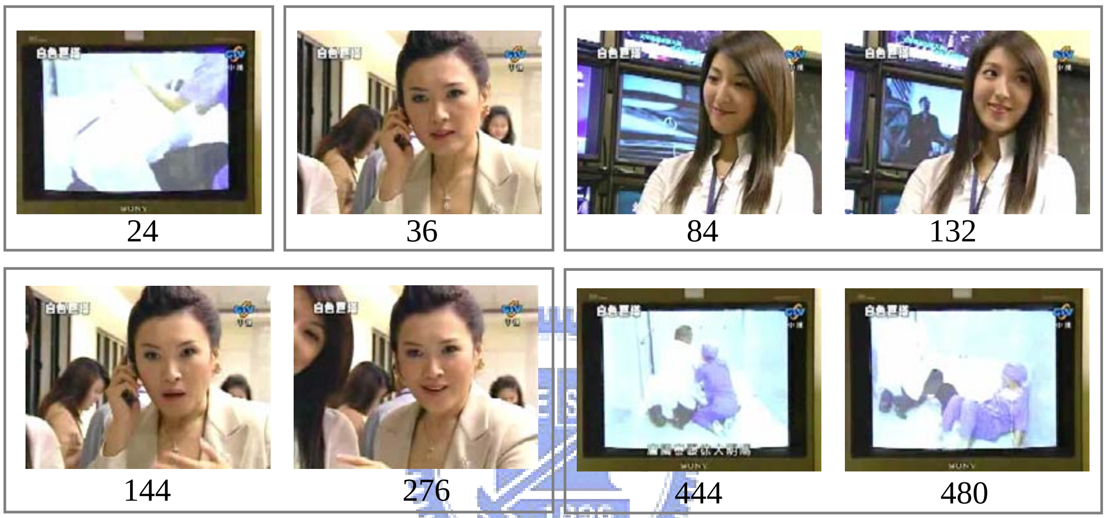
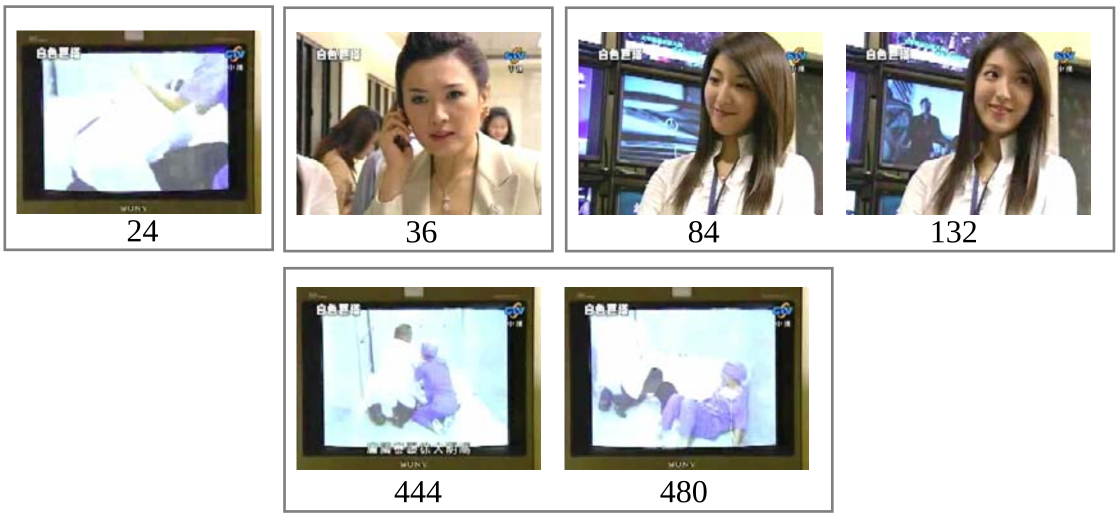


Fig. 4-8. Shot clustering of 'test1'. (a) Original shots. (b) Representative shots.

Fig. 4-9 shows the preserved shots for a given video clip ‘test2’, where more shots are preserved since there are more motions in it. After applying the shot clustering of a video, there are about 44% and 43% save about sport and drama videos, and 83% save about the talk show videos.



(a)



(b)

Fig. 4-9. Shot clustering of ‘test2’. (a) Original shots. (b) Representative shots.

4.3 Luma Feature Similarity Evaluation

In this section we will define the luma similarity measurement of two key frames and of two shots.

For the evaluation of key frames similarity, given key frame F_i, we denote its four sub histograms as $SH(k)^l_{F_i}$, where $k \in [0,3]$, $l \in [0,31]$. Then the similarity between F_i and F_j is given by (4-3) with $T_{diff} = 132.8$.

$$KeySim(F_i, F_j) = T_{diff} - \sum_{k=0}^{3}\sum_{l=0}^{31}\left|SH(k)^l_{F_i} - SH(k)^l_{F_j}\right| \tag{4-3}$$

At the shot level, given luma features of two shots, L_m and L_n. Assume there are M key frames ($F^i_m, i = 1, \dots, M$) contained in L_m and N key frames ($F^j_n, j = 1, \dots, N$) contained in L_n. Then the similarity between L_m and L_n is given by (4-4). If Eq. 4-4 is less than or equal to zero, it means these two shots are dissimilar.

$$ShotLumaSim(L_m, L_n) = \max_j \sum_{i=1}^{M} KeySim(F^i_m, F^j_n). \tag{4-4}$$

CHAPTER 5

MOTION FEATURE EXTRATION

Compared with lumas, motion can offer better temporal information in video descriptions and it is suitable in videos which have a lot of motions. Since two kinds of motions can be found from a video, i.e., object motion and camera motion. Motion feature can be extracted through two kinds of methods [4-5], i.e., object-based and camera-based ones. For the first one, most methods assume that all the desired objects have been manually extracted from video sequences. Then, modeling techniques are proposed for capturing desired object motion characteristic, i.e., the one for modeling objects' trajectories. However, the process of object extraction is expansive. For the camera-based ones, camera motion parameters are often estimated by using the camera motion model and be categorized to form a motion feature. Nevertheless, the camera motion estimation also takes time.

