Technical University of Crete
Department of Electronics Engineering & Computer Engineering

"Segmentation of compressed video using partially uncompressed information"

Doulaverakis Haralampos
December 2003

Accepted on the recommendation of

Prof. Dr. Michalis Zervakis,	Thesis Advisor
Prof. Dr. Nikos Sidiropoulos,	Co-Examiner
Assoc. Prof. Dr. Euripides Petrakis,	Co-Examiner

Table of contents

1. Introduction

The advances in multimedia compression technology and the significant increase in computer performance together with the growth of Internet, have led to wide use and availability of digital video. Applications and needs such as digital libraries, video on demand, interactive TV generate and use a large amount of video data. This fact coupled with the vast information that video data encapsulate has led to the development of tools that can efficiently index, search, browse and retrieve relevant material.

Temporal video segmentation is the first step to automatic annotation and indexing of video sequences, as shown in Figure 1.1. It is intended to divide the sequence into a set of meaningful segments, known as shots, which are the basic elements for the process of indexing. Each shot is then represented by selecting keyframes, a frame that best describes the content of a shot, and indexed using temporal and special features extracted by these frames.

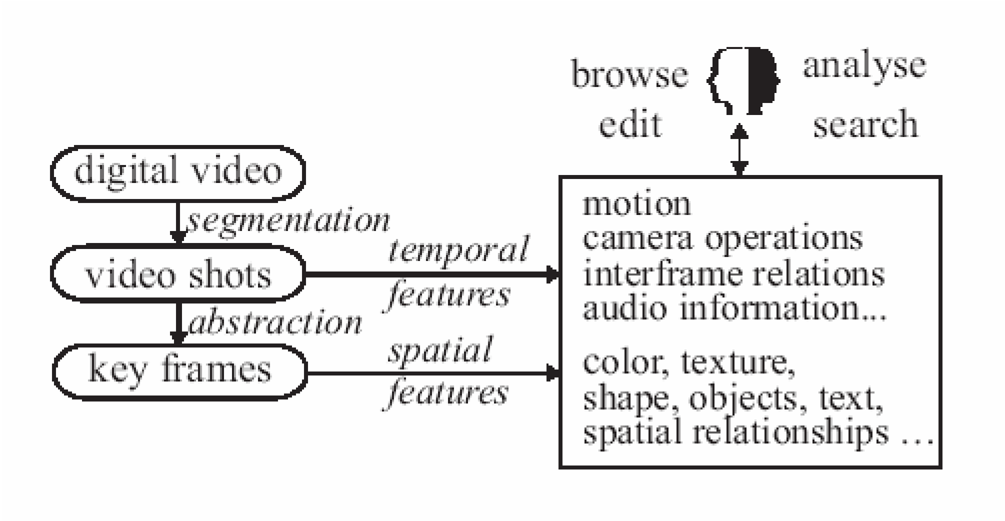


Figure 1.1: The steps of content-based retrieval

A *shot* is defined as an unbroken sequence of frames taken form one camera. When this sequence is broken a *shot transition* is declared. There are two basic types of shot transition, abrupt and gradual. *Abrupt changes* are also known as *cuts* and they occur in single frame and they occur when the camera stops and restarts.

Gradual transitions are mainly the result of video editing process also known as cinematic effects. There are 3 basic types of gradual transitions, dissolves, fades (in and out) and wipes. Dissolves occur when a shot is superimposed to another and the first gets dimmer while the second gets brighter. Fade in is a slow increase in brightness starting from a black frame and fade out is the opposite, a slow decrease of brightness resulting in a black frame. A wipe is the result of a line passing though the frame and is as if the new shot is placed over the old one.

Gradual transitions are generally more difficult to detect than abrupt ones mainly because the interchange between shots lasts more than one frames and the frame to frame differences are small. This is does not apply in cuts as the change is done in one frame and the content in most cases changes dramatically.

Camera movement is also important to be detected. Keyframes selected for these type of shots, panning and zooming, must be selected using another philosophy. In a pan sequence the content of the frames changes substantially through time and more than one keyframe need to be selected. In a zooming shot the frames at the beginning or end of the zoom can be selected as representative for the entire shot.

1.1 Previous work

Much work and research is done for scene change detection directly on the MPEG compressed domain. Arman et al [10] used the DCT coefficients of the I frames to construct a vector for each frame. He then used the inner product of these vectors for each pair of consecutive frames. If the value of the inner product exceeded a given threshold then a cut was declared. However this method provided small temporal resolution (it could not indicate the exact frame the cut took place) and it cannot detect gradual transitions.

Yeo [7] used DC-images derived from the frames of the MPEG stream and calculated frame to frame pixel differences. He then used a sliding window method to determine cuts and gradual transitions were detected by making use of the special form that these transitions had in the difference plot.

Other work [11] includes taking into account the total number of intracoded, backward and forward predicted MBs of a frame. A cut on a P frames will be declared if the number of intracoded MBs is large or in a B frame the number of backward predicted MBs is large. Special conditions apply for a cut in a I frame.

In [12] the MB coding mode and bitrate information were used to determine cuts only.

Zhang, Low and Smoliar [14] used a pair-wise DCT coefficients comparison of I frames to determine possible transitions. They then used the number of motion vectors in P and B frames to determine the presence of a cut. A small number of motion vectors indicate a cut. Gradual transitions were found by using a two pass strategy on I frames, which also provided low temporal resolution.

2. MPEG-1 Video Coding Standard

2.1 Background and structure of MPEG-1 standards activities

The development of digital video technology in the 1980s has made it possible to use digital video compression invarious kinds of applications. The effort to develop standards for coded representation of moving pictures, audio, and their combination is carried out in the Moving Picture Experts Group (MPEG). MPEG is a group formed under the auspices of the International Organization for Standardization (ISO) and the International Electrotechnical Commission (IEC). It operates in the framework of the Joint ISO/IEC Technical Committee 1 (JTC 1) on Information Technology, which was formally Working Group 11 (WG11) of Sub-Committee 29 (SC29). The premise is to set the standard for coding moving pictures and the associated audio for digital storage media at about 1.5 Mbit/s so that a movie can be compressed and stored in a CD-ROM (Compact Disc – Read Only Memory). The resultant standard is the international standard for moving picture compression, ISO/IEC 11172 or MPEG-1 (Moving Picture Experts Group - Phase 1). MPEG-1 standards consist of 5 parts, including: Systems (11172-1), Video (11172-2), Audio (11172-3), Conformance Testing (11172-4), and Software Simulation (11172-5). We will focus only on the video part.

The activity of the MPEG committee started in 1988 based on the work of ISO JPEG (Joint Photographic Experts Group) and CCITT Recommendation H.261: "Video Codec for Audiovisual Services at px64 kbits/s". Thus, the MPEG-1 standard has much in common with the JPEG and H.261 standards. The MPEG development methodology was similar to that of H.261 and was divided into three phases: Requirements, Competition, and Convergence. The purpose of the Requirements phase is to precisely set the focus of the effort and determine the rule for the competition phase. The document of this phase is a "Proposal Package Description" and a test methodology. The next step is the competition phase in which the goal is to obtain state of the art technology from the best of academic and industrial research. The criteria are based on the technical merits and the trade-off between video quality and the cost of implementation of the ideas and the subjective test. After the competition phase, various ideas and techniques are integrated into one solution in the convergence phase. The solution results in a document called the simulation model. The simulation model implements, in some sort of programming language, the operation of a reference encoder and a decoder. The simulation model is used to carry out simulations to optimize the performance of the coding scheme. A series of fully documented experiments called core experiments are then carried out. The MPEG committee reached the Committee Draft (CD) status in September 1990 and the Committee Draft (CD 11172) was approved in December 1991. International Standard (IS) 11172 for the first three parts was established in November 1992. The IS for the last two parts was finalized in November 1994.

2.2 MPEG-1 target applications and requirements

The MPEG standard is a generic standard, which means that it is not limited to a particular application. A variety of digital storage media applications of MPEG-1 have been proposed based on the assumptions that the acceptable video and audio quality can be obtained for a total bandwidth of about 1.5 Mbits/s. Typical storage media for these applications include CD-ROM, DAT (Digital Audio Tape), Winchester-type computer disks, and writable optical disks. The target applications are asymmetric applications where the compression process is performed once and the decompression process is required often. Examples of the asymmetric applications include video CD, video on demand, and video games. In these asymmetric applications, the encoding delay is not a concern. The encoders are needed only in small quantities while the decoders are needed in large volumes. Thus, the encoder complexity is not a concern while the decoder complexity needs to be low in order to result in low-cost decoders.

The requirements for compressed video in digital storage media mandate several important features of the MPEG-1 compression algorithm. The important features include normal playback, frame-based random access and editing of video, reverse playback, fast forward / reverse play, encoding high-resolution still frames, robustness to uncorrectable errors, etc. The applications also require MPEG-1 to support flexible picture sizes and frame rates. Another requirement is that the encoding process can be performed in reasonable speed using existing hardware technologies and the decoder can be implemented using small number of chips in low cost.

Two of the main characteristics of the MPEG-1 compression algorithm are bi-directional motion compensated prediction and motion compensated prediction with half-pixel accuracy. They are explained in the sections to follow.

2.3 Bi-directional motion compensated prediction

MPEG-1 allows the previous video frame is used as the reference frame for the motion compensated prediction (forward prediction). But it also allows the future frame to be used as the reference frame for the motion compensated prediction (backward prediction), which can provide better prediction. For example, as shown in Figure 2.1, if there are moving objects, and if only the forward prediction is used, there will be uncovered areas (such as the block behind the car in Frame N) for which we may not be able to find a good matching block from the previous reference picture (Frame N-1). On the other hand, the backward prediction can properly predict these uncovered areas since they are available in the future reference picture, i.e. frame N+1 in this example. Also shown in the figure, if there are objects moving into the picture (the airplane in the figure), these new objects cannot be predicted from the previous picture, but can be predicted from the future picture.

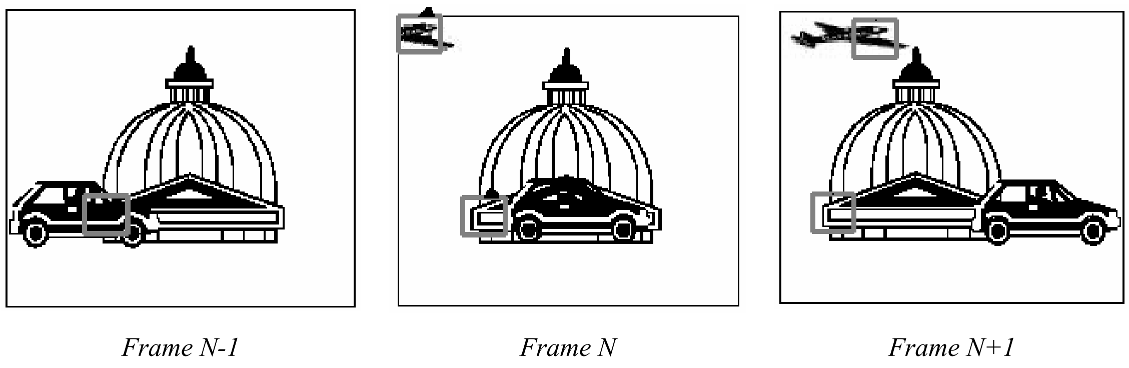


Figure 2.1: A video sequence showing the benefits of bi-directional prediction

2.4 Motion compensated prediction with half-pixel accuracy

A moving object often moves to a position which is not on the pixel-grid but between the pixels. MPEG-1 allows half-pixel-accuracy motion vectors. By estimating the displacement at a finer resolution, we can expect improved prediction and, thus, better performance than motion estimation with integer-pixel accuracy. As shown in Figure 2.2, since there is no pixel-value at the halfpixel locations, interpolation is required to produce the pixel-values at the half-pixel positions. Bi-linear interpolation is used in MPEG-1 for its simplicity. The motion estimation is performed only on luminance blocks. The resulting motion vector is scaled by 2 and applied to the chrominance blocks. This reduces the computation but may not necessarily be optimal. Motion vectors are differentially encoded with respect to the motion vector in the preceding adjacent Macroblock. The reason is that the motion vectors of adjacent regions are highly correlated, as it is quite common to have relatively uniform motion over areas of picture.

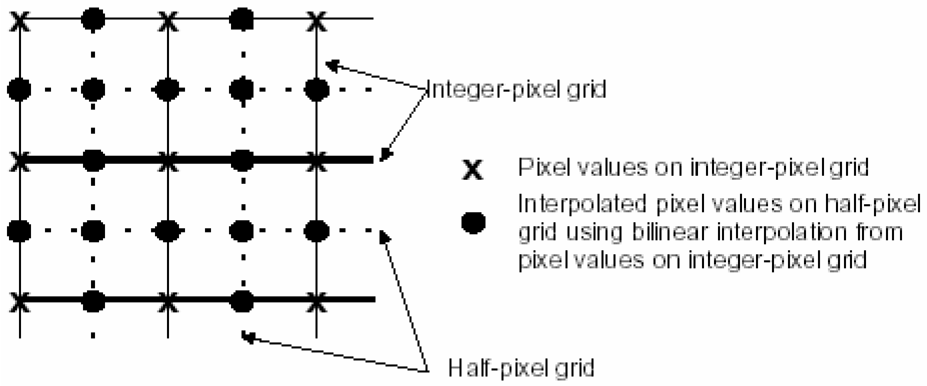

Integer-pixel grid
X Pixel values on integer-pixel grid
● Interpolated pixel values on half-pixel grid using bilinear interpolation from pixel values on integer-pixel grid
Half-pixel grid

Figure 2.2: Half-pixel motion estimation

2.5 MPEG-1 video structure

2.5.1 Source Input Format (SIF)

The typical MPEG-1 input format is the Source Input Format (SIF). SIF was derived from CCIR601, a worldwide standard for digital TV studio. CCIR601 specifies the Y Cb Cr color coordinate where Y is the luminance component (black and white information), and Cb and Cr are two color difference signals (chrominance components). A luminance sampling frequency of 13.5 MHz was adopted. There are several Y Cb Cr sampling formats, such as 4:4:4, 4:2:2, 4:1:1, and 4:2:0. In 4:4:4, the sampling rates for Y, Cb, and Cr are the same. In 4:2:2, the sampling rates of Cb and Cr are half of that of Y. In 4:1:1 and 4:2:0, the sampling rates of Cb and Cr are one quarter of that of Y. The positions of Y Cb Cr samples for 4:4:4, 4:2:2, 4:1:1, and 4:2:0 are shown in Figure 2.3.

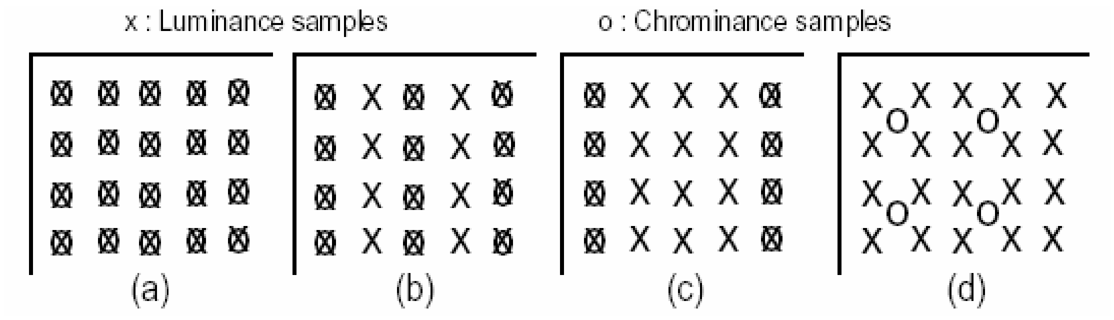


Figure 2.3: Luminance and chrominance samples in (a) 4:4:4 format (b) 4:2:2 format (c) 4:1:1 format (d) 4:2:0 format.

Converting analog TV signal to digital video with the 13.5 MHz sampling rate of CCIR601 results in 720 active pixels per line (576 active lines for PAL and 480 active lines for NTSC). This results in a 720x480 resolution for NTSC and a 720x576 resolution for PAL. With 4:2:2, the uncompressed bit-rate for transmitting CCIR601 at 30 frames/s is then about 166 Mbits/s. Since it is difficult to compress a CCIR601 video to 1.5 Mb/s with good video quality, in MPEG-1, typically the source video resolution is decimated to a quarter of the CCIR601 resolution by filtering and sub-sampling. The resultant format is called Source Input Format (SIF) which has a 360x240 resolution for NTSC and a 360x288 resolution for PAL. Since in the video coding algorithm, the block-size of 16x16 is used for motion compensated prediction, the number of pixels in both the horizontal and the vertical dimensions should be multiples of 16. Thus, the four left-most and right-most pixels are discarded to give a 352x240 resolution for NTSC systems (30 frames/s) and a 352x288 resolution for PAL systems (25 frames/s). The chrominance signals have half of the above resolutions in both the horizontal and vertical dimensions (4:2:0, 176x120 for NTSC and 176x144 for PAL). The uncompressed bit-rate for SIF (NTSC) at 30 frames/s is about 30.4 Mbits/s.

2.5.2 Group Of Pictures (GOP) and I-B-P Pictures

In MPEG, each video sequence is divided into one or more groups of pictures (GOPs). There are three types of pictures defined in MPEG-1: I-, P-, and B-pictures. They are shown in Figure 2.4. Each GOP is composed of one or more pictures; one of these pictures must be an I-picture. Usually, the spacing between two anchor frames (I- or P-pictures) is referred to as M, and the spacing between two successive I-pictures is referred to as N. In Figure 4, M=3 and N=9.

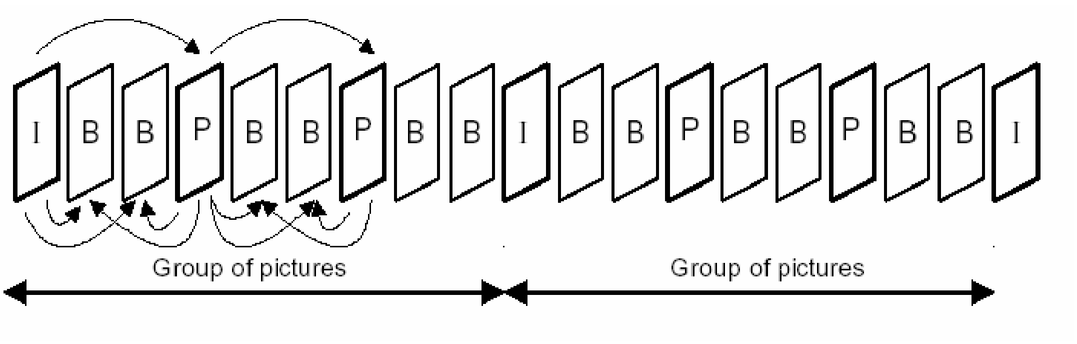


Figure 2.4: MPEG Group Of Pictures

I-pictures (Intra-coded pictures) are coded independently with no reference to other pictures. I-pictures provide random access points in the compressed video data, since the I-pictures can be decoded independently without referencing to other pictures. With I-pictures, an MPEG bit-stream is more editable. Also, error propagation due to transmission errors in previous pictures will be terminated by an I-picture since the I-picture does not have a reference to the previous pictures. Since I-pictures use only transform coding without motion compensated predictive coding, it provides only moderate compression.

P-pictures (Predictive-coded pictures) are coded using the forward motion-compensated prediction from the preceding I- or P-picture. P-pictures provide more compression than the I-pictures by virtue of motion compensated prediction. They also serve as references for B-pictures and future P-pictures. Transmission errors in the I-pictures and P-pictures can propagate to the succeeding pictures since the I-pictures and P-pictures are used to predict the succeeding pictures.

B-pictures (Bi-directional-coded pictures) allow Macroblocks to be coded using bi-directional motion compensated prediction from both the past and future reference I- or P-pictures. In the B-pictures, each bi-directional motion compensated Macroblock can have two motion vectors: a forward motion vector which references to a best matching block in the previous I- or P-pictures, and a backward motion vector which references to a best matching block in the next I- or P-pictures as shown in Figure 2.5. The motion compensated prediction can be formed by the average of the two referenced motion compensated blocks. By averaging between the past and the future reference blocks, the effect of noise can be decreased. B-pictures provide the best compression compared to I- and P-pictures. I- and P-pictures are used as reference pictures for predicting B-pictures. To keep the structure simple and since there is no apparent advantage to use B-pictures for predicting other B-pictures, the B-pictures are not used as reference pictures Hence, B-pictures do not propagate errors.

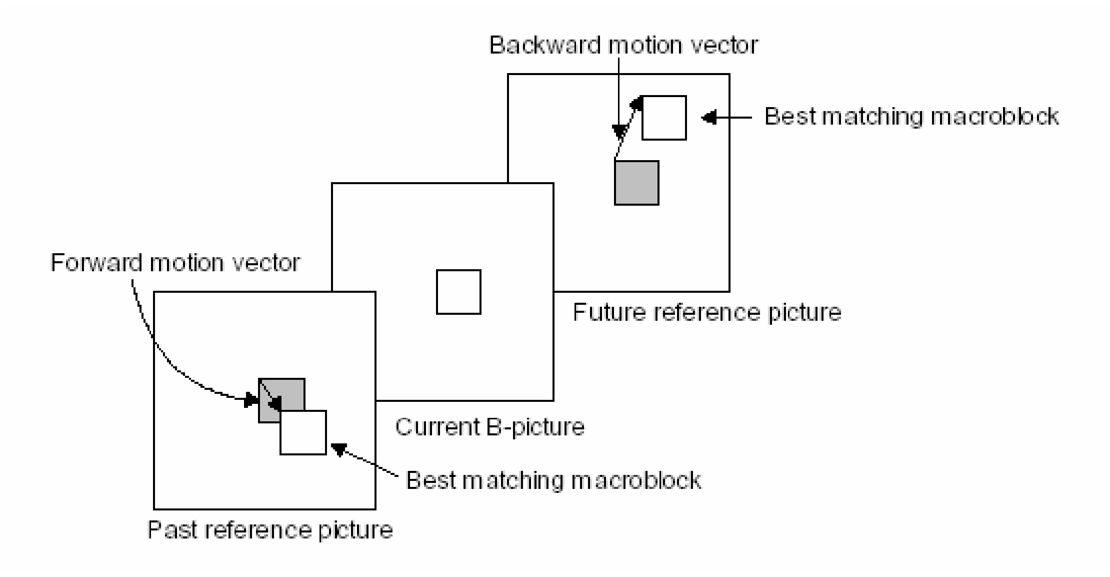


Figure 2.5: Bi-directional motion estimation

The trade-off of having frequent B-pictures is that it decreases the correlation between the previous I- or P-picture and the next reference P- or I-picture. It also causes coding delay and increases the encoder complexity. With the example shown in Figure 2.4 and Figure 2.6, at the encoder, if the order of the incoming pictures is 1, 2, 3, 4, 5, 6, 7, … , the order of coding the pictures at the encoder will be: 1, 4, 2, 3, 7, 5, 6, … . At the decoder, the order of the decoded pictures will be 1, 4, 2, 3, 7, 5, 6, … . However, the display order after the decoder should be 1, 2, 3, 4, 5, 6, 7. Thus, frame memories have to be used to put the pictures in the correct order. This picture re-ordering causes delay. The computation of bi-directional motion vectors and the picture-re-ordering frame-memories increase the encoder complexity.

In Figure 2.6, two types of GOPs are shown. GOP1 can be decoded without referencing other GOPs. It is called a Closed-GOP. In GOP2, to decode the 8th B- and 9th B-pictures, the 7th P-picture in GOP1 is needed. GOP2 is called an Open GOP which means the decoding of this GOP needs to reference other GOPs.

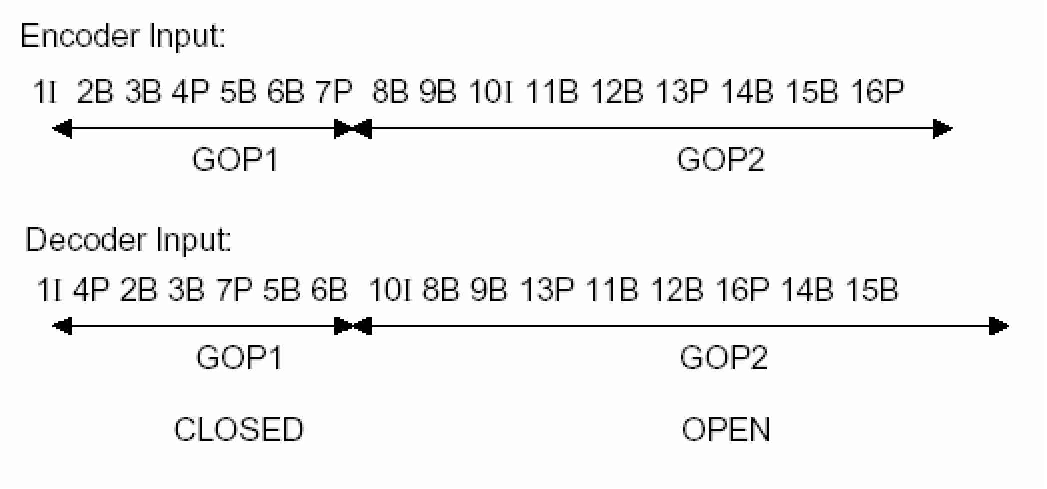

Encoder Input:
1I 2B 3B 4P 5B 6B 7P 8B 9B 10I 11B 12B 13P 14B 15B 16P
GOP1
GOP2
Decoder Input:
1I 4P 2B 3B 7P 5B 6B 10I 8B 9B 13P 11B 12B 16P 14B 15B
GOP1
GOP2
CLOSED
OPEN

Figure 2.6: Frame reordering

2.5.3 Slice, Macroblock and Block structures

An MPEG picture consists of slices. A slice consists of a contiguous sequence of Macroblocks in a raster scan order (from left to right and from top to bottom). In an MPEG coded bit-stream, each slice starts with a slice-header which is a clear-codeword (a clear-codeword is a unique bit-pattern which can be identified without decoding the variable-length codes in the bit-stream). Due to the clear-codeword slice-header, slices are the lowest level of units which can be accessed in an MPEG coded bit-stream without decoding the variable-length codes. Slices are important in the handling of channel errors. If a bit-stream contains a bit-error, the error may cause error propagation due to the variable-length coding. The decoder can regain synchronization at the start of the next slice. Having more slices in a bit-stream allows better error-termination, but the overhead will increase.

A Macroblock consists of a 16x16 block of luminance samples and two 8x8 block of corresponding chrominance samples as shown in Figure 2.7. A Macroblock thus consists of four 8x8 Y-blocks, one 8x8 Cb block, and one 8x8 Cr block. Each coded Macroblock contains motion-compensated prediction information (coded motion vectors and the prediction errors). There are four types of Macroblocks: intra, forward-predicted, backward-predicted, and bi-directionally-predicted Macroblocks. The motion information consists of one motion vector for forward- and backward-predicted Macroblocks and two motion vectors for bi-directionally-predicted Macroblocks. P-pictures can have intra- and forward-predicted Macroblocks. B-pictures can have all four types of Macroblocks. The first and last Macroblocks in a slice must always be coded. A Macroblock is designated as a skipped Macroblock when its motion vector is zero and all the quantized DCT coefficients are zero. Skipped Macroblocks are not allowed in I-pictures. Non-intra coded Macroblocks in P- and B-pictures can be skipped. For a skipped Macroblock, the decoder just copies the Macroblock from the previous picture.

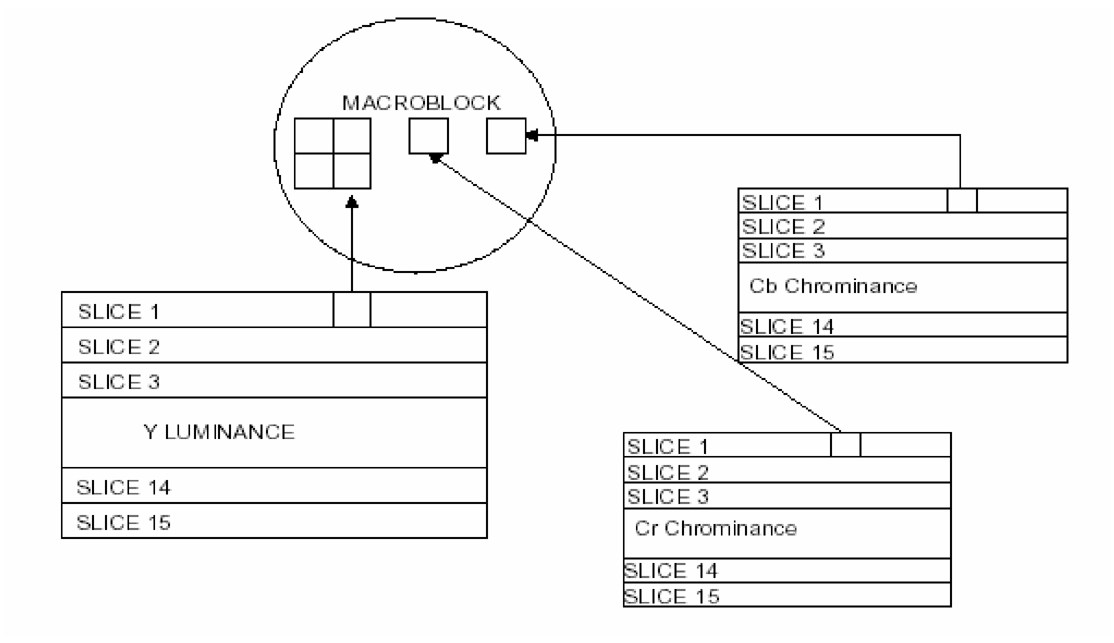
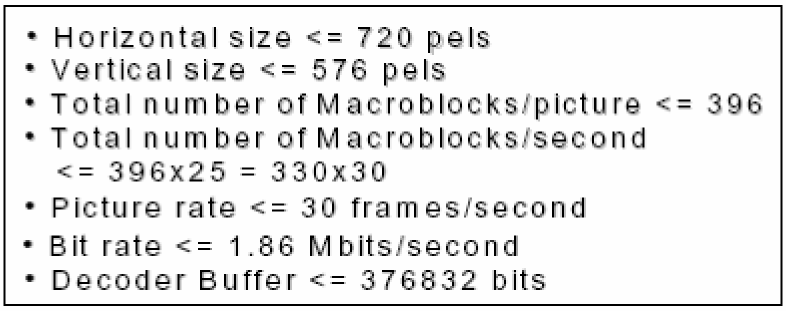


Figure 2.7: Macroblock and slice structures

In order to allow cost effective implementation of user terminals, MPEG-1 defines a Constrained Parameter Set which lays down specific constraints, as listed in Table 1.



- Horizontal size <= 720 pels
- Vertical size <= 576 pels
- Total number of Macroblocks/picture <= 396
- Total number of Macroblocks/second <= 396x25 = 330x30
- Picture rate <= 30 frames/second
- Bit rate <= 1.86 Mbits/second
- Decoder Buffer <= 376832 bits

Table 1: MPEG-1 Constrained Parameter Set

2.6 Simulation Model

MPEG-1 specifies only the syntax and the decoder. Many detailed coding options such as the rate control strategy, the quantization decision levels, the motion estimation schemes, and coding modes for each Macroblock are not specified. This allows future technology improvement and product differentiation. In order to have a reference MPEG-1 video quality, Simulation Models were developed in MPEG-1. A simulation model contains a specific reference implementation of the MPEG-1 encoder and decoder including all the details which are not specified in the standard. The final version of the MPEG-1 simulation model is “Simulation Model 3” (SM3). In SM3, the motion estimation technique uses one forward and/or one backward motion vector per Macroblock with half-pixel accuracy. A two-step search scheme which consists of a full-search in the range of +/- 7 pixels with the integer-pixel precision, followed by a search in 8 neighboring half-pixel positions, is used. The decision of the coding mode for each Macroblock (whether or not it will use motion compensated prediction and intra/inter coding), the quantizer decision levels, and the rate-control algorithm are all specified.

2.7 MPEG-1 video bit-stream structures

As shown in figure 8, there are 6 layers in the MPEG-1 video bit-stream: the video sequence, group of pictures, picture, slice, Macroblock, and block layers.

- A video sequence layer consists of a sequence header, one or more groups of pictures, and an end-of-sequence code. It contains the setting of the following parameters: the picture size (horizontal and vertical sizes), pel aspect ratio, picture rate, bit-rate, the minimum decoder buffer size (video buffer verifier size), constraint parameters flag (this flag is set only when the picture size, picture rate, decoder buffer size, bit rate, and motion parameters satisfy the constraints bound in Table 1), the control for the loading of 64 eight-bit values for intra and non-intra quantization tables, and the user data.

- The GOP layer consists of a set of pictures that are in a continuous display order. It contains the setting of the following parameters: the time code which gives the hours-minutes-seconds time interval from the start of the sequence, the closed GOP flag which indicates whether the decoding operation needs pictures from the previous GOP for motion compensation, the broken link flag which indicated whether the previous GOP can be used to decode the current GOP, and the user data.

- The picture layer acts as a primary coding unit. It contains the setting of the following parameters: the temporal reference which is the picture number in the sequence and is used to determine the display order, the picture types (I/P/B), the decoder buffer initial occupancy which gives the number of bits that must be in the compressed video buffer before the idealized decoder model defined by MPEG decodes the picture (it is used to prevent the decoder buffer overflow and underflow), the forward motion vector resolution and range for P- and B-pictures, the backward motion vector resolution and range for B-pictures, and the user data.

- The slice layer acts as a resynchronization unit. It contains the slice vertical position where the slice starts, and the quantizer scale that is used in the coding of the current slice.

- The macroblock layer acts as a motion compensation unit. It contains the setting of the following parameters: the optional stuffing bits, the macroblock address increment, the macroblock type, quantizer scale, motion vector, and the Coded Block Pattern which defines the coding patterns of the 6 blocks in the macroblock.

- The block layer is the lowest layer of the video sequence and consists of coded 8x8 DCT coefficients. When a macroblock is encoded in the Intra-mode, the DC-coefficient is encoded similar to that in JPEG (the DC coefficient of the current macroblock is predicted from the DC coefficient of the previous macroblock). At the beginning of each slice, predictions for DC coefficients for luminance and chrominance blocks are reset to 1024. The differential DC values are categorized according to their absolute values and the category information is encoded using VLC (Variable-Length Code). The category information indicates the number of additional bits following the VLC to represent the prediction residual. The AC-coefficients are encoded using a VLC to represent the zero-run-length and the value of the non-zero coefficient.

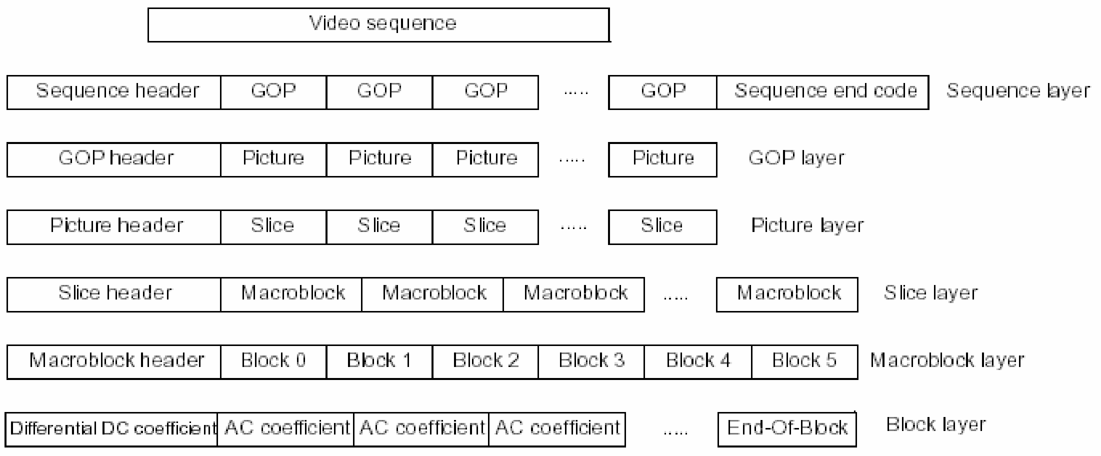


Figure 2.8: MPEG-1 bit-stream syntax layers

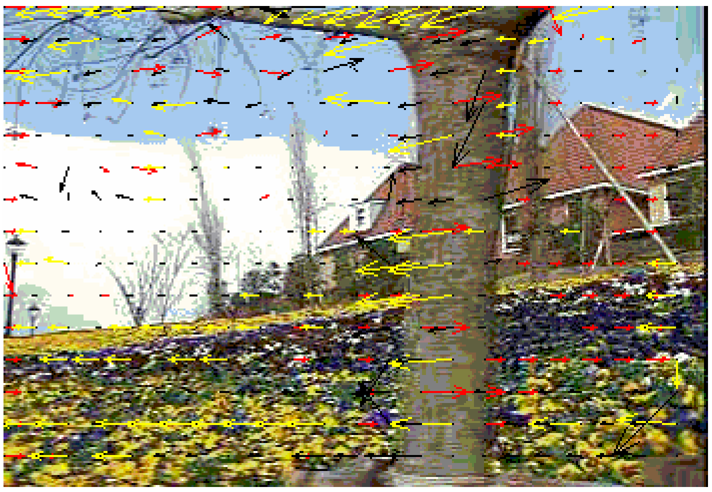
2.7 Summary

MPEG-1 is mainly for storage media applications. Due to the use of B-picture, it may result in long end-to-end delay. The MPEG-1 encoder is much more expensive than the decoder due to the large search range, the half-pixel accuracy in motion estimation, and the use of the bi-directional motion estimation. The MPEG-1 syntax can support a variety of frame-rates and formats for various storage media applications. Similar to other video coding standards, MPEG-1 does not specify every coding option (motion estimation, rate-control, coding modes, quantization, pre-processing, post-processing, etc.). This allows continuing technology improvement and product differentiation.

