DIPLOMA THESIS - PROYECTO FIN DE CARRERA

INGENIERÍA DE TELECOMUNICACIÓN

Segmentation of sport video sequences

Beatriz López García
Supervisor: Olivia Nemethova
Professor: Markus Rupp
Tutor: Francisco Javier Gabiola Ondarra

05-10-2005

Universidad Alfonso X El Sabio

INSTITUT FÜR
NACHRICHTENTECHNIK UND
HOCHFREQUENZTECHNIK

Abstract

Wireless mobile systems of the 3rd generation have limited bandwidth. To support multimedia streaming over such systems requires high compression, which can result in a considerable visual distortion of the video quality. Especially challenging content to be transmitted over mobile networks are the sport videos, most popular of which is the soccer. Soccer match sequences usually contain various scene changes like for example scene cuts, zooming, transitions, wipes, fast motion scenes as well as wide angle panning parts. Task of this project is to enable preprocessing of such soccer videos to make it more robust against the compression impairments. The most important object in a soccer match is the ball. Due to the compression impairments (blurriness), it may happen that in the wide-angle camera shots, where the ball is very small, it disappears, or appears very unclearly. This is annoying for the user. To overcome this, the position of a ball can be found in each frame and it can be replaced by a slightly bigger ball or just sharpened, depending on the required compression ratio. To perform the preprocessing reliably, a robust video segmentation is crucial. Spatial and temporal video segmentation allows recognizing of the critical scenes that need to be preprocessed. Focus of this work is the playfield segmentation according to the straight lines as well as the detection of various scene changes, especially scene cuts, zooming and pans. Investigated methods are of low complexity to enable real-time preprocessing. If possible, algorithms deployed at the video encoder are used to ease the possible implementation into the encoder. In combination with the known position of the ball, the segmentation provides means for the event detection in soccer game.

Acknowledgements

This thesis was completed at Vienna University of Technology. First of all, I sincerely would like to thank my supervisor, Olivia Nemethova, for giving me the opportunity to do this thesis, her support, help and enthusiasm through the duration of the research project. I would also like to express my gratitude to Professor Markus Rupp for his always useful hints and advises. Also, I want to thank the Institut für Nachrichten und Höchfrequenztechnik, for providing me with the facilities I needed for my work.
I would also like to say thank you to the Universidad Alfonso X el Sabio de Madrid, where I studied most of my degree and received a great education, in especial to Francisco Javier Gabiola Ondarra, my supervisor in Spain, for his help with all procedures for validations as well as for his full of wisdom and always interesting lessons. Besides, I thank them for having given me the chance to go to Vienna to complete my degree.
Above all, I really want to thank all of those who helped me along the way and stayed with me, not only during the preparation of the thesis but also during the years of study and work. Especially, I am grateful to my family and friends for their encouragement, enthusiasm, and for always believing in me.

Table of Contents

Table of figures

Introduction

The rapid increase in the adoption of video production devices, such as camcorders , mobiles or digital cameras, along with the advances in video compression and the increasing usage of the Internet and wireless communication, such as 3G and 4G standards, have enabled the entrance in the market of a wide range of applications. One of the most appealing ones is video streaming.
Streaming was pioneered by the RealNetworks company and now has competing versions from Microsoft, Apple and others. It is simply a method of making video, audio and other multimedia available relatively quickly. The advantage of streaming is that it can enable easier and more responsive on-demand access to multimedia resources. Perhaps even more exciting is the possibility of integrating video and audio with other web-based resources, for example communication and quiz tools. But how does video streaming works? In order to play smoothly, video data needs to be available continuously and in the proper sequence without interruption. Until fairly recently, video had to be downloaded in its entirety to before it could be played. With streaming, the file remains on the server. The initial part is copied to a buffer and then, after a short delay, starts to play and continues as the rest of the file is being pulled down. Streaming provides a steady method of delivery controlled by interaction between the receptor and the server. The server regulates the stream according to network congestion and thereby optimises the presentation.
Among the wide range of possibilities that streaming provides us with, we find one of the most successful ones: life sport transmissions. In this diploma thesis, we will focus on the king of them all: soccer, even though most of the improvements could be applied to all of them.

When dealing with this task, two main problems appear:

- Since we are talking about real-time videos, no retransmission can be arranged at the transport layer and thus user datagram protocol (UDP) is used rather than the transport control protocol (TCP). The reason is that once after a frame has been rendered in the device screen, you can not see the previous frames anymore and the time limit to rendering is given by the size of the jitter buffer, which is not more than 3 or 5 seconds in nowadays terminals.

- Limitations in bandwidth are usually a fact, considering more bandwidth is costly. Considering that a full coloured frame is usually recorded at a frequency of 30 frames per second, taking for granted that QCIF (144x176) resolution is used and knowing that a full coloured image is formed by 3 different matrixes of pixels, we

see that the bandwidth (BW) that we need for transmission is about 18Mbps. However, the BW that is usually available for the 3rd generation mobile terminals ranges from 64 to 128 Kbps. Therefore, something should be done in order to solve this problem and that is the implementation of compression techniques. On the other hand, applying of the compression techniques results in a visual quality degradation.

1. Video compression

Compression is a reversible conversion of data to a format that requires fewer bits, usually performed so that the data can be stored or transmitted more efficiently. The size of the data in compressed form (C) relative to the original size (O) is known as the compression ratio (R=C/O). If the inverse of the process, decompression, produces an exact replica of the original data then the compression is lossless. Lossy compression, usually applied to image data, does not allow reproduction of an exact replica of the original image, but has a higher compression ratio. Thus lossy compression allows only an approximation of the original to be generated. For image compression, the fidelity of the approximation usually decreases as the compression ratio increases. The success of data compression depends largely on the data itself and some data types are inherently more compressible than others. Generally some elements within the data are more common than others and most compression algorithms exploit this property, known as redundancy. The greater the redundancy within the data, the more successful the compression of the data is likely to be. Fortunately, digital video contains a great deal of redundancy and thus is very suitable for compression.

1.1. How is compression accomplished ?

There are many popular general purpose lossless compression techniques, that can be applied to any type of data.

- **Run Length Encoding**

Run Length Encoding is a compression technique that replaces consecutive occurrences of a symbol with the symbol followed by the number of times it is repeated. For example, the string 111110000003355 could be represented by 15063252. Clearly this compression technique is most useful where symbols appear in long runs, and thus can sometimes be useful for images that have areas where the pixels all have the same value, like for example cartoons.

- **Relative Encoding**

Relative encoding is a transmission technique that attempts to improve efficiency by transmitting the difference between each value and its predecessor, in place of the value itself. Thus the values 15106433003 would be transmitted as 1+4-4-1+6-2-1+0-3+0+3. In effect the transmitter is predicting that each value is the same as its predecessor and the data transmitted is the difference between the predicted and actual values. Differential Pulse Code Modulation (DPCM) is an example of relative encoding.

- **Huffman Coding**

Huffman coding [41] is a popular compression technique that assigns variable length codes (VLC) to symbols, so that the most frequently occurring symbols have the shortest codes. On decompression the symbols are reassigned their original fixed length codes. When used to compress text, for example, variable length codes are used in place of ASCII codes, and the most common characters, usually *space*, *e*, and *t* are assigned the shortest codes. In this way the total number of bits required to transmit the data can be considerably less than the number required if the fixed length representation is used. Huffman coding is particularly effective where the data are dominated by a small number of symbols.

- **Arithmetic Coding**

Although Huffman [27] coding is very efficient, it is only optimal when the symbol probabilities are integral powers of two. Arithmetic coding does not have this restriction and is usually more efficient than the more popular Huffman technique. Although more efficient than Huffman coding, arithmetic coding is more complex.

- **Lempel-Ziv Coding**

Lempel-Ziv compressors use a dictionary of symbol sequences. When an occurrence of the sequence is repeated it is replaced by a reference to its position in the dictionary. There are several variations of this coding technique and they differ primarily in the manner in which they manage the dictionary. The most well known of these techniques is the Lempel-Ziv-Welch variation .

1.2. Compression techniques

There are two main techniques [42] to compress video images: spatial or intraframe compression and temporal or interframe compression. Hereafter, a short explanation of both is done:

1.2.1. Intraframe / Spatial compression

Intraframe compression is compression applied to still images, such as photographs and diagrams, and exploits the redundancy within the image, known as spatial redundancy. Intraframe compression techniques can be applied to individual frames of a video sequence.

1.2.1.1. Sub-sampling

Sub-sampling is the most basic of all image compression techniques and it reduces the amount of data by throwing some of it away. Sub-sampling reduces the number of bits required to describe an image, but the quality of the sub-sampled image is lower than the quality of the original. Sub-sampling of images usually takes place in one of two ways. In the first, the original image is copied but only a fraction of the pixels from the original are used. Alternatively, sub-sampling can be implemented by calculating the average pixel value for each group of several pixels, and then substituting this average in the appropriate place in the approximated image. The latter technique is more complex, but generally produces better quality images.

In the example in Figure 1 the pixels in every second row and every second column are ignored. To compensate for this, the size of the remaining pixels is doubled. This is known as pixel doubling.

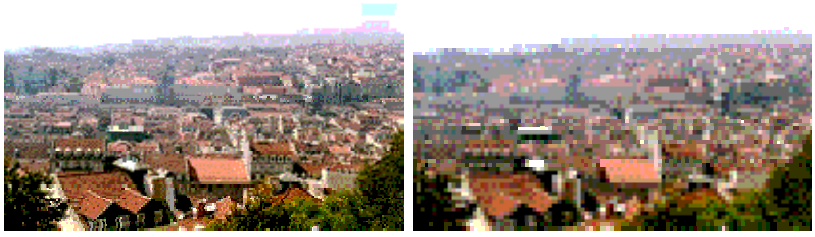


Figure 1: Example of the appliance of subsampling intraframe compression technique

When coding colour images, it is common to sub-sample the colour component of the image by 2 in both directions, while leaving the luminance component intact. This is useful because human vision is much less sensitive to chrominance than it is to luminance, and sub-sampling in this way reduces the number of bits required to specify the chrominance component by three quarters.

Sub-sampling is necessarily lossy, but relies on the ability of human perception to fill in the gaps. The receiver itself can also attempt to fill in the gaps and try to restore the pixels that have been removed during sub-

sampling. By comparing adjacent pixels of the sub-sampled image, the value of the missing in between pixels can be approximated. This process is known as interpolation. Interpolation can be used to make a sub-sampled image appear to have higher resolution than it actually has and is usually more successful than pixel doubling. It can, however, result in edges becoming blurred.

1.2.1.2. Coarse Quantization

Coarse quantization is similar to sub-sampling in that information is discarded, but the compression is accomplished by reducing the numbers of bits used to describe each pixel, rather than reducing the number of pixels. Each pixel is reassigned an alternative value and the number of alternate values is less than that in the original image. In a monochrome image, the number of shades of grey that pixels can have is reduced. Quantization where the number of ranges is small is known as coarse quantization. One example of coarse quantization is Figure 2. In it, the three images have different numbers of colors. The first has thousands of different colored pixels. The middle has 64 different colors and the rightmost uses only 8 different colors.

Figure 2: Example of coarse quantization intraframe compression technique

1.2.1.3. Vector quantization

Vector quantization is a more complex form of quantization that first divides the input data stream into blocks. A pre-defined table contains a set of patterns for blocks and each block is coded using the pattern from the table that is most similar. If the number of quantization levels (i.e.

blocks in the table) is very small, as in Figure 3, then the compression will be lossy. Because images often contain many repeated sections vector quantization can be quite successful for image compression.

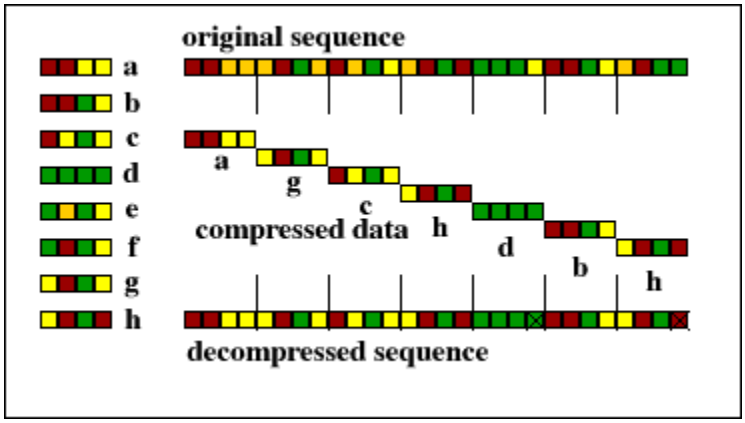


Figure 3: Example of vector quantization intraframe compression technique

In this example of vector quantization a sequence of symbols is divided into blocks of four symbols and the blocks are compared to those in a table. Each block is assigned the symbol in the table entry it most resembles. These symbols form the compressed form of the sequence. On decompression an approximation of the original sequence is generated.

1.2.1.4. Transform Coding

Transform coding is an image conversion process that transforms an image from the spatial domain to the frequency domain. The most popular transform used in image coding is the Discrete Cosine Transform (DCT). Because transformation of large images can be prohibitively complex it is usual to decompose a large image into smaller square blocks and code each block separately.

Instead of representing the data as an array of 64 values arranged in an 8x8 grid, the DCT represents it as a varying signal that can be approximated by a collection of 64 cosine functions with appropriate amplitudes. The DCT represents a block as a matrix of coefficients. Although this process does not in itself result in compression, the coefficients, when read in an appropriate order, tend to be good candidates for compression using run length encoding or predictive coding.

The most useful property of DCT coded blocks is that the coefficients can be coarsely quantized without seriously affecting the quality of the image that results from an inverse DCT of the quantized coefficients. It is in this

manner that the DCT is most frequently used as an image compression technique.

1.2.2. Interframe / Temporal compression

Interframe compression is compression applied to a sequence of video frames, rather than a single image. In general, relatively little changes from one video frame to the next. Interframe compression exploits the similarities between successive frames, known as temporal redundancy, to reduce the volume of data required to describe the sequence.

There are several interframe compression techniques, of various degrees of complexity, most of which attempt to more efficiently describe the sequence by reusing parts of frames the receiver already has, in order to construct new frames.

1.2.2.1 Sub-sampling

Sub-sampling can also be applied to video as an interframe compression technique, by transmitting only some of the frames. Sub-sampled digital video might, for example, contain only every second frame. Either the viewer's brain or the decoder would be required to interpolate the missing frames at the receiving end.

1.2.2.2 Difference Coding

Difference coding, or conditional replenishment, is a very simple interframe compression process during which each frame of a sequence is compared with its predecessor and only pixels that have changed are updated. In this way only a fraction of the number of pixel values is transmitted. The Figure 4 below are successive frames from the table tennis sequence and illustrates how, frequently, very little changes from one frame to the next.

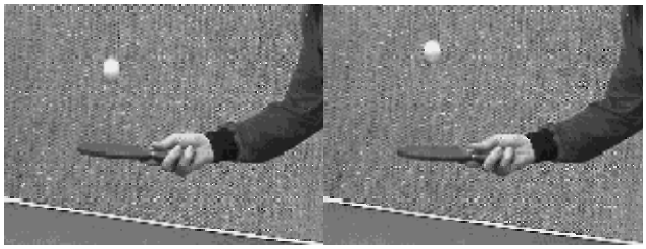


Figure4: Consecutive frames from the Table Tennis sequence.