Based on our observations, in some TV programs such as talk shows, the camera is often still and only object moves none or slightly. Other TV programs, like dramas and sports, they are often blended by camera movements and object movements. As a result, both camera motion and object motion should be used. Motivated by above reasons, a statistic-based scheme is proposed to extract the motion feature which contains both dominant object motions and camera motions from a video shot but with no requirements of object extraction and camera parameter estimation.

5.1 Motion Feature Extraction

For motion feature extraction in one shot, we construct a motion histogram of each P-frame in advance. According to the motion histogram analysis, we extract and combine the dominant object motion vector(s) (ODMV) and dominant camera motion vector(s) (CDMV) of each P-frame. Fig. 5-1shows the flowchart of the proposed motion feature extraction.

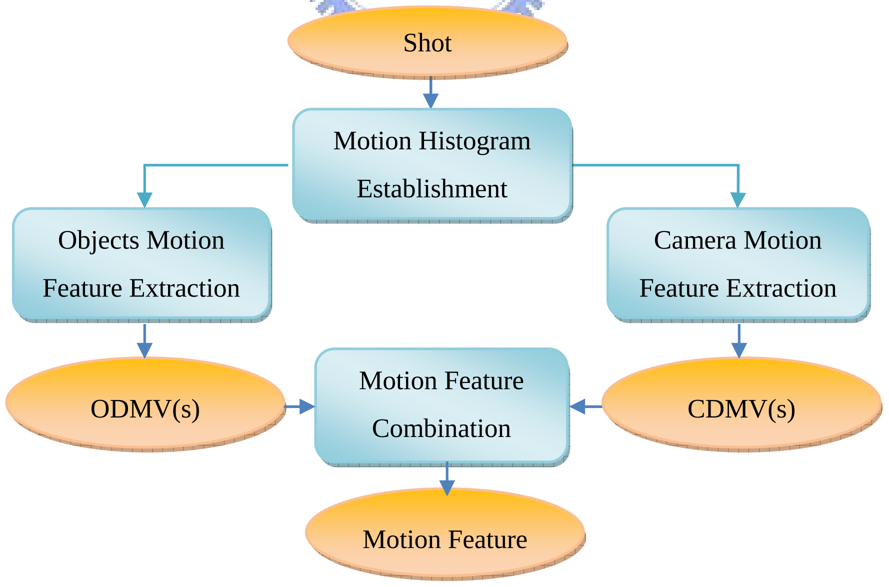

Shot
Motion Histogram Establishment
Objects Motion Feature Extraction
Camera Motion Feature Extraction
ODMV(s)
Motion Feature Combination
CDMV(s)
Motion Feature

Fig. 5-1. Flowchart of motion feature extraction.

Before the introduction of the proposed motion feature extraction, we define a structure to represent a motion feature of a shot. Motion feature is connected by consecutive motion feature points (MFP), each of which is a P-frame. Although B-frames also have rich motion vectors, they might be predicted from two different reference frames. It is more complex to analyze the motion feature in a B-frame. Therefore, we only analyze the motions of MFP of each P-frame. Then, each MFP could contain ODMV(s), CDMV(s) or no dominant motion (NULL) (see Fig. 5-2).

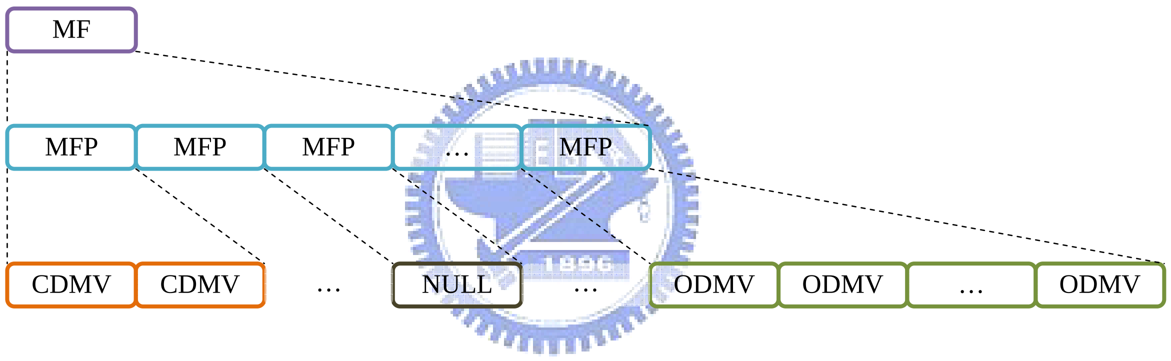


Fig. 5-2. Motion feature of one shot.