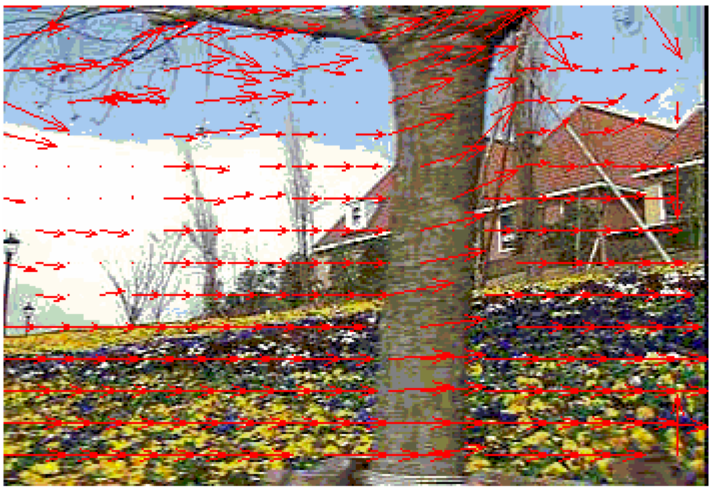
3. Motion vector normalization

A MB can have zero, one or two motion vectors depending on its frame type and whether the MB is intracoded, forward- or backward predicted, or bidirectionally predicted, respectively. Moreover, these motion vectors can be forward predicted or backward predicted with respect to reference frames which may or may not occur adjacent to the frame that the MB belongs to. A problem occurs if an I frame with no motion vectors to a B frame with primarily bidirectionally predicted MBs, or even two B frames, one of which is primarily forward predicted and the other primarily backward predicted, need to be compared. Therefore a more uniform set of motion vectors, independent of frame type and direction of prediction is required. The three types of frames (**I**, P, B) with their corresponding motion vectors are illustrated in Figure 3.1.

(a)



(b)



(c)

Figure 3.1: (a) I frame, no motion vectors are present, (b) B frame, yellow color: backward predicted, red: forward predicted, black: bidirectionally predicted, (c) P frame, only forward predicted motion vectors

This approach involves representing each motion vector as a backward predicted vector with respect to the next frame, independently of frame type. The set of motion vectors (or flow) for each frame then represents the direction of motion for each MB with respect to the next frame. It should be noted that not all MBs will have this flow vector associated with them. But the number of such MBs is rarely large enough to affect our analysis. Across cuts or breaks, most of the MBs are not expected to have flow information. The first step in deriving the flow is to analyze the frame-type pattern (i.e. the pattern of I, P and B frames) in the MPEG stream [8].

3.1 For clips containing only I and P frames

If there are only P frames between I frames then flow can be derived for each of the frames between two consecutive I frames (a GOP), including the I frames themselves, except for the last P frame for which we have no information about its relationship with the I frame that follows it. The flow for an I frame that is followed by a P frame is the set of forward predicted motion vectors of the P frame after inversion. Intuitively if a MB in the P frame is displaced by a motion vector *(x,y)* with respect to a MB in the I frame then it is logical to resume that the latter MB is displaced by a motion vector *(-x,-y)* with respect to the MB in the P frame (Figure 3.2). The same reasoning is applied to the flow estimation of the MBs of a P frame that is followed by another P frame (Figure 3.3). The MPEG stream however does not contain any information relating a P frame to the I frame that follows it, unless B frames are present between them.

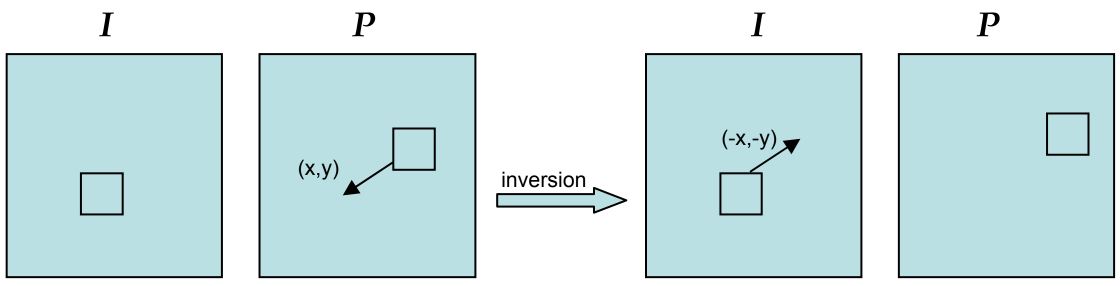


Figure 3.2: Flow between an I frame and its proceeding P frame

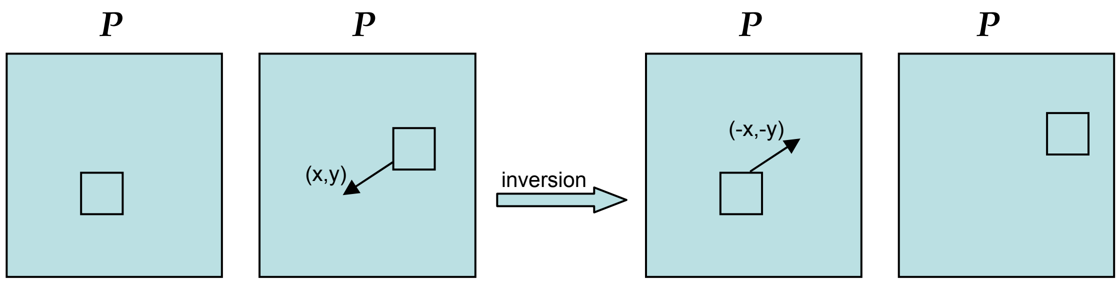


Figure 3.3: Flow between a P frame and its proceeding P frame

3.2 For clips containing B frames

Most MPEG streams contain B frames between consecutive reference frames. Let two consecutive reference frames be R_i and R_j . Let the B frames between them be denoted by $B_1,\ldots,B_n$, where n is the number of B frames these two reference frames (typically n=2). The first is to derive the flow between the first reference frame R_i and its next frame B_1 using the forward predicted motion vectors of B_1. This case is similar to the I-P case discussed above. The inverses of the forward predicted motion vectors form the form the flow vectors for the MBs of R_i.

Similarly using the backward predicted motion vectors of frame B_n with respect to R_j, the flow is derived for frame B_n (Figure 3.4). There is no need to invert the motion vectors here since the flow vectors essentially are backward predicted vectors. Flow for R_j will be derived when R_j is analyzed with the reference frame following R_j.

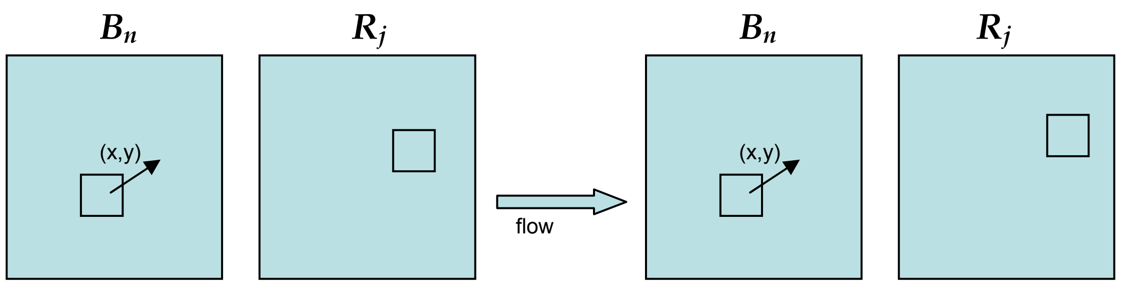


Figure 3.4: The backward predicted vectors of Bn are used as is to derive the flow for it

We have not yet considered the cases where a MB in B_1 does not have a forward predicted motion vector with respect to R_i, or the case where a MB in B_n does not have a backward predicted motion vector with respect to R_j. In the former case we look for successive until we find a frame, say B_k, in which the corresponding MB has a valid forward predicted motion vector and we use the inverse of that vector. Since this vector is predicted from k frames earlier, we scale it down by a factor of k (Figure 3.5). If we are not able to find such a B frame, we tag the flow vector as undefined.

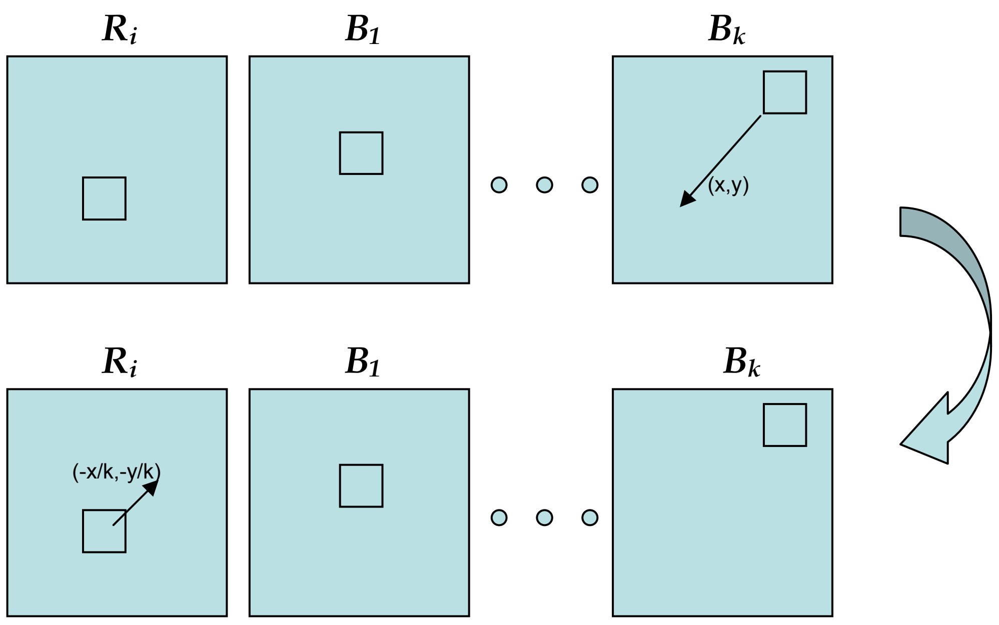


Figure 3.5: Flow derived for R_i from the forward predicted vector of B_k

Similarly, in the latter case, we look at preceding B frames until we find a B frame with a valid backward predicted motion vector, which is similarly scaled down by the number of frames over which it was predicted before being assigned (Figure 3.6).

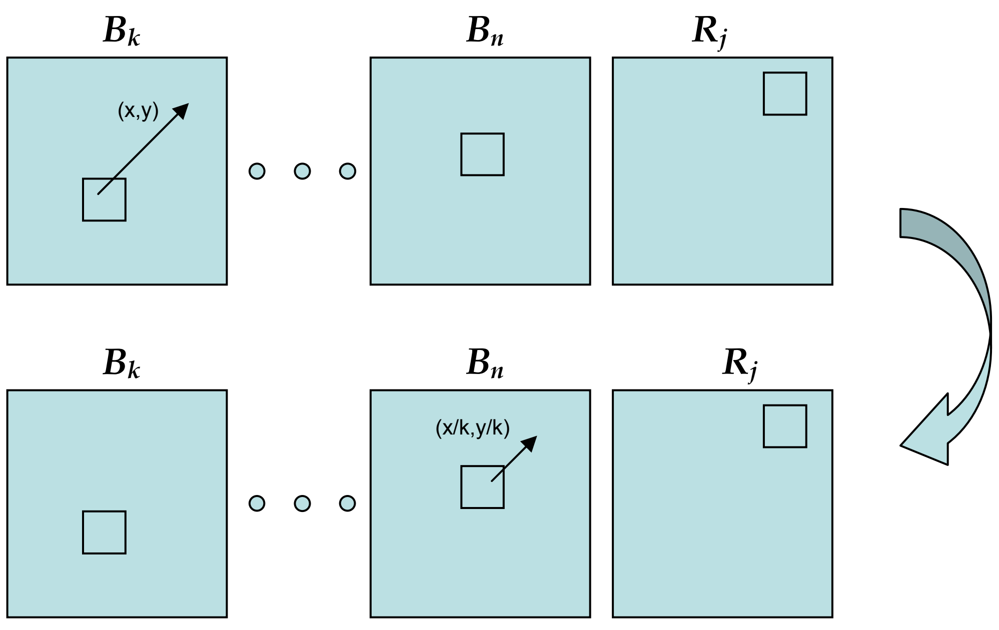


Figure 3.6: Flow derived for B_n from the backward predicted vector of B_k

The next step is to determine the flow between the consecutive B frames. Obviously there is no direct interaction between such consecutive B frames in the MPEG stream, in contrast to the aforementioned flow derivation step involving a reference frame. Flow between successive B frames is derived by analyzing corresponding MBs in those B frames and their motion vectors with respect to their reference frames. We want to find the vector from a MB in one B frame, say *B₁*, to the corresponding MB in the next B frame, say *B₂*. Since each MB in each B frame can be of one of three types, namely forward predicted (F), backward predicted (B), or bidirectionally predicted (D), there exist nine possible combinations. We can represent these nine pairs by FF, FB, FD, BF, BB, BD, DF, DB, and DD. Each of these nine combinations is considered individually and the flow is estimated between them by analyzing each of the motion vectors with respect to the reference frame.

Let the forward predicted motion vector of the current MB in B_1 be denoted by $\overrightarrow{B_1R_i}$, and let the forward predicted motion vector of the corresponding MB in B_2 be denoted by $\overrightarrow{B_2R_i}$. If we denote the flow between the MBs of the B frames by $\overrightarrow{B_1B_2}$, then we have the relationship

$$\overrightarrow{B_1B_2} = -(\overrightarrow{B_2R_i} - \overrightarrow{B_1R_i})$$

from which $\overrightarrow{B_1B_2}$ can be obtained easily. Since only the forward predicted motion vectors $\overrightarrow{B_1R_i}$ and $\overrightarrow{B_2R_i}$ are required, this covers the cases of the MB having patterns FF, FD, DF or DD (Figure 3.7).

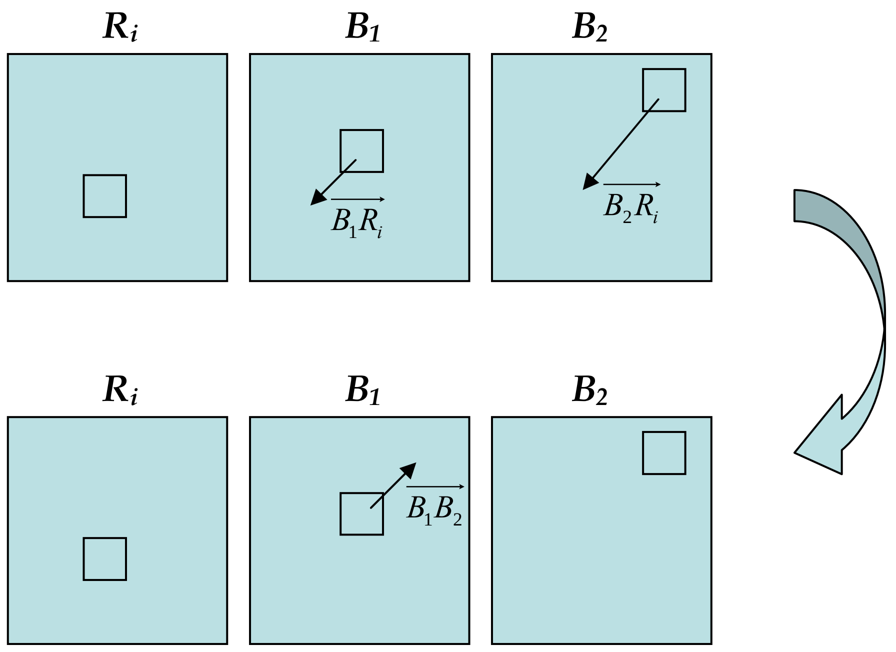

R_i
B_1
B_2
$\overrightarrow{B_1R_i}$
$\overrightarrow{B_2R_i}$
R_i
B_1
B_2
$\overrightarrow{B_1B_2}$

Figure 3.7: Flow derived for B_1 from the forward predicting vectors of B_1 and B_2

If the MB pair has patterns BB, BD or DB, we can find $\overrightarrow{B_1B_2}$ by using the backward predicted motion vectors of B_1 and B_2 with respect to R_j. Let the backward predicted motion vector of the current MB in B_1 be denoted by $\overrightarrow{B_1R_j}$ and let the backward predicted motion vector of the corresponding MB in B_2 be denoted by $\overrightarrow{B_2R_j}$. Then we have

$$\overrightarrow{B_1B_2} = \overrightarrow{B_1R_j} - \overrightarrow{B_2R_j}$$

from which $\overrightarrow{B_1B_2}$ can be obtained easily (Figure 3.8).

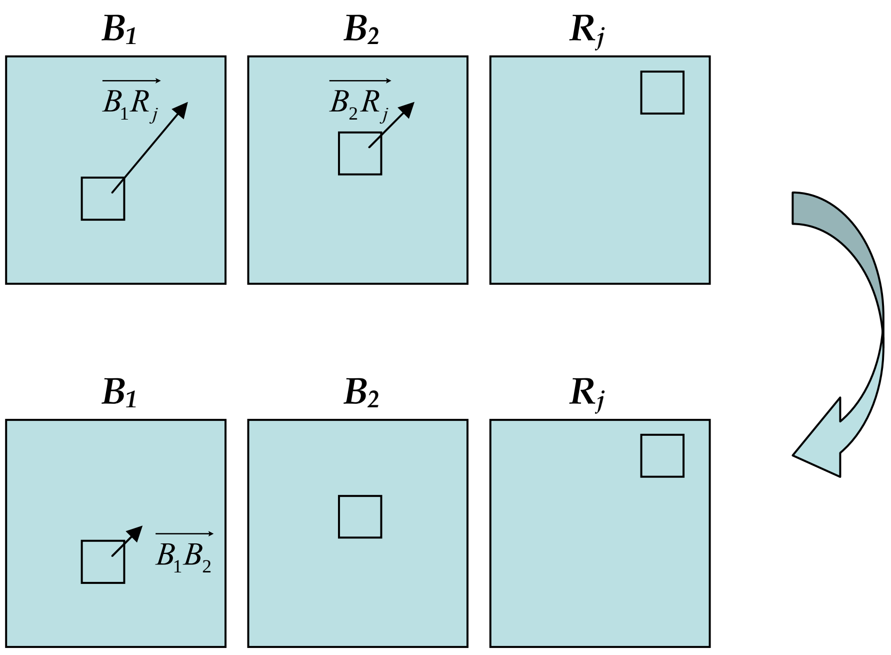

B_1
B_2
R_j
$\overrightarrow{B_1R_j}$
$\overrightarrow{B_2R_j}$
B_1
B_2
R_j
$\overrightarrow{B_1B_2}$

Figure 3.8: Flow derived for B_1 from the backward predicted motion vectors of B_1 and B_2

The only remaining cases to consider are FB and BF. Clearly, for the case of FB, the flow $\overrightarrow{B_1B_2}$ is undefined, because this pattern is an indication of the presence of a cut between the two B frames.

For the BF case the flow vector $\overrightarrow{R_iB_1}$ is computed, for the corresponding MB of R_i using the scale-down technique explained above. Then using the forward predicted motion vector of B_2, $\overrightarrow{B_2R_i}$, $\overrightarrow{B_1B_2}$ is calculated as

$$\overrightarrow{B_1B_2} = -(\overrightarrow{R_iB_1} + \overrightarrow{B_2R_i})$$

The process is shown in Figure 3.9.

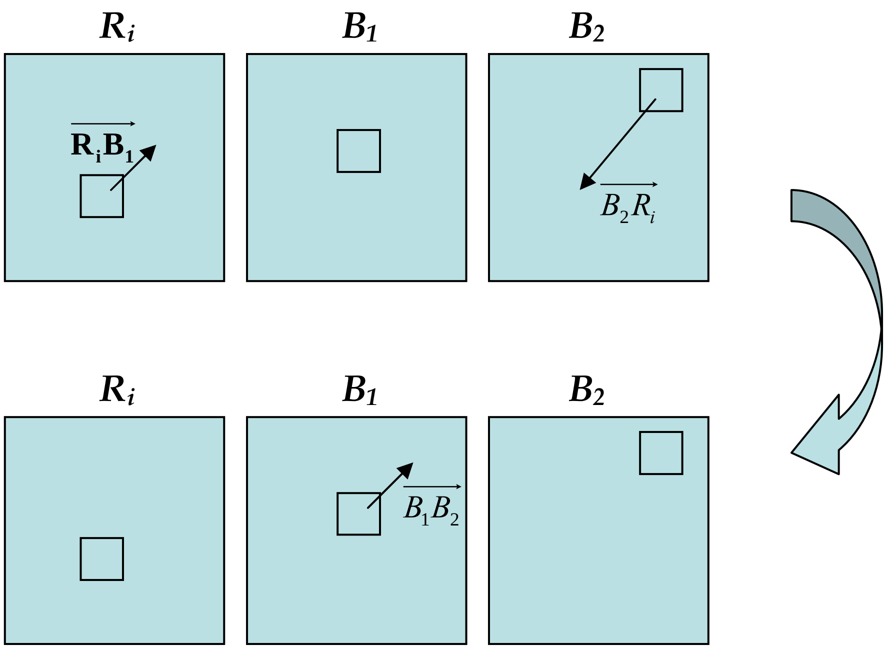

R_i
B_1
B_2
$\overrightarrow{R_iB_1}$
$\overrightarrow{B_2R_i}$
R_i
B_1
B_2
$\overrightarrow{B_1B_2}$

Figure 3.9: Flow derived for B_1 using flow information from R_i and the forward predicted vector of B_2

Similarly the flow vector $\overrightarrow{B_2R_j}$ is found for the MB in B_2 using the scale-down technique. Using the backward predicted motion vector of B_1, $\overrightarrow{B_1R_j}$, $\overrightarrow{B_1B_2}$ is calculated as

$$\overrightarrow{B_1B_2} = \overrightarrow{B_1R_j} - \overrightarrow{B_2R_j}$$

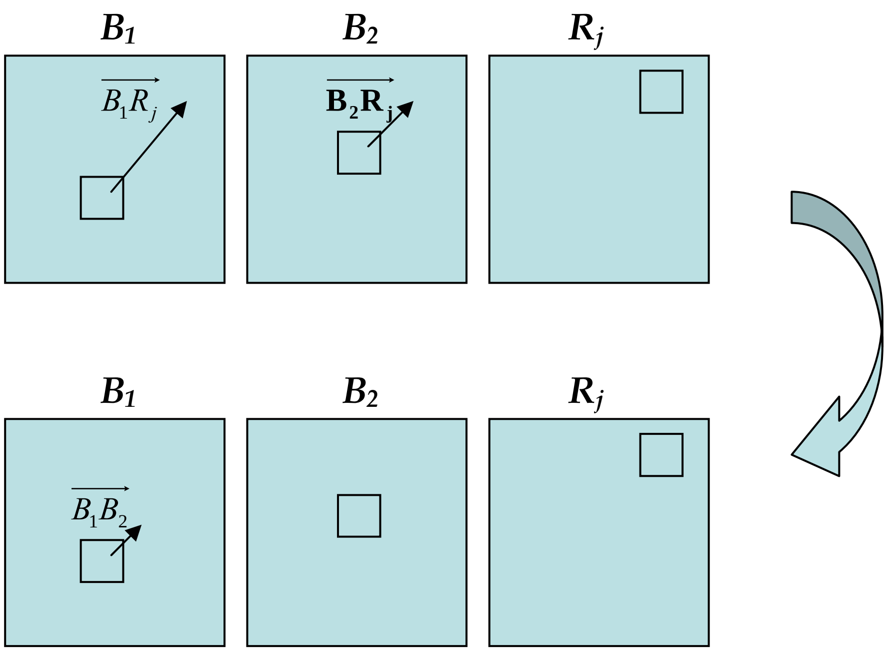
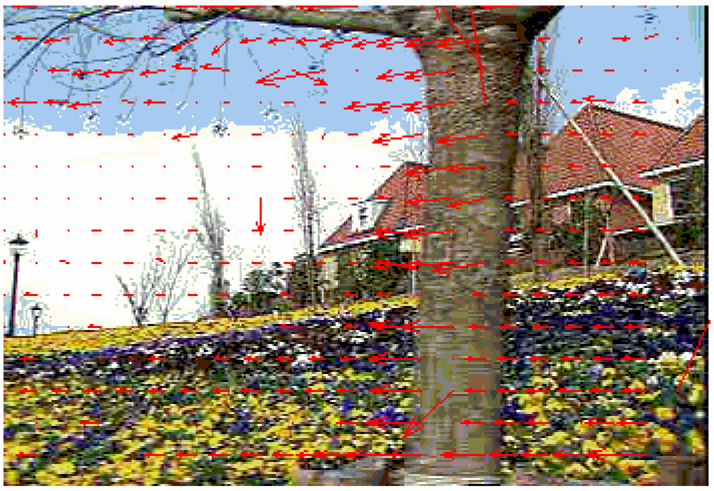

B_1
B_2
R_j
$\overrightarrow{B_1R_j}$
$\overrightarrow{B_2R_j}$
B_1
B_2
R_j
$\overrightarrow{B_1B_2}$

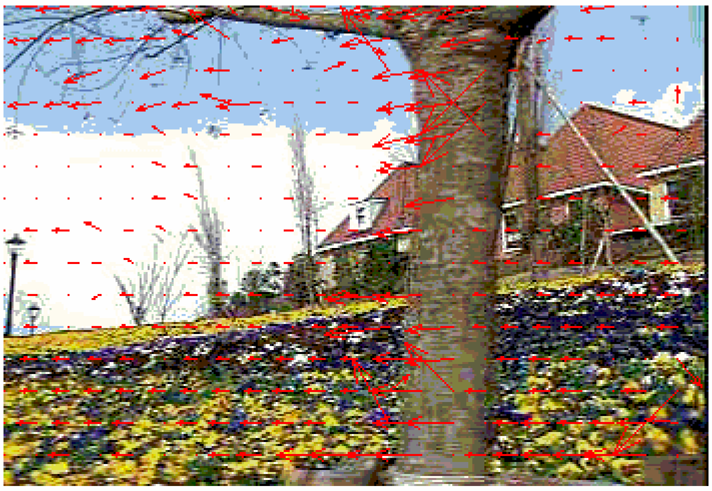
Figure 3.10: Flow derived for B_1 using flow information from B_2 and the backward predicted vector of B_1

Since the vectors have been estimated with respect to the reference frames using the scale down technique, the average of these two vectors is taken to yield a better estimate of the actual $\overrightarrow{B_1B_2}$.

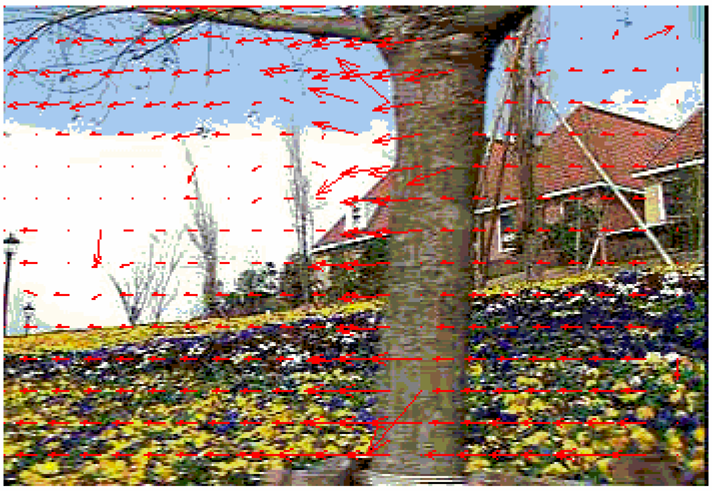
By using the methods described above and after the processing of the MPEG stream a uniform set of motion vectors is acquired, one in which in every frame (I, B or P type) the motion vectors are with respect to the next frame. The result of this process is shown in Figure 3.11 below, where the three frames that were shown earlier in this chapter having different types of motion vectors or none at all, now have the same uniform set.



(a)



(b)



(c)

Figure 3.11: Result of motion vector normalization, (a) I frame, (b) B frame, (c) P frame

4. DC Sequence extraction from the MPEG stream

4.1 Introduction

The international standard for video compression, MPEG, has been widely used both for storage and transmission purposes. While MPEG can reduce significantly the number of bits needed to represent a video sequence without appreciable degradation of image quality, the compressed format does not lend itself to easy image processing. With the growing importance of maintaining huge amounts of video information in applications such as in Digital Libraries, efficient algorithms need to be developed to operate on compressed video for fast visual processing, indexing, browsing, searching and retrieval.

The DCT based compression used for encoding I-frames and intra-coded macroblocks induces a multiresolution structure on the spatial images; thus permitting reconstruction of images of smaller resolution, without the need for full-frame decompression. This chapter examines the reconstruction of such reduced images. For predictively coded P-frames and bidirectionally interpolated B-frames, the situation is more involved. In this chapter, we derive exact expressions and approximations that yield very fast and accurate reconstruction. Minimal decoding is performed to extract the reduced images of various resolution. We demonstrate how such reduced images can still capture global information useful for image processing operations.

4.2 Compression method and DCT

The two-dimensional Discrete-Cosine Transform (DCT) is used as the basis of compression of each I-frame and of the residue images from P frames and B frames. The image is first grouped into 8x8 blocks, and then fed to the Forward DCT. The Forward and Reverse DCT are defined as follows:

$$c(i,j)=\frac{1}{4}k(i)k(j)\left[\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y)\cos\frac{(2x+1)i\pi}{16}\cos\frac{(2y+1)j\pi}{16}\right] \qquad (4.1)$$

$$f(i,j)=\frac{1}{4}\left[\sum_{x=0}^{7}\sum_{y=0}^{7}k(x)k(y)c(x,y)\cos\frac{(2i+1)x\pi}{16}\cos(\frac{(2j+1)y\pi}{16}\right] \quad (4.2)$$

where *i,j=0,1,…,7* and

$$k(i)=\begin{cases}\frac{1}{\sqrt{2}}, & i=0\\ 1, & otherwise\end{cases} \quad (4.3)$$

In terms of matrix notation we can write

$$C=TFT^t$$
$$F=T^tCT$$

where the 8x8 matrices $C=c(i,j)$, $F=f(i,j)$, and T is the DCT matrix with entries $t(i,j)$ given by

$$t(i,j)=\frac{1}{2}k(j)\cos\frac{(2i+1)j\pi}{16} \quad (4.4)$$

The DC term $c(0,0)$ is related to the pixel values $f(i,j)$ by

$$c(0,0)=\frac{1}{8}\sum_{x=0}^{7}\sum_{y=0}^{7}f(x,y) \quad (4.5)$$

which is 8 times the average intensity of the block. The other terms, $c(i,j)$, for $0<i+j<64$ are the AC terms.

4.3 Preliminaries

Given an image *f* of size NxM, we denote each pixel in the lth *BxB* block by $x_l(i,j)$ and its DCT by $c_l(i,j)$. We define a reduced image *f(n)* to be of type R_3, if

1. it is of size $\frac{N}{8} \times \frac{M}{8}$

2. the DCT coefficients $c_l^{(3)}$ of the l_{th} block of size $\frac{B}{8} \times \frac{B}{8}$ are formed as follows:

$$c_l^{(3)}(i,j) = \frac{1}{8} c_l(i,j) H_3(i,j), \qquad i,j = 0,1,\ldots,\frac{B}{8}-1 \tag{4.6}$$

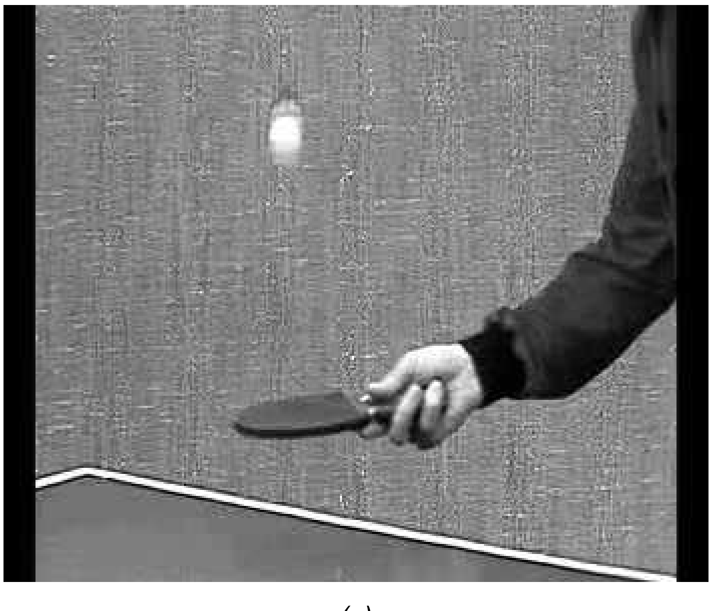
where

$$H_3(i,j) = \begin{cases} 1 & ,i = j = 0 \\ 0 & ,otherwise \end{cases} \tag{4.7}$$

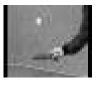
Condition (2) means that the reduced image is formed by retaining selected terms of the original DCT coefficients of f.

Figure 4.1 shows an image of a frame in an MPEG sequence with its original resolution of 320x288 and its reduced DC image with dimensions 40x36. It can be seen that important content information is still well preserved in the DC image.

The following have been proposed by Yeo [7].



(a)



(b)

Figure 4.1: (a) Original frame from the "Tennis" sequence, (b) reduced DC image of that frame

4.4 Reconstruction of type R_3 image: DC image and DC sequence

A reduced image of type R_3 is reduced 8 times in each dimension. This corresponds to using one pixel to represent every 8x8 block. In the DCT domain, only the DC coefficient of each block is retained, with $H_3(0,0) = 1$. We shall call such image a DC image and the sequence of such images a DC sequence. It is clear that the DC image is obtained from blockwise averaging. While it is much smaller than the original image, it retains significant amount of “global" information.

The extraction of the DC image from an I frame is trivial. We will now show how the DC image corresponding to a P frame or B frame can be exactly extracted directly from the MPEG compressed streams.

4.4.1 Exact extraction of DC sequence

The extraction of exact values of DCT coefficients in a P frame from an I frame follows the approach described by Chang and Messerschmitt, where the DCT coefficients of a motion-compensated block are recovered from the DCT coefficients of the anchor frame. The computation of the DCT coefficients of a new arbitrary position image block from the DCT coefficients of the four original neighbouring blocks takes the form of pre-multiplying and post-multiplying of those blocks with appropriate matrices. Suppose in Figure 4.1, P_{ref} is the current block of interest, P_1, P_2, P_3, P_4 are the four original

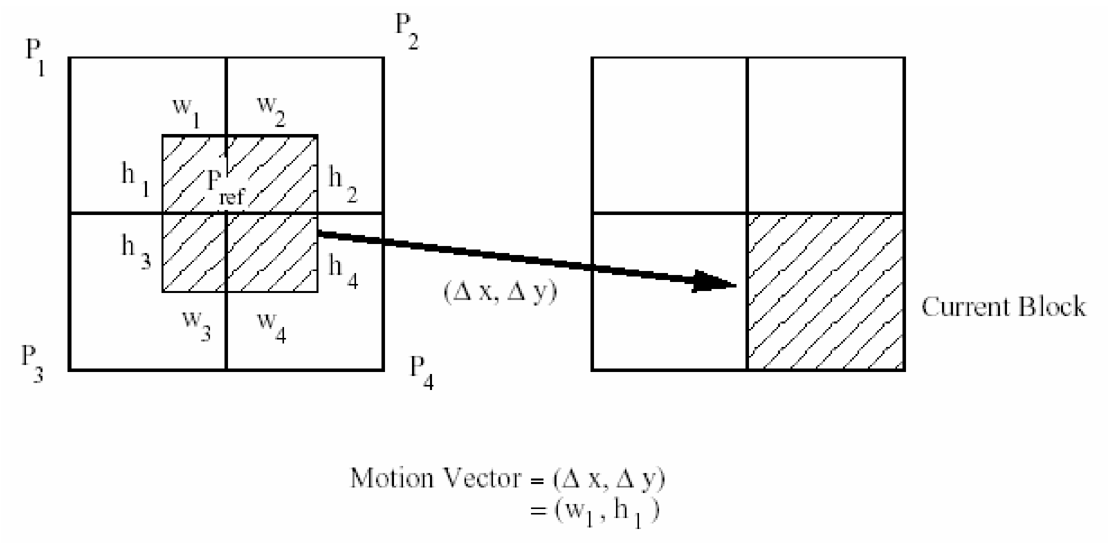


Figure 4.2: Reference block P_{ref}, motion vectors and original blocks

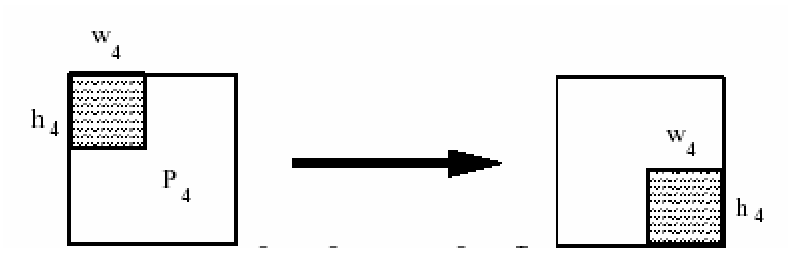
neighbouring blocks from which P_{ref} is derived and the motion vector is *(Δx, Δy).* If we each block as an 8x8 matrix, then we can describe in the spatial domain through matrix multiplication

$$P_{ref} = \sum_{i=1}^{4} S_{i1} P_i S_{i2} \tag{4.8}$$

where S_{ij} are matrices like

$$L_n = \begin{bmatrix} 0 & 0 \\ I_n & 0 \end{bmatrix} \text{ or } R_n = \begin{bmatrix} 0 & I_n \\ 0 & 0 \end{bmatrix}$$

Each I_n is an identity matrix of size n. An example of such an effect of S_{ij} on P is demonstrated on Figure 4.3



$$\begin{bmatrix} 0 & 0 \\ I_{h4} & 0 \end{bmatrix} P_4 \begin{bmatrix} 0 & I_{w4} \\ 0 & 0 \end{bmatrix}$$

Figure 4.3: Pre and Post matrix multiplications to move subblocks

There are four possible locations of the subblock of interest: upper left, upper right, lower left, lower right. The actions in term of matrices are tabulated is Table 4.1 below.