If the coding is required to be lossless then every changed pixel must be updated. There is an overhead associated with indicating which pixels are to be updated, and if the number of pixels to be updated is large, then this overhead can adversely affect compression.

Two modifications can alleviate this problem somewhat, but at the cost of introducing some loss. Firstly, the intensity of many pixels will change only slightly and when coding is allowed to be lossy, only pixels that change significantly need be updated. Thus, not every changed pixel will be updated. Secondly, difference coding need not operate only at the pixel level, but at the block level.

1.2.2.3 Block Based Difference Coding

If the frames are divided into non-overlapping blocks and each block is compared with its counterpart in the previous frame, then only blocks that change significantly need be updated. If, for example, only those blocks of the Table Tennis frame that contain the ball, lower arm and bat were updated, the resulting image might be an acceptable substitute for the original.

Updating whole blocks of pixels at once reduces the overhead required to specify where updates take place. The 160x120 pixels in the Table Tennis frame can be split into 300 8x8 pixel blocks. Significantly fewer bits are required to address one of 300 blocks than one of 19200 individual pixels. If pixels are updated in blocks, however, some pixels will be updated unnecessarily, especially if large blocks are used. Also, in parts of the image where updated blocks border parts of the image that have not been updated, discontinuities might be visible and this problem is worse when larger blocks are used. Clearly the choice of block size must be an informed one so as to achieve the best balance between image quality and compression.

Block Based Difference Coding can be further improved upon by compensating for the motion between frames. Difference Coding, no matter how sophisticated, is almost useless where there is a lot of motion. Only objects that remain stationary within the image can be effectively coded. If there is a lot of motion or indeed if the camera itself is moving, then very few

pixels will remain unchanged. Even a very slow pan of a still scene will have too many changes to allow difference coding to be effective, even though much of the image content remains from frame to frame. To solve this problem it is necessary to compensate in some way for object motion.

1.2.2.4 Block Based Motion Compensation

Block based motion compensation, like other interframe compression techniques, produces an approximation of a frame by reusing data contained in the frame's predecessor. This is completed in three stages.

First, the frame to be approximated, the current frame, is divided into uniform non-overlapping blocks and motion vectors are obtained for each block. If the target block and matching block are found at the same location in their respective frames then the motion vector that describes their difference is known as a *zero vector*.

Finally, when coding each block of the predicted frame, the motion vector detailing the position (in the past frame) of the target block's match is encoded in place of the target block itself. Because fewer bits are required to code a motion vector than to code actual blocks, compression is achieved.

During decompression, the decoder uses the motion vectors to find the matching blocks in the past frame, which it has already received, and copies the matching blocks from the past frame into the appropriate positions in the approximation of the current frame, thus reconstructing the image.

However, the effectiveness of compression techniques that use block based motion compensation depends on the extent to which the following assumptions hold:

- Objects move in a plane that is parallel to the camera plane. Thus the effects of zoom and object rotation are not considered, although tracking in the plane parallel to object motion is.

- Illumination is spatially and temporally uniform. That is, the level of lighting is constant throughout the image and does not change over time.
- Occlusion of one object by another, and uncovered background are not considered.

Bidirectional motion compensation uses matching blocks from both a past frame and a future frame to code the current frame. A future frame is a frame that is displayed after the current frame. Bidirectional compression

is much more successful than compression that uses only a single past frame, because information that is not to be found in the past frame might be found in the future frame. This allows more blocks to be replaced by motion vectors. Bidirectional motion compensation, however, requires that frames be encoded and transmitted in a different order from which they will be displayed.

1.3. Video codecs

A number of video codecs are available that work in either hardware or software. A few examples are explained below:

•MPEG: MPEG stands for the Moving Picture Experts Group. MPEG is an ISO/IEC working group, established in 1988 to develop standards for digital audio and video formats. There are five MPEG standards being used or in development. Each compression standard was designed with a specific application and bit rate in mind, although MPEG compression scales well with increased bit rates. They include:

•MPEG-1: Designed for up to 1.5 Mbit/sec, it is the standard for the compression of moving pictures and audio. This was based on CD-ROM video applications, and is a popular standard for video on the Internet, transmitted as .mpg files. In addition, level 3 of MPEG-1 is the most popular standard for digital compression of audio known as MP3. MPEG-1 is also the standard of compression for VideoCD, the most popular video distribution format thoughout much of Asia.

• MPEG-2 : Designed for between 1.5 and 15 Mbit/sec. Standard on which Digital Television set top boxes and DVD compression is based. It is based on MPEG-1, but designed for the compression and transmission of digital broadcast television. The most significant enhancement from MPEG-1 is its ability to efficiently compress interlaced video. MPEG-2 scales well to HDTV resolution and bit rates, obviating the need for an MPEG-3.

• MPEG-4 : Standard for multimedia and Web compression. MPEG-4 is based on object-based compression, similar in nature to the Virtual Reality Modeling Language. Individual objects within a scene are tracked separately and compressed together to create an MPEG4 file. This results in very efficient compression that is very scalable, from low bit rates to very high. It also allows developers to control objects

independently in a scene, and therefore introduce interactivity.

Figure 5: Original frame (left) and frame compressed with MPEG-4
Size original frame: 21.8 MB. Size compressed frame: 916KB

• MPEG-7 : This standard, currently under development is also called the Multimedia Content Description Interface. When released, the group hopes the standard will provide a framework for multimedia content that will include information on content manipulation, filtering and personalization, as well as the integrity and security of the content. Contrary to the previous MPEG standards, which described actual content, MPEG-7 will represent information about the content.

• MPEG-21 : Work on this standard, also called the Multimedia Framework, has just begun. MPEG-21 will attempt to describe the elements needed to build an infrastructure for the delivery and consumption of multimedia content, and how they will relate to each other.

•M-JPEG (Motion JPEG): JPEG stands for Joint Photographic Experts Group. It is also an ISO/IEC working group, but works to build standards for continuous tone image coding. This scheme basically combines a series of JPEG still images that are rapidly displayed to form a moving image.

• Cinepak : This is an older codec that was developed by Radius Corp. for playing back video on slow-speed CD-ROM drives. Cinepak provides relatively high quality and good performance.

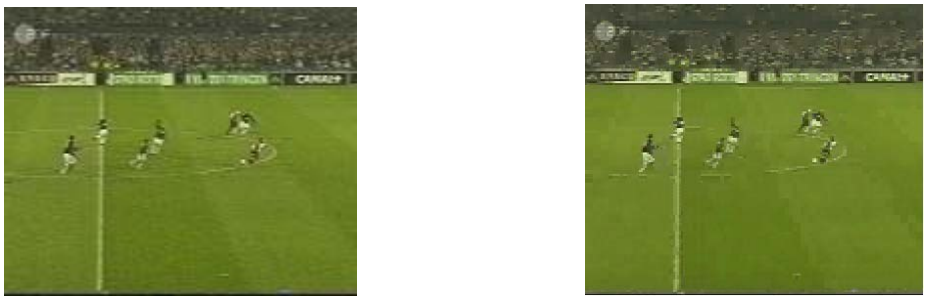


Figure 6: Original frame (left) and frame compressed with Cinepak
Size original frame: 21.8 MB Size compressed frame: 888KB

◦ Indeo Video 3.2 : An Intel-developed 24-bit video codec that first appeared in the 1980s. Like Cinepak, this codec was designed for video playback on slow CD-ROMs.

• Indeo Video Interactive 4.1 : This codec provides higher quality than the earlier Indeo versions. It uses wavelet technology.

• H.261: It is an ITU standard designed for two-way communication over ISDN lines (video conferencing) and supports data rates which are multiples of 64Kbit/s. The algorithm is based on DCT and can be implemented in hardware or software and uses intraframe and interframe compression. H.261 supports CIF and QCIF resolutions.


Figure 7: Original frame (left) and frame compressed with H261
Size original frame: 21.8 MB *Size compressed frame: 1MB*

◦ H.263: It is based on H.261 with enhancements that improve video quality over modems. It supports CIF, QCIF, SQCIF, 4CIF and 16CIF resolutions..


Figure 8: Original frame (left) and frame compressed with H263
Size original frame: 21.8 MB *Size compressed frame: 396KB*

◦ H264: H.264 uses the latest innovations in video compression technology to provide incredible video quality from the smallest amount of video data. H.264 delivers the same quality as MPEG-2 at a third to half the data rate and up to four times the frame size of MPEG-4 Part 2 at the same data rate. H.264 achieves the best-ever compression efficiency for a broad range of applications, such as broadcast, DVD, video conferencing, video-on-demand, streaming and multimedia messaging. And true to its

advanced design, H.264 delivers excellent quality across a wide operating range, from 3G to HD and everything in between.

- VDOnet VDOwave : Developed by VDO Corporation, this codec was designed to deliver video that is optimized for the Internet.

- RealSystem G2 : RealSystem is famous for its Internet streaming multimedia products. Its codecs offer outstanding quality at all bit rates and under lossy network conditions. The new RealAudio G2 music codec enhances frequency response by 80 percent for 28.8 modem users. RealAudio and RealVideo files now automatically scale to all bandwidths and bit rate is dynamically adjusted to match available bandwidth, eliminating rebuffering.

As we can see, after compression and decompression, image loses a great percentage of its quality. Since our main point in these ball game retransmissions is to ease any prospective user to know clearly where the ball and what exactly is happening at any time of the match, we can clearly see that will not be possible unless something else is done. Spatial segmentation, together with temporal segmentation solves this problem accurately.

With help of the spatial segmentation, as we will explain in the following chapters, event detection can be accomplished. We assume the video preprocessing application as introduced in [44]. This application performs ball search in each original sequence frame and if the ball is small enough, it will be replaced by larger or at least sharper version. After applying of this method, the compression will not result in loss of the ball from the picture. The side-effect of such preprocessing is that we know the position of the ball exactly in every frame. If we additionally can recognize the scene changes and the structure of the playfield, given by the straight lines, it is enough to recognize some high-level events. We will be able to tell on which side of the playield the ball occurs and detect the ball presence in the goalmouth – the goal. This leads to the possibility of a fully automatic video segmentation that can be for instance used at the operator to select the most important scenes for sending to the customers per MMS (multimedia messaging service).

Nevertheless, temporal segmentation is needed to perform this method reliably, as it will enable the indexing of the video sequence, so that each sequence can be more isolated and, therefore, recognition performance is improved.

2. Spatial segmentation

Processing of sports videos, such as detection of events, analysis or summarization, ease the possibility of delivering sports videos also over narrow-band networks, such as wireless mobile communication networks, the focus of this thesis, and Internet since the valuable semantics just occupy a small portion of the whole content. The problem with this kind of videos, as it was already exposed in previous chapters, is that its value decays abruptly after a relatively short period of time and therefore, processing should be made automatically in real-time and, at the same time, results obtained must be semantically meaningful.

Proper semantic analysis in these cases can be generally achieved by the use of cinematic and object based features [1]. When talking about cinematic features, we refer to the ones that correspond to the filmic video elements and result from the application of cinematographic rules and conventions to filmmaking. Shot-boundaries, shot-types, shot length, shot transitions and slow-motion replays are the ones we often see in sports video processing. Object-based features are color, texture, shape, motion, and high-level features of semantic objects, such as players, referee, field objects, and ball. As cinematic features are easier to compute, which is desirable for time, power, and bandwidth-limited applications, and common to multiple sports, which is imperative for generic processing, they are deployed before object-based features for fast and generic summarization while the use of both cinematic and object-based features results in computationally more efficient descriptor instantiation.

This chapter proposes an object tracking algorithm for field objects, specifically straight playground lines based on the appliance of the Sobel filter in order to detect the edges and on the Hough transform to get parameters of the tracked line. We employ object detection following one purpose: detect, or at least increase the likelihood of the existence of some sports video events in specific shot segments. As a result, we will be able then to implement video indexing as we will know now how to segment the video effectively.

2.1. Event detection versus line tracking

Our main aim with spatial segmentation is being able to recognize some events after straight lines recognition has been made. When we refer to a straight line, we mean the ones that can be described by a linear formula of type $y = ax + b$, where a is the slope and b the shift. However, elliptical lines will be ignored in this work since the straight ones are the most important in terms of spatial segmentation of the playfield.
It is important to take into account that the existence of a program that detects where the ball exactly is, already explained in the previous chapter is taken for granted. Once tracking of the ball and line has been achieved, three groups of events can then be guessed.

- First group refers to those shots in which three parallel lines are present, one of them more outlined than the others (goalmouth). When this occurs, it means that the goalmouth is somewhere in the picture. In Figure 9 we can see an example.

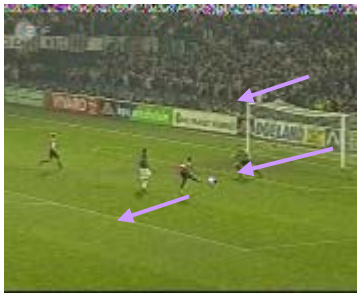


Figure9: Three parallel lines that can be used for the detection of the goalmouth Presence in the picture

Considering we know where the ball is, we can inquire what is more likely to happen: A goal attempt or a goal scored, the beginning of the play again once the referee has stopped the ball, these some are examples of events that can be recognized after detection of the line and the ball has been performed.

- Second group involves shots with just one straight line on them. That is the playground centre line. This way we can allocate the ball in one side of the football pitch or the other. Figure 10 demonstrates graphically what we mean.

Figure 10: Centre line subdividing the playground into two halves, each belonging to one competing team

- Third group, finally, includes events in which no line can be seen in the shot. This means we can be referring to a close-up shot, for example, of a player shooting the ball or even, like in Figure 11, of something that has nothing to do with the game itself such a group of supporters.

Figure 11: An example of close-up shot.

2.2. Edge Recognition

In sports videos, objects of interest correspond to person objects, such as referee and players, field objects that have special semantics, such as endzone in football, painted region in basketball, or penalty-box in soccer, and the ball.

A large number of object recognition strategies exist.

The simplest ones take a model of the image they intend to search for and systematically, compare possible regions with some considerable degrees of freedom. A similarity metric based on the sum of absolute differences (SAD) of both the model and the image can be used. To make the process faster and easier, search is usually done in a coarse to fine

manner and work is implemented just with the gray values of the image, which correspond to the mean value of the three matrixes that dispose a full-colored image.
Analyzing responses or results, we can conclude that normalized cross correlation of gray values is invariant to linear brightness changes but however, does not work well in cases of clutter, occlusion or nonlinear contrast changes. In the case of sum of similarity metrics applied to gray values, it does not seem an effective method for any of the changes, even though it can be made moderately improved by computing the similarity measure in a statistically robust manner.