5.1.1 Motion Histogram

Motion histogram is constructed for the analysis of motion vectors distribution of each P-frame. Since each motion vector in H.264 is 1/4 pixel accuracy, and the search range defined in our system is ± 16 in both horizontal and vertical directions. We round the motion vector of each non-overlapping 4×4 block to its nearest integer motion vector and create a 2D motion histogram (MH) with size 33×33 ($16 \times 2 + 1$).

The MH is extended to the size of 35 × 35 for operation convenience (see Fig. 5-3(a)) and normalized to the range of 0~255.

5.1.2 Objects Motion Analysis and Feature Extraction

Based on our observation, most object motions are centralized in the center of the MH. For object motion extraction, an object motion matrix (OMM) which is built by copying values from (-2, -2) to (2, 2) of its MH (see Fig. 5-3(b)). The higher the value of the bin is, the more blocks are with the same motion vector in the P-frame. However, the (0, 0) bin whose value is the number of blocks nearly not moving is not taken into concern.

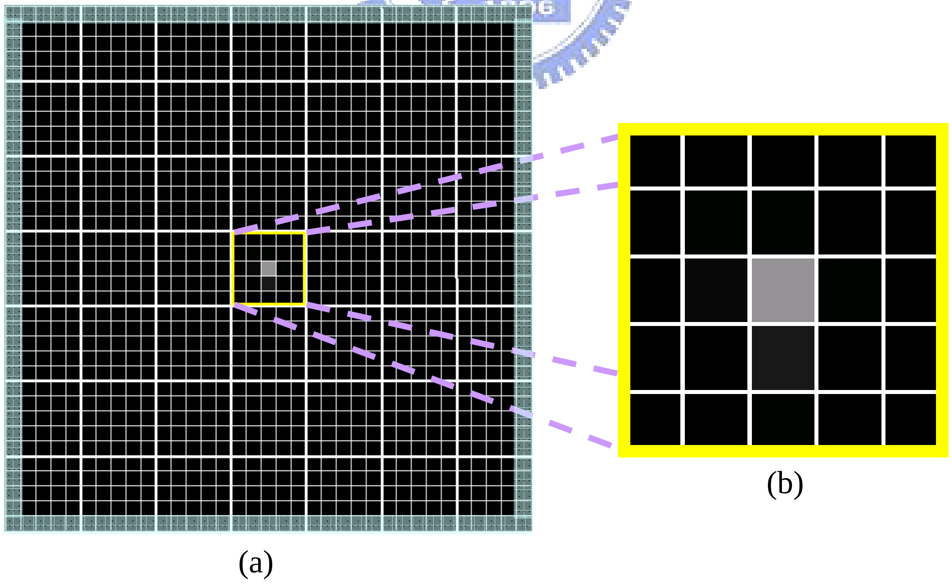


Fig. 5-3. The MH and the OMM. (a) The MH with size 35 × 35, the gray center point is (0,0), the top-left bin is (-17,-17), and the bottom-right is (17,17). (b) 5 × 5 OMM.

Since videos with the same content but captured by different devices may have different encoding sequence such as “P...P...I”, “P...I...P” or “I…P…P”. To make sure that the OMM sequence is not interrupt by an I-frame. We must predict the OMM of each I-frame which has no motion information in its bit stream. Since objects motions are consecutive in temporal domain, the successive OMMs also reflect the consecution of object motions. We could use the fact to predict the OMM of an I-frame. In the Fig. 5-4, assume we want to predict OMM_i of I-frame, i, and we have OMM_{p_b} of the previous P-frame, p_a, and OMM_{p_a} of the next P-frame, p_b. Each bin in OMM_i would be set as the average value of the corresponding bin in OMM_{p_b} and OMM_{p_3}. The predicting formula is defined as follows:

$$OMM_i(x,y) = round\left(\frac{OMM_{p_a}(x,y) + OMM_{p_b}(x,y)}{2}\right), \quad (5\text{-}1)$$

where $x, y = -2, \dots, 2$. Then, we use a static threshold to determine if a bin in the OMM is a dominant motion vector, however, some errors might happen. We give an example shown in Fig. 5-5 to illustrate this error. The dotted line in the figure is the predefined threshold. Consider the most top-left bin called bin0 in each OMM. Only bin0 whose value greater than the threshold is considered as a dominant motion vector, thus bin0 of OMM_a, OMM_c, OMM_d, *and* OMM_e are dominant ones. But actually, the value of bin0 in OMM_b is just a little less than the given threshold and its neighboring bin0s are all larger than the threshold. The bin0 of OMM_b should to be denoted as a dominant one.