Subblock	**Position**	S_{i1}	S_{i2}
P_1	lower right	$\begin{bmatrix} 0 & 0 \\ I_{h1} & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & I_{w1} \\ 0 & 0 \end{bmatrix}$
P_2	lower left	$\begin{bmatrix} 0 & I_{h2} \\ 0 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & I_{w2} \\ 0 & 0 \end{bmatrix}$
P_3	upper right	$\begin{bmatrix} 0 & 0 \\ I_{h3} & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 0 \\ I_{w3} & 0 \end{bmatrix}$
P_4	upper left	$\begin{bmatrix} 0 & 0 \\ I_{h4} & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & I_{w4} \\ 0 & 0 \end{bmatrix}$

To proceed we denote the 2D DCT of block P by $DCT(P)$, i.e.

$$DCT(P) = TPT^t \tag{4.9}$$

where T is the DCT matrix defined in section 4.2. By noting the relationship

$$DCT(AB) = DCT(A)DCT(B) \tag{4.10}$$

we can apply 2D DCT to each side of (4.8) thus obtaining

$$DCT(P_{ref}) = \sum_{i=1}^{4} DCT(S_{i1})DCT(P_i)DCT(S_{i2}) \tag{4.11}$$

In the cases when one of Δx or Δy is a multiple of the blocksize (i.e. 8), one of $DCT(S_{i1})$ and $DCT(S_{i2})$ reduces to I, the identity matrix.

Note that while we are only interested in DC sequences, we have to recover all the DCT coefficients in a P frame because P frames are used as anchors for prediction of successive B frames and P frames. In the case of B frames, which are not used as anchor frames, we can use less computation. In particular, the DC value of $DCT(P_{ref})$ is given by

$$\begin{aligned}(DCT(P_{ref}))_{00} &= \sum_{i=1}^{4}\left(\sum_{m=0}^{7}\sum_{l=0}^{7}(DCT(S_{i1}))_{0m}(DCT(P_i))_{ml}(DCT(S_{i2}))_{l0}\right)\\ &= \sum_{i=1}^{4}\left(\sum_{m=0}^{7}\sum_{l=0}^{7}w_{ml}^{i}(DCT(P_i))_{ml}\right)\end{aligned} \tag{4.12}$$

where we denote the *(i,j)* component of P by $(P)_{i,j}$ and write $w_{ml}^{i} = (DCT(S_{i1}))_{0m} \times (DCT(S_{i2}))_{l0}$. The interpretation of (4.12) is that the DC value of block P_{ref} is a linear combination of DCT coefficients of P_1, P_2, P_3, P_4 with weights given by w_{ml}^{i}. Thus, for each motion vector, in order to construct a DC value in a B frame, we need to perform a maximum of 256 multiplications. In the case of a P frame, all of the AC coefficients must be reconstructed as well.

4.4.2 Approximate extraction of DC sequence

It is clear from (4.12) that it is computationally expensive to reconstruct a DC coefficient for a B frame and even more so to reconstruct all the DCT coefficients of a P frame.

To reduce the computation of reconstructions of DC values, we propose here a method known as *first order approximation.* We shall assume without loss of generality that $0 \le \Delta x \le 8$ and $0 \le \Delta y \le 8$. Thus, in figure 4.2, $h_1 = h_3 = \Delta x, w_1 = w_3 = \Delta y, h_2 = h_4 = 8 - h_1, w_2 = w_4 = 8 - w_1$. We denote by $DC(P)$ the DC value of P. The DC values under first order approximation are denoted by $DC(P)^1$.

To compute the first order approximation, we weigh the contributions from the 4 neighbouring DC values with the ratio of overlaps of the block P_{ref} with each of the block $P_1,...,P_4$:

$$DC(P_{ref})^1 = \sum_{i=1}^{4}\frac{h_i w_i}{64}DC(P_i)$$

Here 4 multiplications are performed for the DC value.

Two interesting observations can be made regarding the first order approximations.

Observation 1: The coefficient w_{00}^{i} in (4.12) equals $\frac{h_i w_i}{64}$.

This follows simply from the fact that w_{00}^{i} is

$$\left(\frac{1}{8}\sum_{n,m}(S_{i1})_{n,m}\right)\left(\frac{1}{8}\sum_{n,m}(S_{i2})_{n,m}\right)=\frac{h_i w_i}{64}$$

Rewriting (4.12) as

$$DC(P_{ref})=\sum_{i=1}^{4}\frac{h_i w_i}{64}DC(P_i)+c$$

where

$$c=\sum_{i=1}^{4}\left(\sum_{(m,l)\neq(0,0)}w_{ml}^{i}(DCT(P_i))_{ml}\right) \tag{4.13}$$

we can view that reconstruction of $DC(P_{ref})$ as a sum of the first order approximation and an error term c that does not depend on the DC coefficients of $P_1,\ldots,P_4$. We shall denote by $\hat{c}=\frac{c}{8}$ the normalized error term. This normalization takes into account the fact that the DC value is 8 times the average intensity of the block.

Observation 2: The maximum absolute normalized error is $\frac{3}{4}\times\max_{x,y} f(x,y)$, where $\max_{x,y} f(x,y)$ denotes the maximum value a pixel takes on. For 8-bit images this value is 255.

Proof : Instead of working directly on (4.13) in the transform domain, we shall work in the spatial domain. Referring to Figure 4.4 we can write the normalized error as

$$
\begin{aligned}
\hat{c} &= \frac{1}{64}\sum_{i=1}^{4}\left(\left(1-\frac{h_i w_i}{64}\right)\sum_{A_i} f(x,y) - \frac{h_i w_i}{64}\sum_{\overline{A}_i} f(x,y)\right) \\
&\le \frac{1}{64}\sum_{i=1}^{4}\left(\left(1-\frac{h_i w_i}{64}\right)\sum_{A_i} f(x,y)\right) \\
&\le \max_{x,y} f(x,y) \times \sum_{i=1}^{4}\left(1-\frac{h_i w_i}{64}\right)\frac{h_i w_i}{64} \\
&\le \frac{3}{4}\max_{x,y} f(x,y)
\end{aligned}
$$

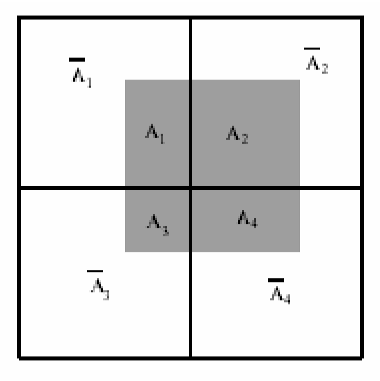


Figure 4.4: Block arrangement

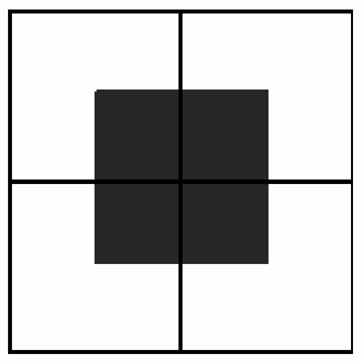


Figure 4.5: Worst case block arrangement

Equality is attained in all the inequalities if

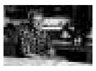
$$
f(n,m) = \begin{cases} 0 & ,(n,m) \in \overline{A}_i \\ \max_{x,y} f(x,y) & ,(n,m) \in A_i \end{cases}
$$

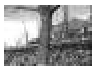
and $h_i = h_j$ and $w_i = w_j$, for all i,j=1,2,3,4.

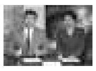
So the worst case approximation occurs when exactly ¼ of P_{ref} overlaps with each P_i with values $\max_{x,y} f(x,y)$ in P_{ref} and 0 elsewhere. This situation is shown in Figure 4.5. In practice however, such arrangement rarely occurs. Experimental results show that DC sequence constructed using first order approximation is good.

The error depends on variations of values in the blocks and $(\Delta x, \Delta y)$. At the extreme, if each of the 4 blocks is constant, then approximation is exact. There will also be no error when $\Delta x = \Delta y = 0$. In addition, as we have just seen, errors could be large when $\Delta x = \Delta y = 4$.

Below DC images are shown which were extracted using first order approximation from the original frames. The results show that the quality of these reduced images is very good.







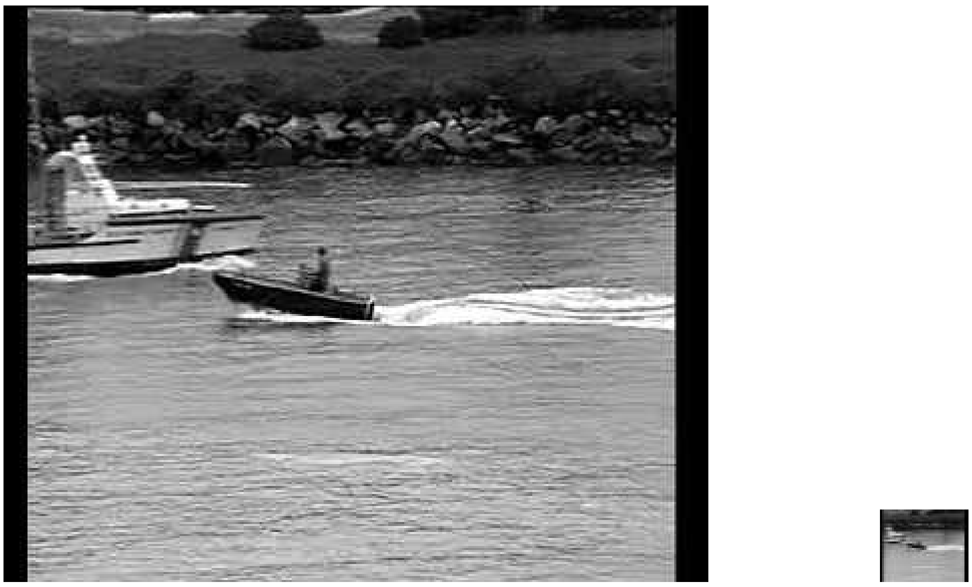


Figure 4.6: Examples of DC images and their original frames

5. Segmentation based on intensity

5.1 Introduction

An approach to video segmentation is though intensity information gathered from the frames of the video. However in an MPEG stream, extracting the intensity information from the frames, through full decoding is not an efficient way of doing it. This due to the fact that much time has been spent on the coding of the video, from uncompressed form to the compressed MPEG form. The process of fully decoding the stream just for analyzing it, not for playback, is computationally expensive and leads to waste of valuable time.

That is why the search for faster algorithms for compressed video processing has led to the use of reduced DC images, extracted as explained in Chapter 3.

The DC images used in this analysis are extracted from P and B frames through motion compensation and approximation. For I frames the DC coefficients for every block are known exactly, since they are intracoded. Each image intensity values range from 0 to 255 in grayscale. They are actually 8-bit images. If $X \times Y$ is the size of the frames for the video, then the size of the DC images is of dimensions $M \times N = \frac{X}{8} \times \frac{Y}{8}$ as one pixel in the reduced image corresponds to one 8x8 block of the original frame. This greatly reduces complexity and computation time for every activity that involves processing.

From now on we will only work with these DC images and when we mention “frames” we will mean these reduced images. This is done for convenience matters. It is also true because the DC images capture the important content information of every frame. So they are considered as frames of reduced dimensions.

5.2 Histogram computation

The metric which will be used for the analysis is the histogram difference of consecutive frames, taken in pairs. So we consider the interframe difference of histograms in a way similar to that of the previous chapter where we used the interframe difference of direction histograms.

The histogram is computed as following:

For every DC image extracted with dimensions MxN, the histogram of that image is computed which is actually a vector with elements taken as

$$H_k(i) = \frac{n_i}{M \times N}, \ i=0,\ldots,255$$

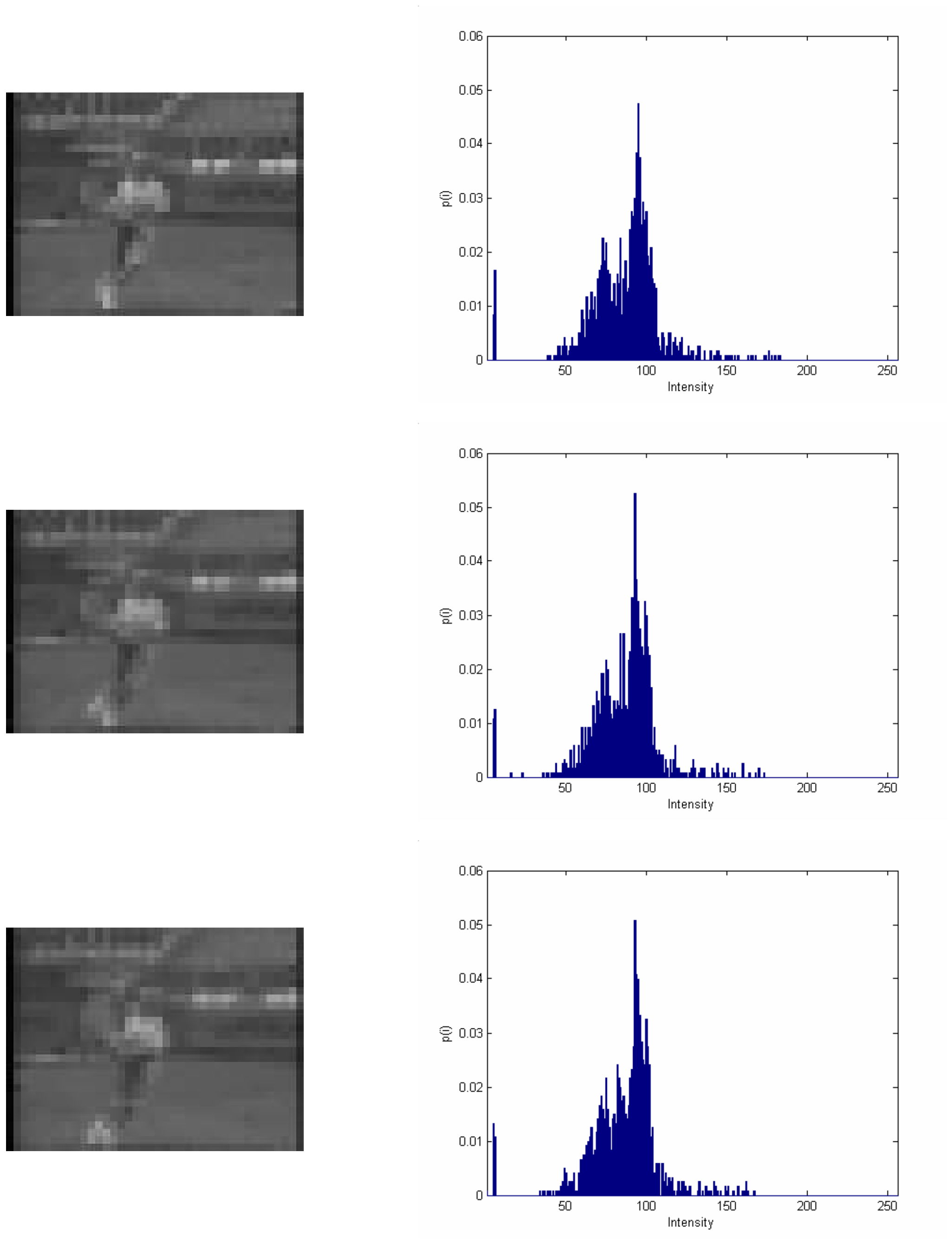
where i is the intensity value, n_i is the number of pixels with intensity i, k is the frame number for the video, $1 \le k \le TotalFrames$ and $TotalFrames$ are the total frames of the video. Clearly the elements of $H_k(i)$ range is

$$0 \le H_k(i) \le 1$$

and

$$\sum_{i=0}^{255} H_k(i) = 1, \qquad 1 \le k \le TotalFrames$$

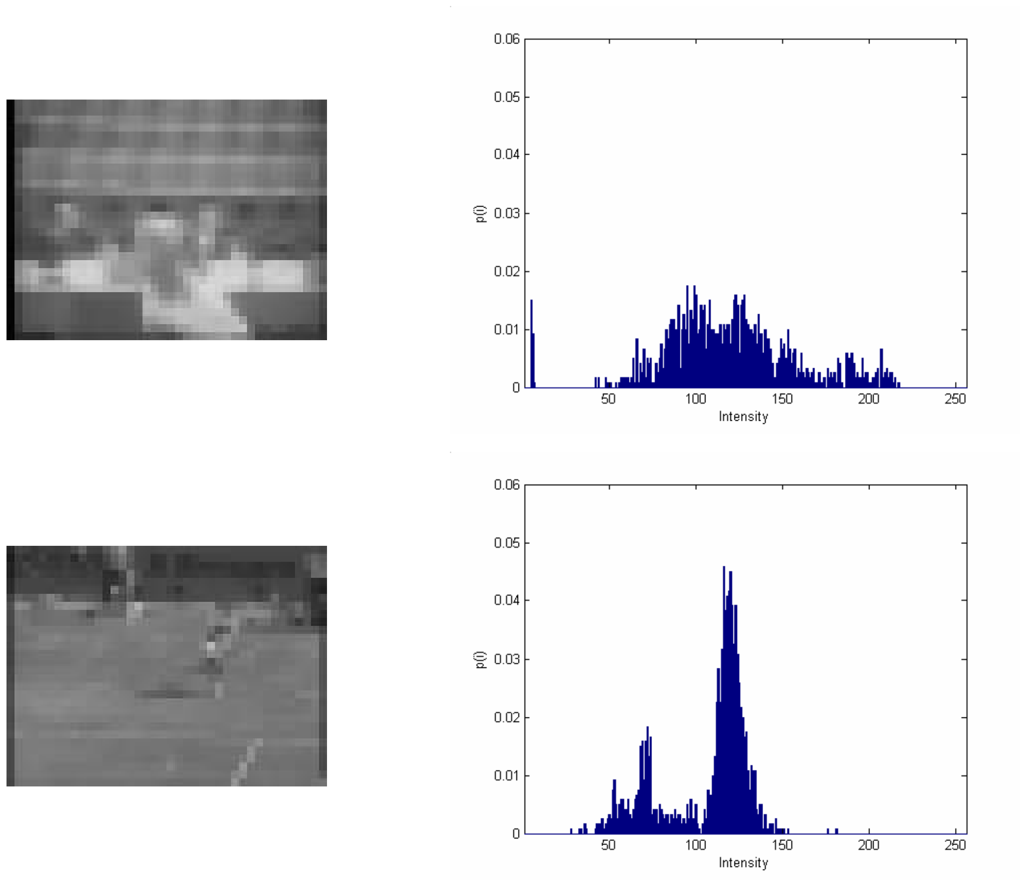
Frames that belong to the same shot will have a similar histogram. Even in sequences where we have new objects getting inside the shot, like panning or zooming, the frame to frame differences will not be large. This is illustrated in Figure 5.1 where there are three frames of the same shot and their corresponding histogram on the side. It is clear the histograms are very much alike.



(a) *(b)*

Figure 5.1: (a) Three consecutive DC images, (b) their corresponding histograms

On the other hand, when there is an abrupt shot change the histogram of the frames just before and just after the shot are very different. This is because the content of the video changes and so does the histograms. The only case where the histograms of two shots are alike is when the shots have the similar content or similar colors but this is a very rare scenario. The histograms of two frames belonging to different shots are shown in Figure 5.2



(a) *(b)*

Figure 5.2: Histograms of frames belonging to different shots

5.3 Interframe difference of histograms

By taking the interframe difference of histograms D for consecutive frames as

$$D(j) = \sum_{i=0}^{255} \left(\left| H_k(i) - H_{k-1}(i) \right| \right)$$

where

$$0 \le j \le TotalFrames - 1$$

$$2 \le k \le TotalFrames$$

We expect large peaks at the time where shot cuts occur. Figure 5.3 shows a plot of D for a video. Indeed the cuts are well distinguished as peaks in the plot and can be easily identified with the use of a proper threshold as described in the next section.

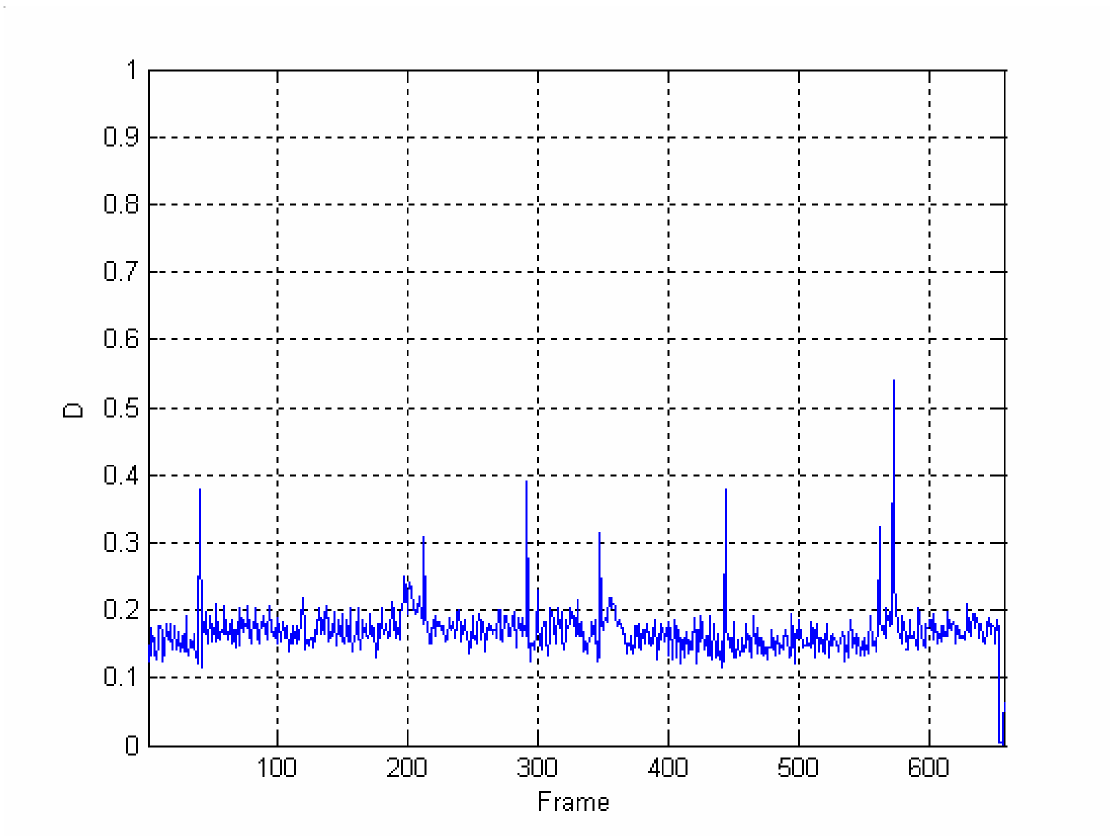
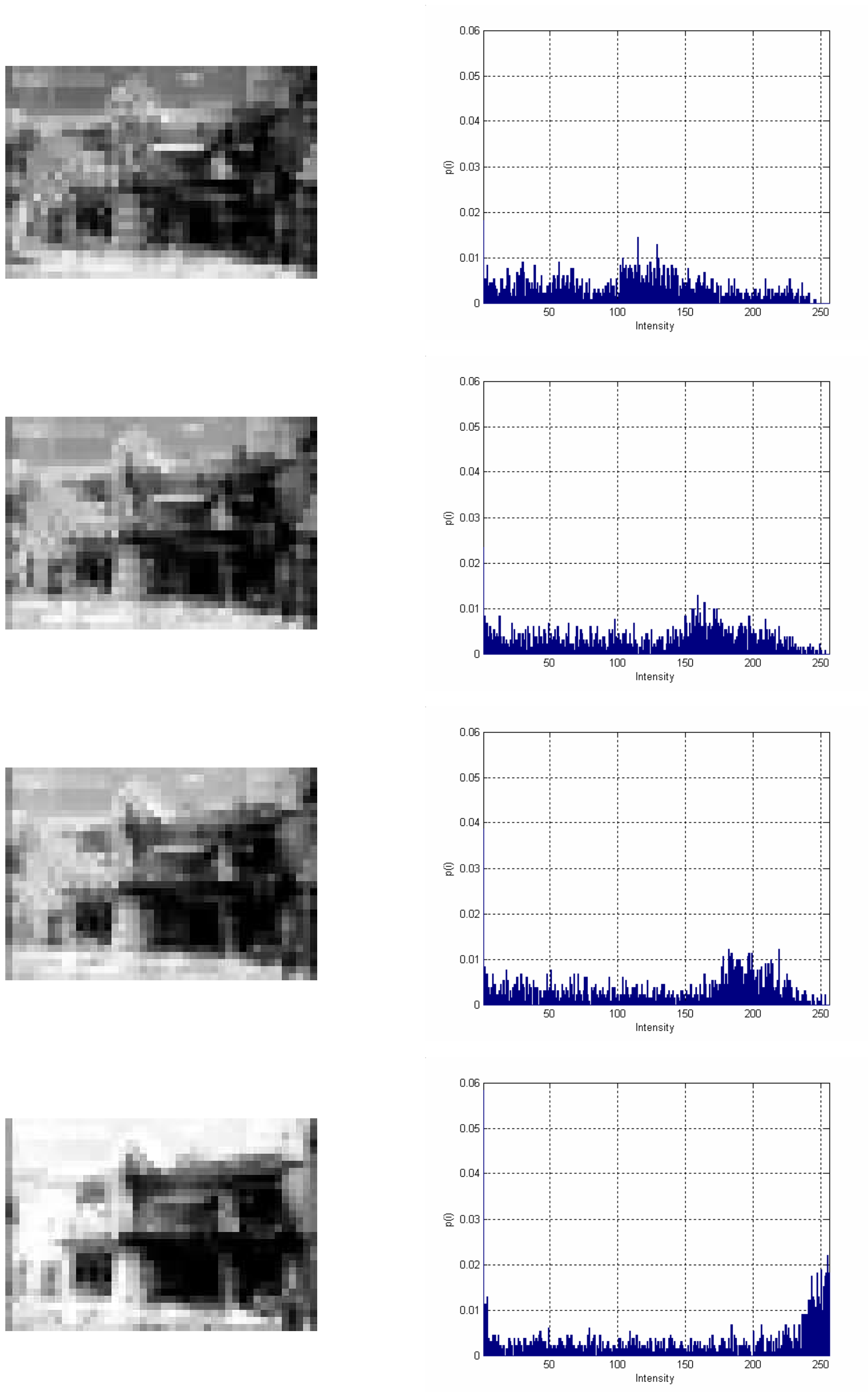


Figure 5.3: Plot of the interframe difference D. The cuts appear as peaks. The interframe difference is normalized to 1.

In the case of gradual changes such as dissolves or fades, the content of the frames as time proceeds changes as time proceeds. It is during that time where the interframe differences in content are large. This can be seen in Figure 5.4 where the histograms of the frames that belong to a gradual change are shown



(a) (b)

Figure 5.4: Histograms of frames belonging to a dissolve (gradual change). Notice how the histogram changes through time.

As it can be seen in Figure 5.5, the frames that belong to a gradual change have a larger interframe difference of histograms than frames that belong to the same shot. So a gradual change in the video will not be presented as a peak in the plot but rather as large values for a period of time, namely the duration of the change.

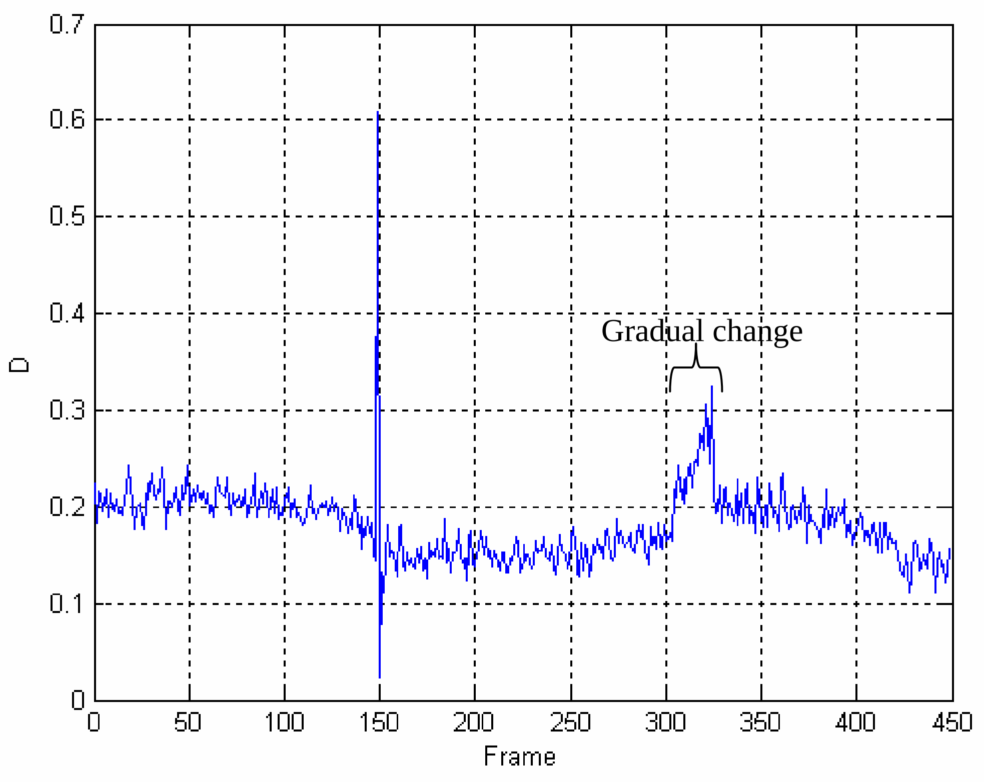


Figure 5.5: Plot of the interframe difference. The gradual change is between the frames 304-330.

For the detection of cuts and gradual changes we will use the threshold which was described in §5.1.4, the sliding window threshold which will be little modified to be used for intensities and interframes difference of intensity histograms.

5.4 Automatic selection of thresholds

A key feature for an accurate and effective video segmentation algorithm is applying the appropriate threshold. Much work and research has been done in finding the appropriate threshold for various video processing states. For detecting scene changes A. M. Alattar used the variance of pixel values or Zhang et al.[14] used twin comparison.

The adoption of automatic selection of a threshold is based on the frame to frame difference over a video source. More precisely, not the exact frame to frame difference but the difference between the histograms of consecutive frames. For the same scene, the interframe differences of histograms should be exactly the same for the whole length of the scene, provided that there is no camera movement. The only reason why there is a variation in the difference can only be due to noise, specifically noise introduced by video production equipment, from digitizing the original video signal or even because the few objects are perfectly still in a scene.

All the above three sources of noise can be described as Gaussian and hence the variation of interframe differences in a video stream can be introduced two factors: Gaussian noise and variations from abrupt cuts, gradual transitions and camera movements.

If σ is the standard deviation and μ is the mean value of frame-to-frame differences then the only departure from μ for frames that lie in the same scene can only be due to Gaussian noise. Thus for most of the frames in the shot, the probability integral

$$P(x) = \int_0^x \frac{1}{\sqrt{2\pi\sigma}} \cdot e^{\frac{|x-\mu|^2}{s\sigma^2}}\, dx$$

will be valid. This means that most interframe differences will fall in the range of 0 to $\mu+\alpha\sigma$ where α is a value constant for the whole stream. So a threshold that can be established is

$$T_a = \mu + \alpha\sigma \qquad (1)$$

Any difference that falls out of this range can only be due to scene changes, abrupt or gradual, if of course the noise introduced is considered to be Gaussian as said above.

5.4.1 Twin Comparison

Twin Comparison was proposed by Zhang[14] and is based on the thoughts made in §5.4. It uses two thresholds, T_b and T_s, for the detection of abrupt cuts and gradual transitions respectively. T_b is defined as

$$T_b = \mu + a_t \cdot \sigma$$

where μ and σ are the overall mean and variance of the interframe histogram difference D, and α_t is constant that is user selected and does not change for the whole video. If the difference exceeds the value of T_b then a scene change is declared at that point.

T_s is defined as

$$T_s = \beta \cdot \mu$$

where β is again a user selected constant. In the case of gradual transitions the values of the difference are expected to be greater than T_s for a number of frames which is the duration of the transition. If the gradual transition starts at frame m and has a duration of k frames then we have that

$$D_n \geq T_s, \qquad m \leq n \leq m + k$$

where D_n is the value of interframe difference at point n. The gradual transition is declared if

$$D_m + \sum_{i=m+1}^{i=m+k} \left| D_i - D_{i-1} \right| > T_b$$

Figure 5.6 shows how the two thresholds Tb, Ts are used for cut and gradual change detection.

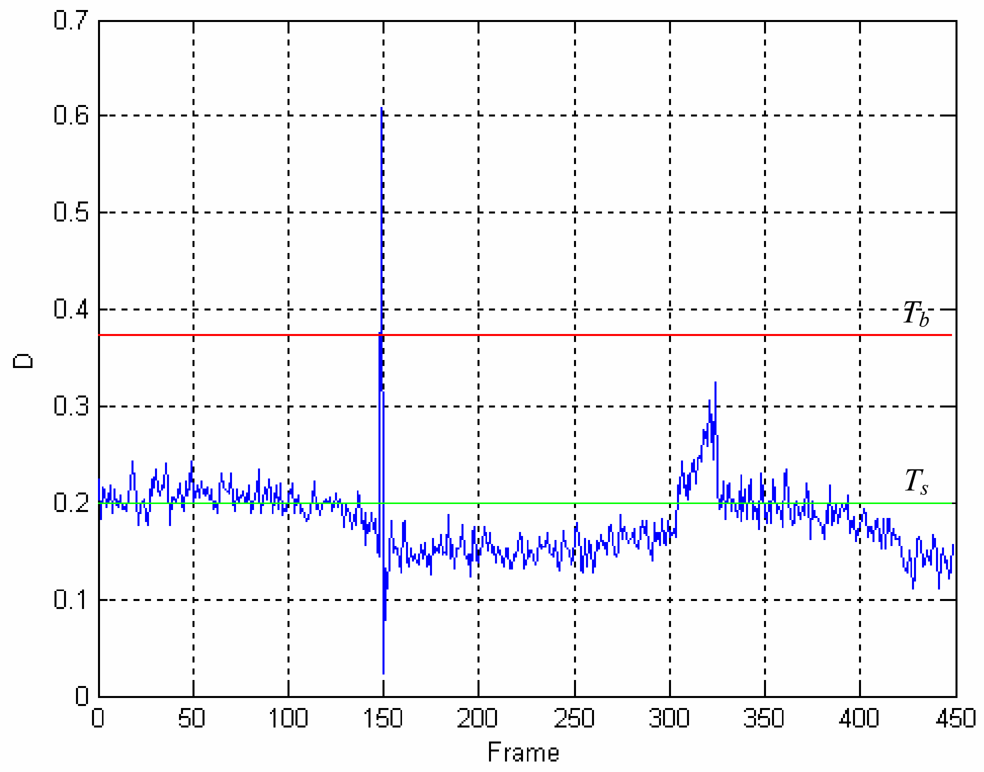


Figure 5.6: Twin Comparison and the two thresholds T_b, T_s

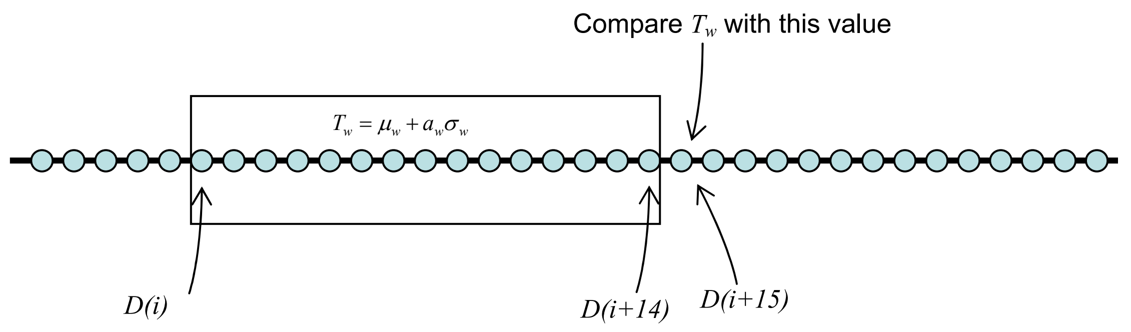
5.4.2 Sliding window threshold

We construct a window with size 15. This means that 15 values of D may be in the window in every moment. Let $T_w(i)$ be the threshold value, $\mu_w(i)$ and $\sigma_w(i)$ be the mean and variance for the data $v(i)$ inside the window. The variable i is in the range of $1 \leq i \leq Frames$, where $Frames$ is the total number of frames in the MPEG stream. We place the window on the difference data and we compute $\mu_w(i)$ and $\sigma_w(i)$. Thus $T_w(i)$ becomes $T_w(i)=\mu_w(i)+\alpha_w\sigma_w(i)$. The parameter α_w is constant in the whole process. Typical values range between 3.5 and 6. We then compare $T_w(i)$ with the next difference value outside the window. This means that suppose we have inside the window the differences [$D(k),\ldots,D(k+14)$] and the threshold is $T_w(k+14)$ then we compare it with $D(k+15)$. If $D(k+15)$ is smaller than $T_w(k+14)$ then $D(k)$ is removed from the window, $D(k+15)$ is placed and the new $\mu_w(k+15)$ and $\sigma_w(k+15)$ are computed.

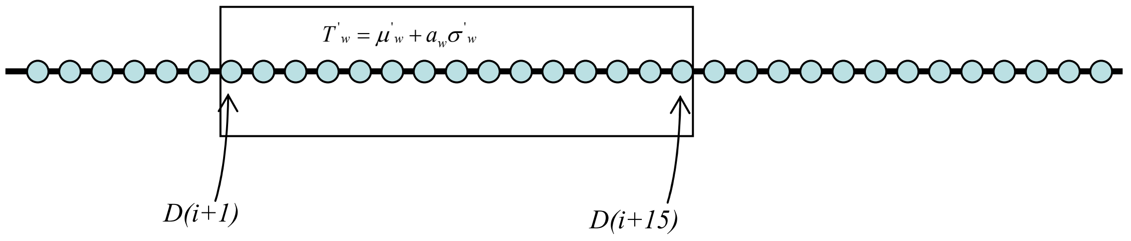
The threshold becomes then $T_w(k+15) = \mu_w(k+15) + \alpha_w \sigma_w(k+15)$. If $D(k+15)$ is greater than $T_w(k+14)$ then the threshold remains constant to $T_w(k+14)$ and the window just progresses until there is a value, say $D(l)$, which is smaller than the threshold. In that case the previous process is continued.
This goes on for the whole length of the video. The algorithm can be described in steps:

1. Construct a window with the first 15 histogram difference $D(i)$ values
2. Calculate the mean μ_w and variance σ_w for the window
3. Calculate the threshold T_w as $T_w = \mu_w + \alpha_w \sigma_w$
4. Compare T_w with the next histogram difference value $D(k)$ outside the window
5. If $T_w > D(k)$ then add $D(k)$ in the window, discard the first value of the window and compute the new mean and variance for it. Calculate the new threshold $T_w^{'} = \mu_w^{'} + \alpha_w \sigma_w^{'}$.
6. if $T_w < D(k)$ do not do any calculations. Just move the window to the right until a histogram difference value $D(l)$ is smaller than the threshold T_w. Then goto step 5.
7. Repeat for all the frames in the stream.