Other more complicated methods use the edges of the objects for matching. However, the most of them may be grouped into two categories, gradient based methods and Laplacian methods[2]. The gradient method, implemented for instance with Sobel, Roberts or Prewitt filter among others, detects the edges by looking for the maximum in the first derivative of the image. The Laplacian method, like with Marr Hildreth filter, searches for zerocrossings in the second derivative of the image to find edges. Below, some of the well-known edge detection methods are shortly described:

2.2.1 SOBEL

This filter extracts and enhances edges and contours in an image by expressing intensity differences (gradients) between neighboring pixels as an intensity value. This is done by combining the difference between the top and bottom rows in a neighborhood, with the difference between the left and right columns, using the following formula:

$$G_{jk} = |G_x| + |G_y| \tag{1}$$

$$G_X = F_{j+1,k+1} + 2F_{j+1,k} + F_{j+1,k-1} - (F_{j-1,k+1} + 2F_{j-1,k} + F_{j-1,k-1}) \tag{2}$$

$$G_Y = F_{j-1,k-1} + 2F_{j,k-1} + F_{j+1,k-1} - (F_{j-1,k+1} + 2F_{j,k+1} + F_{j+1,k+1}) \tag{3}$$

where (j, k) are the coordinates of each pixel F_{jk} in the Image F, and G_x corresponds to the horizontal edge component and G_y to the vertical edge component. This is equivalent to a convolution using the masks,

$$\text{X mask} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \quad (4) \qquad \text{Y mask} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad (5)$$

In neighborhoods where there is no difference among values in the neighborhood the pixel's intensity is set to 0 (black); where there is the greatest possible difference, the pixel is set to 255 (white). The results are similar to the Roberts filter: highlighted edges against a dark background , but the Sobel filter is less sensitive to image noise. This generally results in an image with smoother, more pronounced outlines of only the principal edges.

2.2.2 ROBERTS

This filter extracts and enhances fine edges in an image by expressing the differences between neighboring pixels (cross pairs) as an intensity value. In neighborhoods where there is no difference among values in the neighborhood the pixel's intensity is set to 0 (black); where there is the greatest difference, the pixel is set to 255 (white). Intermediate levels of gray reflect varying amounts of difference. The result is an image in which edges and contours are highlighted against a dark background. Unlike most other filters, which use an odd-sized square neighborhood, the Roberts filter operates upon a 2 x 2 neighborhood. Since this neighborhood has no center, the pixel in the upper left-corner is the one replaced with a new value. Its filtered value is calculated using the following formula:

The Roberts function returns an approximation to the Roberts edge enhancement operator for images:

$$G_{jk} = |F_{jk} - F_{j+1,k+1}| + |F_{j,k+1} - F_{j+1,k}| \quad (6)$$

where (j, k) are the coordinates of each pixel F_{jk} in the Image and G_x corresponds to the horizontal edge component and G_y to the vertical edge component.. This is equivalent to a convolution using the masks,

$$\begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \text{and} \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \quad (7)$$

where the underline indicates the current pixel F_{jk}. The last column and row are set to zero.

The Roberts filter enhances all edges within an image, even those introduced by noise.

2.2.3 PREWITT

The Prewitt operator measures two components. The vertical edge component is calculated with kernel *Gy* (10) and the horizontal edge component is calculated with kernel *Gx* (9) . The formula:

$$G_{jk} = |G_x| + |G_y| \quad (8)$$

gives an indication of the intensity of the gradient in the current pixel.

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad (9) \qquad Gy = \begin{pmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{pmatrix} (10)$$

Depending on the noise characteristics of the image or streaming video, edge detection results can vary. Gradient-based algorithms such as the Prewitt filter have a major drawback of being very sensitive to noise. The size of the kernel filter and coefficients are fixed and cannot be adapted to a given image. An adaptive edge-detection algorithm is necessary to provide a robust solution that is adaptable to the varying noise levels of these images to help distinguish valid image content from visual artifacts introduced by noise.

2.2.4 MARR HILDRETH

The Laplacian or Marr Hildreth filter is an edge filter that accentuates an image by modifying a pixel's value to exaggerate its intensity difference from its neighbors. The idea is to compute the laplacian [20] of an image, and then mark the zero-crossings of the laplacian as edges. It produces an image with harsh intensity transitions, and results in an image with only edges of high contrast visible. One of the advantages of the Marr-Hildreth is the simplicity, and the fact that all edges detected are closed loops, as we can see in *Figure 4*. The biggest problem is that it is extremely sensitive to noise. The usual solution is to convolve the kernel with a Gaussian [20], which is a low pass-filter that removes very high frequencies caused by noise. It is also separable since it can be decomposed in a product of two filters, one depending only on the variable *x* and the other depending only on *y*. Some problems are alleviated, but the edges are then blurred and become more curved, preventing the line detection step further. Its results are very similar to those of a Hi-Pass Filter. It produces an image with harsh intensity transitions, and results in an image with only edges of high contrast visible.

When just one of the two masks *Gx* or *Gy* of the filters is used, horizontal or vertical edges, respectively, will be obtained. Although they are not purely edge detection methods but a partial result of any of the above, we have included them here as different edge filters since sometimes its properties are needed.

- ***HORIZONTAL EDGE***: This edge filter accentuates the horizontal edges in an image by highlighting pixels with significant intensity differences from those above and below it. It produces an image with just its horizontal edges visible against a flat background. Horizontal edge filtering is used when horizontal features need to be extracted from an image.

- ***VERTICAL EDGE***: This edge filter accentuates the vertical edges in an image by highlighting pixels with significant intensity differences from those to the left and right of it. It produces an image with just its vertical edges visible against a flat background. Vertical edge filtering is used when vertical features need to be extracted from an image.

Some examples of the previous methods can be seen in Figures 4 and 5. As we can see below in Figure 12, where Sobel, Roberts, Prewitt and Marr-Hildreth filtering are applied to the same image, the laplacian method is the one that gives poorer results concerning noise followed by far by Roberts filtering. Another aspect to consider about Marr-Hildreth's technique results is that, due to the use

of a low pass filter, a noticeable distortion of the real position of the facial features is clearly distinguished.

In Figure 13, we can see how the association of Sobel and Prewitt filtering with vertical edge and horizontal edge one, respectively, makes just certain edges being shown: vertical edges in detriment to horizontal in first case and horizontal to vertical in second case.
Since Prewitt and Sobel provide us with the best results so far, we will use Sobel filtering to perform line tracking.

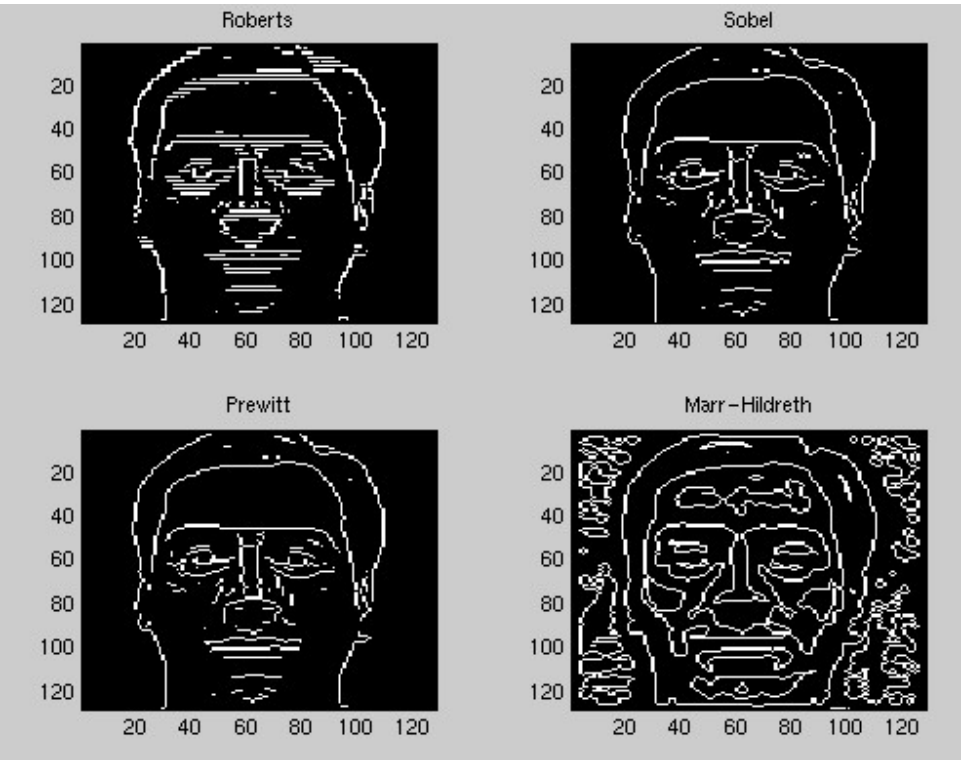


Figure 12: Comparison of different edge detection methods

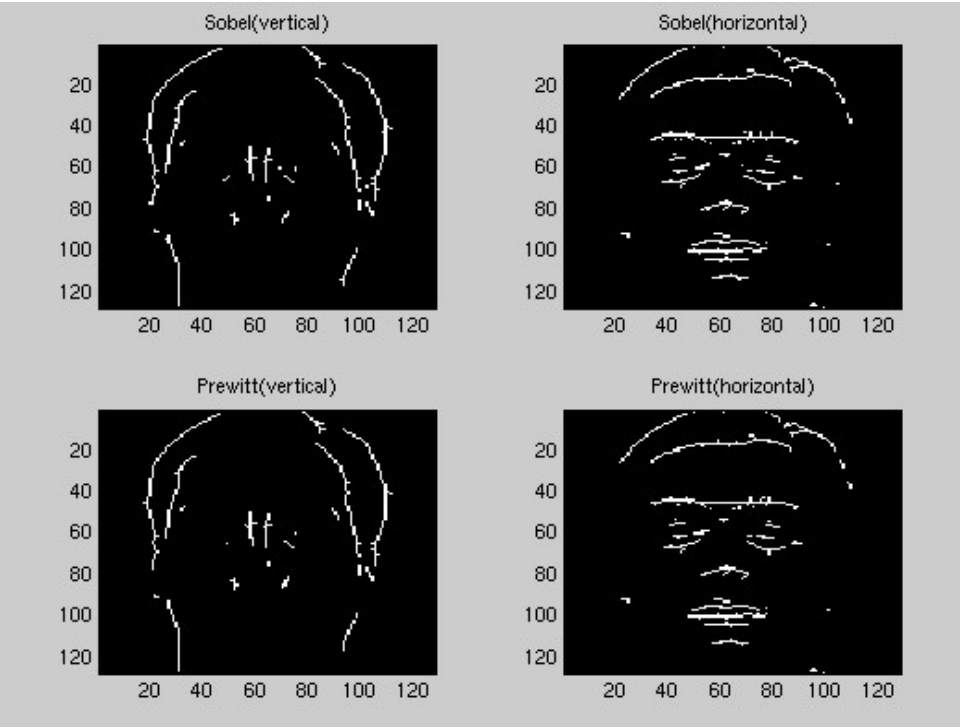


Figure 13: Association of Sobel and Prewitt filter with horizontal and vertical filter

2.3. Line Parameters tracking: Hough transformation

The Hough transform [3] is a standard tool in image analysis that allows recognition of global patterns in an image space by recognition of local patterns (ideally a point) in a transformed parameter space. It is particularly useful when the patterns one is looking for are sparsely digitized, have holes and/or the pictures are noisy.

The basic idea of this technique is to find curves that can be parameterized like straight lines, polynomials, circles, etc., in a suitable parameter space. Although the transform can be used in higher dimensions the main use is in two dimensions to find, e.g. straight lines, centres of circles with a fixed radius, parabolas $y = ax^2 + bx + c$ with constant c, etc.

In our case, it will be used to detect straight lines, so we will explain now the basic principle of the straight-line recognition:

First of all, we must apply any kind of edge detection technique, such as the ones we saw before and then thresholding to keep only pixels with a strong edge gradient, then the image may look like Figure 14.

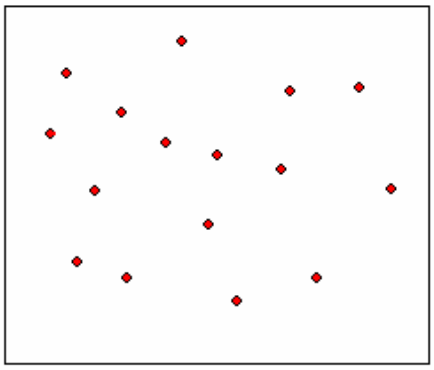


Figure 14: Example of results after an edge detection technique is applied (pixels marked in red for outlining)

A straight line connecting a sequence of pixels can be expressed in the form $y = ax + b$. If we are able to evaluate values for *a* and *b* so that the line passes through a number of the pixels that are set, then we have a usable representation of a straight line. The Hough transform takes the above image and converts into a new image (what is termed) in a new space. In fact, it transforms each significant edge pixel in (x,y) space into a straight line in this new space.

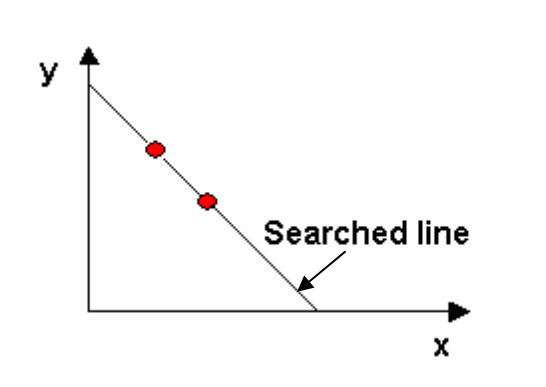


Figure 15: How Hough Transformation works searching for lines within pixels

Clearly, many lines go through a single point (*x,y*), e.g. a horizontal line can be draw through the point, a vertical line, and all the lines at different angles between these. However, each line will have a slope (*a*) and intercept (*b*) such that the above equation holds true. Then for every (*x,y*) point, each element that lies on the corresponding line in the (*a,b*) accumulator array can be incremented. So that after the first point in the (*x, y*) space has been processed, there will be a line in the (*a,b*) array. This plotting in the (*m, c*) array is done using an enhanced form of Bresenham's algorithm, which will plot a wide, straight line so that at the end crossing lines are not missed. How this algorithm works is explained shortly afterwards. At the end of processing all pixels, the highest value in the

(*a,b*) accumulator array indicates that a large number of lines cross in that array at some points (*a',b'*). The value in this element corresponds to the same number of pixels being in the straight line in the *(x,y)* space and the position of this element gives the equation of the line in the *(x,y)* space. .

2.3.1 BRESENHAM'S ALGORITHM

The main goal of this algorithm is to draw a segment between (x_o, y_o) and (x_1, y_1), with x_o, x_1, y_o, y_1 being integer pixel coordinates. This will be done with considerations of efficiency and robustness. The general idea is to start from (x_o, y_o) and to track the line from pixel to pixel using only integer arithmetics. For this, the equation of the segment y = ax + b will be used. The algorithm was first introduced by Bresenham [21]. A description can also be found in [22], Chapter 3.

1. First step is to subdivide the plane in eight sectors so that, for example, the first sector will be the following:

$$\begin{cases} x_1 - x_0 > 0 \\ y_1 - y_0 > 0 \\ a = \frac{y_1 - y_0}{x_1 - x_0} = \frac{\Delta y}{\Delta x} < 0 \end{cases} \quad (11)$$

2. Now we need to determine whether, considering we have the pixel (x_k, y_k) outlined or switched on , in our case because represents part of an edge, the algorithm should switch on $(x_k + 1, y_k)$ or $(x_k, y_k + 1)$. To answer this question, the error associated with each operation needs to be evaluated. The choice that minimizes this error is the correct one.