… B P B

0	0	0	0	1
0	1	0	0	0
0	26	106	0	1
0	8	4	1	1
0	0	0	0	0

OMM_{p_b}

B I B

0	0	0	0	1
0	2	2	0	0
0	21	114	2	1
0	8	5	1	1
0	0	2	0	0

OMM_i

B P B…

0	0	0	0	0
0	3	3	0	0
0	15	121	4	1
0	7	6	1	0
0	0	3	0	0

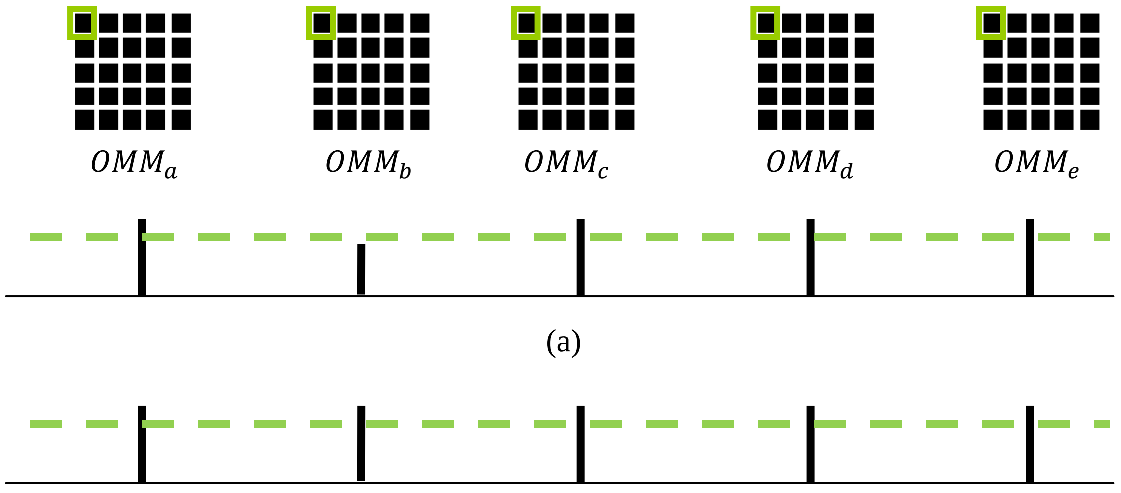
OMM_{p_3}

Fig. 5-4. OMM Prediction of an I-frame.

To solve this problem, we apply a 1D median filter with size 3 × 1 for each bin temporally, after that, the bin0 of OMM_b is considered as a dominant one. The ODMV(s) is then fetched from each OMM by

$$(x,y) \text{ is } \begin{cases} \text{a ODMV}, & \text{if } \mathrm{OMM}(x,y) \geq \mathrm{T}_O \\ \text{not a ODMV}, & \text{otherwise} \end{cases} \tag{5-2}$$

where $x, y = -2, .., 2$ and $T_O = 24$.



(b)

Fig. 5-5. Case of lost dominant bins. (a) The dotted line is the predefined threshold, bin0 of OMM_b is not dominant. (b) After filtering, bin 0 of OMM_b is dominant.

Finally, we give an example of the relation between the object movement and the corresponding OMMs. In Fig. 5-6, the actor in the right side is moving to the left from fame 39 to frame 54 in 0.5 seconds. The value of bin (0, -1) of each OMM becomes larger than others while the actor moving from right to left.

5.1.3 Camera Motion Analysis and Feature Extraction

In many kinds of the TV programs, it seems not enough if we only extract the object motion feature, especially in those videos with lots of camera movements. Based on our observation, when a camera movement occurs, it often accompanies with a lot of motion vectors appearing in nearing directions.

frame 39(P)

frame 42(P)

0	1	1	0	1
0	3	0	3	1
0	26	⇐106	9	7
0	8	5	2	3
0	0	0	1	1

frame 45(P)

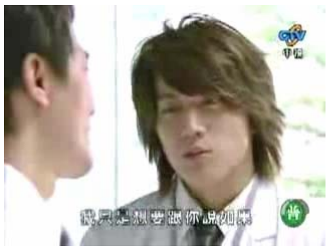
0	0	0	0	1
0	1	0	0	0
0	26	⇐106	0	1
0	8	4	1	1
0	0	0	0	0

frame 48(I)

0	0	0	0	1
1	3	3	0	0
0	26	⇐121	4	1
1	12	6	2	1
0	0	3	0	0

frame 51(P)

0	0	0	0	0
0	3	3	0	0
0	15	⇐121	4	1
0	7	6	1	0
0	0	3	0	0


frame 54(P)

0	0	0	0	0
0	1	2	0	0
0	15	⇐111	2	1
0	4	6	1	0
0	0	1	0	0

Fig. 5-6. Object movement and the corresponding OMMs.

For camera motion feature extraction, in order to represent these nearly motion vectors in a more compact direction, the 35 × 35 MH is quantized into a 7 × 7 camera motion matrix (CMM). The same prediction scheme is applied to predict the CMM of each I-frame. After that, a 3 × 1 median filter is also applied to each bin of the CMM temporally. Finally, a suitable threshold is given to fetch CDMV(s) from each CMM.

$$(x, y) \text{ is } \begin{cases} \text{a CDMV,} & \text{if } \mathrm{CMM}(x, y) \geq T_C \\ \text{not a CDMV,} & \text{otherwise} \end{cases} \tag{5-3}$$

where $x, y = -3, \ldots, 3$ and $T_C = 3T_O$. We also give an example of the relation between the camera movement and the corresponding CMMs. In Fig. 5-7, the camera is tilling up from fame 0 to frame 15 in 0.5 seconds. The value of bin (1, 0) of each CMM becomes larger than others while the camera moving up.

5.1.4 Object and Camera Motion Combination

In most cases, the camera often follows the main object to make sure that the object will be captured by camera and still be in the center of frame. The motion of object is then mostly canceled by the camera motion and becomes very small. As a result, for each MFP, when more than one dominant camera motion occurs, the CDMVs are selected; otherwise the ODMVs are selected.

frame 0(I)

frame 3(P)

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	207	0	0	0
0	0	0	36	2	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

frame 6(P)

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	207	1	0	0
0	0	0	36	2	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

frame 9(P)

0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	207	1	0	0
0	0	0	36	1	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0
0	0	0	0	0	0	0

frame 12(I)

0	0	0	0	0	0	0
0	0	0	95	0	0	0
0	0	1	207	1	0	1
0	0	1	39	1	1	5
0	0	0	0	0	1	1
0	0	0	0	0	0	0
0	0	0	0	0	0	0

frame 15(P)

0	0	0	0	0	0	0
0	0	0	95	0	0	0
0	0	1	194	1	0	1
0	0	0	39	1	1	5
0	0	0	0	0	0	1
0	0	0	0	0	0	0
0	0	0	0	0	0	0

Fig. 5-7. Camera movement and the corresponding CMMs.