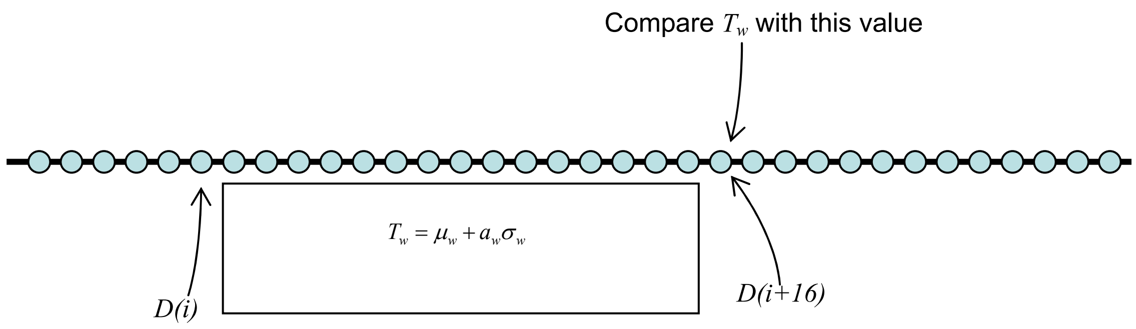
It should be cleared out that for the first 14 values of the histogram differences the threshold is 0 and no comparison is made. This is done because the window needs to have 15 data instances to operate. So $T_w(i) = 0$, for $0 \le i \le 14$. The thresholding scheme is shown in Figure 5.7.



(a)



(b)



(c)

Figure 5.7: (a) An instance of the window. The threshold is compared with the data outside the window.

(b) The threshold is larger than the data so the window moves to the right by discarding the first data D(i) and inserting the last D(i+15). The new threshold is calculated.

(c) If the threshold is smaller that the data in (a) then no other data is entered in the window and the threshold is unchanged. It moves to the right until the threshold meets a smaller data value.

If we apply the threshold in the data of Figure 5.5 then we have the result shown in the Figure 5.8 below

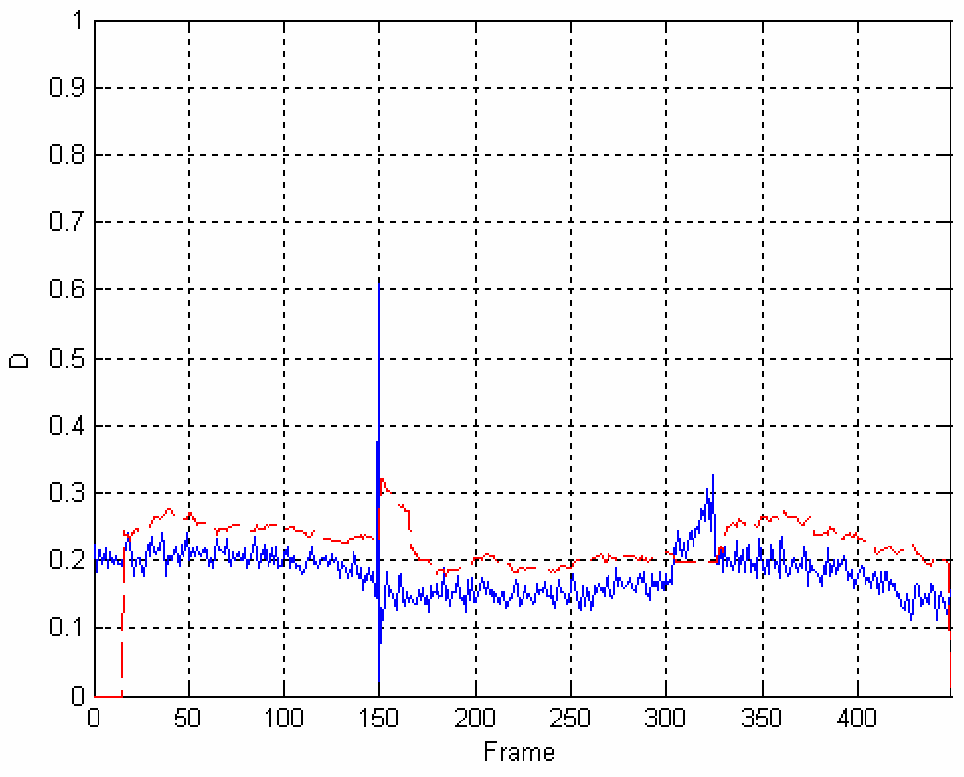


Figure 5.8: The threshold is applied on D. The gradual change is detected.

5.4.3 Adaptive threshold

Another threshold that was used was the adaptive threshold. It's main characteristic is that it can adapt to the signal and can detect large changes with great efficiency. It is constructed in the way explained below.

Three quantities are used for the computation of the threshold, namely μ_a, λ_a and σ_a where μ_a is the adaptive mean, λ_a is the adaptive second moment and σ_a is the adaptive standard deviation. The formulas for each quantity are:

$$\mu_a(i) = \mu_a(i-1) - c \cdot \left(\mu_a(i-1) - D(i)\right) \qquad \text{(Eq. 1)}$$

$$\lambda_a(i) = \lambda_a(i-1) - c \cdot \left(\lambda_\alpha(i-1) - (D(i))^2\right) \qquad \text{(Eq. 2)}$$

$$\sigma_a(i) = \sqrt{\left|\mu_a(i)^2 - \lambda_a(i)\right|} \qquad \text{(Eq. 3)}$$

with

$$\mu_a(1) = D(1)$$

$$\lambda_a(1) = (D(1))^2$$

$$\sigma_a(1) = \sqrt{\mu_a(1)^2 - \lambda_a(1)} = 0$$

c is the controlling coefficient and it determines how sensitive the threshold will be to signal changes. We used a value of 0.05 for c.

The threshold is computed as

$$T_a(i) = \mu_a(i) + a_a \cdot \sigma_a(i)$$

where a_a is a user selected constant. It is applied on the histogram difference in a similar way to the sliding window threshold.

For the first 15 values no comparison is made and the threshold is left to adapt to the difference signal. After these 15 values we compare the current value of the threshold $T_a(i)$ with the next difference value $D(i+1)$. If the latter is smaller than $T_a(i)$ then the new value of the threshold is computed from the above equations, $T_a(i+1) = \mu_a(i+1) + a_a \cdot \sigma_a(i+1)$.

If the difference value $D(i+1)$ is larger than $T_a(i)$ then threshold remains constant and $T_a(i+1) = T_a(i)$ for every consecutive difference value that is above $T_a(i)$ until a value $D(j)$ is smaller. Then the threshold becomes $T_a(j) = \mu_a(j) + a_a \cdot \sigma_a(j)$ where μ_a and σ_a are computed from equations (1) and (3). The process is shown schematically in Figure 5.9.

If we apply the adaptive threshold on a histogram difference we have the result of Figure 5.10.

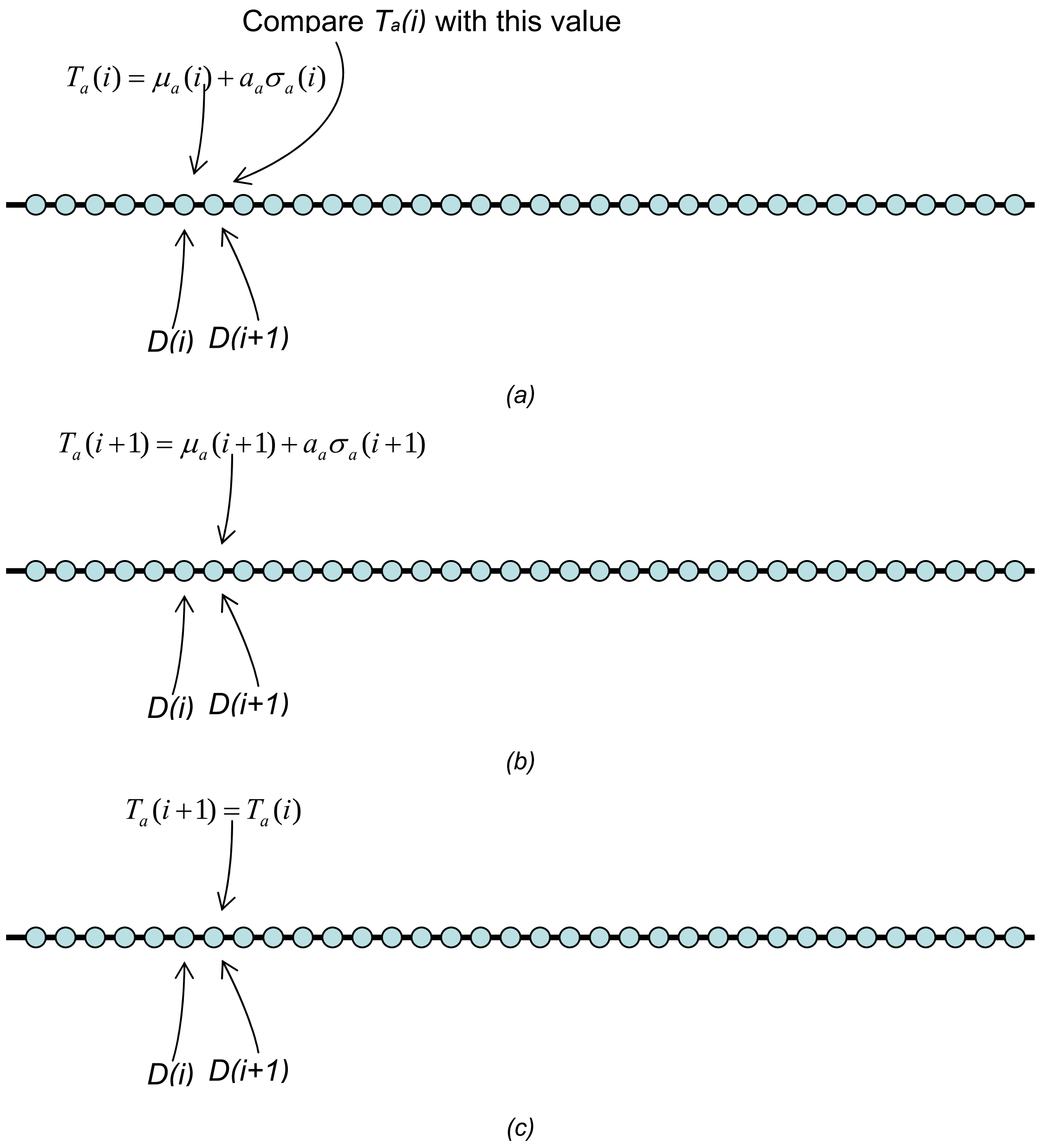

Compare $T_a(i)$ with this value
$T_a(i) = \mu_a(i) + a_a\sigma_a(i)$
D(i)
D(i+1)
(a)
$T_a(i+1) = \mu_a(i+1) + a_a\sigma_a(i+1)$
D(i)
D(i+1)
(b)
$T_a(i+1) = T_a(i)$
D(i)
D(i+1)
(c)

Figure 5.9: (a) An instance of the threshold. It is compared with the data outside the window.
(b) The threshold is larger than the data so the new T_a value is computed based on the formulas
(c) If the threshold is smaller that the data in (a) then the threshold is unchanged. It moves to the right until it meets a smaller data value.

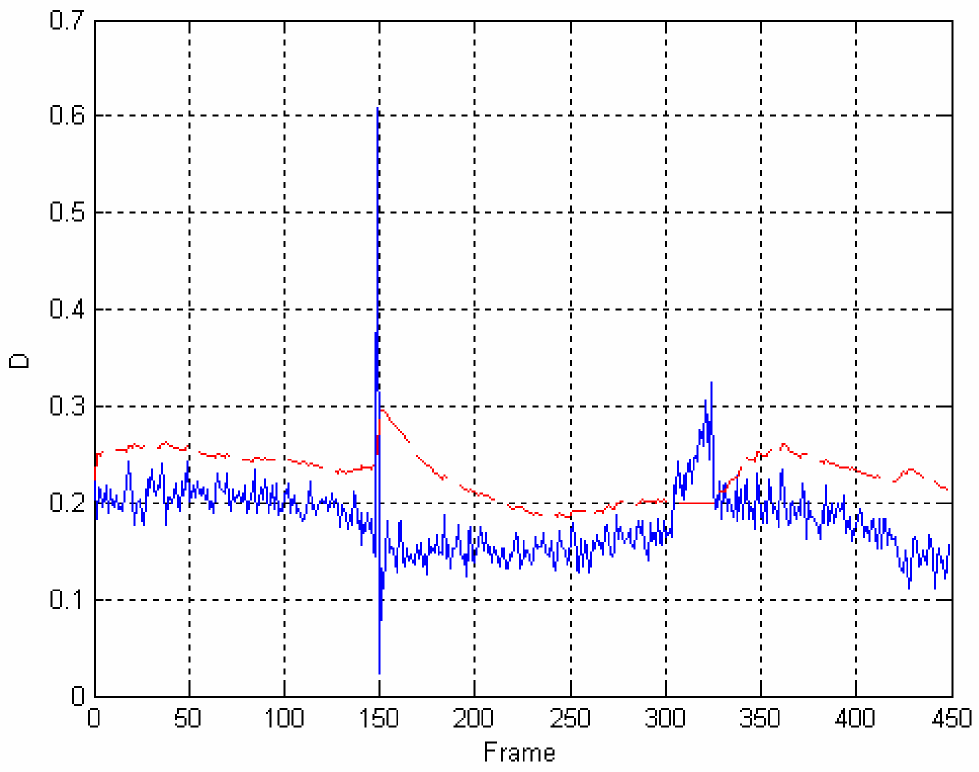


Figure 5.10: The appliance of the adaptive threshold on a histogram difference signal. Notice how it detects the cut at frame 149 and the gradual transition at frames 304-326

6. Camera motion classification and scene segmentation based on information from motion vectors

In this chapter we study techniques that can be used for scene segmentation and camera motion classification. We focus on panning and zooming detection, camera motion change detection which shows us when the camera changes direction or stops moving, abrupt cut and gradual transition detection based solely on motion information. Different approaches are used for each case, summarized in Table 6.1.

Detection of:	Method used
Panning, zooming	Variance of direction histogram
Camera change	Interframe difference of direction histograms
Abrupt cuts	Use of undefined flow vectors
Gradual transitions	Variance of length of flow vectors

Table 6.1: Methods used in this chapter

After the transformation of the motion vectors described in the previous section we can now proceed to the analysis of the information that these vectors provide for the MPEG stream. Although for the decoder they show how a MB has moved from one frame f_n to another f_{n+1}, to us they actually describe how an object has moved in that time interval. This movement can be one or multiple objects moving alone in the frame (Figure 6.1) or the whole scene moving to one direction as this happens when the camera is moving (Figure 6.2). The latter case is what we are interested in. The process of detecting this type of motion and classifying it is described in the section below.

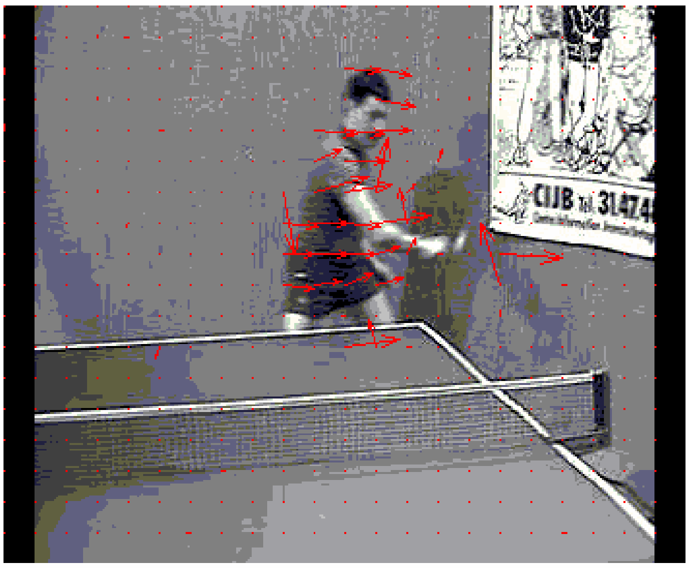
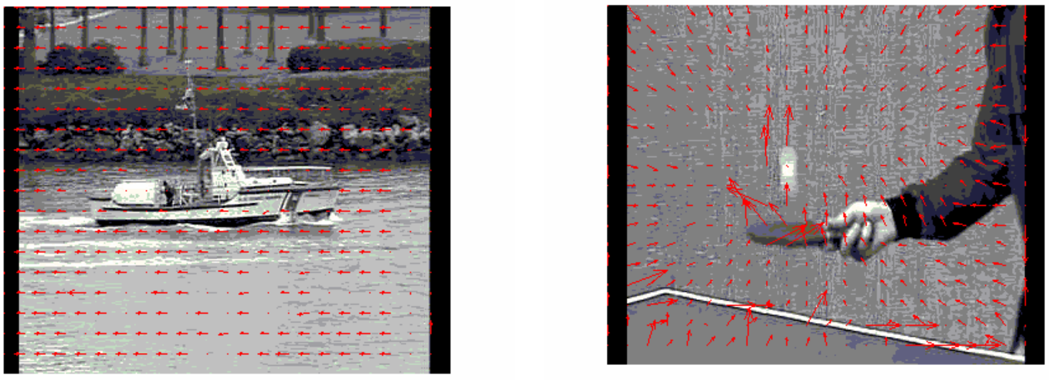


Figure 6.1: Object motion. Outlier vectors due to noise are visible



(a) *(b)*

Figure 6.2: (a) Panning sequence, (b) Zooming sequence

6.1 Camera motion detection

Before we proceed to explaining the process of camera motion detection we should first make clear some facts that play a big role in our analysis and have much influence in the results.

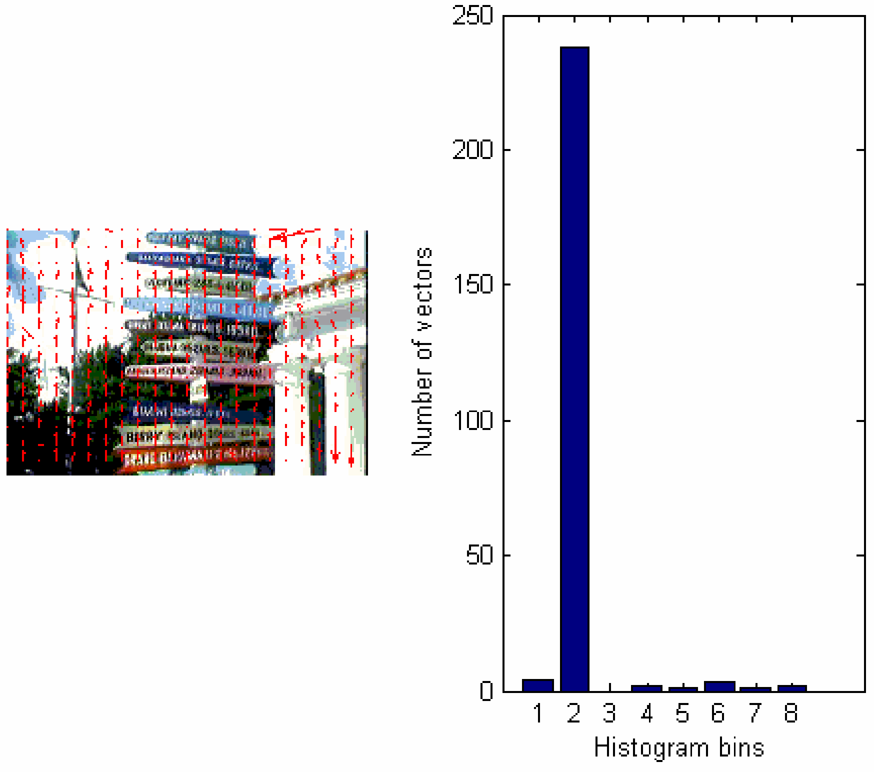
The MPEG compression algorithm for P and B frames uses the block matching method to compute a motion vector. This is done by finding the best matching MB in the current frame regarding the reference frame. If a matching MB is not found then it is intracoded and no motion vector is assigned to it. If the MB has not moved at all then it is marked as skipped and the motion vector that is assigned to it is 0. While this method performs well in terms of visual quality, it can provide us false movement information primarily due to noise. For example a MB may have not moved at all but in the presence of noise the algorithm will assign a vector to it. These cases should be taken under consideration as they can have a great impact in the analysis process.

6.1.1 Construction of the direction histogram

Each motion vector is described by two numbers *(u, v)* showing the horizontal and vertical displacement respectively, of the corresponding MB. We compute the angle of every vector in the frame as

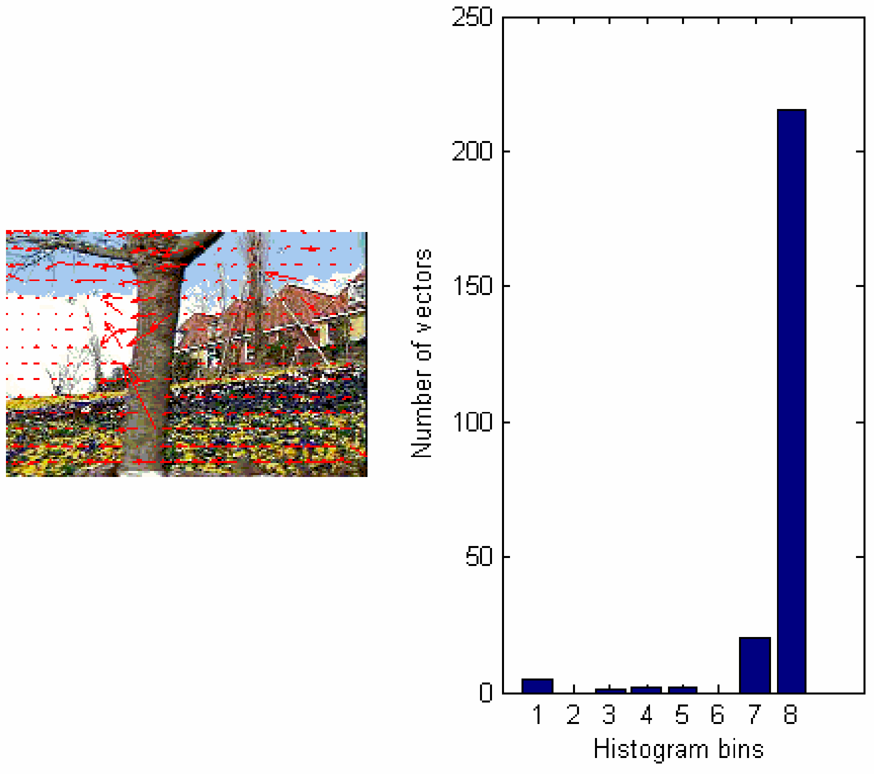
$$\theta = \tan^{-1}\left(\frac{v}{u}\right)$$

We end up with a set of angles for every frame in the range of $[0,2\pi]$. The angle for each vector can take a large number of values. Since we are interested in describing and analyzing the global motion in a frame, making calculations with every angle as it is can result to false conclusions. This is due to the fact that although many motion vectors may be describing the same movement of an object, for example, there could be small variations in the direction between them as an effect of noise. By quantizing them into a smaller set of values we achieve a better description of global motion and also reduced complexity in the calculations. That is why they are quantized into eight values which are the range of $[0,2\pi]$ divided every $\pi/4$ sections as shown in Figure 6.3. They are then placed in an eight bin histogram, with each bin representing $\pi/4$ radians. After the above process we end up having a direction histogram, one for every frame in the stream which gives us a good description about the motion that exists in it. Figure 6.4 shows an example of different histograms obtained from various streams and frames.



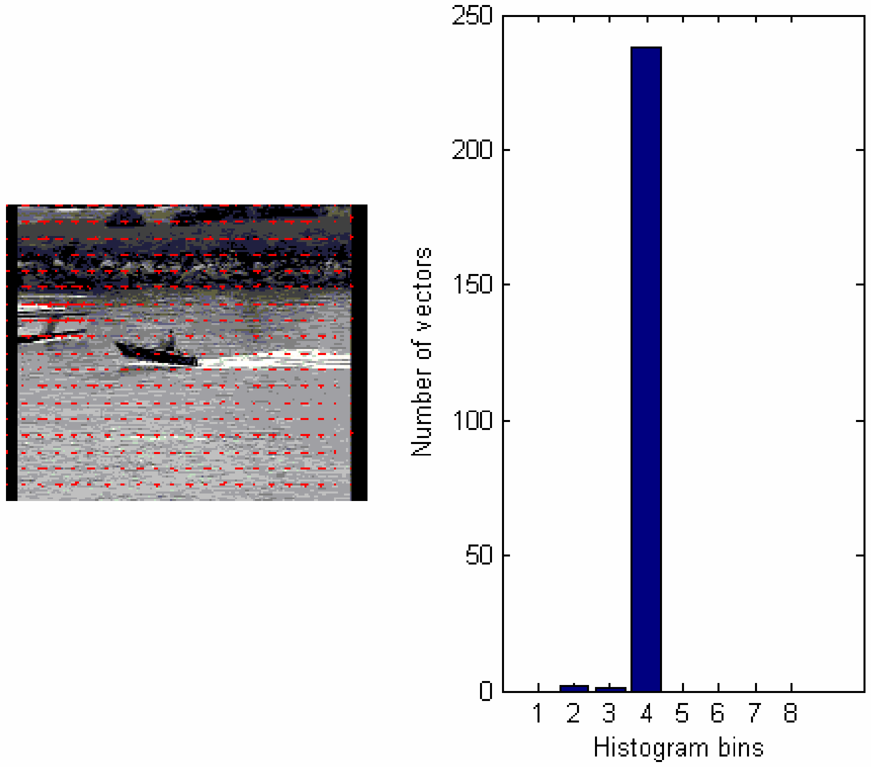
250
200
150
100
50
0
Number of vectors
1 2 3 4 5 6 7 8
Histogram bins

(a)

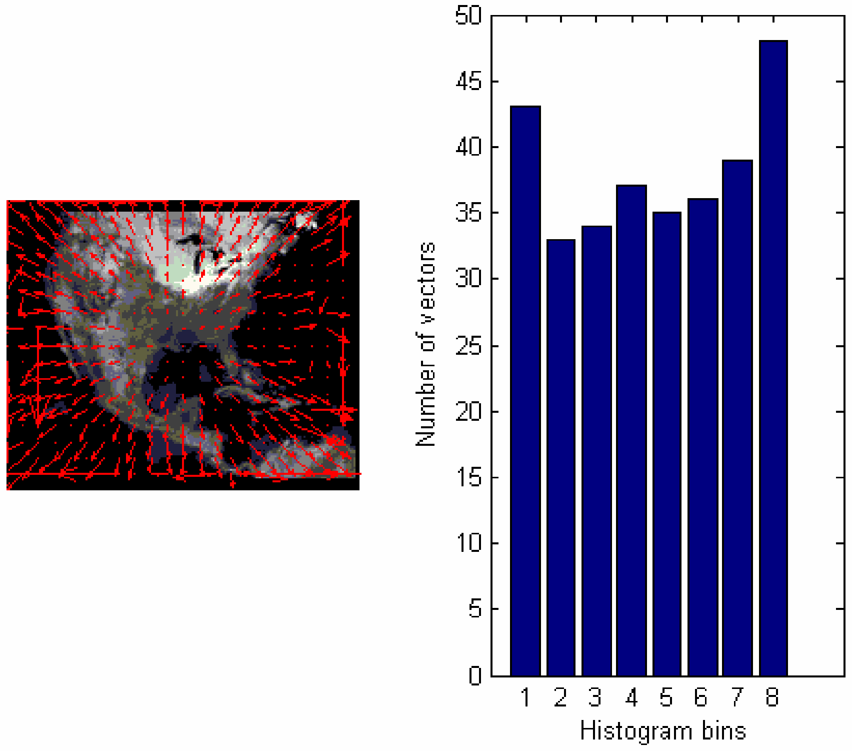


250
200
150
100
50
0
Number of vectors
1 2 3 4 5 6 7 8
Histogram bins

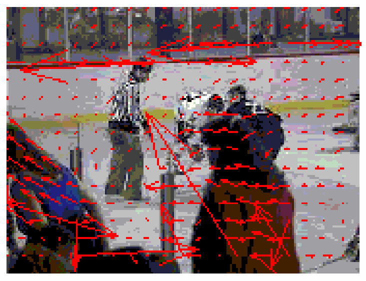
(b)

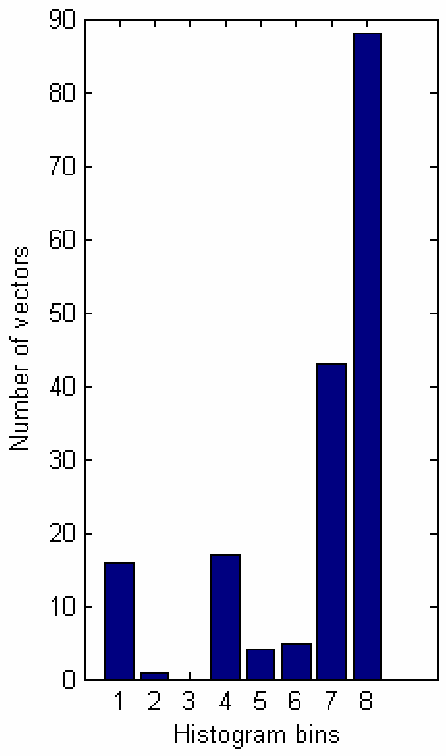

250
200
150
100
50
0
Number of vectors
1 2 3 4 5 6 7 8
Histogram bins

(c)


50
45
40
35
30
25
20
15
10
5
0
Number of vectors
1 2 3 4 5 6 7 8
Histogram bins

(d)





(e)

Figure 6.4: various frames and the corresponding histograms (a) tilt, vertical camera movement, (b) right pan, (c) left pan, (d) zoom in, (e) moving objects

It is clear from Figure 6.4 that during a pan most motion vectors are gathered in one bin which shows the direction of the movement while in a zoom they are spread equally along the bins. This is reasonable because when the camera is zooming there is a center of focus and the vectors are pointing towards or outwards this center depending on the movement. Static scenes with various objects moving show a different distribution in the histogram which is neither uniform nor gathered in one bin.

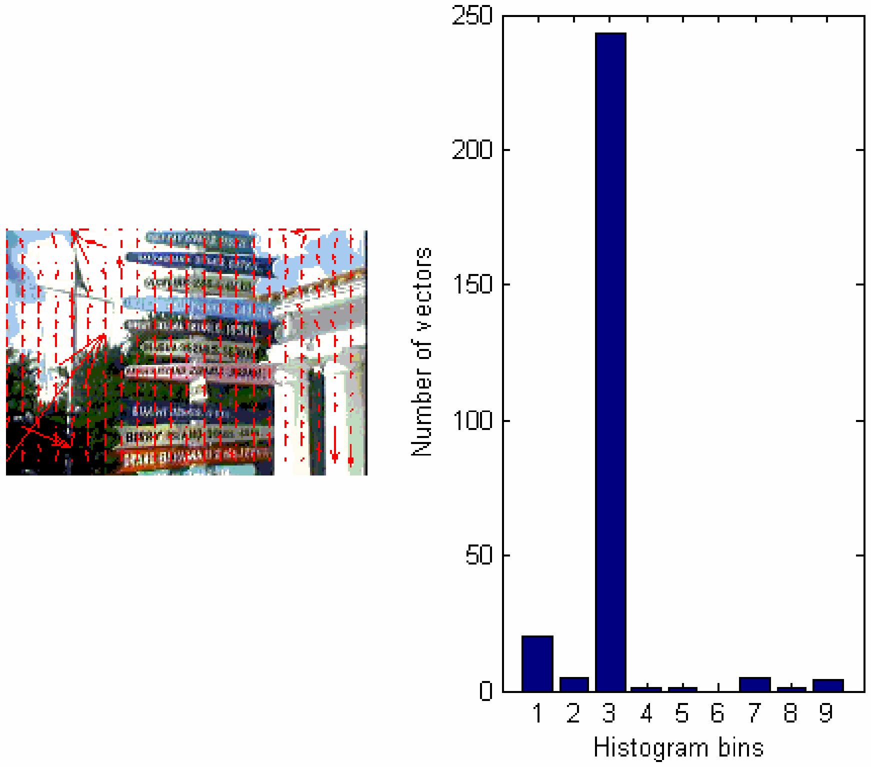
In the construction of these histograms only vectors with a relatively large value of magnitude were used. To be more precise a threshold was put on the length of the motion vectors to be taken under consideration. This threshold was set to

$$T_L = 1$$

Only vectors with length greater than T_L were taken under consideration. This was done because some MB are skipped and as a result their motion vector is 0. They don't have an angle as their coordinates u,v are 0. But also vectors with small magnitude, smaller than T_L, provide no essential information for motion and in many cases the information they provide is false as such little displacement should not be taken under consideration. They usually appear in static scenes where the camera is still and they are a product of noisy input. Their length might be small but because we examine only the direction their presence in the histogram would result in misjudgments.

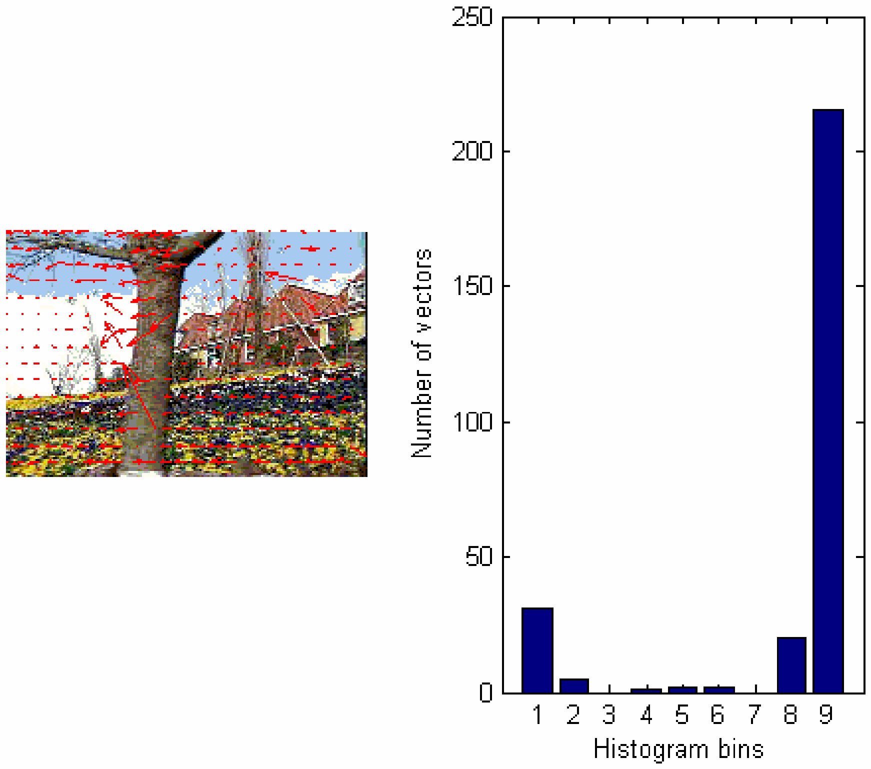
An important fact to be taken under consideration is that not all frame histograms in the video have the same number of motion vectors. In chapter 4 it was mentioned that some flow vectors would be marked as undefined because the algorithm could not correspond them to any MPEG motion vector (backward or forward predicted). This is the case where between two frames there is a cut and no match was found or even between consecutive frames in the same scene where a MB might be assigned a flow vector but in the next frame the corresponding MB might be marked us undefined as no match was found. This is a common case in MPEG and is one of the major problems in motion analysis as it generates inconsistencies in the smooth movement of objects and scenes.

By taking under consideration that vectors with magnitude zero or smaller than T_L actually describe the absence of motion in a frame it is clear that they should be used in the construction of the direction histogram. They are all gathered in one separate bin which will describe MBs that are not moving or do not change through time. The result of this addition is shown in Figure 6.5 for the same frames that are shown in Figure 6.4.