   - **Error evaluation between y and** y_k

$$e_1 = y - y_k = \frac{\Delta y}{\Delta x} \cdot (x_k + 1) + y_0 - y_k \quad (12)$$

   - **Error evaluation between y and** $y_k + 1$

$$e_2 = y_k + 1 - y = y_k + 1 - \frac{\Delta y}{\Delta x} \cdot (x_k + 1) + y_0 \quad (13)$$

We can state now, then, that:

$$\text{if } e_1 > e2, \text{ select } y_k + 1, \text{ else, select } y_k$$

3. For knowing if $e_1 > e_2$, we need to compute the sign of $e_1 - e_2$. Considering:

$$e_1 - e_2 = (2 \cdot \frac{\Delta y}{\Delta x} \cdot (x_k + 1) - 2y_k + 2y_0 - 1) \quad (14)$$

$$\text{Let } P_k = \Delta x \cdot (e_1 - e_2) :$$

$$P_k = 2\Delta y \cdot x_k - 2\Delta x \cdot y_k + \overbrace{2\Delta x \cdot y_0 - 2\Delta x)}^{\text{constant}} \quad (15)$$

$$P_{k+1} - P_k = 2\Delta y \cdot (x_{k+1} - x_k) - 2\Delta x \cdot (y_{k+1} - y_k)$$

Thanks to (1), we know that $x_{k+1} = x_k + 1$.
And if $(e_1 - e_2) > 0$, then $y_{k+1} = y_k + 1$.
if $(e_1 - e_2) < 0$, then $y_{k+1} = y_k$.

Hence:

$$P_{k+1} - P_k = \begin{cases} 2\Delta y & \text{if} \quad (e_1 < e_2) \\ 2\Delta y - 2\Delta x & \text{if} \quad (e_1 > e_2) \end{cases} \quad (16)$$

4. The only thing left is the explanation of how the algorithm is implemented. The algorithm 1 is given for the first eighth of the plane (11). The error is updated for each step using (16).
The global algorithm comes by considering symmetry of the problem. The changes to the algorithm are easily found by switching the *x* and *y*, decrementing x_k and / or y_k instead of incrementing. Special care is needed at boundaries (when $\Delta x = \Delta y$, $\Delta x = 0$, and $\Delta y = 0$).

Algorithm 1 Bresenham Algorithm (for $a \in]0,1[$)

Input
(x_0, y_0), (x_1, y_1): segment extremities (integer coordinates)
Begin
Δx : integer $\leftarrow x_1 - x_0$
Δy : integer $\leftarrow y_1 - y_0$
P : integer $\leftarrow 2\Delta y - \Delta x$
x_k : integer $\leftarrow x_0$
y_k : integer $\leftarrow y_0$
While $x_k \leq x_1$ **Do**
lighten (x_k, y_k)
$x_k \leftarrow x_k + 1$
If $P \leq 0$
$P \leftarrow P + 2\Delta y_k$
Else
$y \leftarrow y_k + 1$
$P \leftarrow P + 2\Delta y - 2\Delta x$
Endif
EndWhile
End

2.4. Algorithm for Line detection

In this section, an algorithm to detect playground lines will be explained. After its implementation, we will show the results obtained and to what extent it is robust enough.

First of all, we use the Sobel method to detect the edges and, therefore, convolve our video information matrix for every frame with the Sobel masks (4) and (5).

Once the convolution has been made the cardinal coordinates ‘x’ and ‘y’ are obtained. Pixels in which edges are present will be now set to 1 and rest of pixels to 0.But since we want to detect straight lines, we need to filter our results so that not all edges are outlined but just the ones that are interesting for us.

So we need to obtain the angle described by the two components so that we can detect whether the searched object is a straight line by checking the orientation of the pixels.

Once we have compared which pixels have similar angle, then we can obtain the pixels that are more likely to be straight lines. Also, two different thresholds will be fixed in order to filter the results and make them optimal:

- First threshold is to give some degrees of freedom to the difference in the angles, so that intervals in angles that are not completely ‘0’ can be taken into account. Best results were obtained for 0.02.
- Second threshold discriminates values in the black and white image that are inferior to 125. That way, very dark values can be taken apart since playground lines are usually unambiguously detected and thus white in the edge image.

Results although, of course, noise is still present, are much more optimal as they can be seen in the following figures 17 and 19.

Figure 16

Figure 17

Figure 18

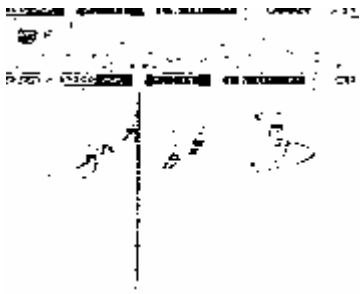


Figure 19

We can see below to what point thresholds are important which pixels we would miss or would take when they are not desired.

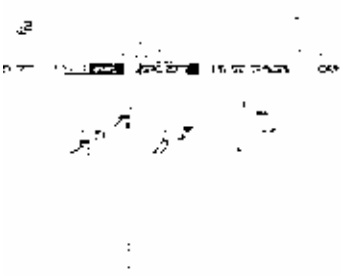


Figure 20
threshold2=150

Figure 21
threshold2=100

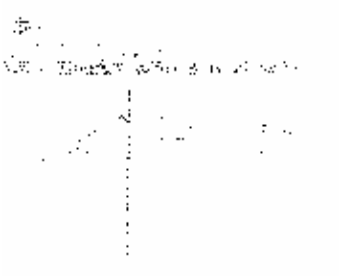


Figure 22
threshold1=0.0001

Figure 23
threshold2=150

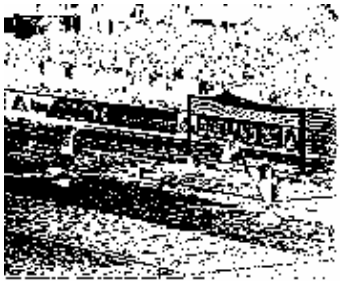


Figure 24
threshold2=100

Figure 25
threshold1: 0.0001

To end the algorithm, just one more step is left: we use function ‘Hough’, so that the line we are looking for is finally tracked. Results can be graphically seen in Figure 26, where we lines detected are outlined in green and beginning and end points are respectively yellow and red.

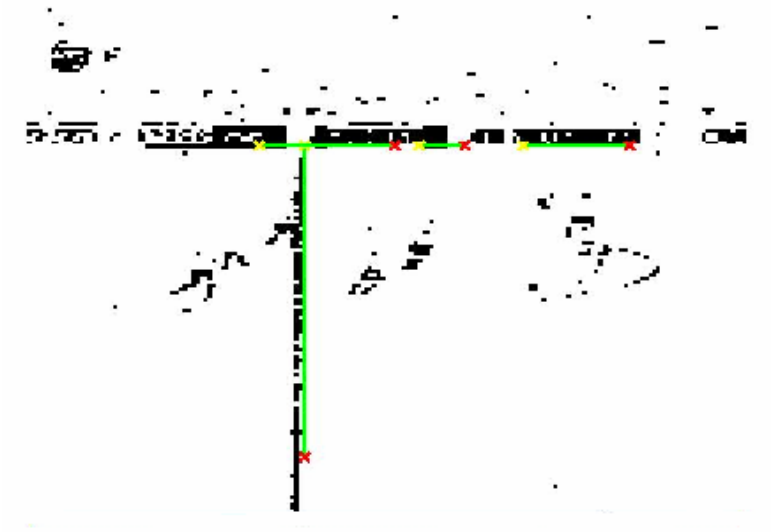


Figure 26: Line recognition performed by Hough transform

3. Temporal segmentation: scene changes

Recently, multimedia information has been made overwhelmingly accessible with the rapid advances in communication and multimedia computing technologies. The decreasing cost of digital storage, coupled with the fact that newer and better video/audio compression formats are being standardised, means that it is now possible for people to build large personal digital video libraries, similarly to such an extent as digital video ones. Such video libraries will be extremely difficult to organise and browse unless tools are available to the user to index and organise the content according to the key events and scenes depicted. A scene in a movie can be defined as a succession of individual shots that are semantically related, understanding a shot as an uninterrupted segment of a video frame sequence with static or continuous camera motion. A scene is an important retrieval unit for users, as they will enable them to search or browse movies based on particular scenes.

Scene detection, therefore, is a useful tool in order to organise the content into important and meaningful segments, being a key issue in high level video processing that is required for applications such as video indexing, browsing, video retrieval etc.

A number of methods for video scene change detection exist in the literature. The matching process between two consecutive frames is the essential part of it. Many of them use the low level global features such as the luminance pixel wise difference, luminance or colour histogram difference to compare two consecutive frames. We will further on explain these different methods in more detail, having grouped them in eight big groups: the based on pixel differences, sum of absolute differences, statistical differences, block differences, histograms differences, edge tracking and compression differences.

The algorithms we have implemented and will, then, be explained through this chapter aims to detect scene cuts in real-time. Since we want to use a non-complex method where optimal results for real time are obtained, we use SAD (Sum of Absolute Differences) to calculate differences between frames. However, different ways to calculate SAD will be discussed comparing which one performs better and showing which results are achieved. Detection of scene cuts will be determined by the setting of a threshold that will outline high peaks above the threshold and identify them as scene cuts. But the kind of threshold plays an important role in detection, so another comparison will be made, but this time between the convenience of using a fixed or a dynamic threshold.

3.1. Types of scene changes

Scene changes may occur in a variety of ways: cuts, gradual transitions such as cross-dissolves, fade-ins or fade-outs,or through various graphical editing effects, like wipes.

A scene cut is a video imagery in which consecutive frames are highly uncorrelated. It represents the simplest scene boundary of them all. We see an example in Figure 27.

Figure 27: Scene cut in four consecutive frames

Gradual transitions, as we said before, include dissolves, fade ins and fade outs.A dissolve is a process whereby one video signal is gradually faded out while a second image simultaneously replaces the original one as we can see in Figure 28. However, a fading is the act of dissolving a video picture to either a color, pattern, or titles. So the clip, then, changes from transparent to fully opaque (or viceversa) to fade in or out, as in Figure 29

Figure 28:Example of a dissolve in a video

Figure 29: Fading in four consecutive frames

Among graphical video effects for scene change, we have wipes. Wipes are video transitions in which the new video physically moves into the frame while displacing the old video.An example of wipes is shown in Figure 30.

Figure 30: Example of a wipe in a video

3.2. Techniques on scene change detection

Unfortunately, it is difficult to compare scene boundary detection methods based on published literature as few studies report quatitative results and those that do involve limited test sequences. Most of them target scene cut detections, even though recenter ones do research also in gradual transition ones. Anyway, the major techniques [4-6] that have been used for scene change detection are pixel differences, sum of absolute differences, block differences, statistical differences, histogram differences, edge tracking and compression differences. Hereafter,we will review them all, trying to discuss their robustness to camera and object motion.

3.2.1. Pixel differences

This method proves to be the easiest of them all, as its basis consists of counting the number of pixels that have changed considerably between two consecutive images, deciding if the difference is higher than a predefined threshold. After counting has been accomplished, we will check if the amount is enough to be considered a scene change comparing it with a second threshold. But the main problem of this method is that it suffers from the variations incurred by camera motion.

One approach of this method [6] adds a supplementary 3x3 averaging filter before the comparison to reduce camera motion and noise effects. Selecting a threshold manually adapted to the input sequence provided good results, although the method was somewhat slow. But the problem with this method is that it is not feasible for real-time applications, since the decision of which threshold to use can just be done once all the data have been received.

A second approach [8] computed what they call chromatic images by dividing the change in gray level of each pixel between two images by the gray level of that pixel

in the second image. But the main problem of this method is that it suffers from the variations incurred by camera and large-sized object motion.

3.2.2. Sum of absolute differences (SAD)

In this method, the absolute mean value of intensity difference between consecutive frames is computed and a large difference is assumed to be a scene change [15]. The difference is calculated from the intensity mean matrix.

$$SAD(n) = \sum_{i}^{N-1} \sum_{j}^{M-1} \left| F_n(i,j) - F_{n-1}(i,j) \right|$$

$$F_n = n^{TH} \quad frame\ of\ \ size\ M \times N \qquad (17)$$

In order to be homogeneous, this is the method we have used in our comparison of techniques at the end of the chapter. We have chosen it because of its simplicity, since our main purpose is to show the difference in results between using a fixed or a dynamic threshold and also between computation of sad calculated separately for the three colored matrix that compose an intensity image matrix or with the single gray or mean one.

The main problem of this method is that it is quite susceptible to noise.

3.2.3. Statistical differences

More robust methods [9-12] calculate statistical measures such as mean or standard deviation. This technique is based on the assumption that the frame intensity variances in a shot do not demonstrate large fluctuations while that of the neighboring shot will be different. Because it is possible for the variance to remain the same as the mean values differ, the use of both intensity mean and variance may be more robust. This method proves to work better when camera and object motion is present but is sensitive to object appearance and disappearance and fast pans and zooms.

In [13], a measure based on the mean and standard deviation of the gray levels in the image is computed. The image , then, is segmented into regions, so it is therefore a hybrid of statistical and block differences kind of method, and statistical measures of the pixels in the regions over the frames are compared. Although it proves to be quite tolerant to noise, it is slow due the complexity of the statistical formulas. It also generates many false detections.

3.2.4. Block differences

Unlike other previous methods, which may be susceptible to local changes, block differences methods have been proposed to handle local changes better. These start with block division for each frame and calculate the differences between the collocated or matching (in the sense of motion compensation) blocks in two frames.

$$Diff(n, n+1) = \sum_{K=1}^{m} W_k * DiffB(n, n+1, k)$$
$$W_k = Weights \ \ for \ the \ difference \ DiffB \qquad (18)$$
$$DiffB = difference \ between \ the \ k^{TH} \ \ block \ \ between \ consecutive \ \ frames$$

In [38], *DiffB* would be set to one when likelihood ratio between the collocated blocks exceeds a certain value and zero when it does not. The likelihood ratio is defined by the mean and variance of intensity features. All W_k values are equal.

One approach [7] consists of, after division of the images into regions has been made, finding the best match for each region in a neighborhood around the region in the other image. The pixels difference for each region was calculated and the weighted sum of the region differences provided the image difference measure. Gradual transitions were detected by generating a cumulative difference measure from consecutive values of the image differences. This functions good in block based video codecs (h.263, h.264, M-4), because we already know the differences and also directions (motion vectors). So, no complexity is added for calculation.

Other hybrid block difference and statistic difference based approaches were inspired by the block based motion compensation used in the ISO MPEG video compression standards. There, just residuals between the motion compensated blocks are encoded for efficient compression. In [16], each frame is partitioned into 12 blocks and for each one the matching block in the previous frame, which results in the minimum intensity difference, is found. Next, intensity difference values $DiffB(n, n+1, k)$ are computed, and deviations, such as the largest and the smallest differences, are trimmed and scene change decision is made by using the remaining blocks. In this scheme, W_k values correspond to the rank order. All in all, block difference algorithms only partly remove the sensitivity to object and camera as fast camera and object motion, for example, cause problems for these approaches. Furthermore, the computation of block statistics and the estimation of motion may be computationally costly.