5.2 Motion Feature Similarity Evaluation

In this section we will define the motion similarity measurement of two MFPs and of two shots.

Given two MFPs, $MFP_q = \{DMV_q^1, \ldots, DMV_q^W\}$ with W dominant motion vectors (DMV) and $MFP_s = \{DMV_s^1, \ldots, DMV_s^X\}$ with X DMVs. The similarity between MFP_q and MFP_s is given by the following cases:

Case 1: $W \neq 0 \; and \; X \neq 0$

$$MatchScore\left(MFP_q,\ MFP_s\right) = \begin{cases} -1, & if\ \forall(w,x), DMV_q^w \neq DMV_s^x \\ 1, & otherwise \end{cases},$$

where $w = 1, \ldots, W, and\ x = 1, \ldots, X$.

(5-4)

Case 2: $W = 0 \; or \; X = 0$

$$MatchScore\left(MFP_q,\ MFP_s\right) = \begin{cases} 0, & if\ both\ W = 0 \text{ and } X = 0 \\ -1, & otherwise \end{cases},$$

At shot level, given two MFs, $MF_i = \{MFP_i^1, \ldots, MFP_i^M\}$ with M MFPs and $MF_j = \left\{MFP_j^1, \ldots, MFP_j^N\right\}$ with N MFPs, and $M \leq N$. The similarity between MF_i and MF_j is given by (5-4)

$$ShotMotionSim\left(MF_i, MF_j\right) = \max_{n, n+M \leq N} \sum_{m=1}^{M} MatchScore\left(MFP_i^m, MFP_j^{n+m}\right). \quad (5\text{-}5)$$

CHAPTER 6

EXPERIMETAL RESULTS

Our video database consists of 59 videos (about 450 minutes) from three types of entertainment sources (sixteen sport videos, twenty-eight drama videos, and fifteen talk show videos). All videos were processed with the techniques described above to extract their luma and motion features. The length of these source videos is 13’03’’±11’01’’. Besides, there are 51 video clips, not from the source videos, as our queries. The length of these query video clips is 1’42’’±1’22’’. The similarity measurements of two videos with lumas and motions are given as follows.

At video level, given lumas of query clip and source video, V_Q and V_S, assume that there are X luma features ($L_Q^x, x = 1, \dots, X$) contained in V_Q and Y luma features ($L_S^y, y = 1, \dots, Y$) contained in V_S. Then the luma similarity between V_Q and V_S at video level is given by (6-1),

$$VideoLumaSim(V_Q, V_S) = \sum_{x=1}^{X} \max_y ShotLumaSim(L_Q^x, L_S^y). \qquad (6\text{-}1)$$

And for given motions of query clip and source video, V_Q and V_S , assume that there are M motion features $(MP_Q^m, m = 1, \dots, M)$ contained in V_Q and N motion features $(MP_S^n, n = 1, \dots, N)$ contained in V_S . Then the motion similarity between V_Q and V_S at video level is given by (6-2)

$$VideoMotionSim(V_Q\ , V_S\ ) = \sum_{m=1}^{M} \max_n ShotMotionSim(MP_Q^m, MP_S^n). \tag{6-2}$$

On the retrieval process, the relevant videos retrieved from top one to top three are returned as the results. There are fifteen sport clips, nineteen drama clips, and seventeen talk show clips as query. In TABLE 6-1, we give the comparison of not applying the luma calibration and applying the luma calibration on the luma feature extraction process. We return the top three relevant results, and better retrieval result is gotten after applying the luma calibration method. TABLE 6-2 and Fig. 6-1 shows the results from top one to top three of query by lumas with luma calibration. If we only return the most relevant result, there are nine sport videos, seventeen drama videos, and sixteen talk show videos are found. However, if we return the top three relevant results, there are three sport videos and one drama videos are not found. The lumas have a high accuracy to retrieve videos but seem not good enough in videos which have a lot of motions. This is because when motions occur, the variation between two successive frames will become large.

TABLE 6-1. COMPARISION OF NOT USING LUMA CALIBRATION (NO LC) AND USING LUMA CALIBRATION (LC)

Category	*Q#*	*NO LC*	*LC*
Sport	15	10	12
Drama	19	13	18
Talk	17	14	17
All	51	37	47

TABLE 6-2. QUERY BY LUMAS

Category	*Q#*	*Top 1*	*Top 2*	*Top 3*
Sport	15	9	11	12
Drama	19	17	18	18
Talk	17	16	17	17
All	51	42	46	47

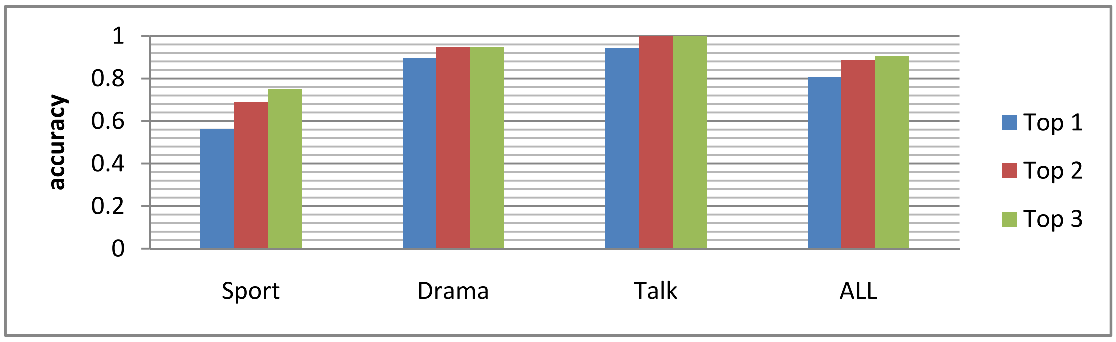


Fig. 6-1. Query by lumas.