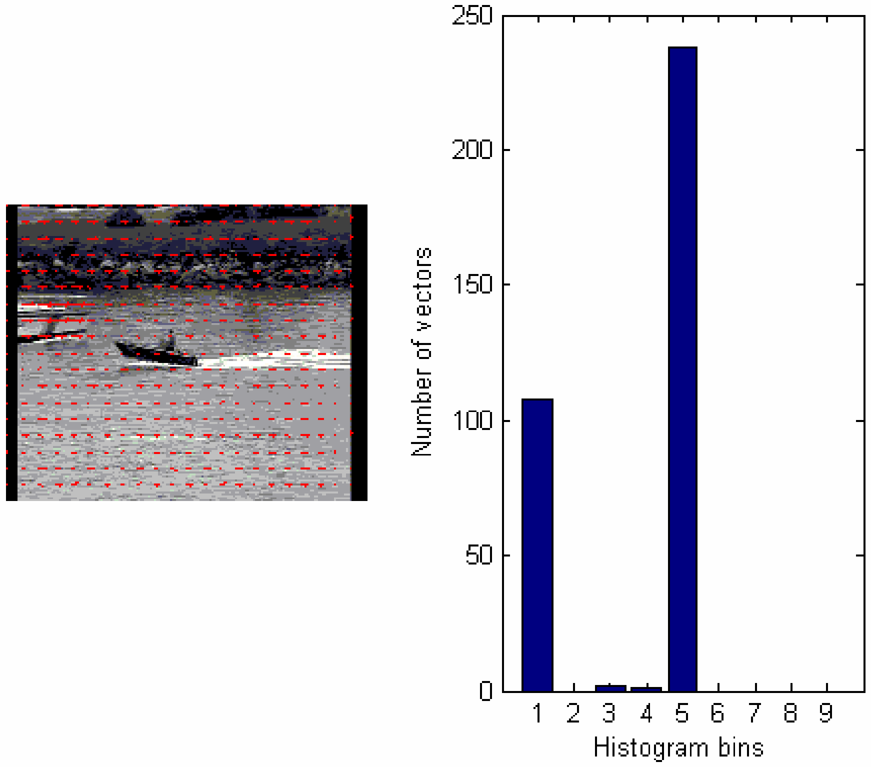
Number of vectors
Histogram bins

(a)

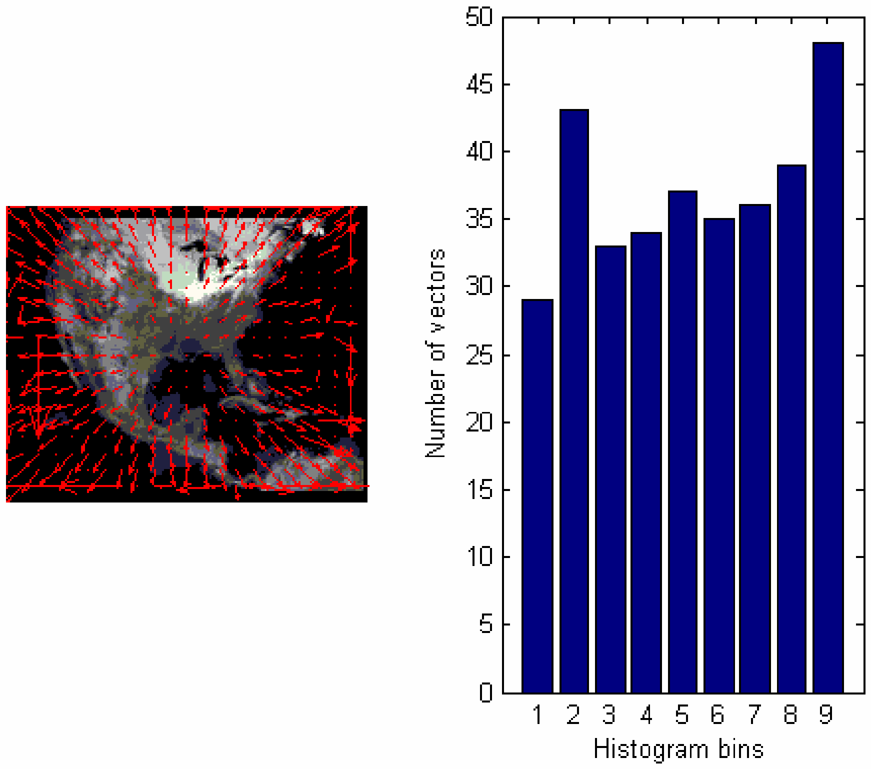


Number of vectors
Histogram bins

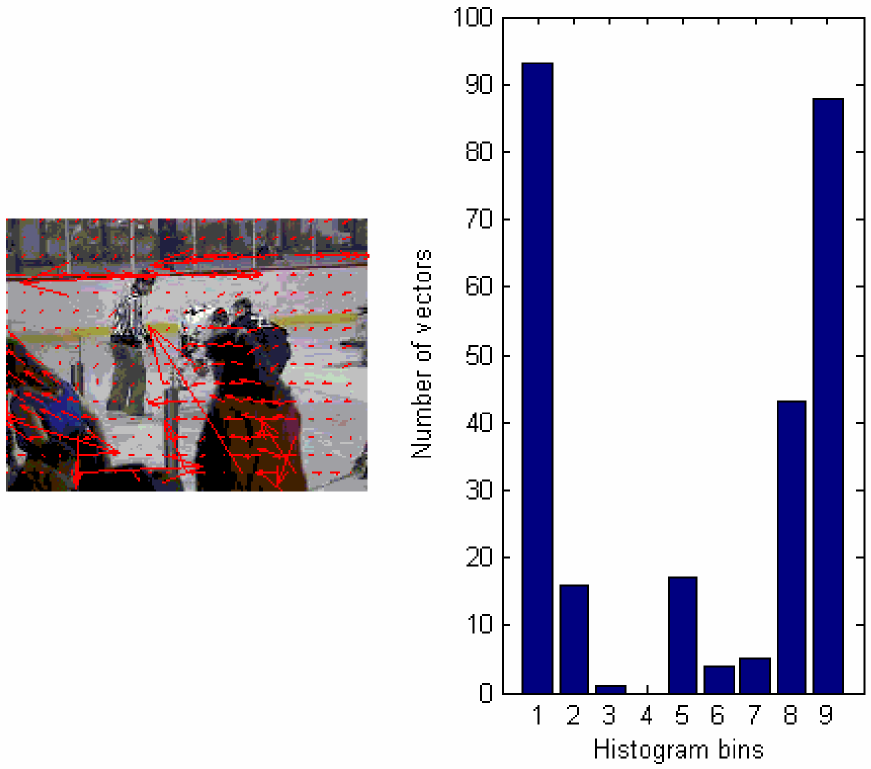
(b)


250
200
150
100
50
0
Number of vectors
1 2 3 4 5 6 7 8 9
Histogram bins

(c)


50
45
40
35
30
25
20
15
10
5
0
Number of vectors
1 2 3 4 5 6 7 8 9
Histogram bins

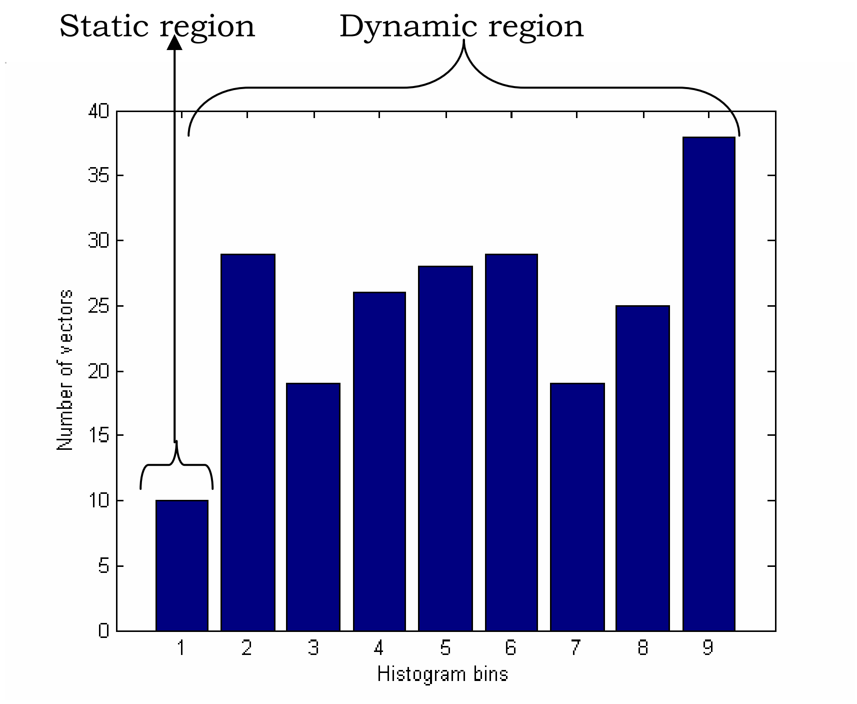
(d)



(e)

Figure 6.5: The histograms shown in Figure 6.4 but now have a separate bin for small vectors

By using the bin that describes the absence of motion in sections in a frame, the histogram is divided into two regions, the dynamic which contains information about the true motion and the static region which shows us how many MBs are skipped and are not changing through time.

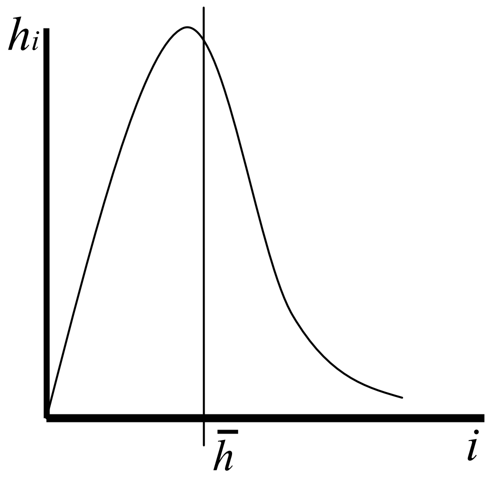


The direction histogram provides a powerful tool in describing camera motion as it gives certain forms of histograms for different kinds of effects. We can see that the bins are most gathered in one bin when we have a pan or tilt, which is the bin of the direction that the camera is advancing. In a zoom sequence the bins are spread as the vectors tend to point to the center of the picture or emerging from it for a zoom out and zoom in respectively. It should also be noticed that there are few small length vectors when we have camera movement, as most of the MBs are predicted from previous or next reference frames. When the camera is still small vectors are many in number and the bins do not have a certain type of distribution. This is strongly depended on the content of the video and the objects that exist in it. This can be made clear from the *(e)* plot of Figure 6.5.

6.1.2 Variance of the histogram

This special form that the direction histogram for different types of camera movement, can be exploited using special tools, to give us the information we need for the classification process. The tool that we use here is the variance of the histogram.

Suppose we have a histogram H with K bins and we want to compute the variance along the vertical axis. We first normalize every bin value so that the sum of K bins will be equal to 1. We end up with



$$p_i = \frac{h_i}{\sum_{i=1}^{K} h_i} \text{ with } i=1,...,K$$

It is clear that $\sum_{i=1}^{K} p_i = 1$. After that we find the mean value along the vertical axes which is

$$\bar{h} = \sum_{i=1}^{K} p_i \cdot i$$

The variance is then defined as

$$\sigma_h^2 = \sum_{i=1}^{K} p_i \cdot (i - \bar{h})^2$$

The minimum variance is expected when the histogram is gathered in one bin, that is the case where we have only one large peak and the rest are zero. This happens because the mean of the histogram will be the peak and the rest p_i will be zero. The sum of these two factors will give a result of 0.

The maximum variance is expected when the bins are evenly spread along the histogram and the p_i value is equal for every bin, $p_1 = p_2 = ... = p_K = \frac{1}{K}$. The mean value will be $\bar{h} = \frac{K}{2}$. The maximum variance is then

$$\sigma_h^2 = \frac{1}{K} \sum_{i=1}^{K} (i - \frac{K}{2})^2$$

After the above analysis and taking under consideration that the direction histogram in a pan sequence is actually a peak and in a zoom sequence it spread in all the bins then by taking the variance of it we can tell what kind of sequences exist in video shot.

To make this analysis we only make calculations on the dynamic region as we are interested only in the motion characteristics of the frame. Figure 6.6 shows the result of the plot. What we have achieved is to describe a whole frame with one number, the variance outcome.

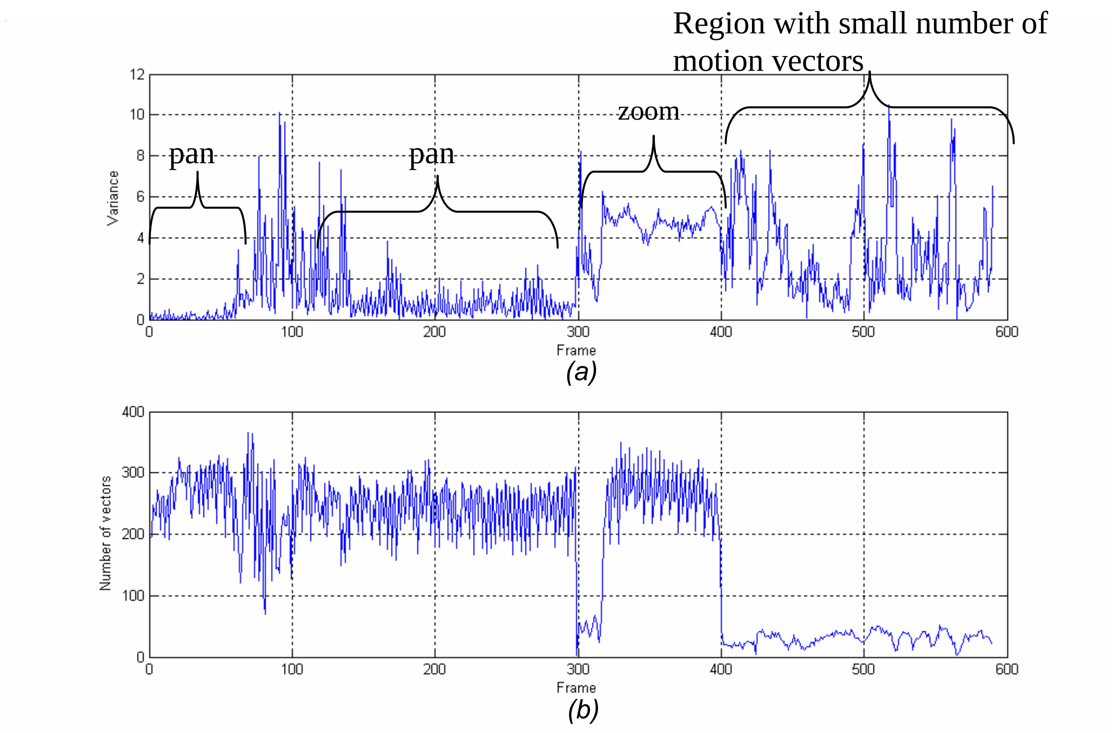


Figure 6.6: (a) Variance plot of a sequence with 2 pans and a zoom. We can see that the region where there is not a large number of motion vectors noise and large variations in the overall variance plot, (b) Number of vectors for the video.

It can be seen that our assumption was right. Pan sequences do give a small variance value and the zoom sequence that is shown in Figure 6.6 has a large variance for the duration of the effect. It is clear however that there are some problematic regions in the plot where we have a large variance value but there is no zooming. This happens because in that area the camera is still. There are little motion vectors and when these frames are used in the analysis, they provide false results that are not reliable for the extraction of information for global motion as it can lead us to conclusions that are wrong.

Because not all frames have the same number of motion vectors special treatment must be applied.

Since we do not want frames with small movement information to contribute much in the variance we introduce the parameter α which is defined as

$$\alpha = 1 - \frac{Undefined + Static}{N} = 1 - \frac{U + S}{N}$$

where U is the number of undefined flow vectors, S is the number of skipped or small motion vectors (the first bin of each histogram) and N is the number of MBs in a frame which is constant for the video. For every frame it applies that

$$U + S \leq N \qquad \text{and}$$

$$0 \leq \alpha \leq 1$$

When U or S are high α will be closer to 0 and when U or S have a low value α will be closer to 1.

We then multiply the variance outcome of each frame with this parameter. So we have

$$\sigma_{h,norm}^2 = \alpha \cdot \sigma_h^2$$

The result will be suppressed variance value for frames with small number of motion vectors and variance close to σ_h^2 for frames with adequate number of motion vectors. The result of this multiplication is shown in Figure 6.7. The region with inadequate motion information has now reduced $\sigma_{h,norm}^2$ which is exactly what we wanted, while useful data for the zoom sequence are retained. Observations about the type of camera movement are now much easier to be extracted.

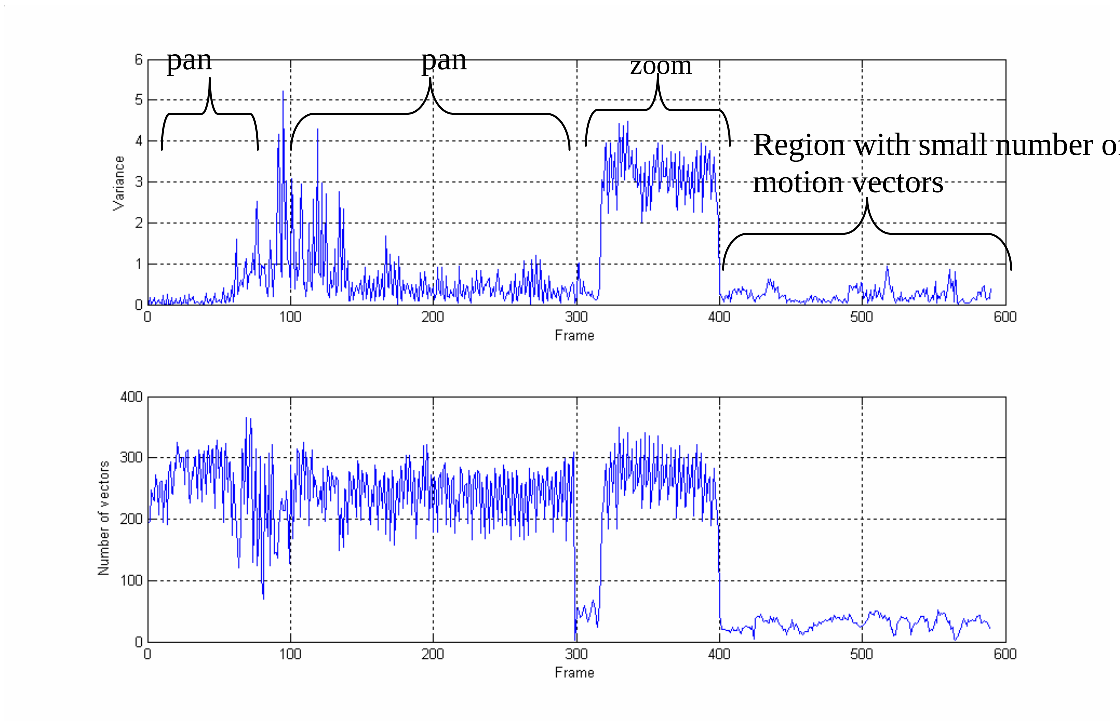


Figure 6.7: Normalized variance

The next step in the detection of camera movement is applying the appropriate threshold.

6.1.3 Selection of thresholds

We shall use the thresholds described in the previous chapter, the sliding window and the adaptive threshold. These threshold are applied in camera motion classification because the techniques used for deriving the histograms and variances for them are similar to the ones used in scene change detection for intensities but also because we want to have a more unified approach to scene segmentation for motion and intensities.

From Figure 6.7 we can see that while the variance for a pan scene is small, when we proceed to a zoom scene it changes abruptly. Below is the presentation of the thresholds but now the terms are mentioned in motion measures.

6.1.4 Sliding window threshold for variance

We construct a window with size 15. This means that 15 values of variance may be in the window in every moment. Let $T_w(i)$ be the threshold value, $\mu_w(i)$ and $\sigma_w(i)$ be the mean and variance for the data $v(i)$ inside the window. The variable i is in the range of $1 \leq i \leq Frames$, where $Frames$ is the total number of frames in the MPEG stream. We place the window on the variance data and we compute $\mu_w(i)$ and $\sigma_w(i)$. Thus $T_w(i)$ becomes $T_w(i)=\mu_w(i)+\alpha_w\sigma_w(i)$. The parameter α_w is constant in the whole process. Typical values range between 3.5 and 6. We then compare $T_w(i)$ with the next variance value outside the window. This means that suppose we have inside the window the variances [$v(k),...,v(k+14)$] and the threshold is $T_w(k+14)$ then we compare it with $v(k+15)$. If $v(k+15)$ is smaller than $T_w(k+14)$ then $v(k)$ is removed from the window, $v(k+15)$ is placed and the new $\mu_w(k+15)$ and $\sigma_w(k+15)$ are computed. The threshold becomes then $T_w(k+15) = \mu_w(k+15) + \alpha_w\sigma_w(k+15)$. If $v(k+15)$ is greater than $T_w(k+14)$ then the threshold remains constant to $T_w(k+14)$ and the window just progresses until there is a value, say $v(l)$, which is smaller than the threshold. In that case the previous process is continued.
This goes on for the whole length of the video. The algorithm can be described in steps:

8. Construct a window with the first 15 histogram variance $v(i)$ values
9. Calculate the mean μ_w and variance σ_w for the window
10. Calculate the threshold T_w as $T_w= \mu_w+ \alpha_w\sigma_w$
11. Compare T_w with the next histogram variance value $v(k)$ outside the window
12. If $T_w>v(k)$ then add $v(k)$ in the window, discard the first value of the window and compute the new mean and variance for it. Calculate the new threshold $T_w^{'} = \mu_w^{'} + \alpha_w\sigma_w^{'}$.
13. if $T_w<v(k)$ do not do any calculations. Just move the window to the right until a histogram variance value $v(l)$ is smaller than the threshold T_w. Then goto step 5.

14. Repeat for all the frames in the stream.

It should be cleared out that for the first 14 values of the histogram variances the threshold is 0 and no comparison is made. This is done because the window needs to have 15 data instances to operate. So $T_w(i) = 0$, for $0 \le i \le 14$. The thresholding scheme is shown in Figure 6.8.

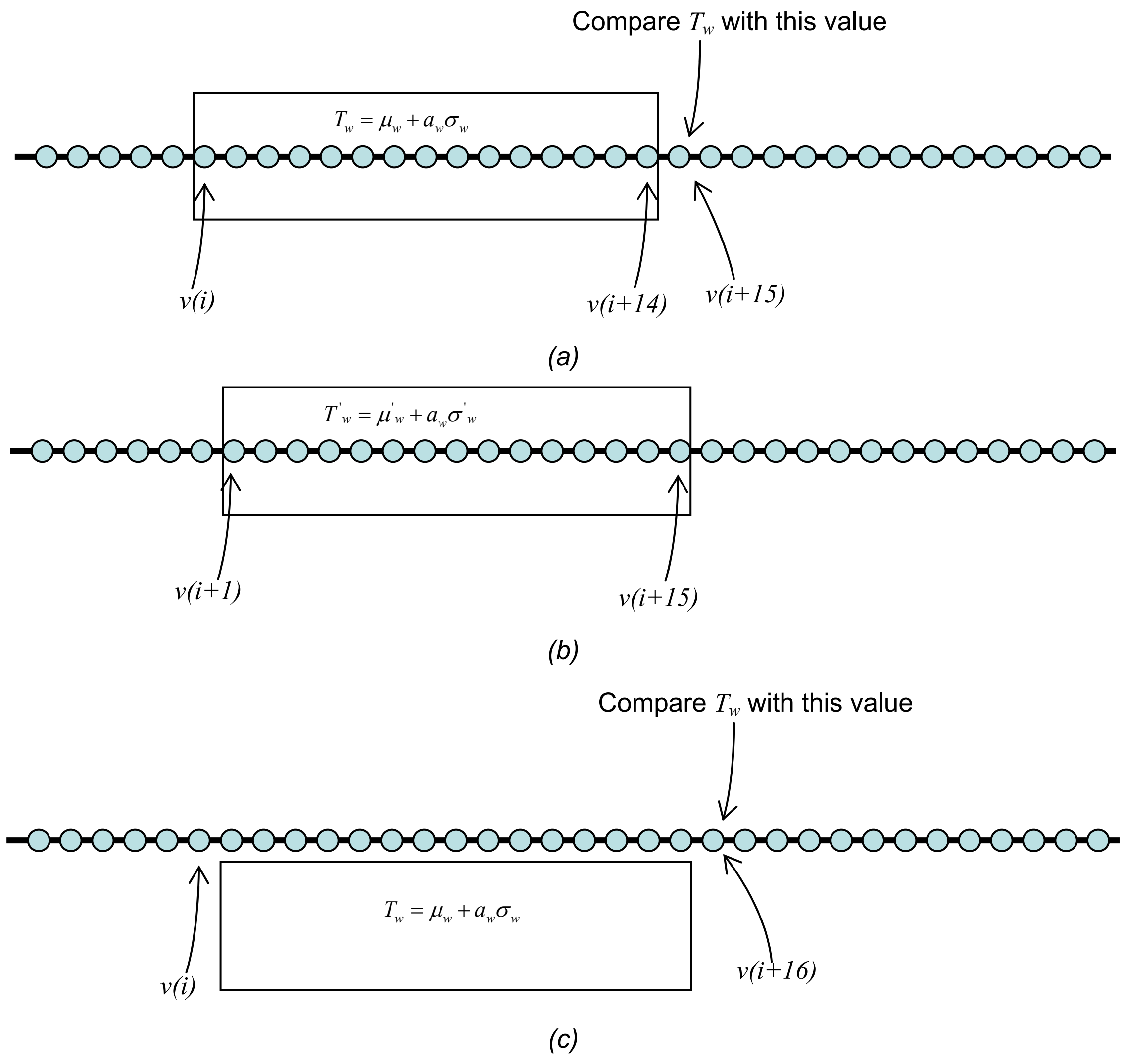

Compare T_w with this value
$T_w = \mu_w + a_w \sigma_w$
v(i)
v(i+14)
v(i+15)
(a)
$T'_w = \mu'_w + a_w \sigma'_w$
v(i+1)
v(i+15)
(b)
Compare T_w with this value
$T_w = \mu_w + a_w \sigma_w$
v(i)
v(i+16)
(c)

Figure 6.8: (a) An instance of the window. The threshold is compared with the data outside the window.
(b) The threshold is larger than the data so the window moves to the right by discarding the first data v(i) and inserting the last v(i+15). The new threshold is calculated.
(c) If the threshold is smaller that the data in (a) then no other data is entered in the window and the threshold is unchanged. It moves to the right until The threshold meets a smaller data value.

The full process of the thesholding is shown in Figure 6.9 when we apply it on the data from Figure 6.7.

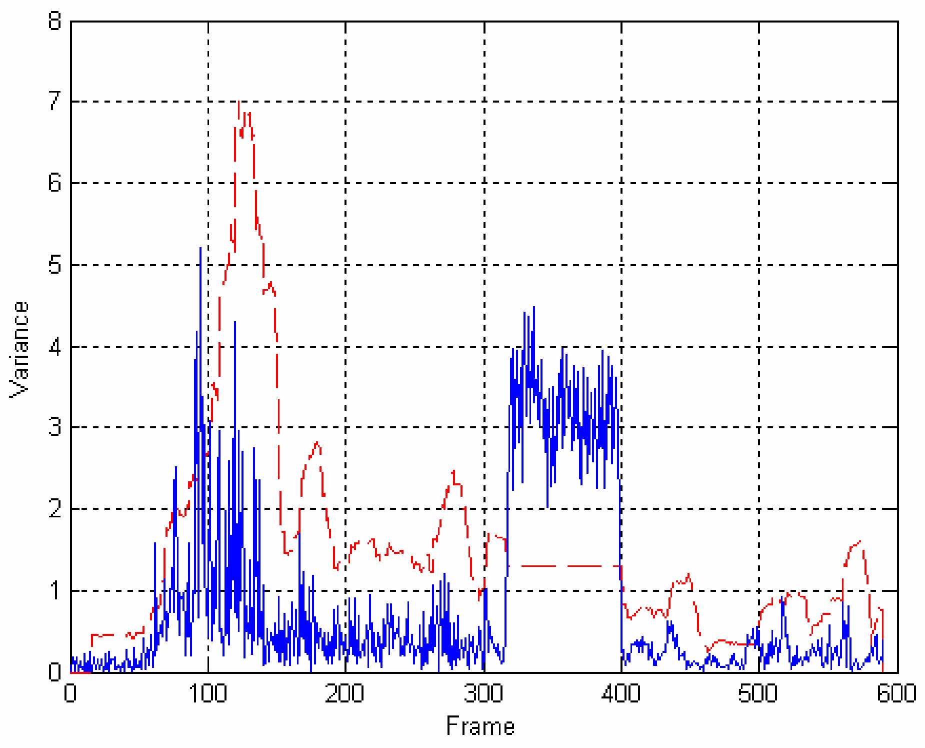


Figure 6.9: The sliding window threshold shown in dashed line. Notice how it stays constant when it enters the zoom sequence

6.1.5 Adaptive threshold

Three quantities are used for the computation of the threshold, namely μ_a, λ_a and σ_a where μ_a is the adaptive mean, λ_a is the adaptive second moment and σ_a is the adaptive standard deviation. The formulas for each quantity are:

$$\mu_a(i) = \mu_a(i-1) - c \cdot \left(\mu_a(i-1) - v(i)\right) \qquad \text{(Eq. 1)}$$

$$\lambda_a(i) = \lambda_a(i-1) - c \cdot \left(\lambda_\alpha(i-1) - v(i)^2\right) \qquad \text{(Eq. 2)}$$

$$\sigma_a(i) = \sqrt{\left|\mu_a(i)^2 - \lambda_a(i)\right|} \qquad \text{(Eq. 3)}$$

with

$$\mu_a(1) = v(1)$$

$$\lambda_a(1) = v(1)^2$$

$$\sigma_a(1) = \sqrt{\mu_a(1)^2 - \lambda_a(1)} = 0$$

The threshold is computed as

$$T_a(i) = \mu_a(i) + a_a \cdot \sigma_a(i)$$

where a_a is a user selected constant. It is applied on the variance in a similar way to the sliding window threshold.

For the first 15 values no comparison is made and the threshold is left to adapt to the variance signal. After these 15 values we compare the current value of the threshold $T_a(i)$ with the next variance value $v(i+1)$. If the latter is smaller than $T_a(i)$ then the new value of the threshold is computed from the above equations, $T_a(i+1) = \mu_a(i+1) + a_a \cdot \sigma_a(i+1)$.

If the variance value $v(i+1)$ is larger than $T_a(i)$ then threshold remains constant and $T_a(i+1) = T_a(i)$ for every consecutive variance value that is above $T_a(i)$ until a value $v(j)$ is smaller. Then the threshold becomes $T_a(j) = \mu_a(j) + a_a \cdot \sigma_a(j)$ where μ_a and σ_a are computed from equations (1) and (3). The process is shown schematically in Figure 6.10.

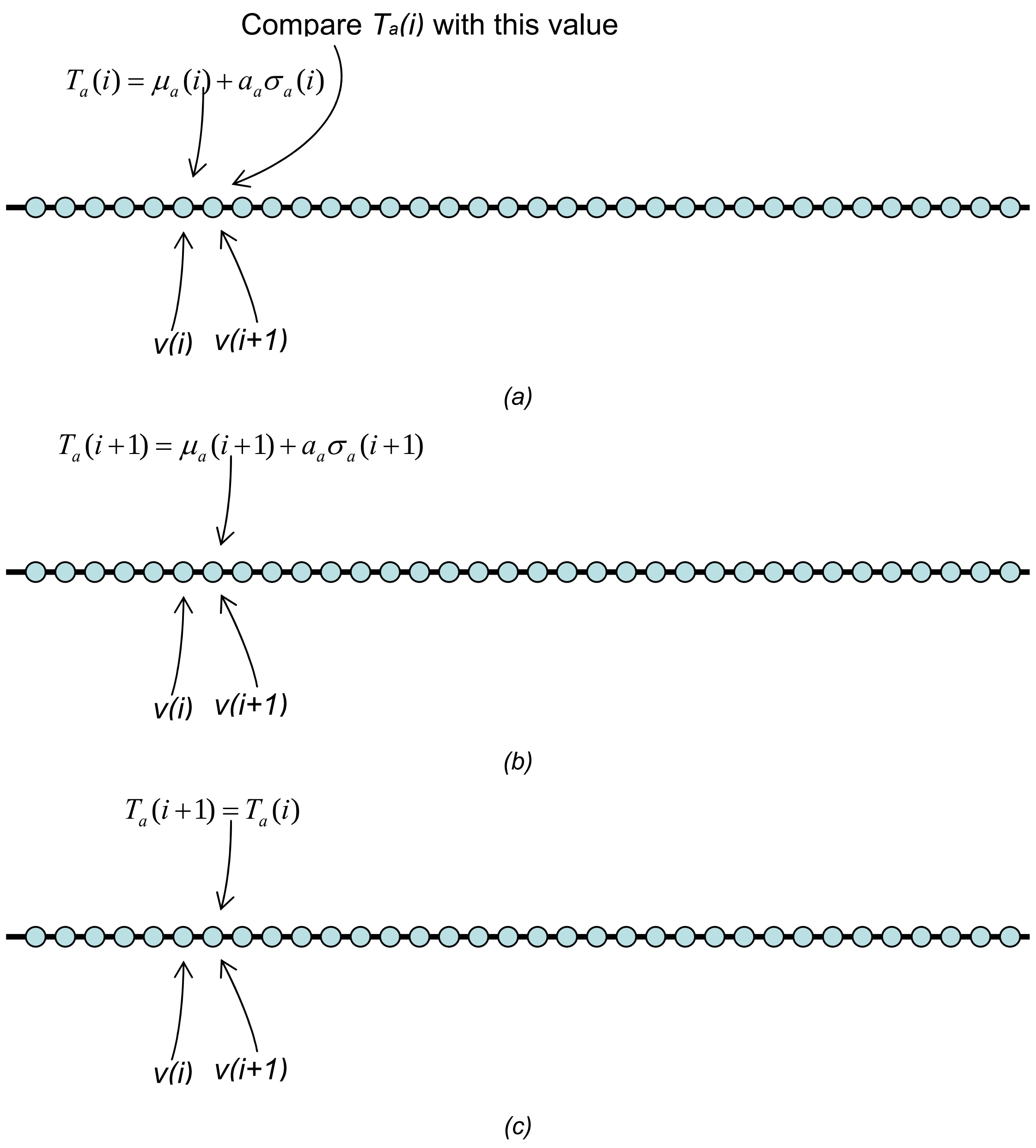

Compare $T_a(i)$ with this value
$T_a(i) = \mu_a(i) + a_a\sigma_a(i)$
v(i)
v(i+1)
(a)
$T_a(i+1) = \mu_a(i+1) + a_a\sigma_a(i+1)$
v(i)
v(i+1)
(b)
$T_a(i+1) = T_a(i)$
v(i)
v(i+1)
(c)

Figure 6.10 : (a) An instance of the threshold. It is compared with the data outside the window.

(b) The threshold is larger than the data so the new T_a value is computed based on the formulas

(c) If the threshold is smaller that the data in (a) then the threshold is unchanged. It moves to the right until it meets a smaller data value.

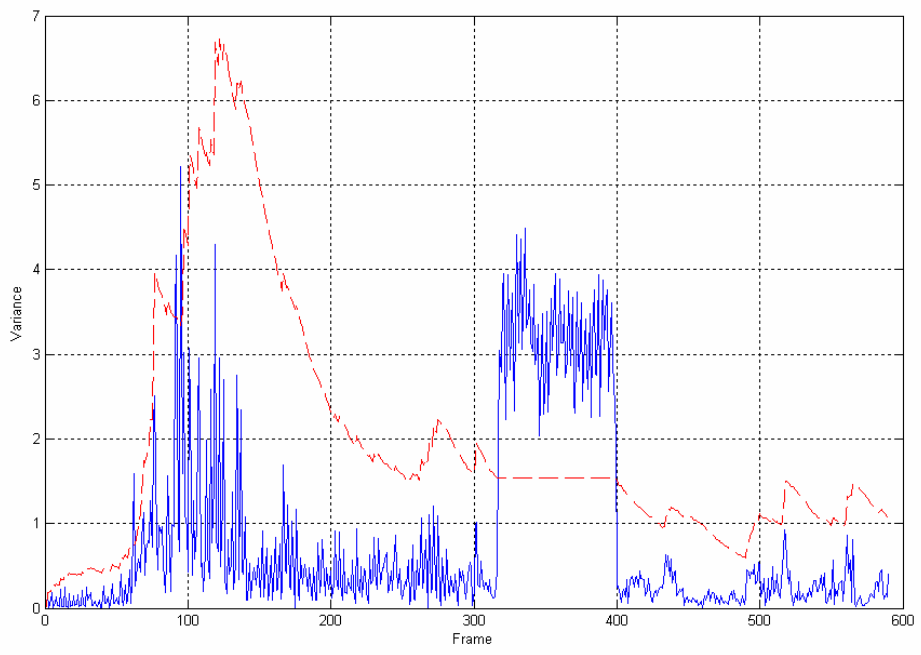


Figure 6.11: The appliance of the adaptive threshold. The threshold is plotted in dashed line.

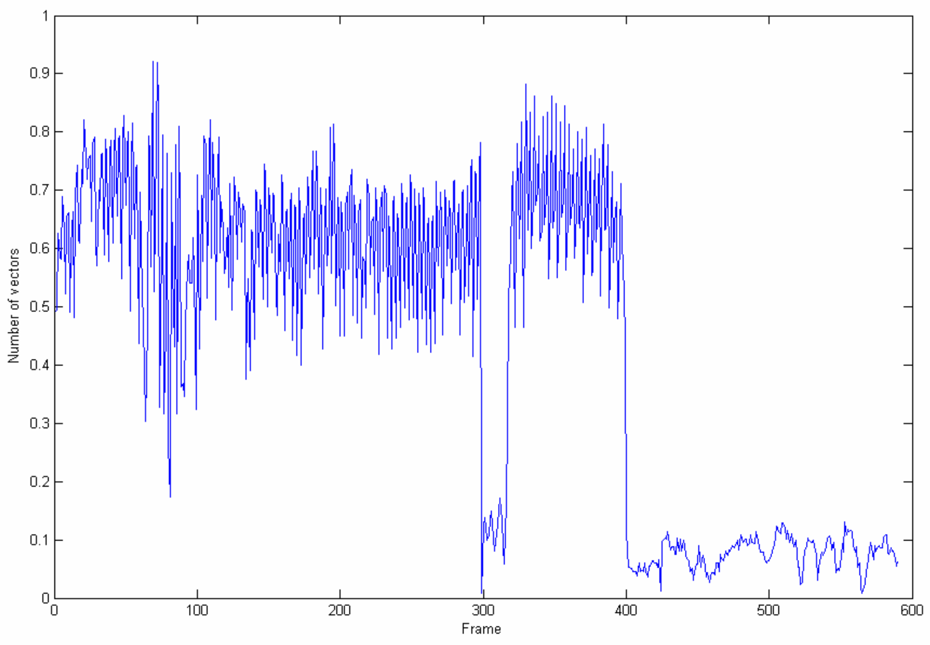
6.1.6 Decision for panning or zooming

Now that the threshold is defined we can proceed to making the decision if a sequence is a zoom sequence, a pan sequence or none of the two. To do this we need another information which is given directly from the analysis we have already made, and this information is the number of motion vectors that exist in the frame to be examined.