3.2.5. Histograms differences

This is the most common method used to detect scene boundaries. Large differences between frames due to object and camera motion may result in intolerable number of false alarms for pixel, block and statistical based methods. A global representation, therefore, has been proposed by using color or intensity histograms. Due mainly to their insensitivity to camera and object motion, histogram representation is reported as the most robust scheme for scene change detection after extensive performance comparisons [14].

Histogram comparisons are usually based on one of the following three distance metrics: bin-to-bin difference (19), chi-square (20) and histogram intersection (21). Among these, histogram intersection is the best of them all. [23]. An interesting result that is observed during the same comparative analysis is that the use of intensity information in the color histograms improves the performance.

$$D(i,i+1) = \sum_{n}^{j=1} |H_i(j) - H_{i+1}(j)| \qquad (19)$$

$$D(i,i+1) = \sum_{n}^{j=1} \frac{|H_i(j) - H_{i+1}(j)|^2}{H_{i+1}(j)} \qquad (20)$$

$$D(i,i+1) = 1 - \frac{\sum_{n}^{j=1} min\,(H_i(j) - H_{i+1}(j))}{\sum_{n}^{j=1} max\,(H_i(j) - H_{i+1}(j))} \qquad (21)$$

with $H_i = i^{TH}$ *histogram of size N*

Histograms are robust to object and camera changes. On the other hand, they are sensitive to intensity variations, such as flashlight and shadow effects. Furthermore, as it is a global measure, two frames with considerably different content may have the same histogram, which results in higher amount of missed detections. One of example of videos in which that happens and, therefore, with which histogram differences does not perform well are sport videos. As a result, this technique was not used in our research.

3.2.6. Edge tracking

To fight against intensity variations, such as illumination changes due to flashlight, edge information can be used.

In [31], Zabih, Miller and Mai compared color histograms, chromatic scaling and their own algorithm based on edge detection. They aligned consecutive frames to reduce the effects of camera motion and compared the number and position of edges in the images. The percentage of edges that enter and exit between two frames was computed and scene changes were recognized by looking for large edge change percentages. Dissolves and fades were identified by looking at their relative values of the entering and existing percentages. The results seemed more accurate at detecting scene cuts than histograms and much less sensitive to motion than chromatic scaling. The main drawback of this edge based technique is the prohibitive computational cost because of motion compensation, edge detection and edge distance computation.

3.2.7. Compression differences

These scene change detection algorithms improve the efficiency in time and memory, since they require only minimal decoding as opposed to spatial domain techniques, which need full decoding of video. Different features are employed:

- DCT coefficients: Arman, Hsu and Chiu [25] found boundaries by comparing a small number of connected regions. By calculating DCT coefficients in JPEG compressed frames and finding differences between these coefficients as a measure of frame similarity. A final speedup was obtained by sampling the frames temporally and using a form of binary search to find the actual boundary. Potential ones were checked using a color histogram difference method. In [24], a subgroup of DCT coefficients for each I-frame are selected and the similarity of these coefficients with those of the next I-frame is calculated. A large frame-to-frame difference is to be considered a cut.

- DC sequences: B- and P-frame DC images cannot be obtained directly by DC coefficients unlike the case for an I-frame. To speed up DC sequence construction, a fast method is proposed in [25] for the P- and B-frame DC image computation by interpolating the DC values from the previously decoded frames and adding the residual. After a DC sequence, e.g., an 8x8 subsampled video, is obtained, any of the scene change detection methods previously presented can be used. In [25], color histograms are used, whereas intensity histograms are used in [26]. When compared [27] and although both methods are accurate and the most robust compressed domain scene boundary detection algorithms, the use of color histograms provides slightly better results.

- Macroblock types (MB-type): It is claimed that a shot-boundary on a B-frame or a P-frame results in conspicuous changes in the MB-type

features. For B-frames, most of the MBs will favor one direction whereas P-frame MBs will be intra-coded [28, 29].

- Size: In [30], difference in size of JPEG compressed frames was used to detect scene changes as a supplement to manual indexing system.

3.3. Thresholding problem

A feature vector computed by the methods that are explained in the previous section should be input to a classifier or a decision block that decides if the feature vector indicates a shot transition or not. In fact, most of the responsibility for succeeding in the video temporal segmentation process is in the thresholding phase as we will see in this chapter.

The easiest way of classification is using of a single global fixed threshold and considering the differences above the threshold as a scene change. The problem with this method is that since sport videos are diverse, a significant error rate is reported.
But the main trouble of using a fixed threshold is that it gets really difficult to find and optimal value, since the election of a too high one can lead to miss certain scene changes while choosing a too low threshold would make us obtain more scene changes than there actually are, that is to say, to false detections. As a result, Zhang et al [33] have proposed a twin-comparison method to detect gradual transitions. Twin-comparison uses two thresholds, high T_h and low T_l thresholds, to detect cuts and gradual transitions. All differences between frames above T_h are defined as cuts. The detection of gradual transitions begins by comparing consecutive frames using any video segmentation methods. Any frame having difference greater than T_l is marked as a potential start frame of a gradual transition. This frame is then compared to subsequent frames and the difference value is accumulated. During a gradual transition, this accumulated difference value will gradually increase. The end frame of the gradual transition is detected when the difference between consecutive frames decreases to less than T_l and the accumulated value has increased to a value greater than T_h .If the difference between consecutive frames drop below T_l before the accumulated value exceeds T_h, then the potential start frame is dismissed. Even though this technique performed better than the one with fixed threshold, it proves to be not good enough for videos like sport ones, where high motion is a fact.

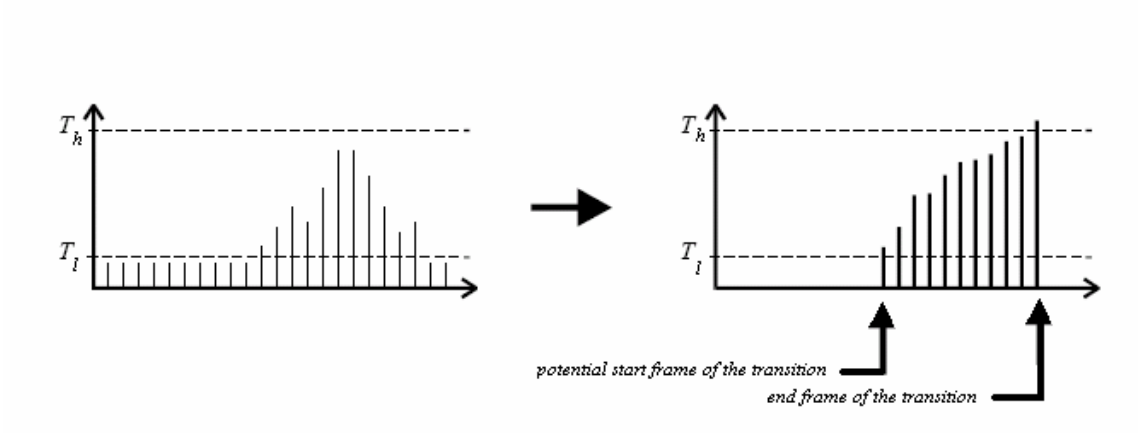


Figure 31: Detection of gradual transition in [33]

Since a global fixed threshold for every video seems to be not an optimal solution, next step was methods which tried to find an optimal static threshold automatically. This way, Yeo and Liu [18] proposed to use locally computed parameters because they assert that scene boundaries result from local activities. Also, another methods based on the Otsu method [34], entropy [35] or histogram differences [36] were used to calculate an adequate threshold.

Further approaches have considered the idea of using a dynamic threshold. In one of them, [37] the standard deviation is first computed and, in addition, a median filter is convolved with all the input data, resulting on a decision based on both deviation and convolution. Although satisfactory results were obtained, this technique is just valid for non-real time applications, as the filter needs as input also information about future frames. A quite recent method with dynamic thresholding [19] was presented where, once differences between frames was computed with SAD and for H.264 codec, a sliding window just with previous frames is used to compute the threshold. This way, this method is feasible for real-time applications. For each sliding window, media and standard deviation is calculated, so that the value of the threshold is obtained from a linear combination of these two parameters as well as the threshold used in the previous window. To avoid false detection immediately after a scene change, an exponentially decaying function based to lower the threshold is used for a certain number of frames. This method, because of its suitability for real-time applications like the ones this thesis aims to and its outstanding results added to its low complexity has been used compared in our thesis with other simpler methods, so that an accurate study of performances for different techniques can be exposed.

3.4. Comparison of methods

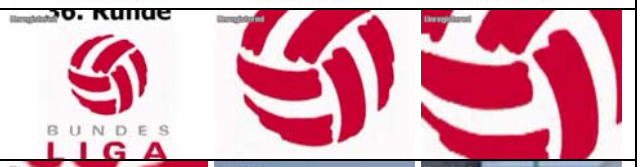
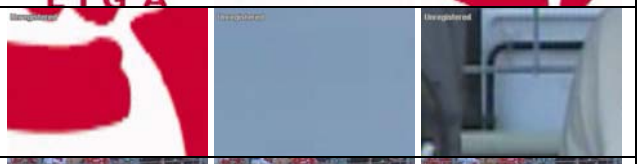
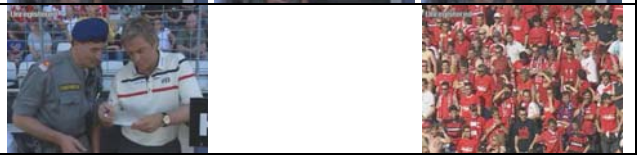
The target of our research is detection of scene cuts for real time applications. First of all an improvement of SAD will be proposed, where computation is made upon the three

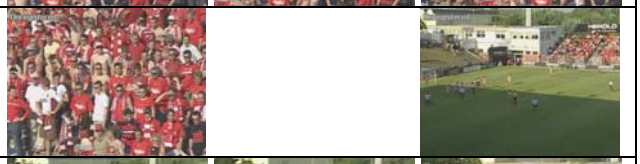
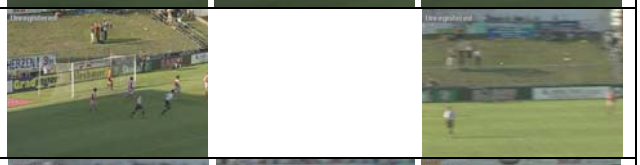
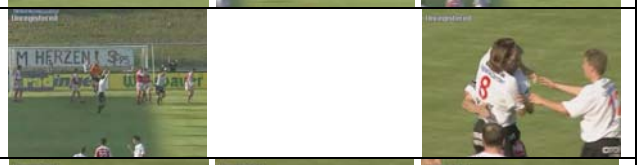
matrixes RGB, one for red intensity, one for green intensity and one for blue intensity, respectively, of which a full colored image is composed. Then, comparisons concerning the threshold will be done, where the method explained before [19] of dynamic threshold will be tested in a video and compared in recall and precision with what would be obtained for the same sequence with a fixed threshold.

The scene cut detection technique we will use is SAD. Its lack of complexity, the good results obtained for detection of scene cuts with it and the willingness to be homogeneous at our comparisons between dynamic and fixed thresholds have made us use this method in detriment to the others.

As we explained before in this chapter, SAD extracts the difference between consecutive frames subtracting the pixel in the image from the corresponding pixel in the previous one and adding all differences obtained to obtain one single value for each pair of frames (17). The simplest approach is calculating it directly in the gray or black and white matrix obtained with the means for each pixel and from the three values in the three matrixes RGB. The results obtained are the following for the first 500 frames of the video 'zibsport.avi'.

To be able to check the optimality of the graphs a full description of the video analyzed comes below:

FRAMES	***EVENT***	***SCREENSHOTS SAMPLE***
0-60	Still scene	
60-69	Fast zoom in	
68-70	Fading	
70-151	Still scene	
151-152	Scene cut	

152-240	Pan	
239-240	Scene cut	
250-310	Pan	
327-328	Scene cut	
328-390	Pan	
452-453	Scene cut	
465-500	Zoom in	

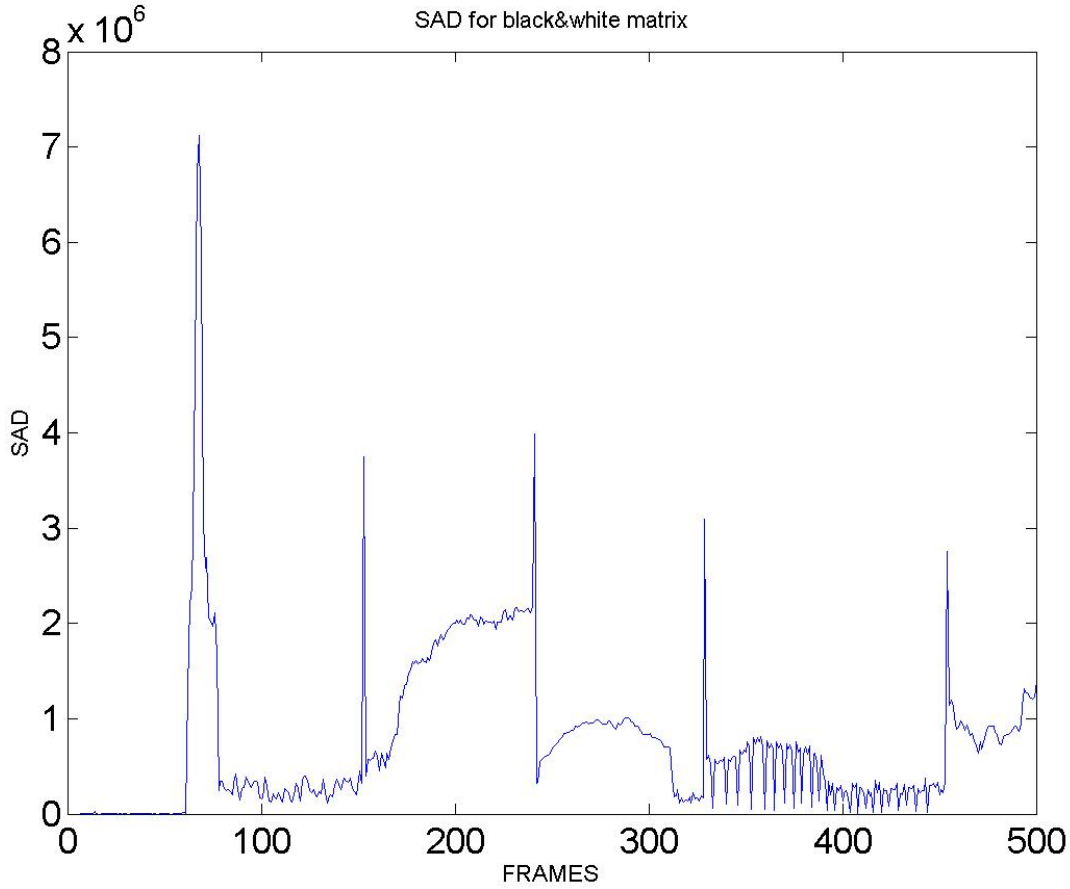


Figure 32: Graph of SAD calculated from the gray scale video over the frame number for the 'zibsport.avi' sequence.