Fortunately, the insufficient of lumas could be complemented by motions. TABLE 6-3 and Fig. 6-2 shows the results of query by motions. If we return the top three relevant results by motions, there are six sport videos, eleven drama videos, and four talk show videos are found. The retrieve accuracy seems not good as the lumas. This is because not every shot has dominant motions but every shot has the luma distribution.

TABLE 6-3. QUERY BY MOTIONS

Category	*Q#*	*Top 1*	*Top 2*	*Top 3*
Sport	15	5	6	6
Drama	19	8	9	11
Talk	17	2	3	4
All	51	15	18	21

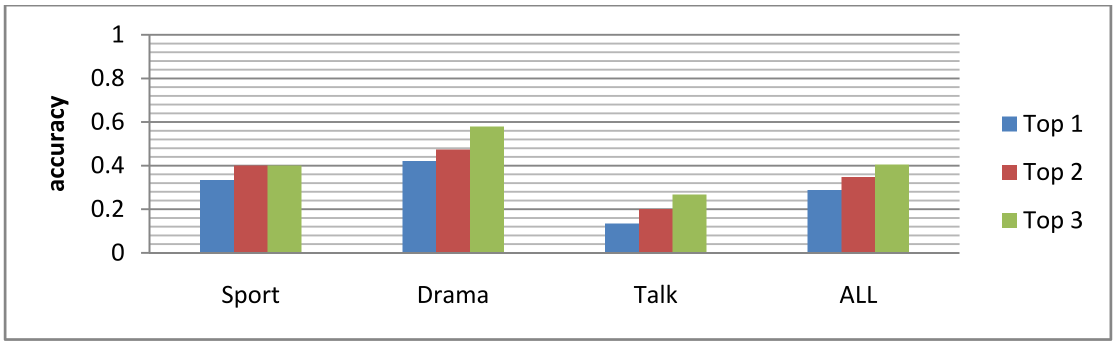


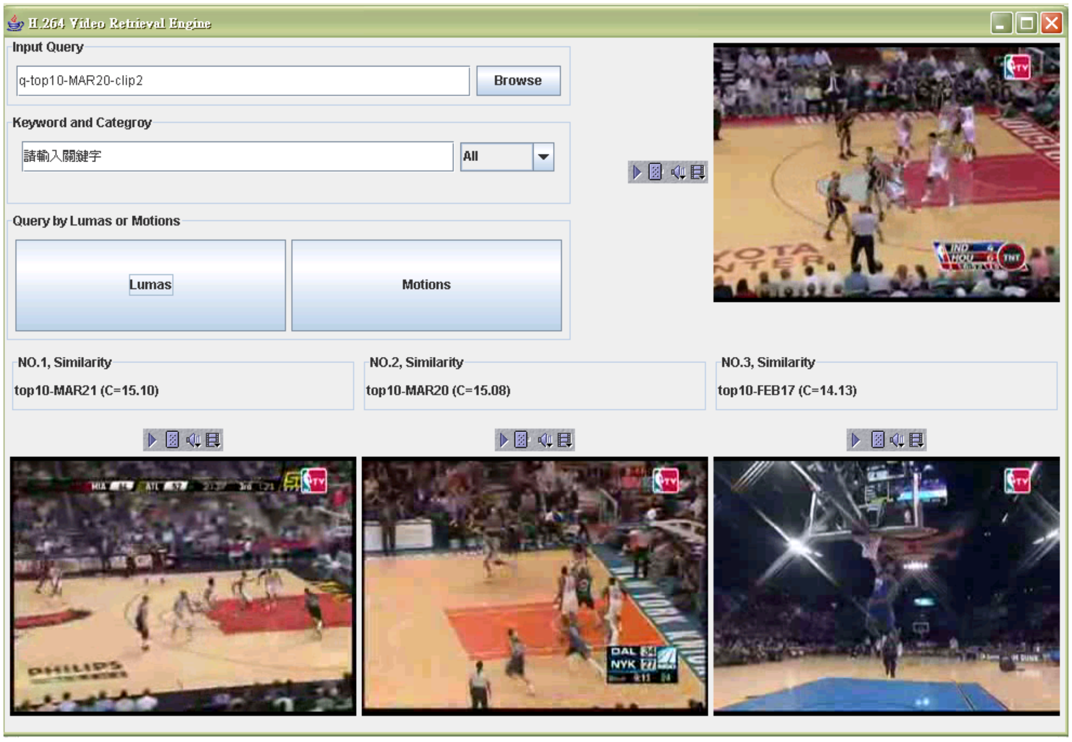
Fig. 6-2. Query by motions.