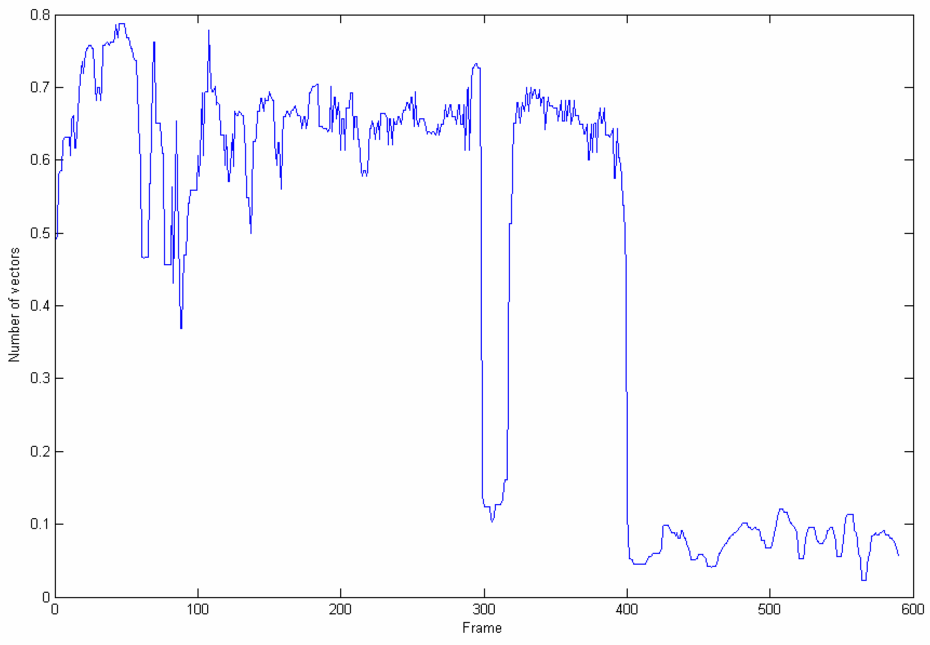
It is clear from Figure 6.2 that during camera movement, the motion is global, which means that it applies for all the MBs in the frame. So a large number of vectors is expected in these scenes. Through experimenting and analysis of various video types we have come to the conclusion that 40% of the total MBs in a scene are motion predicted then there is a good chance that this scene has global motion in it. So we set a threshold of

$$T_v = \frac{n_v}{n_{total}} = 0.4$$

where n_v is the number of vectors in the frame and n_{total} is the total number of MBs in a frame which is constant for the whole video.



(a)



(b)

Figure 6.12: (a) Number of vectors for the MPEG stream in Figure 6.7, (b) Number of vectors after applying a median filter of size 7

Figure 6.12 shows the number of vectors normalized to 1 for the video in Figure 6.7. Before doing any calculations a median filter was applied first to eliminate the rough peaks shown in Figure 6.12a. It can be clearly seen that in the pan and zoom sequence the total number of motion vectors is large for panning and zooming.

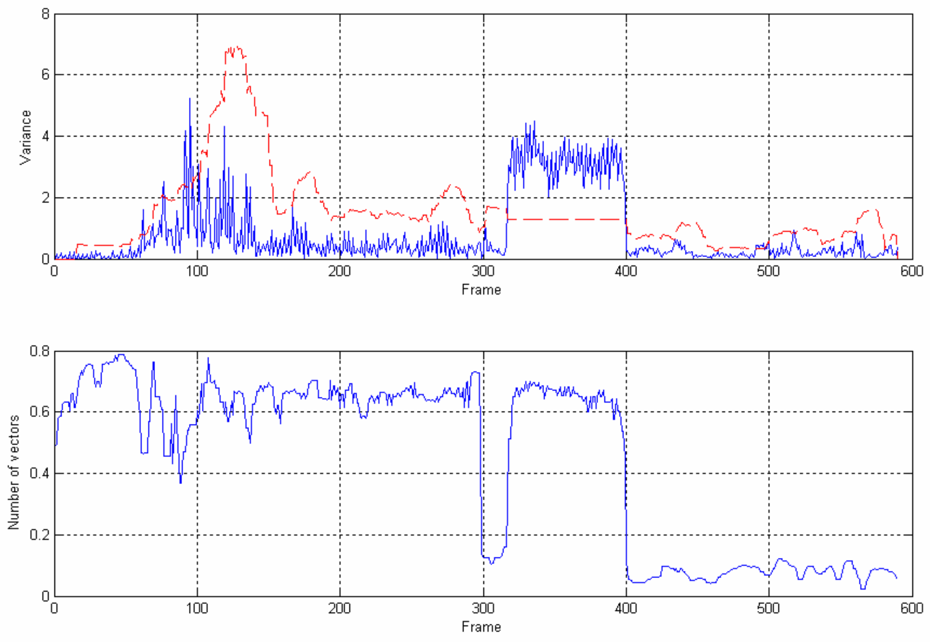


Figure 6.13: Combining the two clues, variance and number of motion vectors

By combining the data given by the variance of the histogram. the total numbe of motion vectors and the sliding window threshold (Figure 6.13), we can now decide about the type of camera movement.

If $\frac{n_v}{n_{total}} > T_v$ and the variance is below the threshold T_W then the frame is declared a pan frame.

If $\frac{n_v}{n_{total}} > T_v$ and the variance is above the threshold T_W then the frame is declared a zoom frame.

If $\frac{n_v}{n_{total}} < T_v$ then the frame is none of the above and we do not make any comparison.

Continuous frames tagged as zoom or pan frames make a zoom or pan sequence accordingly. Pans and zooms with duration smaller than 5 frames are discarded as false detection because the duration is too small for them to be real.

6.2 Detection of changes in camera direction

To detect the changes in camera motion a different approach is required. In a video stream the differences in frame-to-frame motion are small. This is because the motion content does not change so much in so little time (typically for a 30fps MPEG stream the time interval between two consecutive frames is 1/30 sec). This does not apply in the cases of sudden camera movement changes and scene breaks, although scene breaks can be classified in the first category. This means that the direction histogram will be very much alike for two consecutive frames unless we have the two cases described above. So the interframe difference of histograms will remain in small values but it will have peaks (large differences) in the frames where camera motion change exists.

In §6.1.1 we constructed a direction histogram for every frame in the stream. The same will be used for the detection of camera changes. But we cannot take the interframe differences as they are. The reason is that not all histograms have the same number of motion vectors. This depends on the number of undefined flow vectors that a frame has. So by taking the exact frame-to-frame histogram difference could result in false conclusions because if two consecutive histograms do not have the same amount of vectors then by taking their difference we will end up with a large value even if the two frames describe the same motion.

This is why we first normalize the histograms before taking their difference.

$$\overline{H_{f_n}} = \frac{h_i}{M_{f_n}}$$

where $1 \leq i \leq K$, K are the bins of the histogram, h_i is the i-th bin, M_{f_n} is the total number of motion vectors for the frame f_n and $\overline{H_{f_n}}$ is the normalized histogram. After this normalization

$$\sum_{i=1}^{K} \overline{h_i} = 1$$

So we will not actually compare the actual amount of motion vectors but the percentage of them in each bin.

If f_n is the current frame with normalized histogram $\overline{H_{f_n}}$ and f_{n-1} is the previous frame with $\overline{H_{f_{n-1}}}$ then we define the absolute interframe difference of histograms as

$$ID_H(n) = \sum_{i=1}^{K} \left| \overline{H_{f_n}}(i) - \overline{H_{f_{n-1}}}(i) \right|$$

where K are the bins of the histogram and n ranges from 2 to the total frames in the video.

Although in the computation of variance we considered only the dynamic region of the histogram, in the computation of ID_H we consider both the dynamic and the static. This is done because when a frame has a small number of motion vectors then because of the normalization, small changes in a small amount of data will result in large differences as shown in Figure 6.14. Notice how many large peaks the plot has. If the static region is used then in the normalization process the bins with few vectors will be suppressed and will not contribute much to the difference. In 6.14b the large peaks that were not due to change of camera movement have been eliminated.

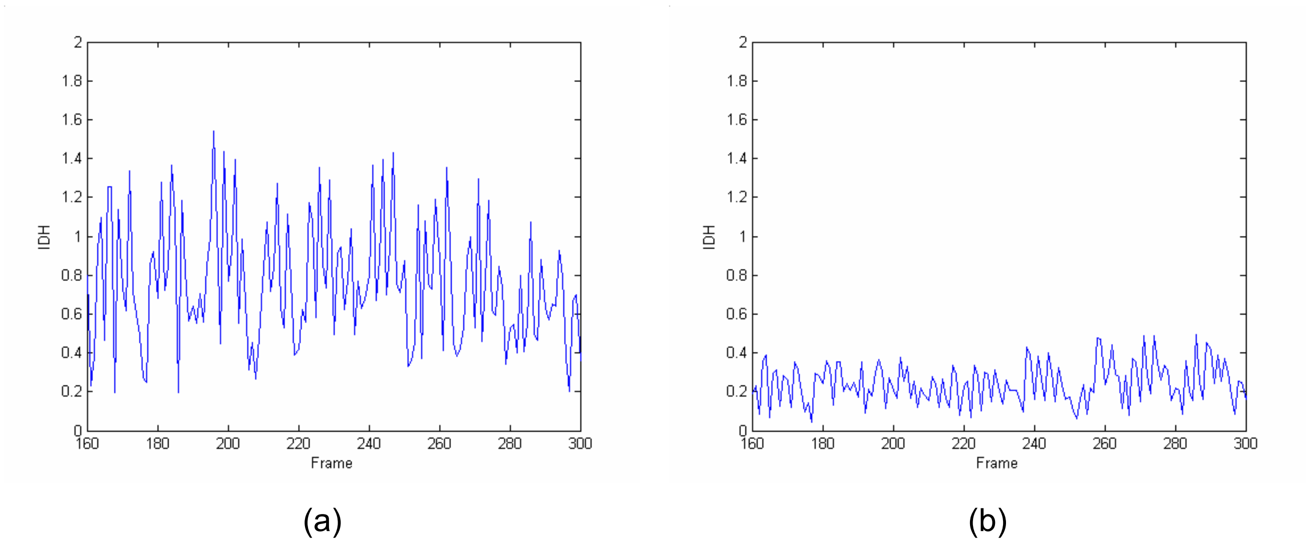


Figure 6.14: (a) Non-normalized interframe difference for a scene with a large static region, (b) The normalized interframe difference for the same scene.

So by using the frame-to-frame histogram difference we have the results shown below in Figure 6.15

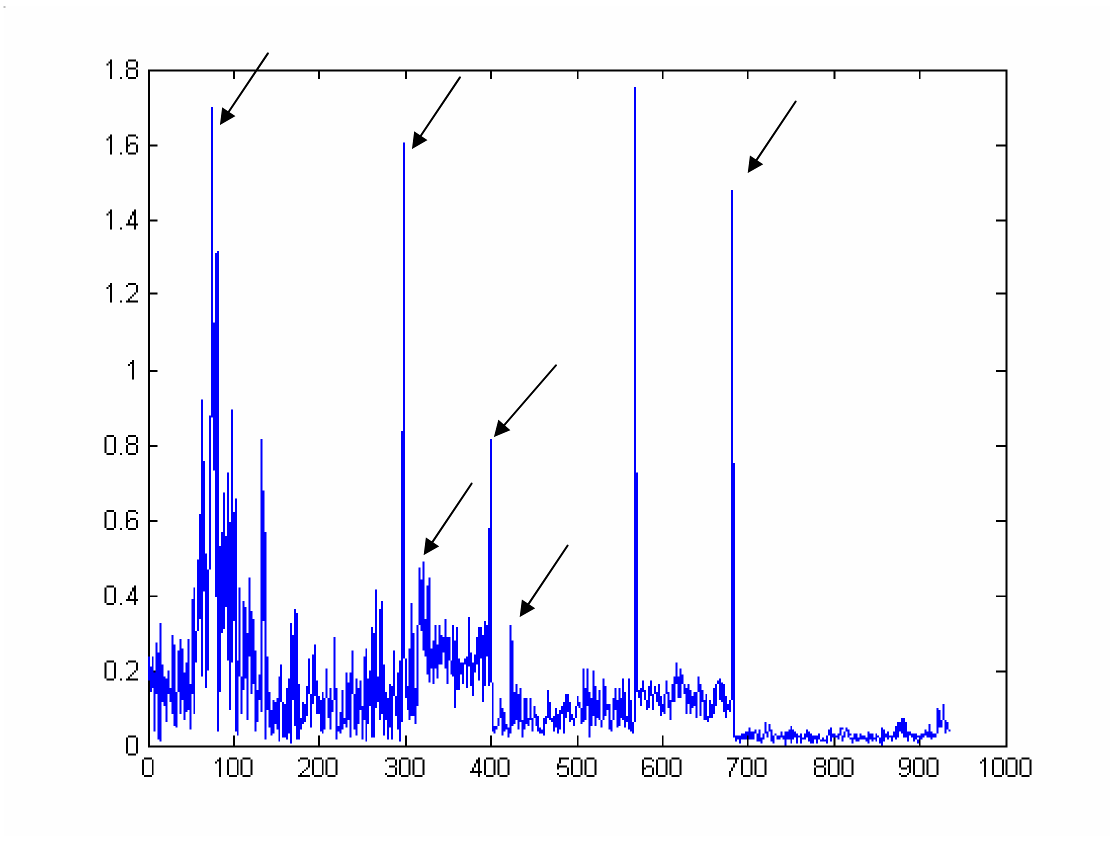


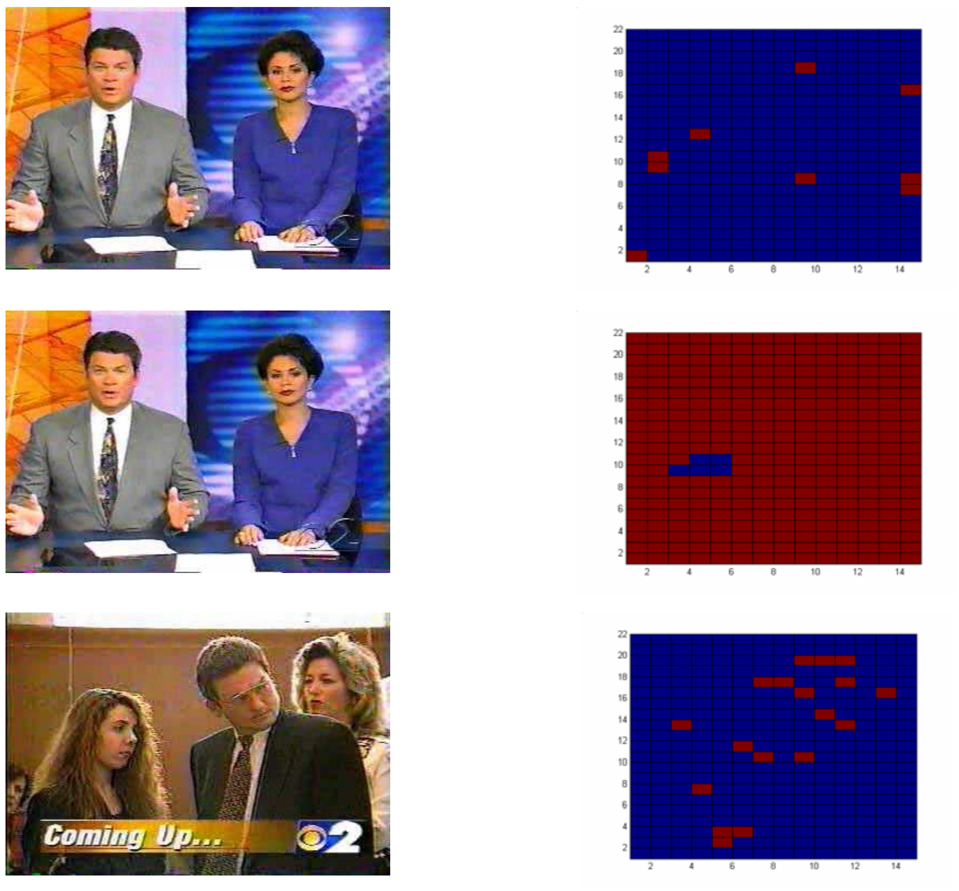
Figure 6.15: The interframe histogram difference of a sequence. The arrows indicate camera motion change

After the differences are acquired we apply on them the sliding window and adaptive thresholds.

6.3 Detection of scene breaks

From the analysis for the normalization of the motion vectors in the MPEG stream, it was explained that not all MBs in a frame will have a flow vector assigned to them. This is done when the algorithm cannot find a corresponding MPEG motion vector for the MB that it currently examines. The reason is that the MB in the frame to be analyzed could not be correlated to a corresponding MB of another reference frame. So it is intracoded.

So a thought is that the more undefined flow vectors are found in a frame, the chance of it being the start of a new scene is greater. Figure 6.16 shows three consecutive frames, where the middle one is actually the scene cut, and on the side is the distribution of the type of flow vectors.



(a) *(b)*

Figure 6.16: Column (a) shows the three consecutive frames and column (b) shows the type of flow vectors. Notice how the middle frame has many undefined vectors (red color) as no match could be found in reference to the next frame.

We can see that there truly is an increase in the amount of undefined flow vectors in the frame just before the cut. So if f_k, f_{k+1} are two consecutive frames with f_k belonging to one scene and f_{k+1} belonging to the next scene, then f_k will have a large number of undefined vectors. The plot of the number of undefined vectors for every frame in the MPEG stream will have large peaks which will indicate the change of scene (Figure 6.17).

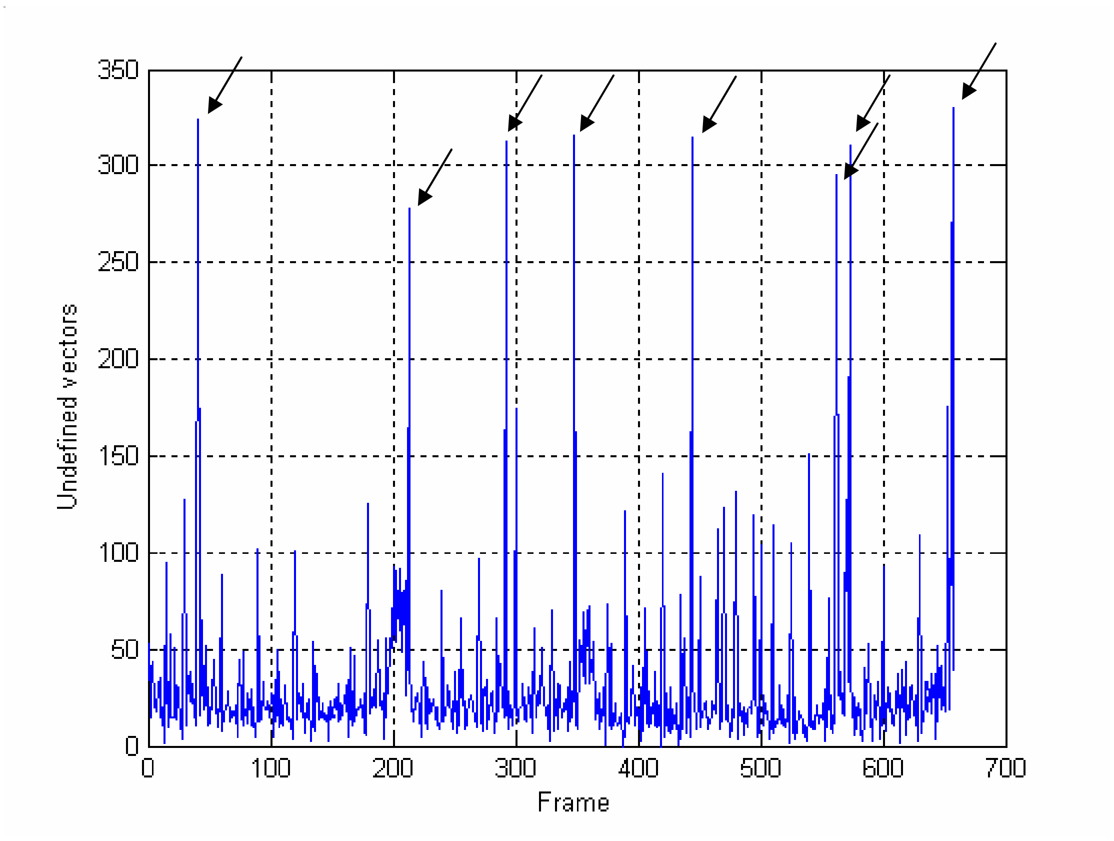


Figure 6.17: Undefined flow vectors for the frames in the video. Arrows indicate scene cuts.

Now we apply the thresholds that were described in §5.4 for detecting the peaks in the plot. If the number of undefined vectors exceeds the threshold then a cut is declared. This technique has shown to have great effectiveness as the cuts are very well defined in the plot.

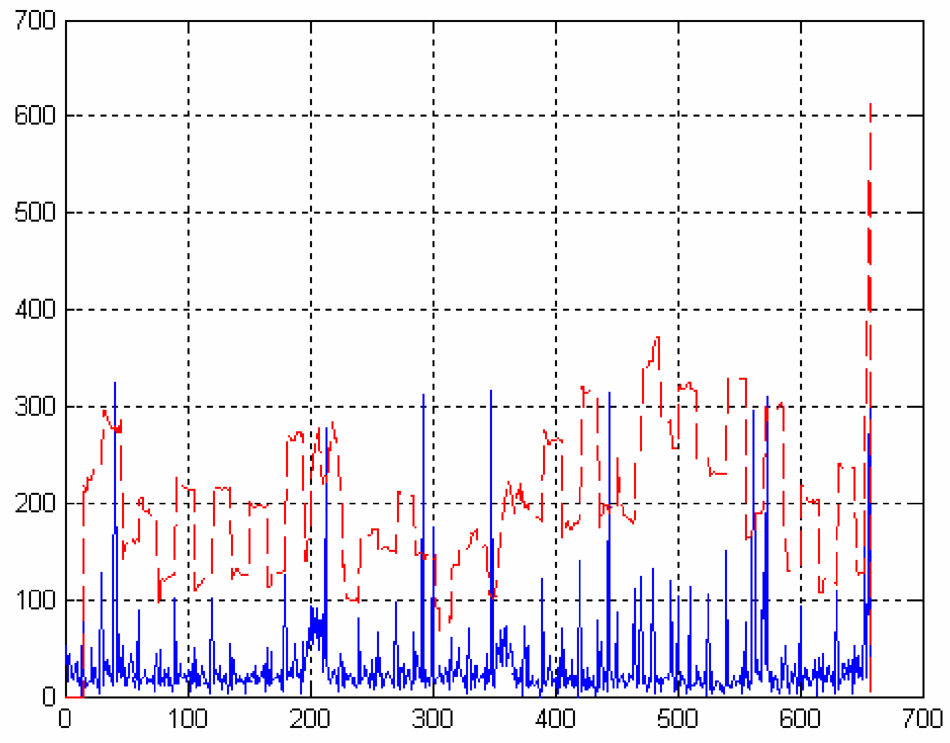


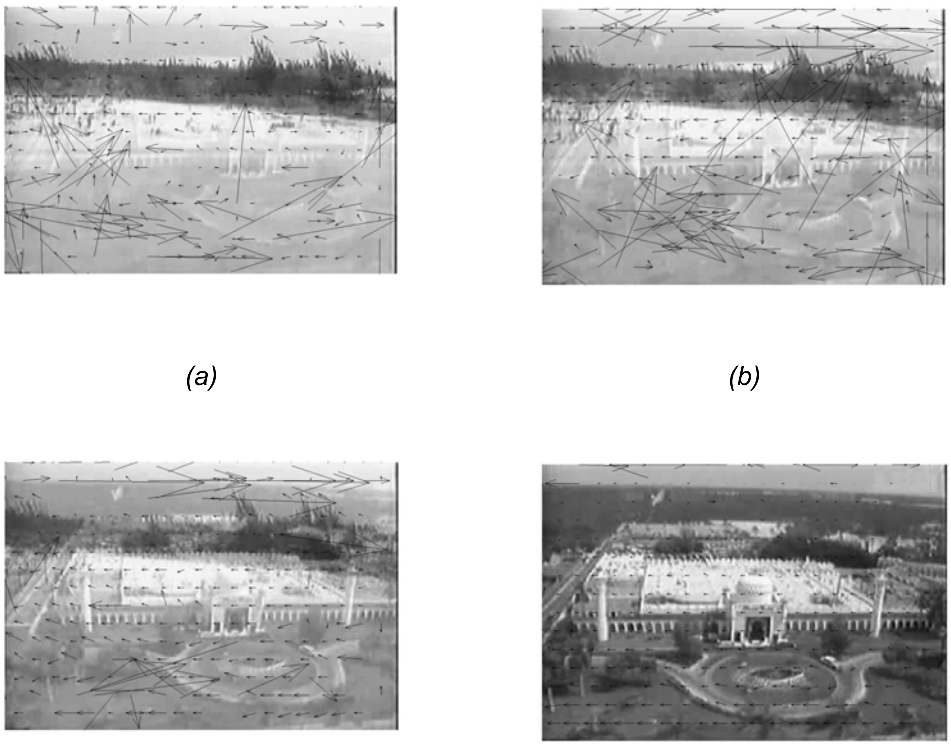
Figure 6.18: The appliance of the sliding window threshold, shown in the dashed line, on the plot of Figure 6.17. The peaks are well distinguished.

6.4 Detection of gradual changes

The detection of gradual changes is not an easy task. Because the transition from one scene to another is not abrupt, but is extended through time, following an approach as in §6.3 will not give good results as the number of undefined vectors will not present peaks.

During the dissolve, as the two scenes blend together, the picture is obscure. The matches that the MPEG finds change constantly through time and as a result the motion vectors behave randomly. So no true motion is described. This behavior of the motion vectors can be exploited to our advantage.

In the dissolve sequence the length of the motion vectors changes constantly and in the same frame it has many variations. This can be seen in Figure 6.18 where a dissolve is displayed.


(a) (b)

(c) (d)

Figure 6.18: Frames belonging to a dissolve sequence. During this gradual transition the motion vectors of a frame behave randomly, trying to find a match for the MB. As a result the variation of their length is great, and can be distinguished from the other scenes where the behavior is smoother.

The length varies from small to large. A dissolve can then be detected if the variance of the length of the motion vectors is measured for the frames in the video. The process is explained below:

We first calculate the length of every flow vector in the frame. Only those with $|v|>1$ are considered, the dynamic region explained in §6.1.1. Then we calculate the percentage of these vectors for the frame as

$$\alpha_v = \frac{n_v}{N},\ 0 \le \alpha_v \le 1$$

where n_v is the number of vectors for the dynamic region and N is the total number of vectors for the frame. N is constant for the video.

We then calculate the variance σ_l for the length of the motion vectors as

$$\sigma_l = \frac{1}{n_v} \sum_{i=1}^{n_v} (l_i - \bar{l})^2$$

where $\bar{l} = \frac{1}{n_v} \sum_{i=1}^{n_v} l_i$

l_i is the length of the i-th vector of the dynamic region. The normalized variance is taken as

$$\sigma_{l,norm} = \alpha_v \cdot \sigma_l$$

This is done to suppress frames with a small number of motion vectors to contribute much in the whole calculation of the vector length variance.

This process is repeated for all the frames in the video stream. The result is shown in Figure 6.19 from a video containing dissolves. The special formation is clearly visible.

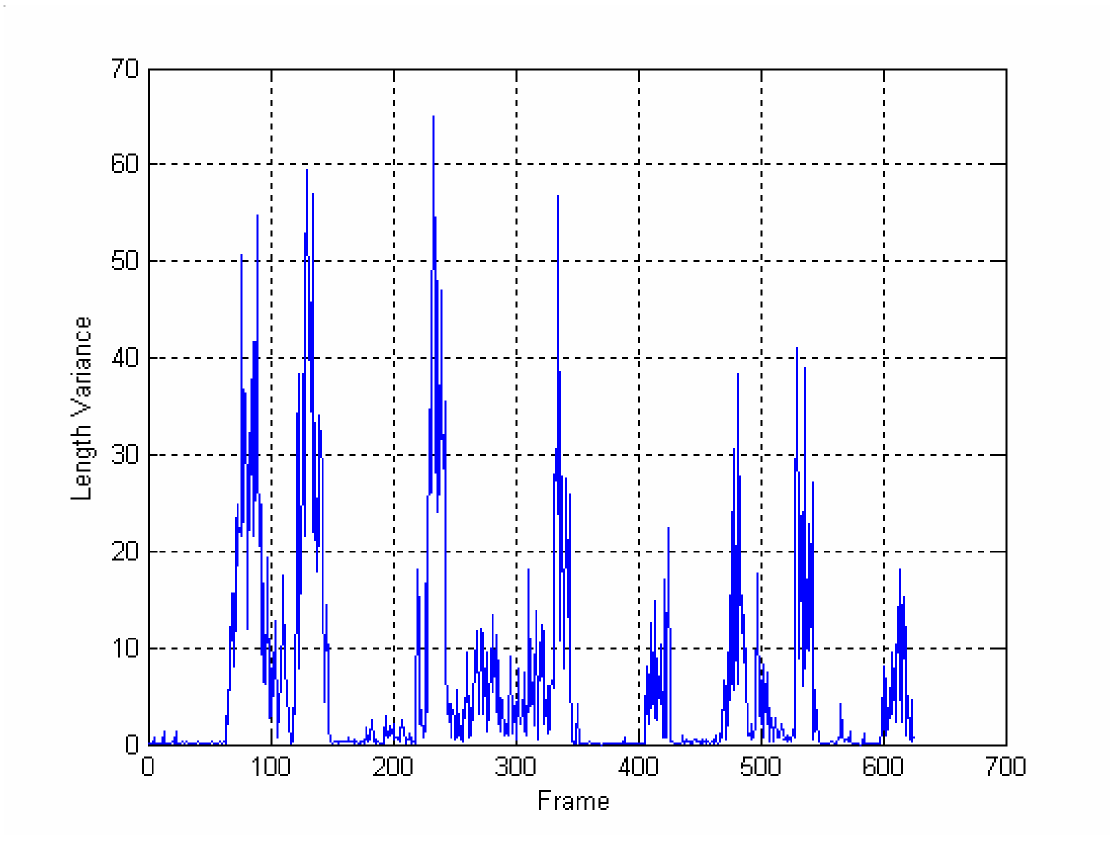


Figure 6.19: Variance of the length of the vectors. The gradual peaks indicate a dissolve

By taking the local mean of the of the length variance we end up with Figure 6.20.

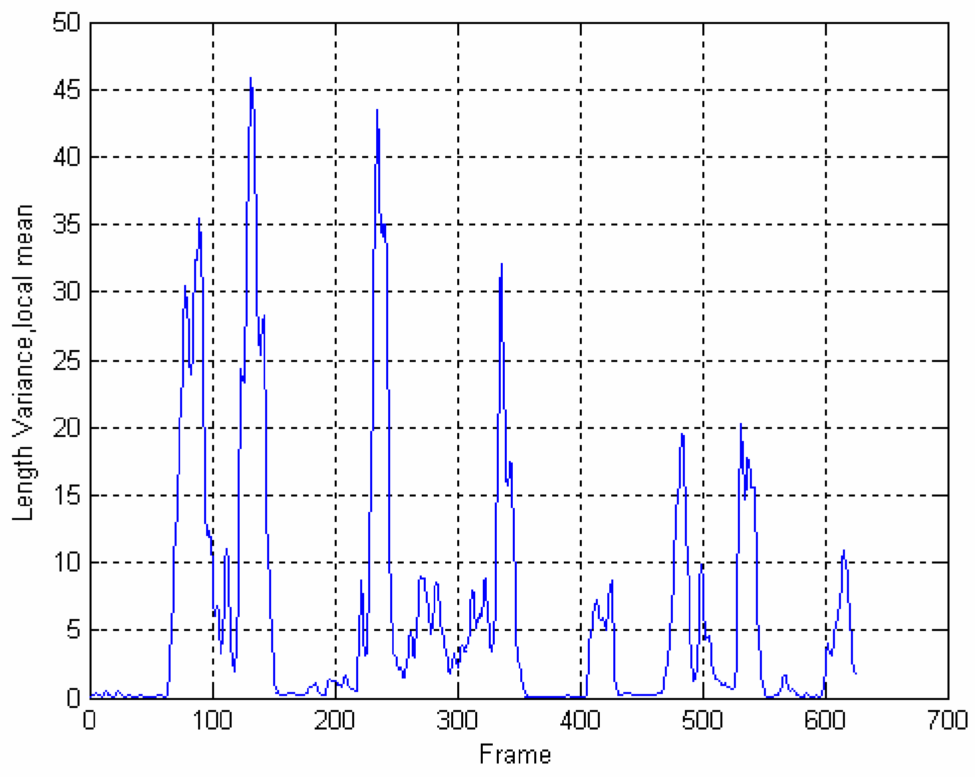


Figure 6.20: By taking the local mean of the variance, we have a more detectable form for dissolves.

The use of the thresholds that were described in the above sections is not applicable here. This is because the variance of the vector length is smoothed as it could not be used directly. As a result the peaks do not rise abruptly but rise in a more gradual way. The sliding window and adaptive thresholds will be very efficient to detect where the variance signal has a value high enough for the frame to be selected as a gradual transition frame. In Figure 6.21 the appliance of the sliding window threshold is shown on the plot of Figure 6.20. To prevent the threshold from dropping to zero values we used a minimum which was set to the overall mean of the variance signal. We can see although some dissolves are recognized others are left undetected.

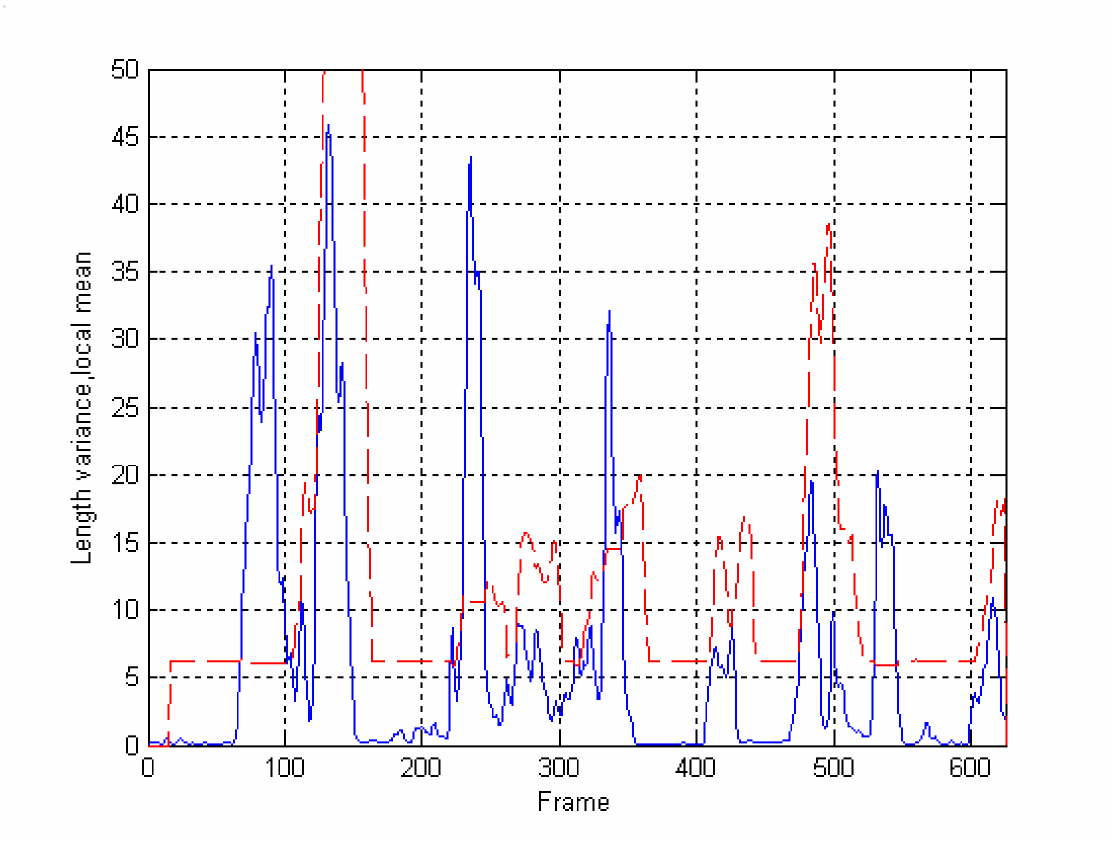


Figure 6.21: Some dissolve sequences while they have large peaks are not detected

This is why a threshold with a fixed value was used. We selected this value to be

$$T_d = 2 \cdot \mu_v$$

where $\mu_v = \frac{1}{TotalFrames} \sum_i Vl(i)$ and Vl is the vector length variance.

If $\sigma_{l,norm} < T_b$ then there is no dissolve

If $\sigma_{l,norm} \geq T_b$ then it is a dissolve sequence

Dissolves with duration smaller than 5 frames are discarded as false detection.