The problem we encounter with this simple method is that a high value in one of the three matrixes and a low value in another for the same pixel gives the same result in the gray matrix as an average value for both cases. For that reason, not that accurate results are obtained in this way. If we perform the calculation of SAD for each of three matrixes separately, we obtain three different SAD values . We have used four different metrics to obtain the final value of SAD. That way, we can check which gives the best results.

- $SAD = SAD_{red} + SAD_{blue} + SAD_{green}$ (22)
- $SAD = \max\left[SAD_{red}, SAD_{blue}, SAD_{green}\right]$ (23)
- $SAD = \text{mean}\left[SAD_{red}, SAD_{blue}, SAD_{green}\right]$ (24)

- $SAD = \sqrt{SAD_{red}^2 + SAD_{blue}^2 + SAD_{green}^2}$ (25)

The first metric (22) , that is, the sum of the three SADs corresponding to the three color components, provides these results:

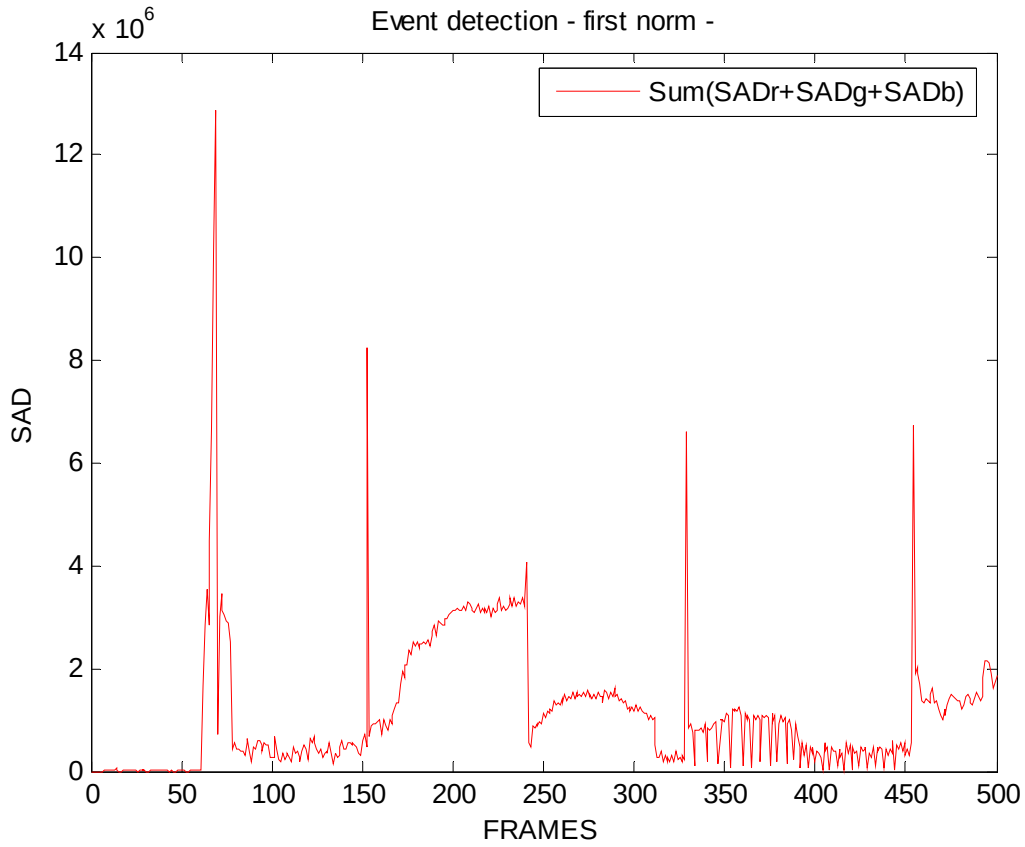


Figure 33: Graph of SAD computed with the sum of the three SADs for the ' zibsport.avi' sequence

The metric where the maximum of all three is taken (23) gives this output:

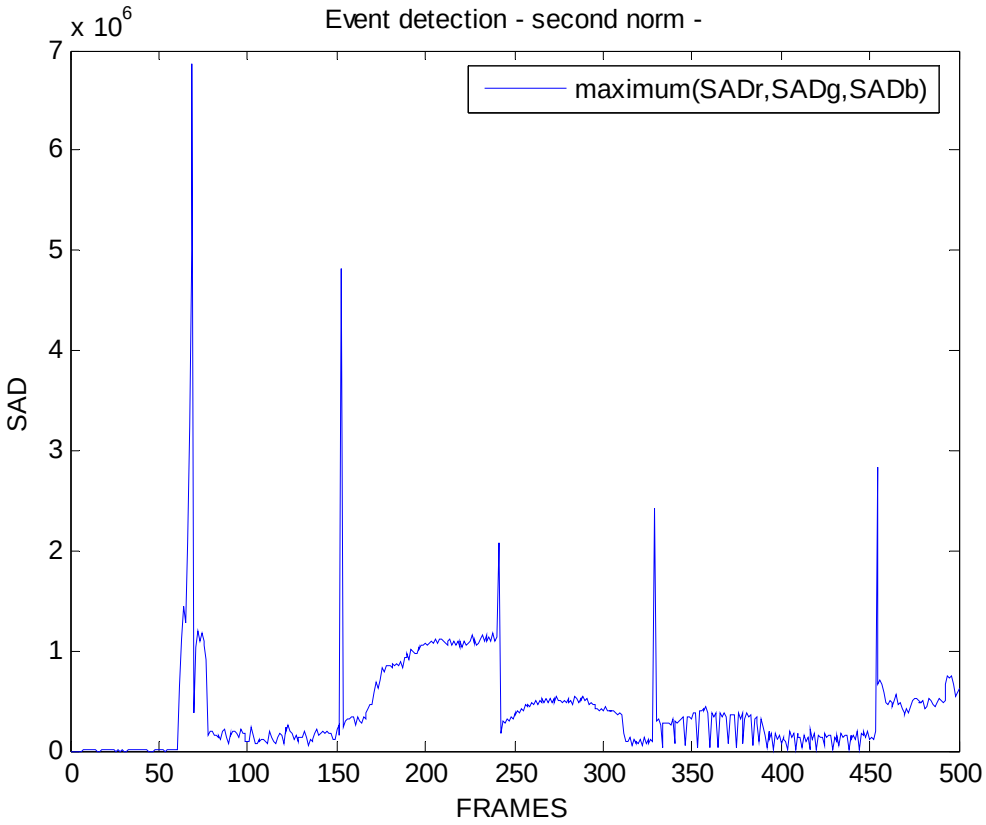


Figure 34: Graph of SAD computed with the maximum of the three SADs for the 'zibsport.avi' sequence

The third metric (24), where we compute the mean, is used with these results:

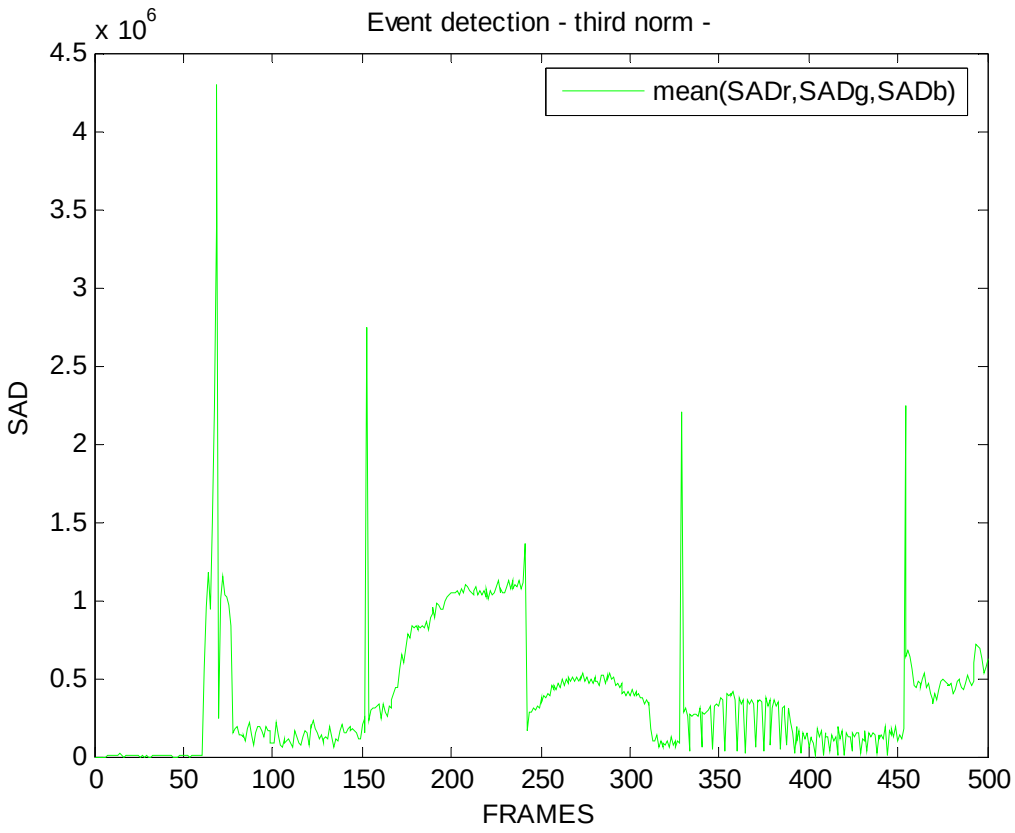


Figure 35: Graph of SAD computed with the mean of the three SADs for the 'zibsport.avi' sequence

Last metric, sum of squared differences (25) gives this output:

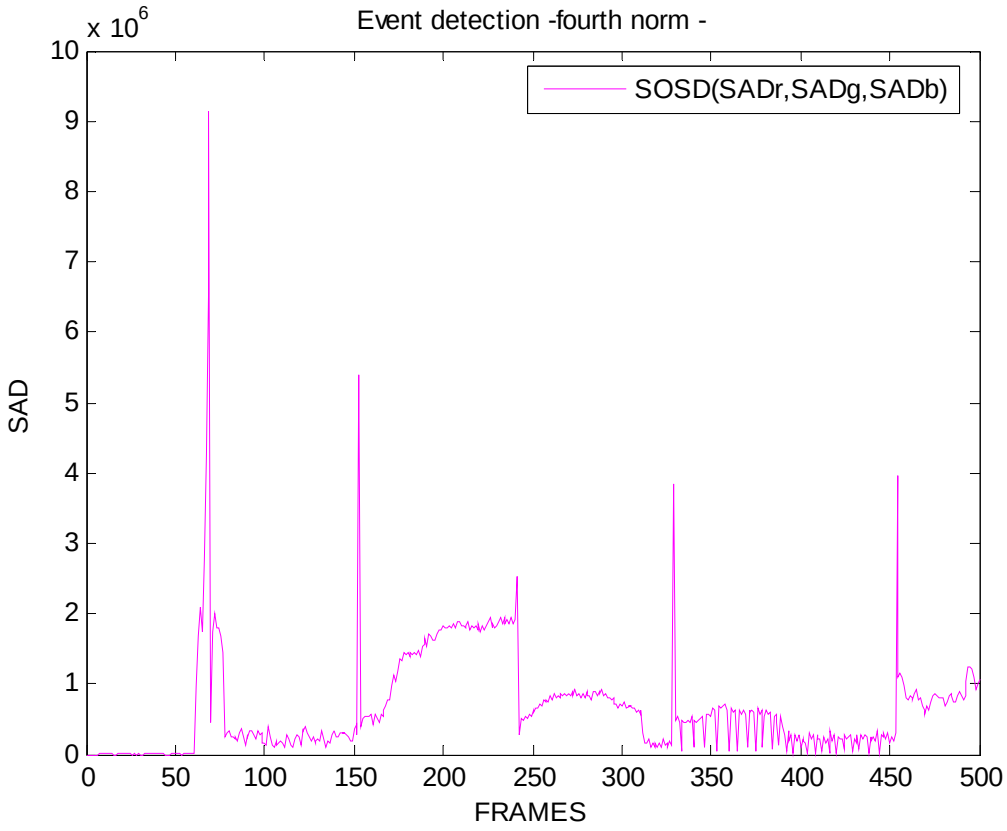


Figure 36: Graph of SAD calculated with the sum of absolute differences for 'zibsport.avi'.

For analyzing the graphs, we state that the higher the values obtained are, the better our results are, since that means detection is, then, easier. Also, narrow and clear peaks is also a characteristic of a nice scene cut recognition figure, for they can be easily distinguished within the sequence. When the comparison of the four metrics is made we see the best results are achieved with the first metric (22), followed by the fourth one or sum of squared differences (25) and equally worst results are obtained with both second (23) and third metrics (24).We see everything clearly in the following image:

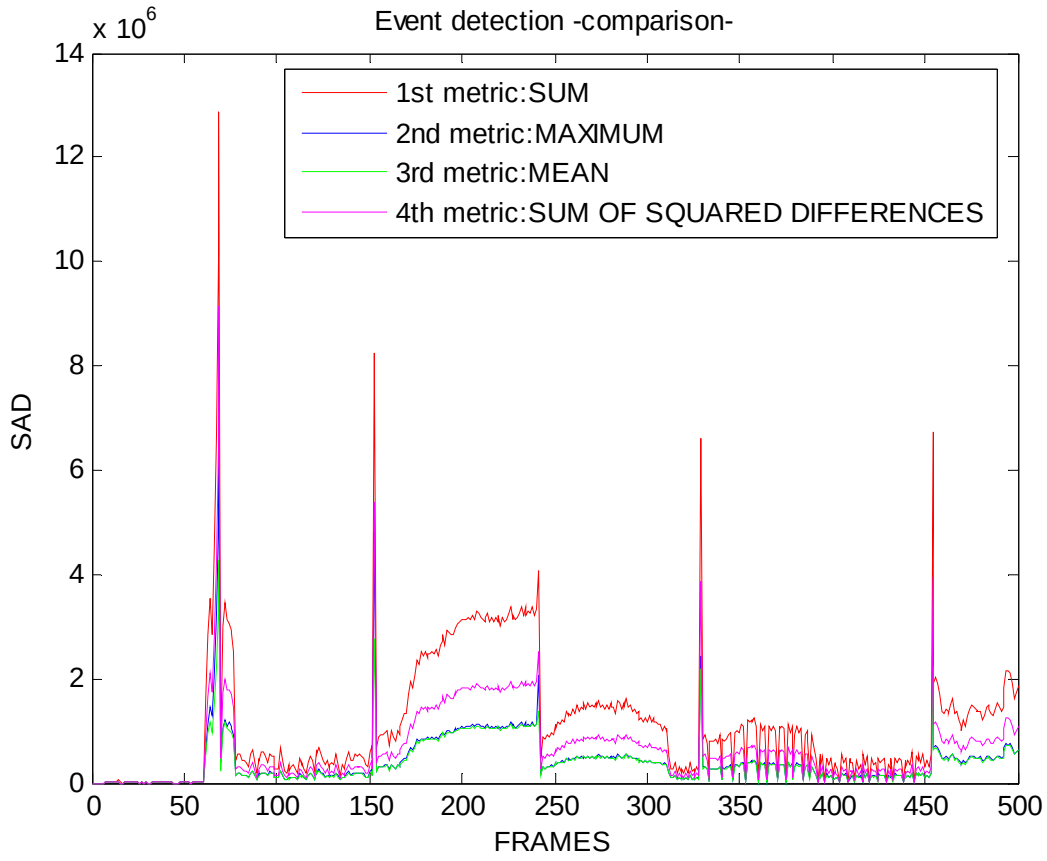


Figure 37: Comparison of the 4 metrics for 'zibsport.avi'

Detection, however, cannot be done, as we explained, without a threshold. First of all, we set a fixed global threshold for that so that values above it will be considered as a scene cut. To see how each calculation of SAD performs with fixed thresholds (the one with gray values matrix and the one with the best metric of RGB matrix), we calculate the recall and the precision parameters defined as follows.