TABLE 6-4 shows the top three relevant result of query by both lumas and motions. There are finally two sport videos and one drama videos are not found. But if we return the top six relevant results, all the desired videos are found. Comprehensively, motions are not good at any kinds of videos but they can complement the insufficient of lumas.

TABLE 6-4. QUERY BY BOTH LUMAS AND MOTIONS (TOP 3)

	Sport	*Drama*	*Talk*	*All*
Q#	15	19	17	51
Found	13	18	17	48

In 450 minutes sources videos, the time of query by lumas is 12 ± 10 seconds. The longest time of query by a sport video clip is 22 seconds, a drama clip is 18 seconds, and a talk shows clip is 12 seconds. The best cases are all below 2 seconds. This shows that the proposed shot clustering method performs a good retrieval time reducing. And for motions, the retrieval time is 12 ± 11 seconds. An example of query by a sport video clip is shown in Fig. 6-3.



q-top10-MAR20-clip2

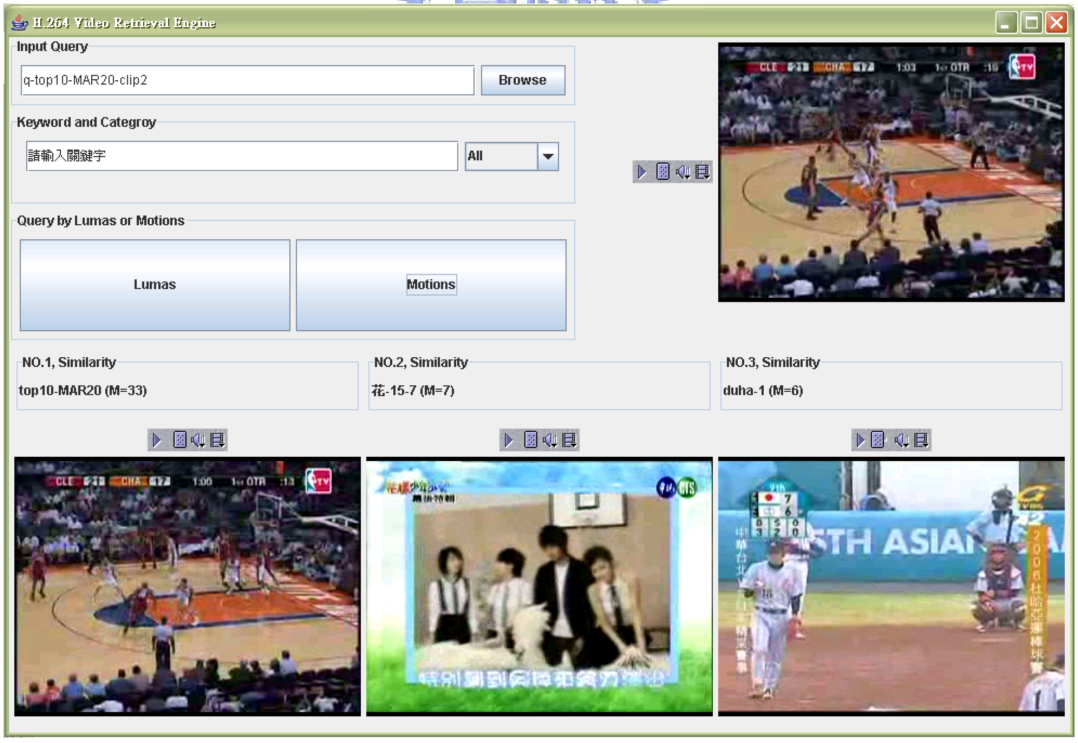
請輸入關鍵字

NO.1, Similarity
top10-MAR21 (C=15.10)

NO.2, Similarity
top10-MAR20 (C=15.08)

NO.3, Similarity
top10-FEB17 (C=14.13)

(a)



q-top10-MAR20-clip2

請輸入關鍵字

NO.1, Similarity
top10-MAR20 (M=33)

NO.2, Similarity
花-15-7 (M=7)

NO.3, Similarity
duha-1 (M=6)

(b)

Fig. 6-3. Query by a sport video clip. (a) By lumas. (b) By motions.

CHAPTER 7

CONCLUTION AND FUTURE WORK

In this thesis, we provide a video retrieval system based on the H.264 videos and no feedback requirement. For the video spatial domain information, luma feature is extracted. Luma calibration is used to fix the problem of color distribution differently and an efficient shot clustering is provided to reduce the video matching time. For the video temporal domain information, a statistic-based motion feature extraction scheme is proposed, which contains both dominant object motions and camera motions but with no object extraction and camera parameter estimation.

The system is efficient and with low cost in video analysis and retrieval. The retrieval process will be more powerful if we add some keywords like video name, and select the category which the video belongs to such as “drama” or “sport”. The above two information is carried in the transport stream. Therefore, when users watch a video trailer from the broadcasting channel and have the desire to find the complete video from the network video database. The combinational information of the trailer will be sent to the video retrieval engine. And the video retrieval engine will return the information of the complete video to users through the network communication channel (see Fig. 7-1).