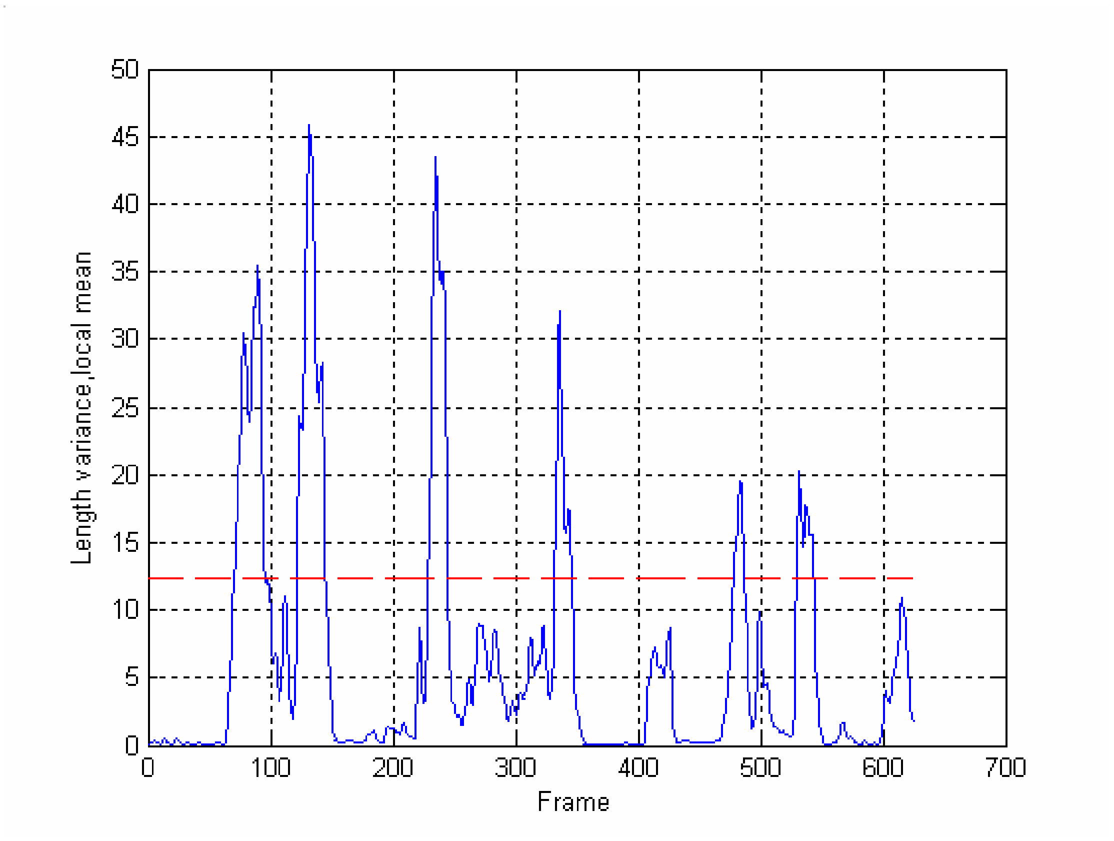


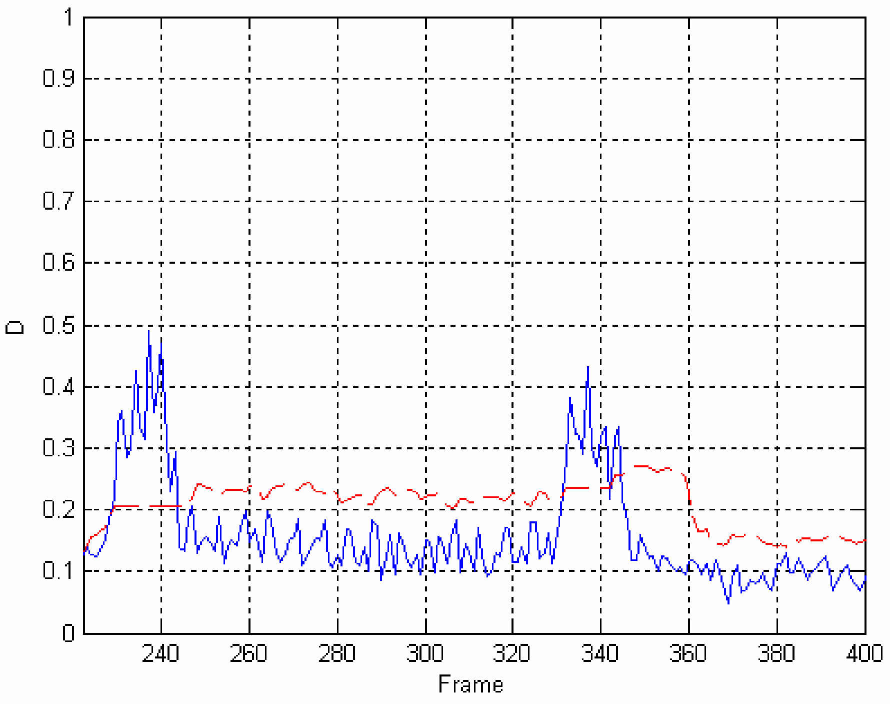
Figure 6.22: Dissolve detection with the static threshold T_d

The same increased variance of motion vector length is also true for zoom shots where the vectors have small magnitude in the center of the frame and large magnitude near the edges. This can lead to false assumptions and it is necessary to exclude these shots from the analysis. Zoom shots can be recognized from the methods used in §6.2.

Although the variance of the vector length can show us where a gradual transition is taking place, it is not an accurate way of detection. Shots with different objects moving and fast camera movements have the same form of variance as in a gradual transition. The use of only motion information for this kind of detection will not give good results because many false alarms and misdetections will exist. For an improved gradual transition identification we propose the combined use of motion and intensity information as explained in the next section.

6.5 Combination of intensity and motion information for the detection of gradual transitions

Although this pattern of motion vectors is observed in almost every gradual transition there are cases, such as many objects moving in the screen, where the motion vectors behave in the same manner. The use of only motion information will result in false positives. To overcome this problem and to further reduce the number of false detections, we propose a combination of motion and intensity information. By using the thresholds described above we can declare a gradual transition if both the histogram difference and length variance of the motion vectors are above the threshold as shown in Figure 6.23.



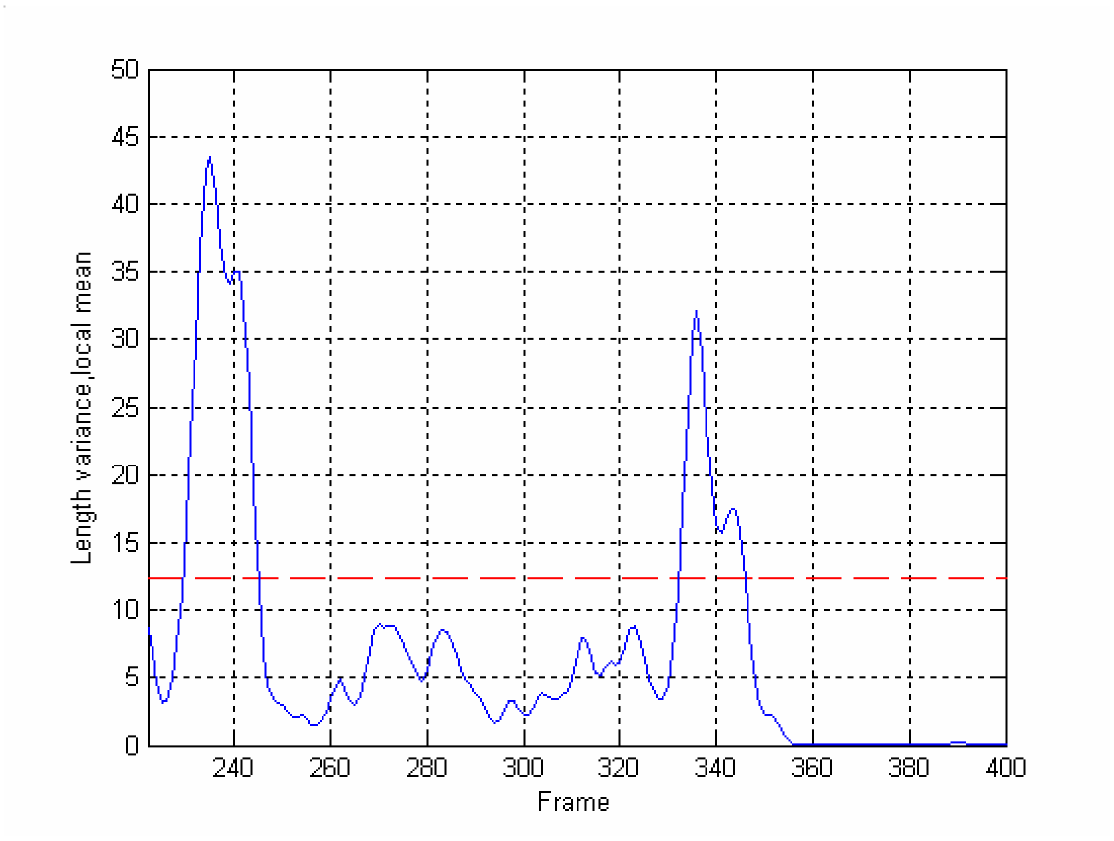


Figure 6.23: A gradual transition is declared if both histogram difference and vector length variance are above the threshold. This method has shown to provide a more accurate way to determine as it increases the number of correctly detected gradual transition sequences thus eliminating false detection.

The combined used of the two different information fields will provide us with a way to identify a gradual transition sequence more accurately as false detections inserted by either motion or intensity fields will be canceled by the double comparison that is made.

It should also be mentioned that because of the similar behavior of the variance in a zoom sequence with the variance of a gradual transition sequence zoom sequences are detected and discarded from the detection. Additionally because in a pan sequence the intensity histogram differs much from frame to frame as objects continuously get in or leave the frame, they are also recognized and discarded from the overall process.

7. Experimental results

By using the methods described in chapters 5 and 6 we can now test our system for efficiency. We have used many different video sequences consisting of 11,671 frames. The types of the video were from news broadcasts, sport videos, cartoons, music videos and movies which cover a wide range of possible videos.

We made tests for panning and zooming detection, camera motion change, cut detection and gradual transitions detection. The panning and zooming detection was made only with the motion based methods of chapter 6 while cut and gradual transition detection was made using both motion based and intensity based methods.

We tested each threshold using different values of the parameter α_w for sliding window threshold, α_a for adaptive threshold and α, β for twin comparison. We expect that for every value of the parameters the algorithm will return a total number of detected changes. Not all of them will be true but the result set will also contain falsely detected changes (cut, gradual, etc).

To compare the efficiency of each threshold we used the *Precision-Recall* diagrams which are widely used in Information Retrieval.

Precision is defined as the fraction of correctly detected changes to the total number of detected changes

$$\Pr ecision = \frac{CorrectChanges}{DetectedChanges}$$

$$0 \leq \Pr ecision \leq 1$$

A *Precision* value close to 1 will indicate that what the algorithm detects as a true change then this is actually a true change and not a false positive.

Recall is defined as the fraction of correctly detected changes to the total number of the actual changes that exist in the video

$$\operatorname{Re} call = \frac{CorrectChanges}{ActualChanges}$$

$$0 \leq \operatorname{Re} call \leq 1$$

A Recall value close to 1 indicates that the algorithm detects every change is the video. Notice however that along with the true changes it might also mark as changes frames that are not actually changes, which means many false positives. A typical form of the precision-recall diagrams is shown in Figure 7.1

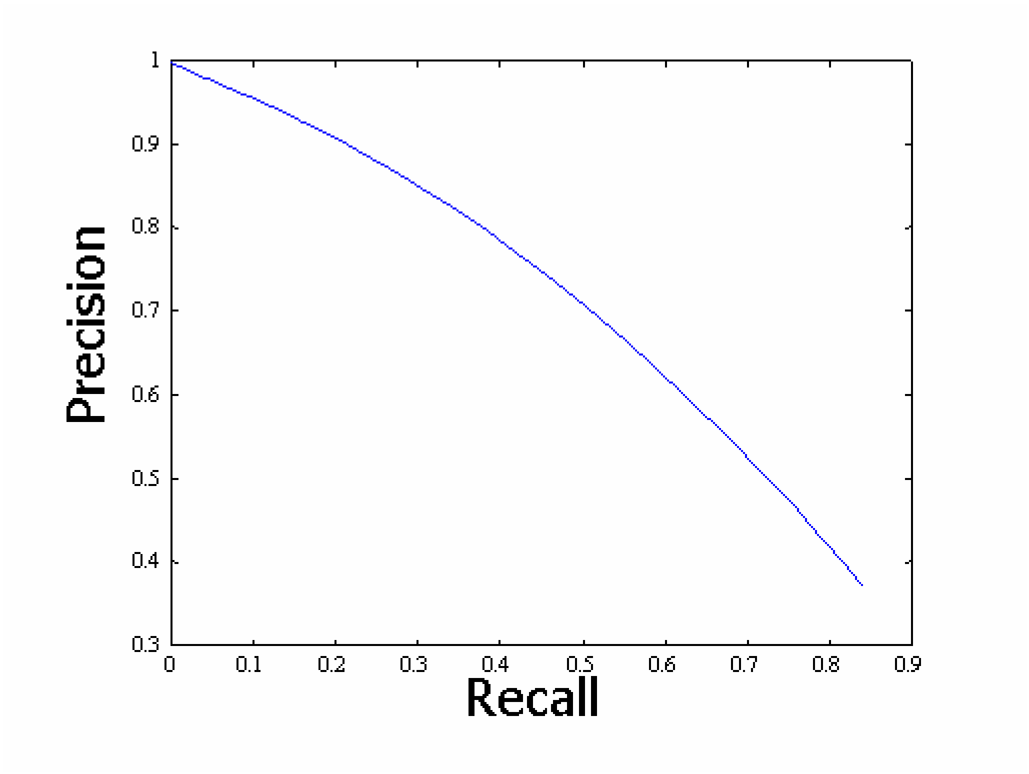


Figure 5.1: A typical figure of the Precision-Recall diagram

The ideal case is an algorithm with *Precision=1* and *Recall=1,* which means that it detects all the changes and what it detects is true.

The videos that we tested were divided into 4 categories: documentary, news, sports and animation. It was found that for each category a different range of the parameter α gave the best results. The reason is that every category has different characteristics which can be summarized in the table below

Documentaries	Many panning and zooming sequences, scenes with a static camera and scenes where the camera is trembling
News	Mainly scenes with a static camera and editing effects (gradual transitions)
Sports	Fast camera movement, panning-zooming sequences
Animation	Low noise in the frames due to the artificially way they are made

7.1 Intensity histogram based methods

7.1.1 Cut detection

Below is the Precision-Recall diagram derived for abrupt cut detection.

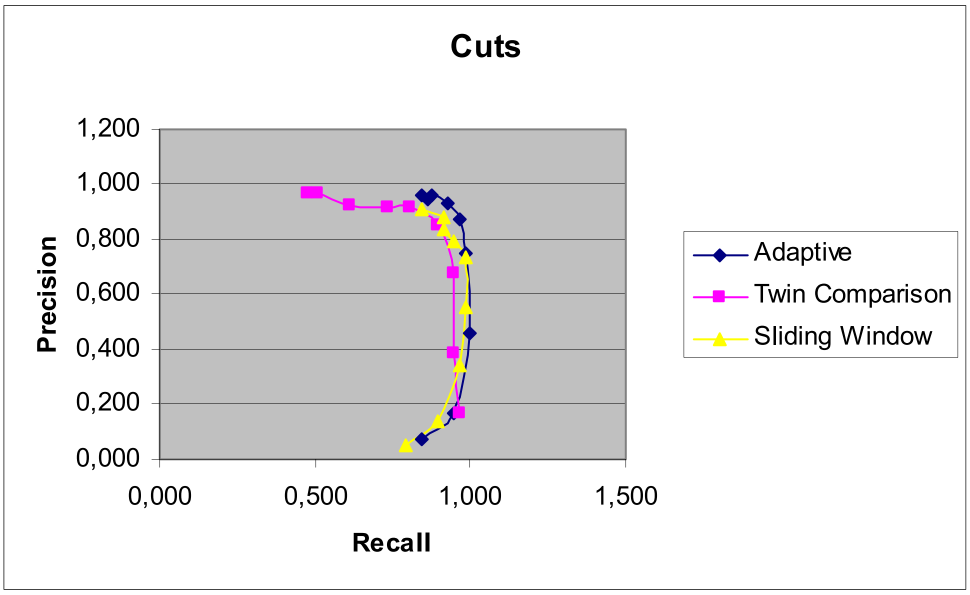


Figure 7.2: Comparison of the three thresholds

The points of the plot which have a large Precision value correspond to high α values, while small Precision corresponds to small α values. The α that were used for adaptive and sliding window were in the range of [2,6] with a 0.5 step, while for twin comparison α was in the range of [1,5] with 0.5 step.

We can observe that for small α values the diagrams have a curve to the left. This is due to the convention that was used for the measurements. If a change was found to have a duration of more than 5 frames then we tagged it as a gradual transition while if the change had a duration smaller than 5 frames we tagged it as cut. When the α parameter had small values, many some cuts would be tagged as gradual transitions, giving a small Recall. As α was higher these cuts are correctly identified and the Recall value becomes larger. This is explained graphically in Figure 7.3

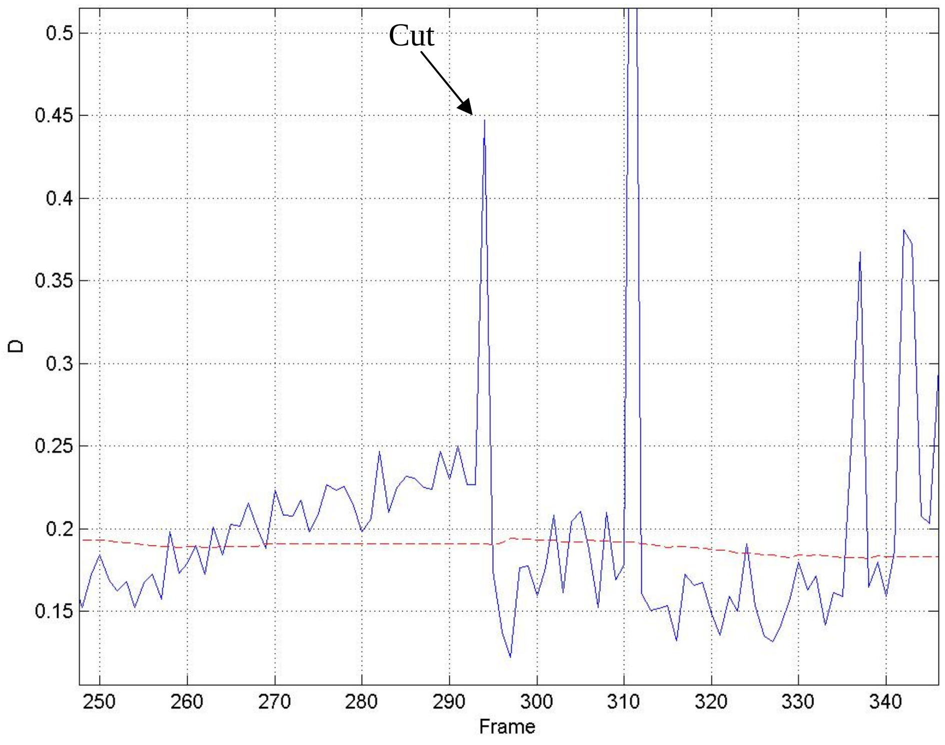


Figure 7.3: For a small value of α for the thresholds, the marked cut will be considered as part of a gradual transition (frames 269-293)

From the diagram we can see that the adaptive performs generally better than the two other thresholds because the curve is higher than the others.

To have a more factual way of comparing the thresholds for the different values of α we introduce the *harmonic mean* which is defined as

$$F = \frac{2}{\frac{1}{\Pr ecision} + \frac{1}{\operatorname{Re} call}}$$

We plot the *harmonic mean* versus α. The α which gives the highest F is the best for each threshold. The result is shown in Figure 7.4

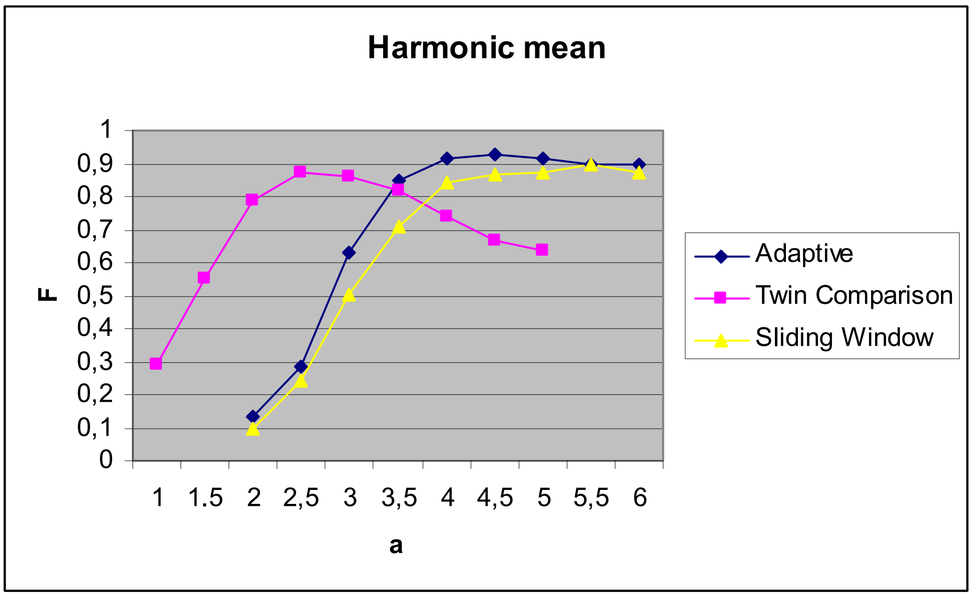


Figure 7.4: The harmonic mean F versus α. From the curves we can see that an α value of 4.5 gives the best results for the adaptive threshold, 5.5 for sliding window and 2.5 for twin comparison

The ranges of α that give the best results for each category of video and for each threshold for abrupt cut detection are:

Category	**Sliding Window**	**Adaptive**	**Twin comparison**
	α	**α**	**α**
Documentary	5 - 6	4 - 4.5	2.5 – 4.5
News	4.5 – 5.5	3 - 4	2.5 – 4
Sports	5.5 - 6	3.5 - 4	2.5 - 3
Animation	5 - 6	4 – 4.5	2.5 – 3

7.1.2 Gradual changes detection

The Precision-Recall diagrams for gradual changes is shown in Figure 7.5

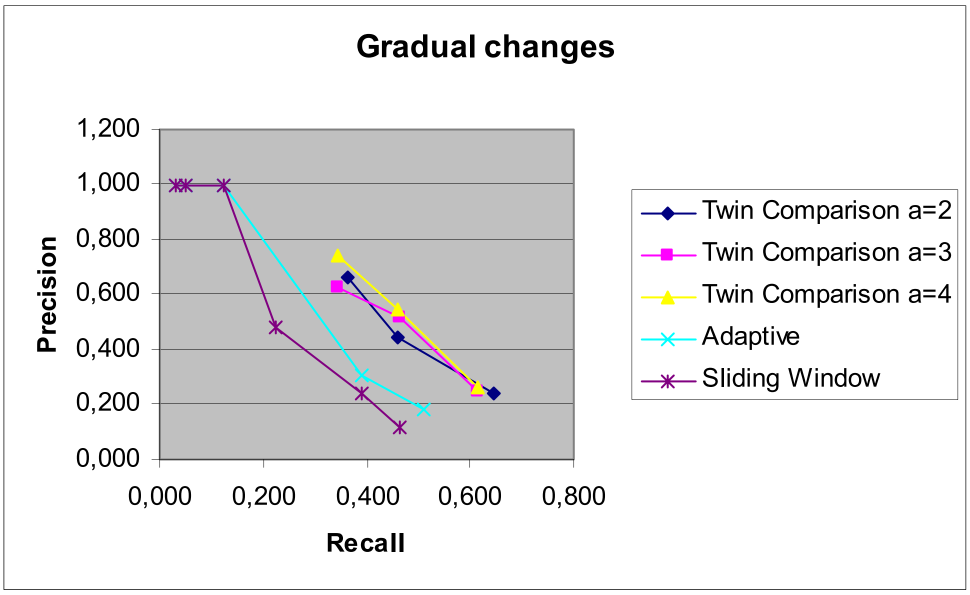
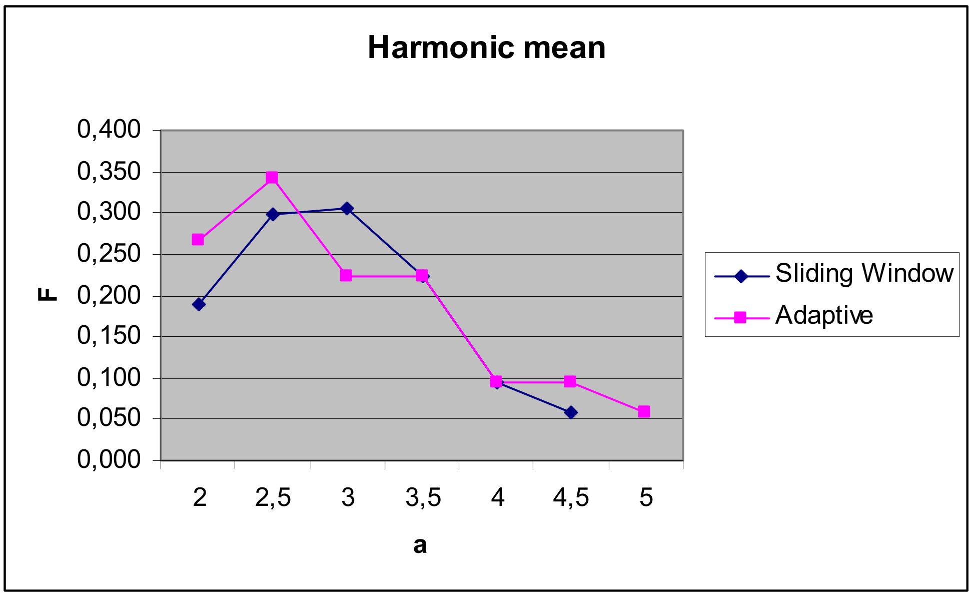


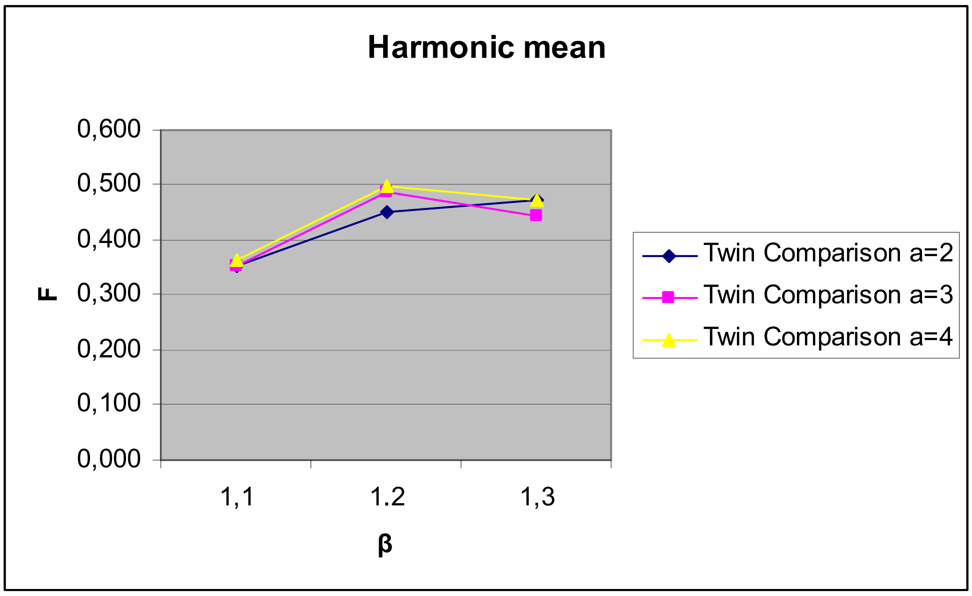
Figure 7.5: Precision-Recall diagrams for gradual changes

We used values of α in the range [2,6] for adaptive and sliding window and for twin comparison we used three values of β [1.1, 1.2, 1.3] for every value of α. It is clear that the twin comparison method performs better than the other two for the DC images as the curves are higher.

The harmonic mean is shown in Figure 7.6



(a)



(b)

Figure 7.6: (a) For the adaptive method α=2.5 and for sliding window α=3 are the best values (b) For twin comparison for α=4 and β=1.2

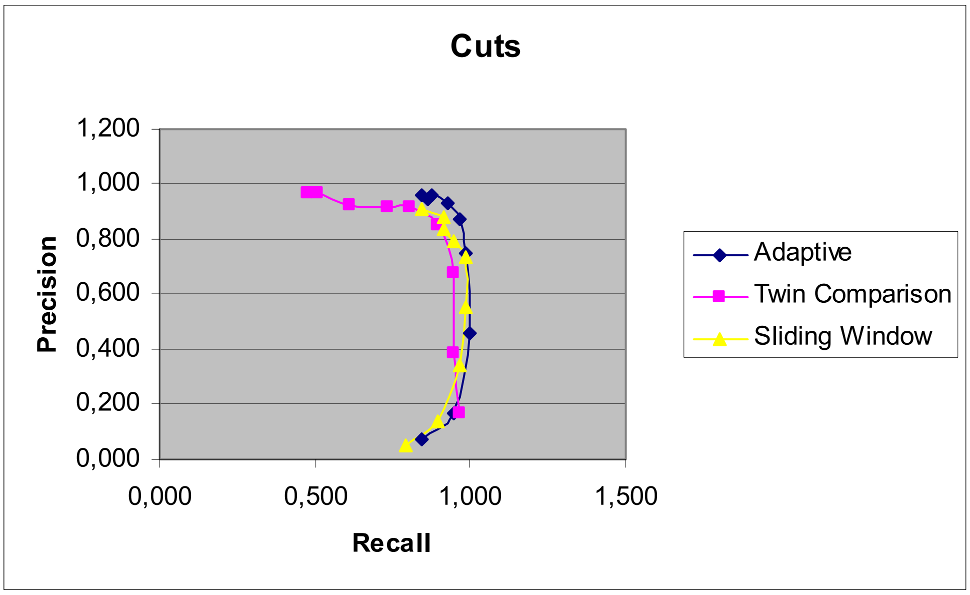
The range of values for α, and β for twin comparison, which give the best results for each category of video are:

Category	Sliding Window	Adaptive	Twin comparison	
	α	α	α	β
Documentary	2.5 – 3	2.5 – 3	2 – 3	1.2
News	2.5 – 3.5	2 – 3	2 – 3	1.1
Sports	3 – 4	3 – 4	3 – 4	1.3
Animation	2.5 – 3.5	2 – 3	2 – 3	1.2

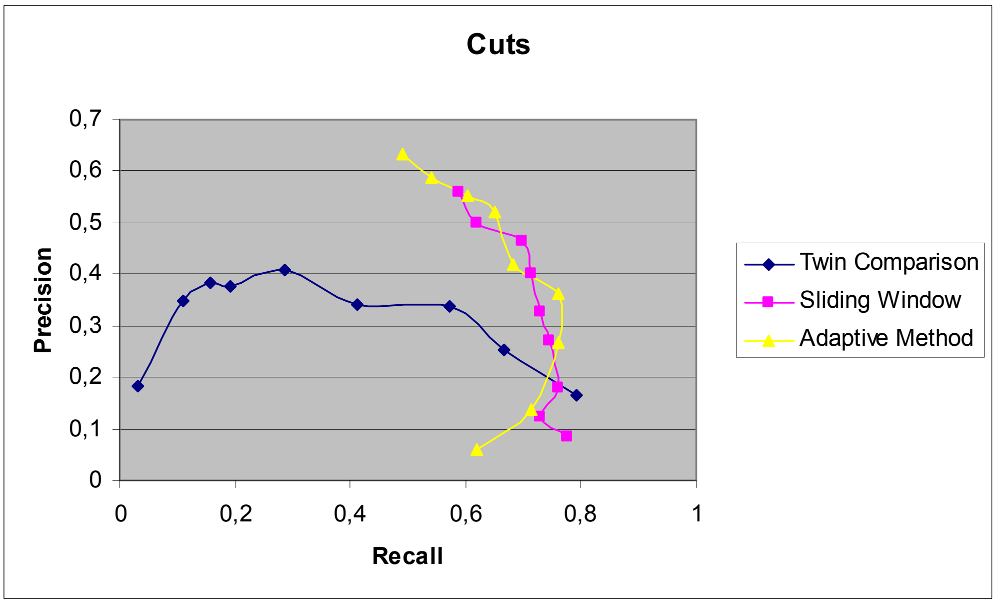
7.1.3 Comparison of histograms methods in the compressed and uncompressed domain

The purpose of this paragraph is to compare the intensity based methods used by us in the compressed to the same methods used in the uncompressed domain. We used the same videos in compressed and uncompressed form and compared the results given from the algorithms.

Cut detection



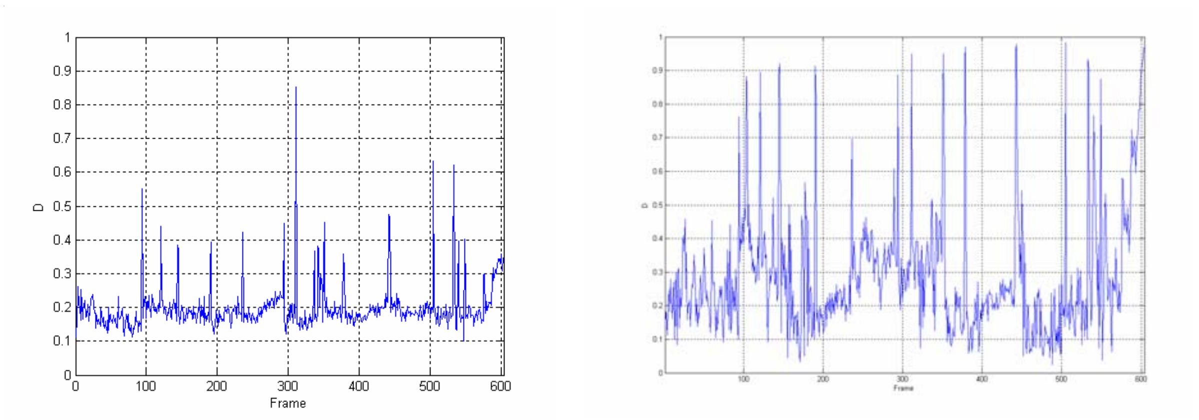
(a)



(b)

Figure 7.7: (a) Compressed domain methods, (b) Uncompressed domain methods

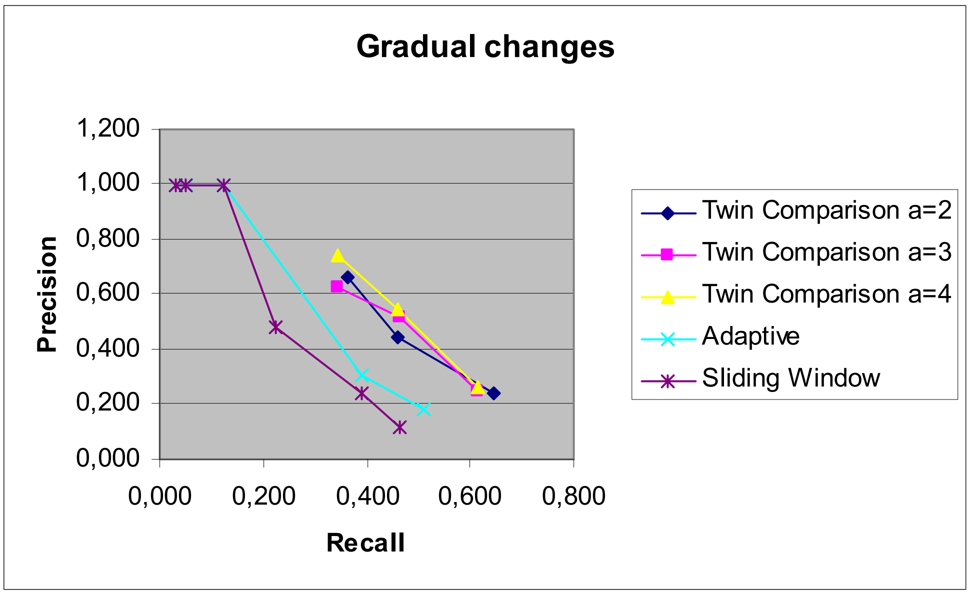
From the diagrams we can see that cut detection performs better in the compressed domain. This is due to the fact that the DC images used from us for the extraction of the intensity histogram are actually the reduced frames of the original video but it is as if a low-pass filter was applied to them. Remember that a pixel in a DC image corresponds to an 8x8 block of the original frame. The result is that in the same scene the changes that are a product of noise do not influence the histogram in a way that this is done in the uncompressed frames. In other words too much detail is not desirable in the case of cut detection.



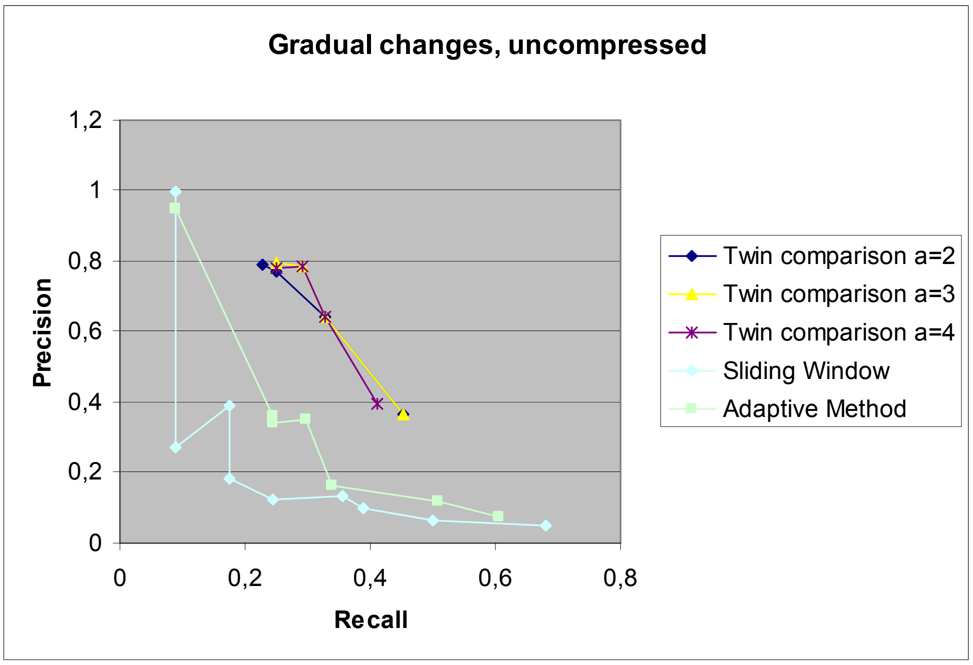
(a) *(b)*

Figure 7.8: The histograms of the same video. (a) is compressed and (b) is uncompressed

Gradual transition detection



(a)



(b)

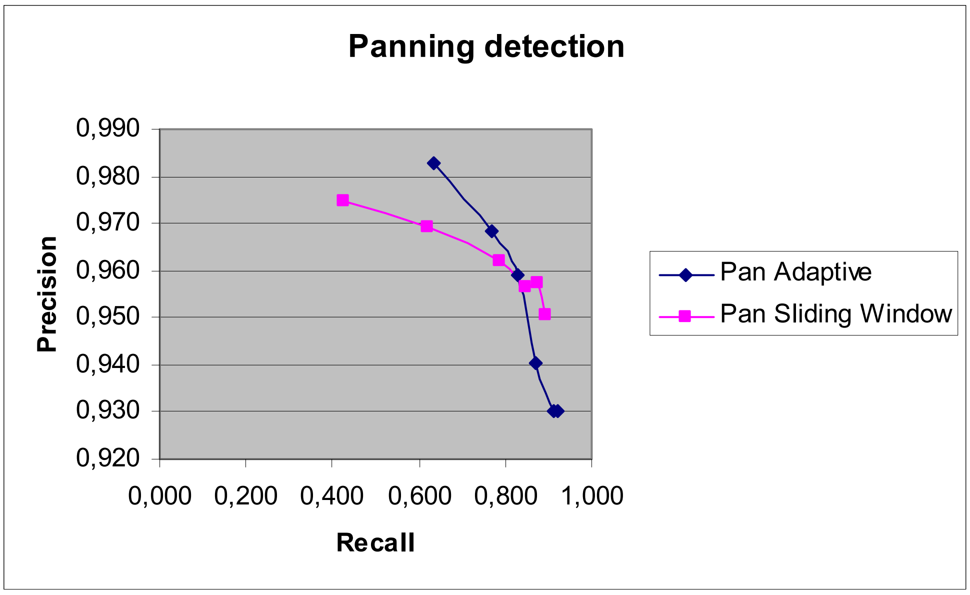
Figure 7.9: (a) is for compressed and (b) is for uncompressed

For the same reasons explained above, due to the smoothing operation on the DC images, the gradual transition detection performs better in the compressed domain than in the uncompressed domain. Gradual transitions are characterized from the slow change between shots. As a result the differences from one frame to another are small and the smoothness of the DC images makes these differences easier to detect as they are distinguished more easily from the histogram difference plot.