Recall and precision are two accepted measurements to determine the utility of an information retrieval system or search strategy. The first one refers to completeness of retrieval, so that the capability of the technique to detect as many events as possible is measured. However, precision gives information about the purity of the retrieval or, in other words, how good the method performs when having to filter distinguish from false to correct detections. Depending on the application, a better value from one or the other or, sometimes, a trade off between the two of them is desired. The relative importance of recall and precision, in particular the latter, depends on the level of the user. Experienced users can work with high recall and low precision, as they are capable of sifting through the information and easily reject irrelevant information. Novice users, on the other hand, need higher precision because they may lack the judgment the experienced audience has. A different way to determine the fit for the task of an information retrieval system is to assess the risk tolerance of the environment. If the tolerance to errors is high and the assignment is not time-critical, it may be acceptable to do false scene cut recognition if as

many correct detections are possible are fulfilled. However, if time is important, and the cost of making a mistake is high, then higher precision is required.

$$\text{Recall} = \frac{N_C}{N_C + N_M} \quad (26) \qquad \text{Precision} = \frac{N_C}{N_C + N_F} \quad (27)$$

where:

N_C = *number of correct* detections

N_F = *number of false* detections

N_M = *number of missed detections*

Zibsport.avi	TRUE	FALSE	MISSED	RECALL	PRECISION
Fixed threshold(GRAY)	29	10	7	0.81	0.74
Fixed threshold (RGB)	32	8	6	0.84	0.8

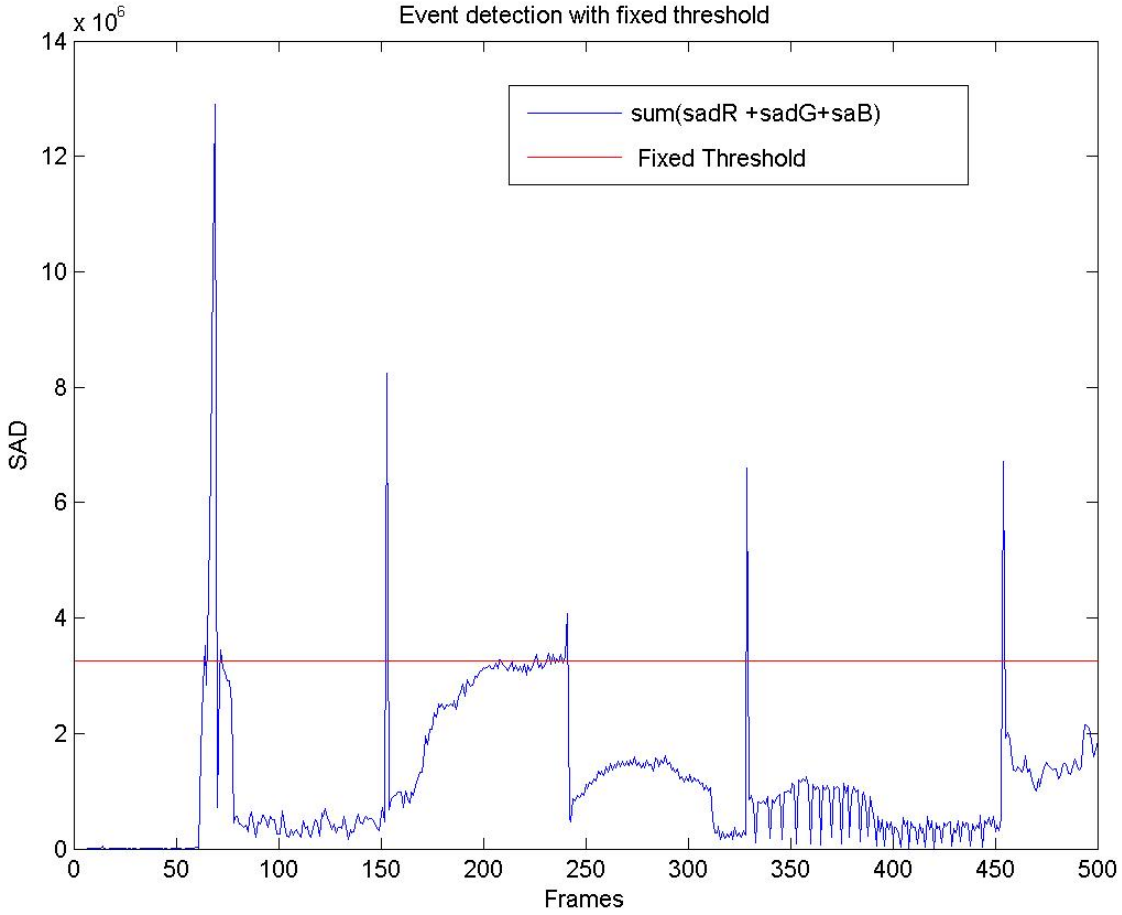


Figure 38: Graph of SAD calculated with the sum of absolute differences and fixed threshold for scene cut detection in ' zibsport.avi'

We can see in the table that the fact of calculating SAD separately for each matrix enables a slight improvement in detection of scene cuts. In spite of that, the use of fixed threshold avoids results as optimals as the ones that can be reached with the dynamic threshold we explained previously in this chapter [19]. Recall and precision for the same video and also for the same number of frames has also been computed below, so that effectiveness of dynamic threshold is proved.

Zibsport.avi	TRUE	FALSE	MISSED	RECALL	PRECISION
Fixed Threshold (GRAY)	29	10	7	0.81	0.74
Fixed threshold (RGB)	32	8	6	0.84	0.8
Dynamic threshold	44	1	1	0.98	0.98

4. Temporal segmentation: Zooming and Pan detection

Content based temporal segmentation is mostly achieved by camera shot detections. Some effects, such as cuts, fades, dissolves and wipes are used to describe when a transition occurs from one shot to another or, in other words, as scene changes. However, camera motion effects such as zooming, panning / tilting, dollying and tracking/booming are used to more efficiently capture the visual features of the scene during a long shot. As camera operations usually explicitly reflect how the attention of the viewer should be directed, the clues obtained are useful also for example for key frame selection at the encoder. For instance, when a camera pans over a scene, the entire video sequence belongs to one shot but the content of the scene could change substantially, thus suggesting the use of more than one key frame. Also, when the camera zooms, the images at the beginning and end of the zoom may be considered as representative of the entire shot. Furthermore, recognizing camera operations allows the construction of salient video stills, static images that efficiently represent video content.

In this chapter, we introduce a method for zoom and pan detection. As we will see, detection of camera motion effects cannot be accomplished with the techniques used for scene change detections since difference between frames will not be as high: in the case of zooming because it happens within the same image, and in panorama because recording is really slow so differences between frames are harder to detect. As a result, new features must be used for these camera effects. Our proposal is obtainment of motion vectors. How they are calculated and the steps that are followed for accurate detection will be explained in this chapter.

4.1. Camera motion effects

First of all, we will make clear what exactly some camera operations are. We will distinguish between [40] zooming, dollying, panning and tiling.

Zooming is the act of changing the scale of a display, generally without changing the amount of screen space it occupies. A zoom in enables the view of the recorded feature or scenery in a more detailed way whereas a zoom out brings a more general sight of it. An example of zooming in can be seen in the Figure 39.

Figure 39: Zoom in

Dollying is similar to zooming, in that the camera moves in toward or away from a subject. However, a dolly shot doesn't use the zoom control at all. In feature film shoots, a dolly is a platform that holds the camera and rides on rails similar to railroad tracks. When filming, one or more people push the dolly, resulting in a smooth shot. When you zoom, the camcorder's lens is simulating the appearance of moving closer to your subject. When dollying, you're moving the camera physically closer. If we compare both effects, the main difference is that zooming does not produce a perspective shift, whereas it does for dollying. It is especially noticeable by looking at the background of the image.

Panning and tilting are similar techniques with which the camera is used to slowly scan a subject horizontally or vertically respectively (see the Figure 41) .In Figure 40 a screenshot of a panorama sequence can be seen.

Figure 40: Pan / panorama shot

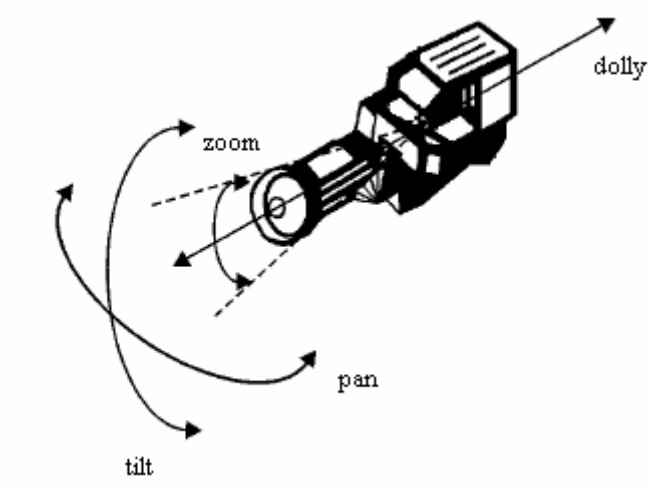


Figure41: Camera motion effects

4.2. Camera Motion Effects detection

Our focus in this chapter, as we said before, is the recognition of camera motion effects, zooming/dollying and panning/tilting concretely. But, since the difference between frames is not that big for those cases, the algorithm we used for scene cut detection does not offer sufficient results. As we can see in the Figure 42, SAD analysis detects an apparent gradual change between frames. However, it will never inform us of whether it is exactly a pan or a zoom what is being detected because they both look the same in the picture. Furthermore, these shots may lead to the false scene cut detections as can be seen in Figure 42.

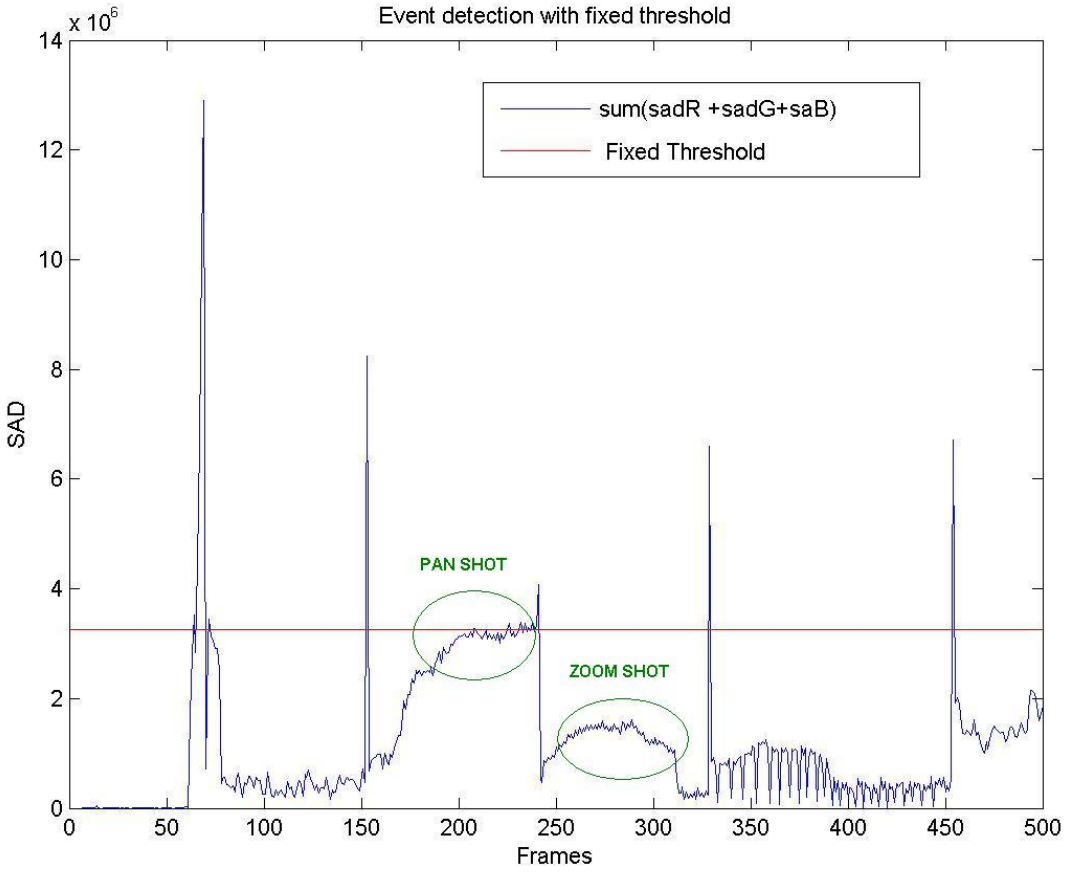


Figure 42: Performance of scene cut detection method for pan and zoom

Other features, therefore, should be taken into account if we want to do an accurate detection. Estimation of motion is the basis of the approaches that are used for this task.

4.3. Motion detection

Let us begin by considering one of the long-standing problems of motion perception: the phenomenon of apparent motion, by which a rapid sequence of movie frames gives rise to the impression of motion. A unified view has emerged that combines insights from several converging lines of investigation. In this view, motion is conceptualized as a kind of slant or orientation in space-time. Any system that is designed to detect this space-time orientation will inevitably experience the "illusion" of apparent motion in the right conditions [39]

There are many types of motion-related problems from the practical point of view. Motion detection is the simplest problem. Moving object detection and localization represent another set of problems. Camera motion produces global motion field in the picture and often an application is only interested in detecting camera motion. The most complex problem is when both the camera and relevant objects are moving, which is the case for the soccer video sequences.

Even though motion analysis is often called dynamic image analysis, it is frequently based on a small number of consecutive frames, sometimes just two or three in sequence. This case is similar to an analysis of static images, and the motion is actually analyzed at a higher level looking for correspondence between pairs of points of interest in sequential images.