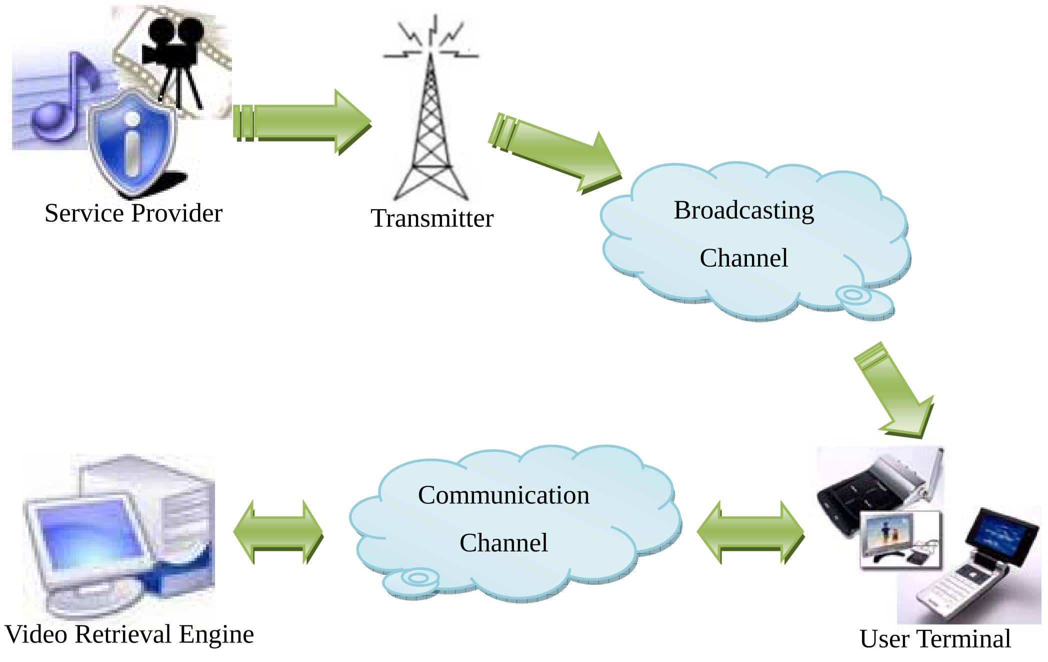


Fig. 7-1. Model of interactive video retrieval service.

Sometimes, videos are added some special lacing or subtitles. The future work we can do is to detect these special effects, remove them to get the original content, and do the analysis and retrieval. Besides, in this thesis, we have analyzed useful and significant features from compressed video stream. There is another interesting issue about how to combine features from other media sources such as audio for higher semantic level video retrieval.

REFERENCES

[1] Y. Rui, T. S. Huang, and S. Mehrotra, " Browsing and retrieving video content in a unified framework," *Multimedia Signal Processing*, pp. 9-14. Dec. 1998.

[2] X. Zhu, A. K. Elmagarmid, X. Xue, L. Wu, and A. Catlin, "InsightVide: Towards Hierarchical Video Content Organization for Efficient Browsing, Summarization, and Retrieval," *IEEE Transactions on Multimedia*, Vol. 7, No.4, pp. 648-666, Aug. 2005.

[3] C. W. Su, H. Y. M. Liao, and K. C. Fan, "A Motion-Flow-Based Fast Video Retrieval System," *IEEE Transactions on Multimedia*, Vol. 7, No. 6, pp. 1106-1113, Dec. 2005.

[4] J.W. Hsieh, S.L. Yu, and Y. S. Chen, "Motion-based video retrieval by trajectory matching," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16, No. 3, pp. 396-408, Mar. 2003.

[5] Y. H. Ho, C. W. Lin, J. F. Chen, and H. Y. M. Liao, "Fast Coarse-to-Fine Video Retrieval Using Shot-Level Spatial-Temporal Statistics," *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 16, No. 5, pp. 642-648, May 2006.

[6] Z. Xiong, X. S. Zhou, Q. Tian, Y. Rui, and T. S. Huang,” Semantic Retrieval of Video: Review of research on video retrieval in meetings, movies and broadcast news, and sports,” *Signal Processing Magazine*, Vol. 23, Issue 2, pp. 18-27, Mar. 2006.

[7] C. Cotsaces, N. Nikolaidis, and L. Pitas, ”Video Shot Detection and Condensed Representation: A Review,” *Signal Processing Magazine*, Vol. 23, Issue 2, pp. 28-37, Mar. 2006.

[8] J. Fan, D. K. Y . Yau, W. G. Aref, and A. Rezqui, “Adaptive motion-compensated video coding scheme towards content-based bit rate allocation,” *Journal of Electronic Imaging,* Vol. 9, No. 4, pp. 521-533, Oct. 2000.

[9] J. Bescos, “Real Time Shot Change Detection Over Online MPEG-2 Video,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 1, No. 4, pp. 475-484, Apr. 2004.

[10] S. C. Pei and Y. Z. Chou, ” Efficient MPEG compressed video analysis using macroblock type information,” *IEEE Transactions on Multimedia*, Vol. 1, No. 4, pp. 321-333, Dec. 1999.

[11] T. Liu, H. J. Zhang, and F. Qi, “A Novel Video Key-Frame-Extraction Algorithm Based on Perceived Motion Energy Model,” *IEEE Transactions on Circuits and Systems for Video Technology*, Vol. 13, No. 10, pp. 1006-1013, Oct. 2003.

[12] W. Wolf, ”Key Frame Selection by Motion Analysis,” in *Proc. IEEE ICASSP* , Vol.2, pp. 1228-1231, May 1996.

[13] I. Richardson, “H.264 and MPEG-4 video compression,” Wiley, 2003.

[14] JVT reference software, JM 11, downloaded from http://bs.hhi.de/.

[15] http://www.youtube.com/

[16] http://www.im.tv/vlog/

[17] http://www.tudou.com/