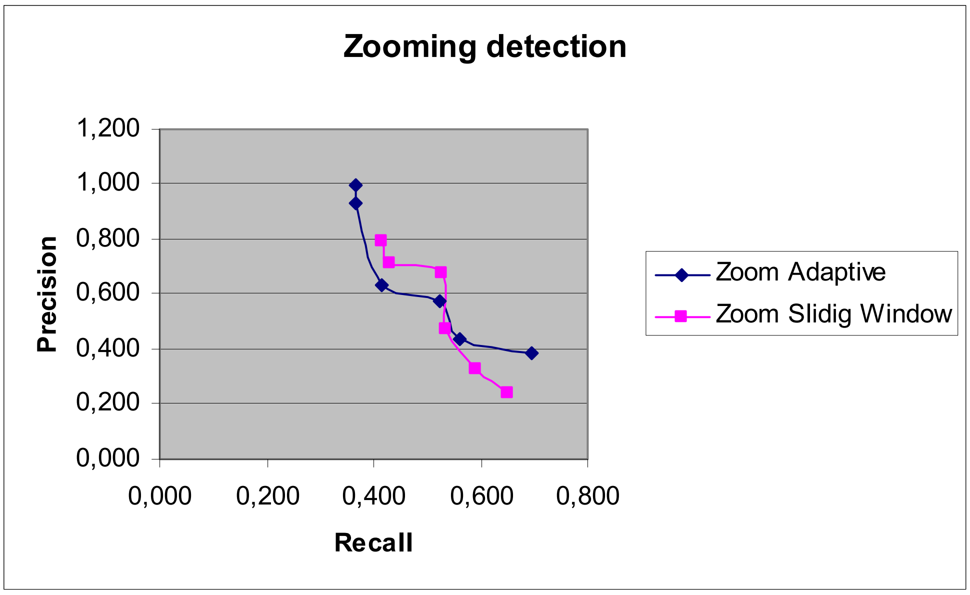
7.2 Motion based segmentation methods and camera motion classification

7.2.1 Panning and zooming detection

We have tested our methods for panning and zooming detection. We used the adaptive and sliding window threshold. Figure 7.10 shows the Precision-Recall diagrams for these thresholds for different values of α.



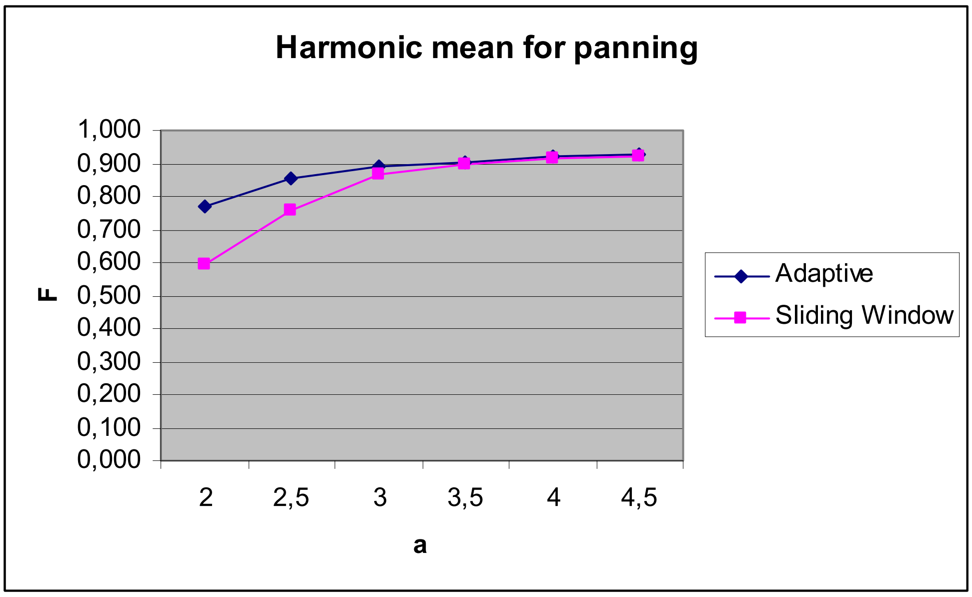
(a)



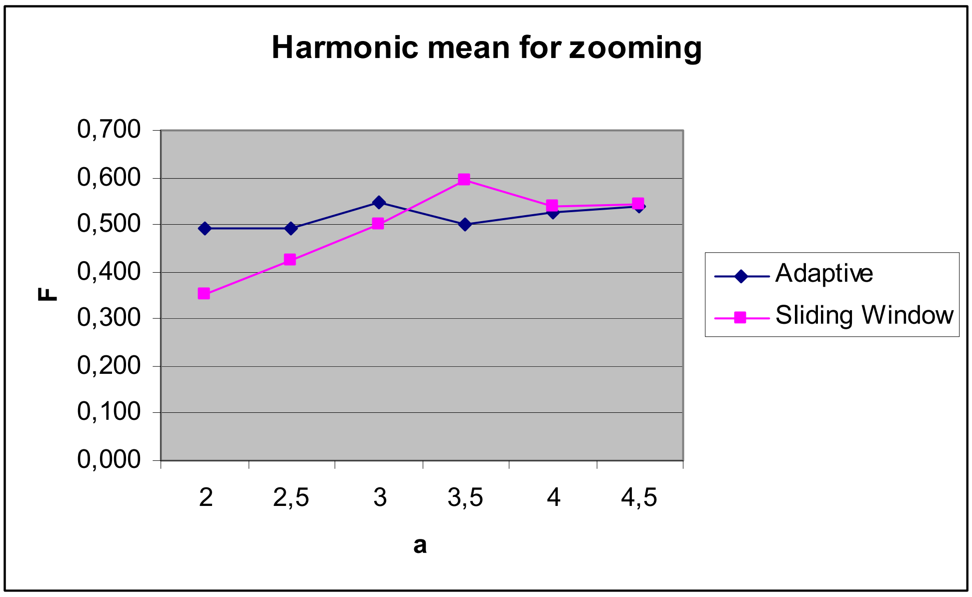
(b)

Figure 7.10: (a) Comparison for panning detection, (b) Comparison for zooming detection

The harmonic mean is plotted in Figure 7.11



(a)



(b)

Figure 7.11: (a) The harmonic mean shows that the adaptive performs better than sliding window and for α=4.5 we have the best results.

(b) For zooming α=3.5 for sliding window gives the highest harmonic mean value

7.2.2 Cut detection

We used all three methods for cut detection through the number of undefined flow vectors mentioned in Ch. 6. The Precision-Recall diagrams are plotted in Figure 7.12 and the harmonic mean is shown in Figure 7.13.

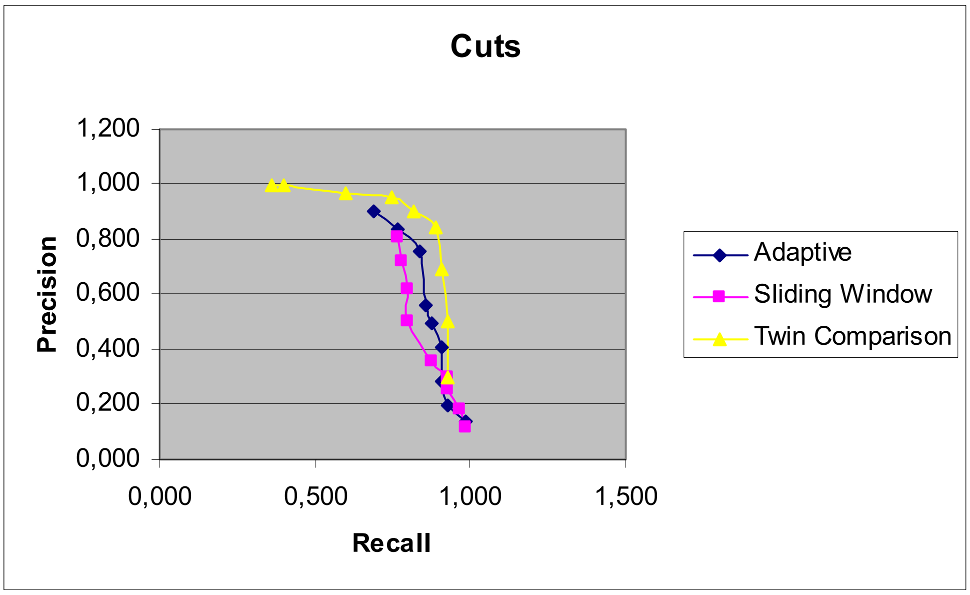


Figure 7.12: Twin comparison gives the best results as its curve is higher than the other two

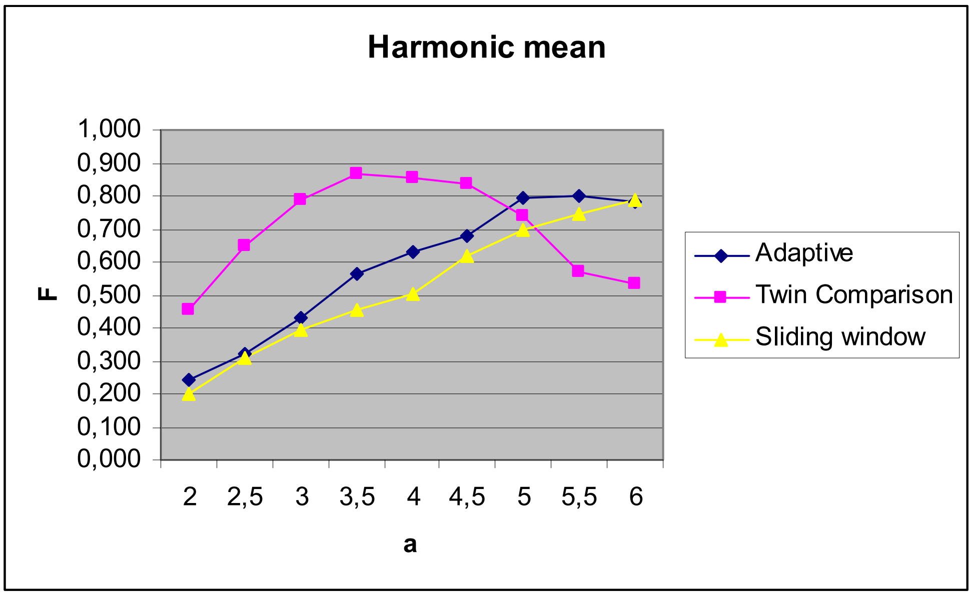
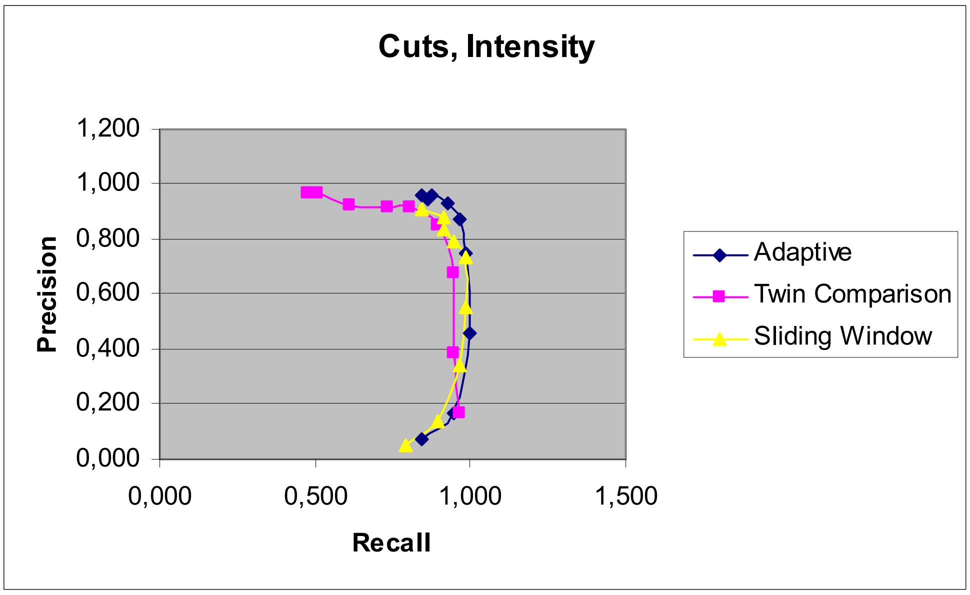
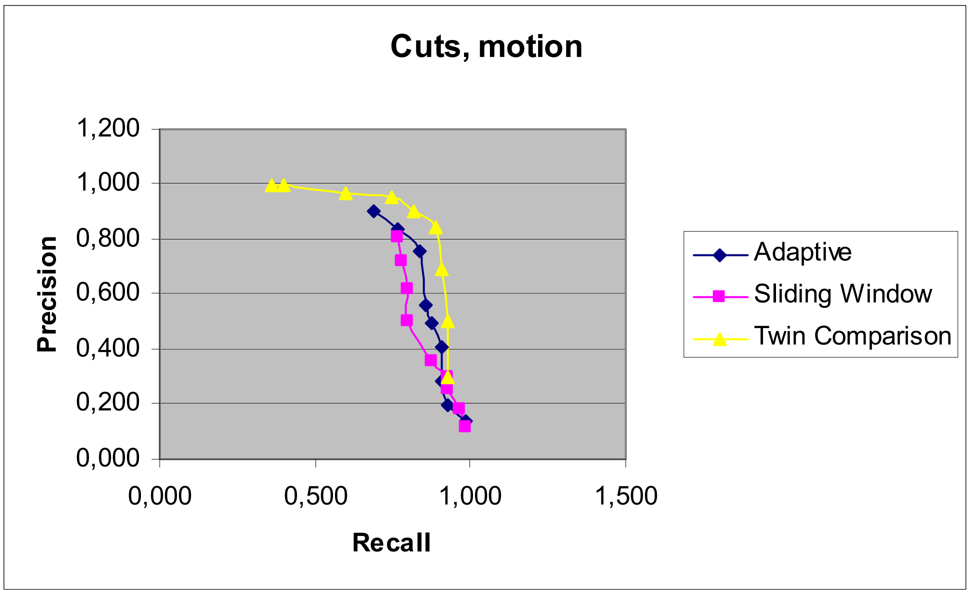


Figure 7.13: The harmonic mean shows that for α=3.5 the twin comparison method gives the highest F value. Sliding window's best α is 6 and for adaptive α=5

Comparing the results obtained for intensity based cut detection of §7.1.1 and the results for motion based cut detection we show the precision recall diagrams of the two methods in Figure 7.14. In Figure 7.15 we plot the harmonic means of the thresholds that give the best results for each method, adaptive for intensities and twin comparison for motion.



(a)



(b)

Figure 7.14: (a) shows the intensity based cut detection and (b) the motion based cut detection. Intensity based cut detection performs better as it gives both Precision and Recall values closer to 1.

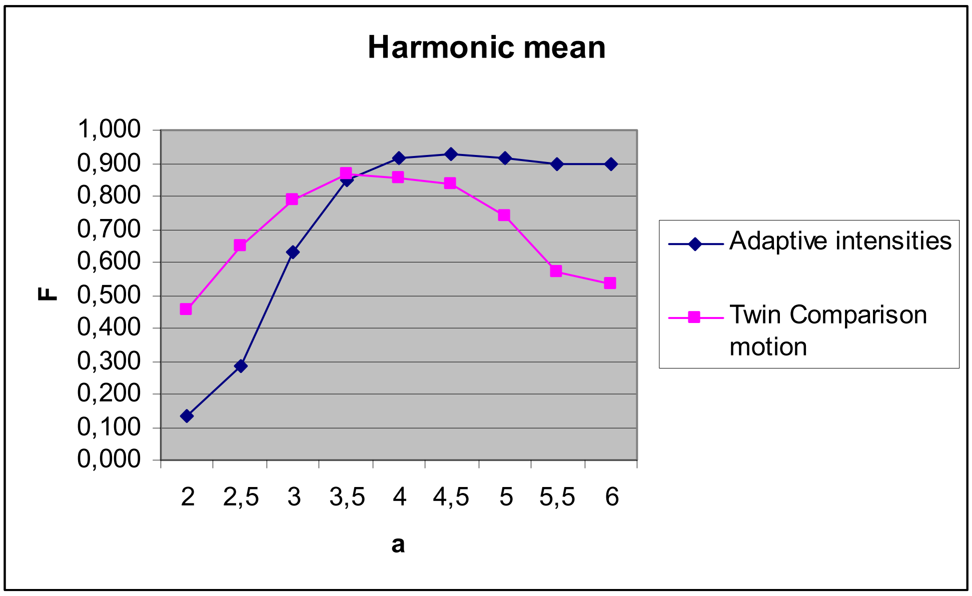
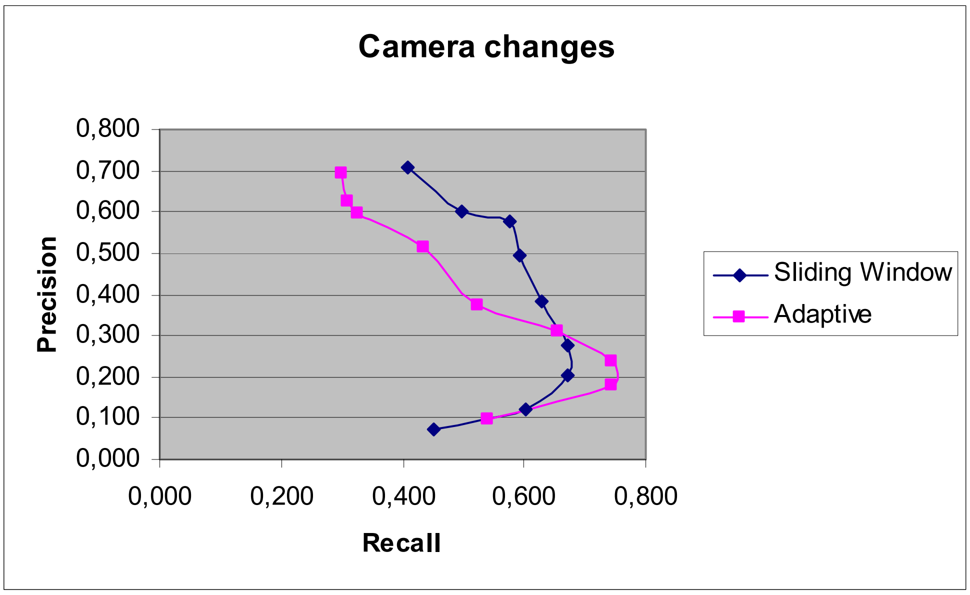


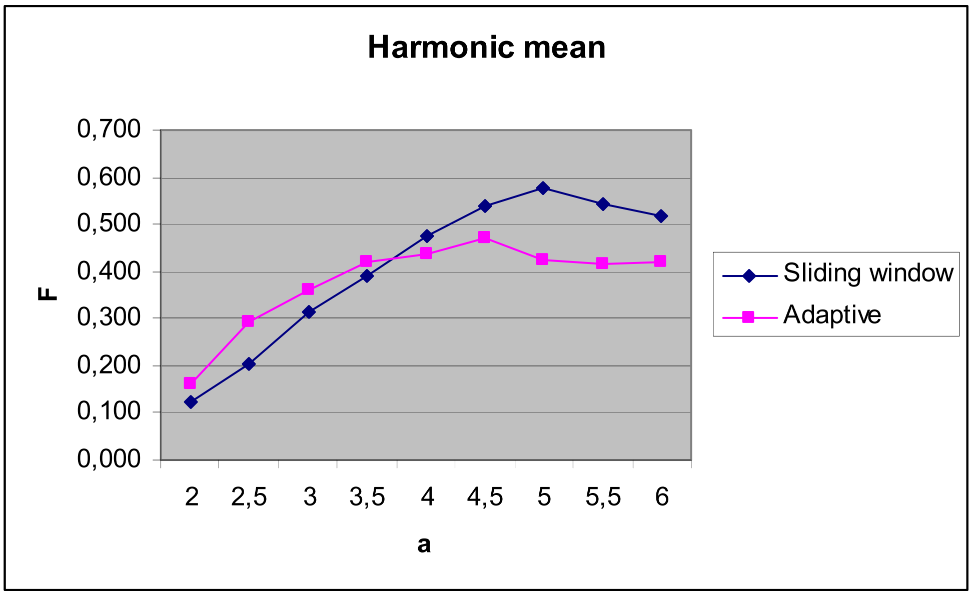
Figure 7.15: The highest harmonic mean value is obtained for α=4.5 for intensity based method

From the above Figures we can see that motion based cut detection performs relatively well. The Twin Comparison threshold is better for this method of detection because we can see from Figure 6.17 of Ch. 6 that the histogram of undefined flow vectors has many peaks apart from those who are due to true cuts in the video stream. Adaptive and Sliding window thresholds will regard the many small peaks as cuts as the first are well above the mean of the histogram and well distinguished. Compared with intensity based cut detection it is not as effective. Using histograms is a more accurate and safe way of detecting abrupt scene changes.

7.2.3 Camera motion change detection



(a)



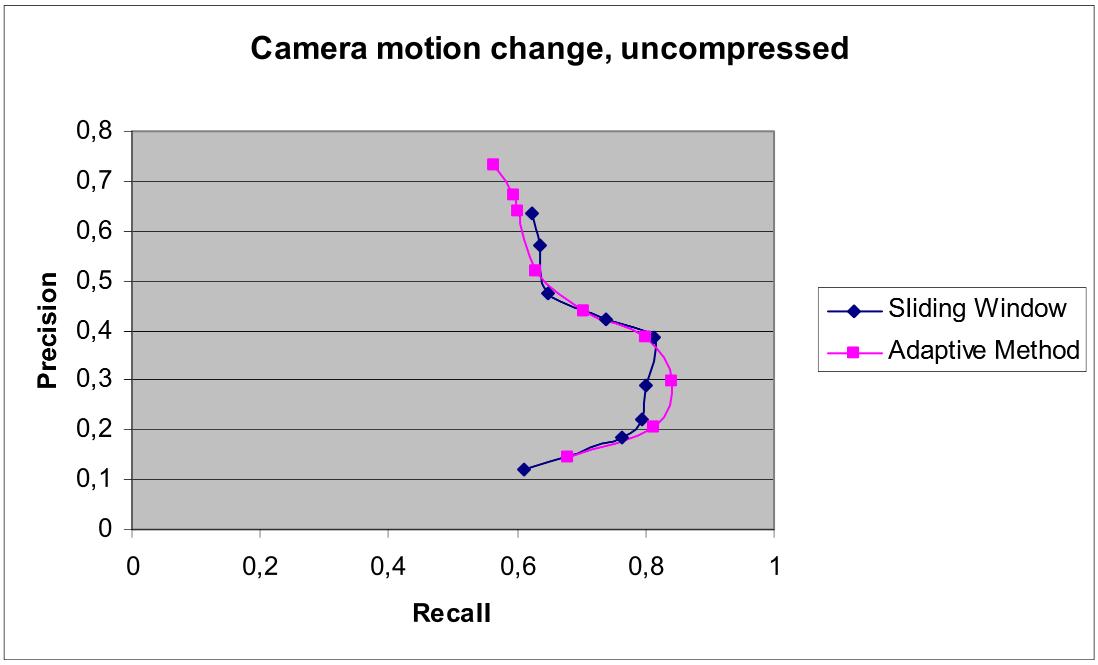
(b)

Figure 7.16: (a) Precision-Recall diagram for camera motion change, (b) Harmonic mean for each. An α value of 5 gives the best results for sliding window

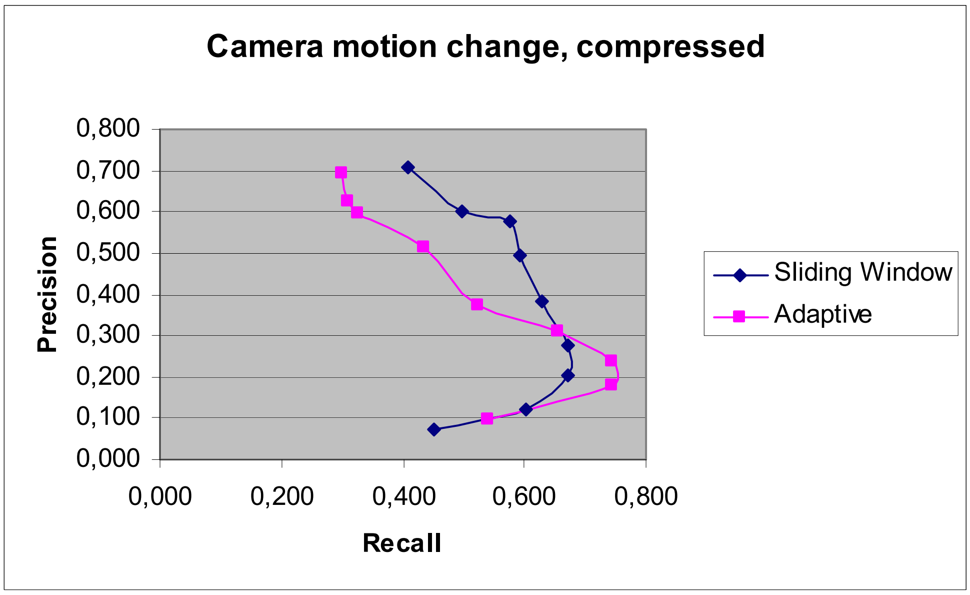
The range of α values that give the best results for every category of video is summarized in the table below

Category	**Sliding Window**	**Adaptive**
	A	**α**
Documentary	3.5 – 4.5	3.5 – 5
News	5 – 5.5	4 – 5
Sports	4.5 – 6	4.5 – 5
Animation	3.5 – 5	4 – 5

The comparison of camera motion change detection methods used in compressed and uncompressed domain is shown in Figure 7.17. In the uncompressed domain the methods give better results. This is due to the fact that motion estimation that was done in the uncompressed domain is much more accurate than the motion vectors of MPEG. Additionally the normalization process that was used in the computation of the motion histograms of Ch. 6 further reduces the accuracy of them.

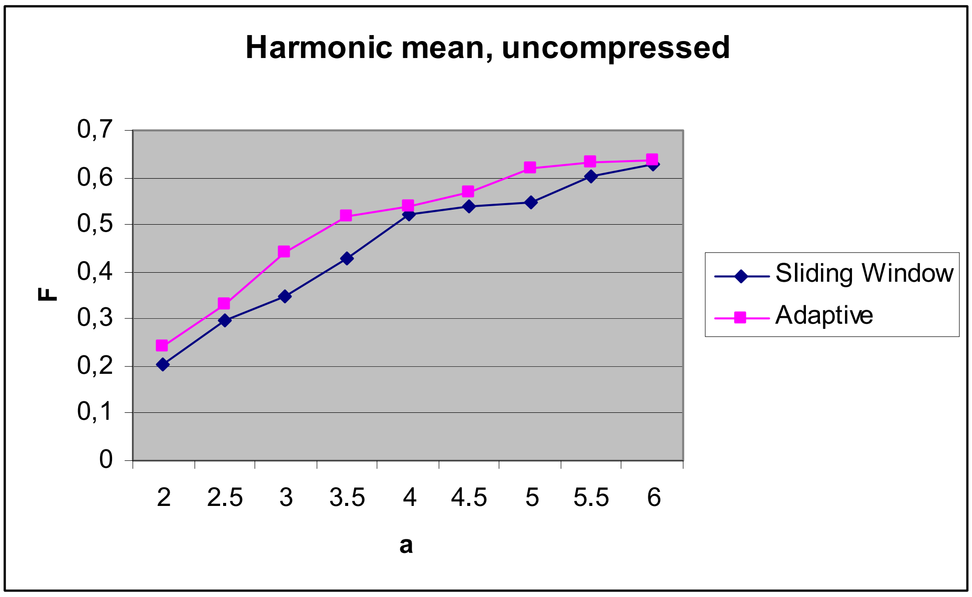


(a)

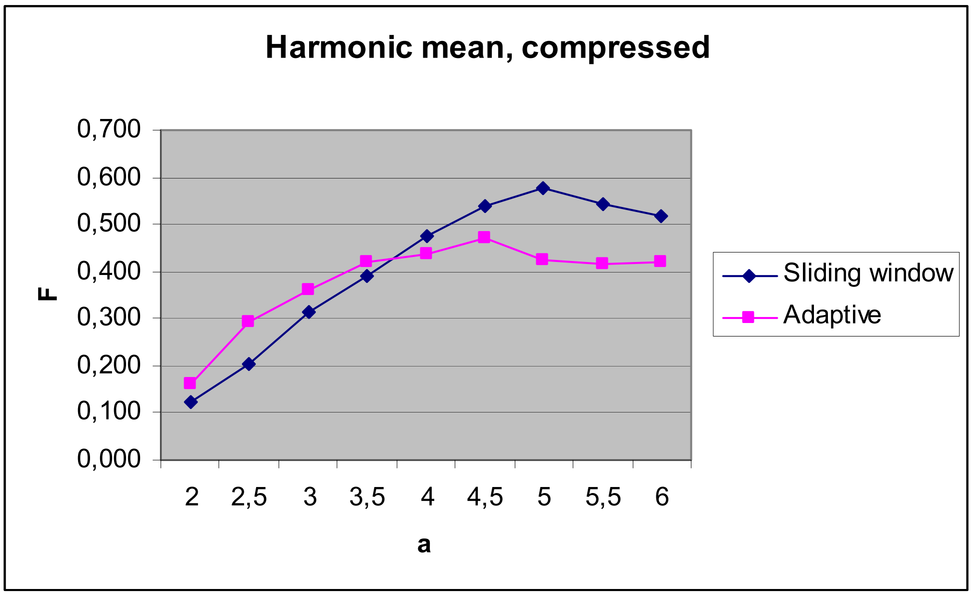


(b)

Figure 7.17: (a) is uncompressed, (b) is compressed



(a)

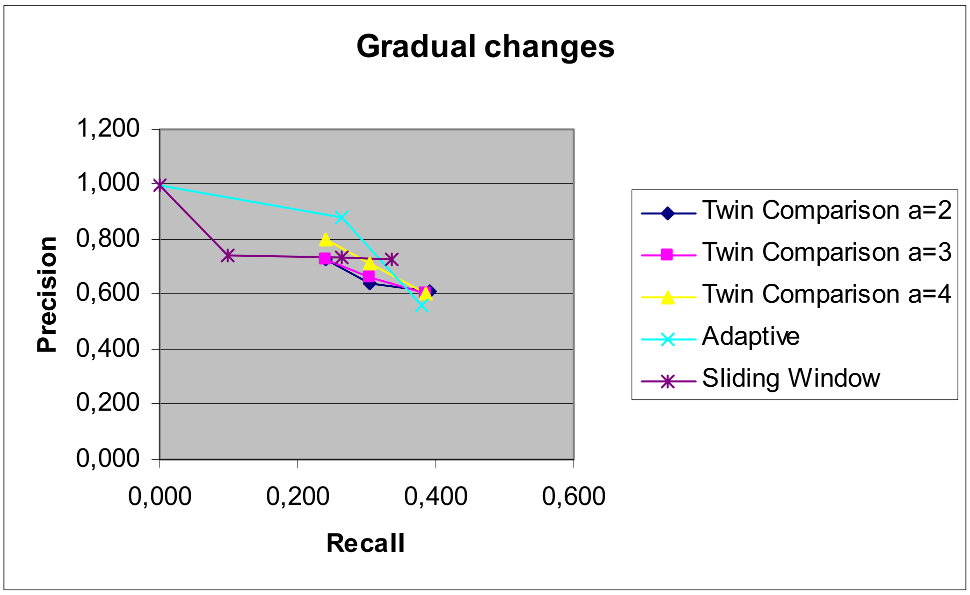


(b)

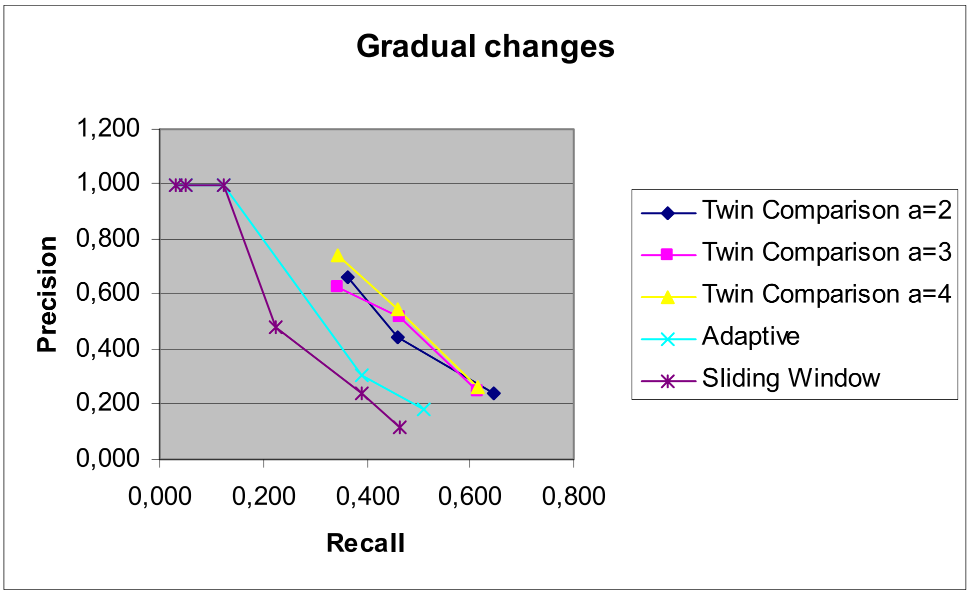
Figure 7.18: Harmonic means for camera change detection

7.2.4 Intensity and motion information combination for gradual transition detection

We next illustrate the results that were obtained for the detection of gradual transitions and compare them with the results of the gradual transition detection based on intensity only. These are shown in Figure 7.19



(a)



(b)

Figure 7.19: (a) Motion and intensity based information, (b) Intensity based information

From the above results we can see that in the case of sliding window and adaptive thresholds we have achieved an increase in the Precision of the gradual transition recognition. This was expected as the many false positives that were introduced from the intensity histogram processing were eliminated from the motion information. We can also distinguish the slight decrease in the Recall value of the thresholds. This is mainly based on the fact that some of our test videos included transitions between two pan or zoom sequences. Because these sequences were taken out of the processing the transition could not be detected.

8. Conclusions and further work

We have proposed new algorithms and thresholding techniques for automatic segmentation of MPEG compressed video sequences. The algorithms use intensity and motion information extracted directly from the compressed domain and from our experimental results they provide a good accuracy in detecting shot changes. The comparison of scene change detection in the compressed and uncompressed domain has shown that abrupt cut detection is much more efficient in the compressed domain. The use of partially uncompressed data from the MPEG stream have shown to perform better because of the smoothed form that the DC images have compared to the frames extracted from the uncompressed video. The same apply for gradual transitions.

Camera motion change detection in the MPEG stream gives relatively good results. It is harder to detect if there are objects moving in the camera field, where the motion vectors do not change abruptly but follow the motion of the object. In the uncompressed domain it performs better mainly because MPEG motion vectors are not very accurate but also because of the intracoded macroblocks in a frame which do not have a motion vector assigned to them.

In the case of gradual transitions, which are harder to detect, the combined use of motion and intensity gives higher precision as the second metric, the motion information, is a reliable indicator for a gradual transition.

As further work of this thesis we propose

- keyframe extraction of the shots identified
- Combination of shots into logical story units

References

[1] Yang Yongsheng, Lin Ming, "A survey on content based video retrieval"

[2] Ba Tu Truog, Chitra Dorai, Sventha Venkatesh, "New enhacements to cut, fade and dissolve detection processes in video segmentation"

[3] Loong-Fah Cheong, "Scene-Based Shot Change Detection and Comparative Evaluation"

[4] M. R. Naphade, R. Mehrotra, A. M. Ferman, J. Warnick, T. S. Huang, A. M. Tekalp, "A high performance shot boundary detection algorithm using multiple cues"

[5] Joan L. Mitchell, "MPEG video compression standard", Chapman & Hall, 1997

[6] Ruggero Milanese, Frederick Deguillaume, Alain Jackot-Descombes, "Video segmentation and camera motion characterization using compressed data"

[7] Boon-Lock Yeo, "Efficient processing of compressed images and video"

[8] Vikrant Kobla, David Doermann, King-Ip Lin, "Compressed domain video indexing techniques using DCT and motion vector information in MPEG video"

[9] Irena Coprinska, Sergio Carrato, "Temporal video segmentation: A survey"

[10] F. Arman, A. Hsu, M-Y Chiu, "Image processing on compressed data for large video databases"

[11] J. Meng, Y. Juan, S.-F. Chang, "Scene change detection in a MPEG compressed video sequence"

[12] J. Feng, K.-T. Lo, H. Mehrpour, "Scene change detection algorithm for MPEG video sequence"

[13] Supavadee Aramvith, Ming-Ting Sun, "MPEG-1 and MPEG-2 Video Standards"

[14] H. Zhang, A. Kankanhalli and S.W. Smoliar, "Automatic Partitioning of Full Motion Video"