The initial hypothesis in measuring image motion,called temporal smoothness, is that the intensity structures of local time - varying image regions are approximately constant under motion for at least a short duration [40]. Formally, if $f(x, y, t)$ is the image intensity function, then:

$$f(x,y,t) \approx f(x+\partial x, y+\partial y, z+\partial z) \quad (28)$$

Expanding this equation in a Taylor series yields:

$$\begin{aligned} f(x+\partial x, y+\partial y, z+\partial z) &= f(x,y,t) + \frac{\partial f}{\partial x}\partial x + \frac{\partial f}{\partial y}\partial y + \frac{\partial f}{\partial t}\partial t + \mathrm{O}(\partial^2) = \\ &= f(x,y,t) + f_x \partial x + f_y \partial y + f_t \partial t + \mathrm{O}(\partial^2) \end{aligned} \quad (29)$$

We can assume that immediate neighborhood of (*x*, *y*) is translated by some small distance (*dx*, *dy*) , called motion vector, during the time interval *dt*; that is, we can find *dx*, *dy*, *dt* such that:

$$f(x+\partial x, y+\partial y, z+\partial z) = f(x, y, t) \quad (30)$$

If *dx*, *dy*, *dt* are very small, the higher order terms in equation (29) vanish and dividing by *dt :*

$$-f_t = f_x \frac{\partial x}{\partial t} + f_y \frac{\partial y}{\partial t} \qquad (31)$$

The goal is to compute the velocity:

$$c = \left(\frac{\partial x}{\partial t}, \frac{\partial y}{\partial t} \right) = (u, v) \qquad (32)$$

f_x, f_y, f_t can be computed or, at least, approximated, from $f(x, y, t)$. The motion velocity can then be estimated as:

$$(33) \quad -f_t = f_x u + f_y v = grad(f)c$$

where *grad*(*f*) is a two-dimensional image gradient. Equation (33) does not specify the velocity vector completely; rather, it only provides the component in the direction of the highest gradient. This phenomenon is known as the aperture problem. For example, the velocity of a surface that is homogenous or containing texture with a single orientation cannot be recovered optically.

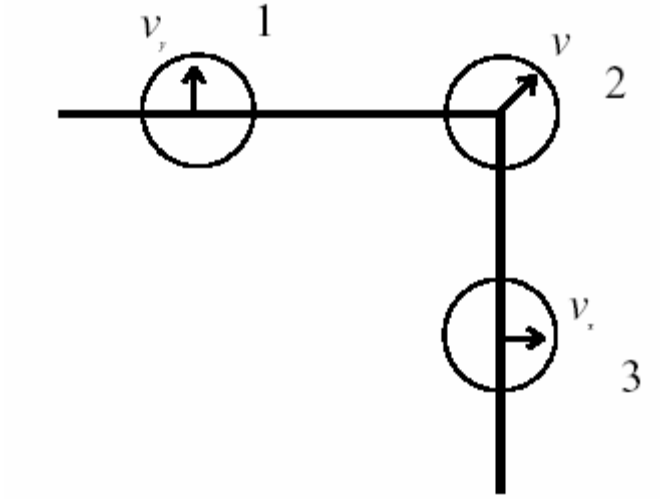


Figure 43: Aperture problem, ambiguous motion [43]

Through apertures 1 and 3 only normal motions of the edges forming the square can be estimated, due to a lack of local structure. Inside aperture 2, at the corner point, the motion can be fully measured as there is sufficient local structure; both normal motions are visible.

Optical flow is only approximation of the true motion in the image. There are many problems in calculation and interpretation of optical flow. These problems are extensively discussed in [43]. In this paper there is also a survey of different methods of calculating the optical flow, which is mostly important for the 3D scene representation. Different methods fall typically into categories: intensity-based differential methods, multi-constraint methods, frequency-based methods, correlation-based methods, multiple motion methods, and temporal refinement methods. The boundaries between each class of methods are not always clear [43]. Calculation of the complete optical flow in the image is computationally very complex and time consuming operation. Hence, in many real world applications complete optical flow is not an option and many different approaches have been developed, which are either application specific or approximations of the actual optical flow.

In scene change detection, object and camera motion can result in large differences that can be mistaken for shot breaks. Object and camera motion characteristics can also be used for classifying the scene type.

4.4. State of the art in camera operation recognition

As stated previously, at the stage of temporal video segmentation gradual transitions have to be distinguished from false positives due to camera motion. In the context of content-

based retrieval systems, camera operation recognition is also important for key frame selection, index extraction, construction of salient video stills and search narrowing.

Historically, motion estimation was extensively studied in the field of computer vision and image coding and used for tracking of moving objects, recovering of object movement, and motion compensation coding. Below we review approaches for camera operation recognition related to shot partitioning and characterization.

4.4.1 Methods based on MVs

Before explaining methods where motion vectors are used for detection, we should, first of all, make clear what they are and how they can be computed.

4.4.1.1. What are Motion Vectors (MV)

Motion vectors are patterns that provide estimation of the velocity of the pixels between two frames in magnitude and direction**.** Camera movements exhibit specific patterns in the field of MVs, as shown in Figure 44 and 45. Therefore, many approaches for camera operation recognition are based on the analysis of MV fields.

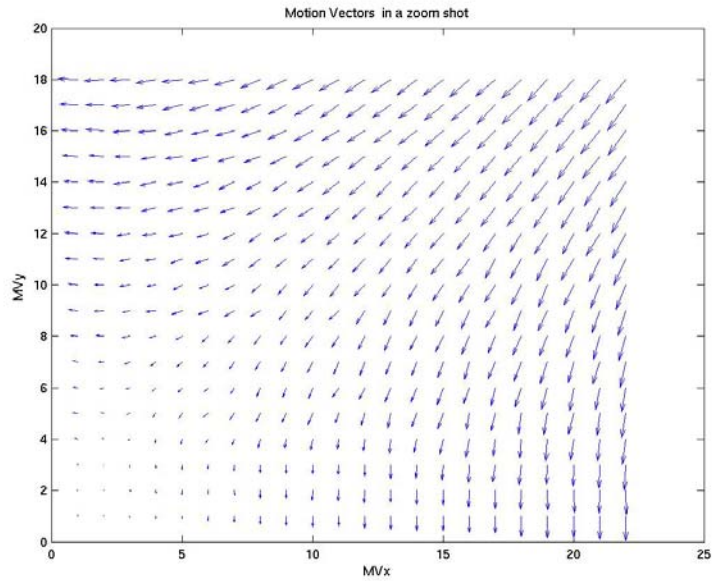


Figure 44: Motion Vectors in a zoom shot

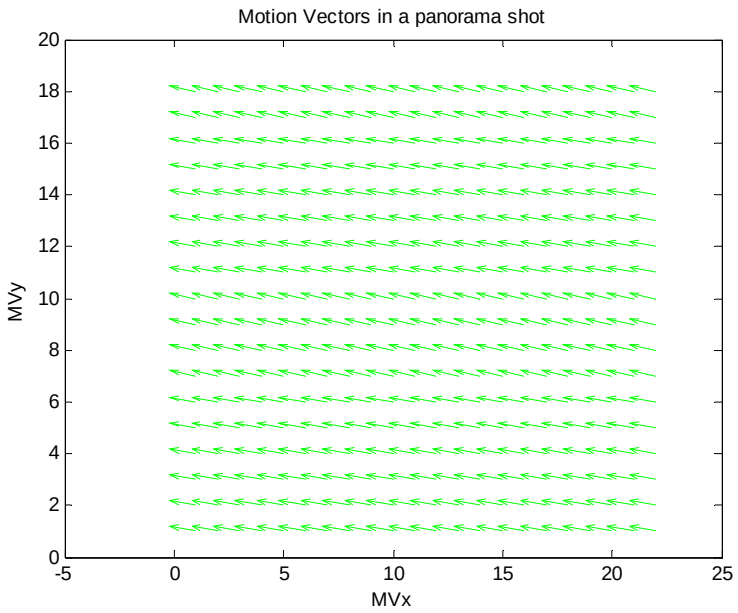


Figure 45: Motion Vectors in a panorama shot

4.4.1.2. How to calculate MV

The easiest way to compute motion vectors consist of dividing the image in blocks (B) of pixels and check the correspondence of every block with a searching region (SR) in the previous frame. To determine the searching region, normally, the blocks in the same position and around mine but in the previous frame are taken. The current block is then compared to all the blocks in the searching region and the least of all differences is then computed. The indexes of the block where that happens correspond to the end coordinates of the motion vectors. We can see it mathematically described in the equation below:

$$[MVx,MVy] = \arg\min_{x,y\in SR} \sum_{i,j\in B} \left| F_{n-1}(x+i,y+j) - B(i,j) \right| \qquad (34)$$

where F_{n-1} corresponds to the previous frame and B to the current block.

4.4.1.3. Methods with MV for camera operation detection

Zhang *et al.* [42] apply rules to detect pan/tilt and zoom in/zoom out. During a pan most of the MVs will be parallel to a modal vector that corresponds to the movement of the camera. This is expressed by the following inequality:

$$(35) \quad \sum_{b=1}^{N} \left| \theta_b - \theta_m \right| \le T$$

Where θ_b is the direction of the MV for block *b*, θ_m is the direction of the modal vector, *N* is the total number of blocks into which the frame is partitioned and *T* is a threshold near zero. In the case of zooming, the field of MVs has focus of expansion (zoom in) or focus of contraction (zoom out). Zooming is determined on the basis of "peripheral vision", i.e. by comparing the vertical components v_k of the MVs for the top and bottom rows of a frame, since during a zoom they have opposite signs. In addition, the horizontal components u_k of the MVs for the left-most and right-most columns are analyzed in the same way. Mathematically these two conditions can be expressed in the following way:

$$\begin{aligned} \left| v_k^{top} - v_k^{bottom} \right| &\geq \max(\left| v_k^{top} \right|, \left| v_k^{bottom} \right|) \\ \left| u_k^{left} - u_k^{right} \right| &\geq \max(\left| u_k^{left} \right|, \left| u_k^{right} \right|) \end{aligned} \qquad (36)$$

When both conditions are satisfied, a zoom is declared. The problem is that a combination of zoom and pan is difficult to detect, since it considers them separately.

Another approach [39]evaluates the orientation of block based motion vectors to identify global pan and zoom motion.For that, first of all each video frame is partitioned in blocks of 8x8 pixels. For each block, the angle of the motion vector is computed and, by clustering of the MV, regions of angles with the same orientation are described. The biggest and second biggest regions are obtained so that the percentages covered by both of them separately are computed. Also, the overall standard deviation of the largest region is calculated. Once we have these three parameters, the detection is done by fixing three different thresholds: the first one to check that the standard deviation is below one value in all zoom and pan detections for every frame, the second to check that the sum of both percentages is above that threshold in case of pan and the third to check that this sum of percentages in zoom detections is below this third quantity.This method is the basis we have used in the development of ours, where we make detection simpler with also optimal results.

4.4.2. Methods based on the Hough Transform and MV

Akutsu *et al.* [43] characterize the MV patterns corresponding to the different types of camera operations by two parameters: the magnitude of MVs and the divergence/convergence point. The algorithm has three stages. During the first one, a block matching algorithm is applied to determine the MVs between successive frames. Then, the spatial and temporal characteristics of MVs are determined. MVs are mapped

to a polar coordinate space by the Hough transform. A Hough transform of a line is a point. A group of lines with point of convergence/divergence (x_o, y_o) is represented by a curve $\rho = x_o \sin\varphi + y_o \cos\varphi$ in the Hough space. The least squares method is used to fit the transformed MVs to the curve represented by the above equation. There are specific curves that correspond to the different camera operations, e.g. zoom is characterized by a sinusoidal pattern, pan by a straight line. During the third stage these patterns are recognized and the respective camera operations are identified. The approach is effective but also noise sensitive and with high computational complexity.

4.4.3. Methods based on Decision Trees and MV

An alternative approach for detecting camera operations is proposed by Patel and Sethi [44]. They apply induction of *decision trees* (DTs) [45] to distinguish among the MV patterns of the following six classes: stationary, object motion, pan, zoom, track and ambiguous. DTs are simple, popular and highly developed technique for supervised learning. In each internal node a test of a single feature leads to the path down the tree towards a leaf containing a class label. To build a decision tree, a recursive splitting procedure is applied to the set of training examples so that the classification error is reduced. To classify an example that has not been seen during the learning phase, the system starts at the root of the tree and propagates the example down the leaves. After the extraction of the MVs from the MPEG stream, Patel and Sethi generate a 10-dimensional feature vector for each P frame. Its first component is the fraction of zero. MVs and the remaining components are obtained by averaging the column projection of MV directions. The results in [45] have shown high classification accuracy at a low computational price. We note that as only MVs of P frames are used, the classification resolution is low. In addition, there are problems with the calculation of the MV direction due to the discontinuity at 0º/360º.

The above limitations are addressed in [46] where a neural network supervised algorithm is applied. Given a set of pre-classified feature vectors (training examples), *Learning Vector Quantization (LVQ)* [47] creates a few prototypes for each class, adjusts their positions by learning and then classifies the unseen examples by means of the nearest-neighbor principle. While LVQ can form arbitrary borders, DTs delineate the concept by a set of axis-parallel hyperplanes which constrains their accuracy in realistic domains. In comparison to the approach of Patel and Sethi, one more class (dissolve) is added and the MVs from both P and B frames are used to generate a 22-dimensional feature vector for each frame. The first component is calculated using the number of zero MVs in forward, backward and interpolated areas. Then, the forward MV pattern is subdivided in 7 vertical strips for which the following 3 parameters are computed: the average of the MV direction, the standard deviation of the MV direction and the average of MV modulus. A technique that deals with the discontinuity of angles at 0º/360º is proposed for the calculation of the MV direction. Although high classification accuracy is reported, it was found that the most difficult case is to distinguish dissolve from object motion. In [48] MV patterns are classified by an integration between DTs and LVQ. More specifically, DTs are viewed as a feature selection mechanism and only those parameters that appear

in the tree are considered as informative and used as inputs in LVQ. The result is faster learning at the cost of a slightly worse classification accuracy.

4.4.4. Methods based on Spatiotemporal Analysis

Another way to detect camera operations is to examine the so called spatiotemporal image sequence. The latter is constructed by arranging each frame close to the other and forming a parallelepiped where the first two dimensions are determined by the frame size and the third one is the time. Camera operations are recognized by texture analysis of the different faces.

In [49] video X-ray images are created from the spatiotemporal image sequence. Sliced *x-t* and *y-t* images are first extracted from the spatiotemporal sequence and are then subject to an edge detection. The process is repeated for all *x* and *y* values, the slices are summed in the vertical and horizontal directions to produce gray-scale *x-t* and *y-t* video X-ray images. There are typical X-ray images corresponding to the following camera operations: still, pan, tilt and zoom. For example, when the camera is still, the video X-ray show lines parallel to the time line for the background and unmoving objects . When the camera pans, the lines become slanted; in the case of zooming, they are spread. We should note that performing edge detection on all frames in the video sequence is time consuming and computationally expensive.

In [50] the texture of 2-dimensional spatiotemporal (2DST) images is analyzed and the shots are divided into sub-shots described in terms of still scene, zoom, and pan. The 2DST images are constructed by stacking up the corresponding segments of the images. The directivity of the textures are calculated by computing the power spectrum by applying the 2-dimensional discrete Fourier transform.

4.5. Algorithm for zoom and pan detection

Our approach consists of a method for zoom and pan detection based on motion vectors. Its basis lies on the method [39] described before and created by Dumitras *et al.*, although the ours combines simplicity, that implies less computationally costly and therefore fast execution and good results.

These are the steps that should be followed in its implementation:

1. First of all, each frame is divided in macroblocks. We define a macroblock as a region containing a fixed number of pixels. Since a QCIF resolution is used, we obtain 396 macroblocks for each frame.

2. Secondly, we compute the motion vectors for each macroblock the way we explained before in Equation 7. To compute the difference, Sum of Absolute

Differences (SAD) is utilized. We use this method for its computational simplicity.

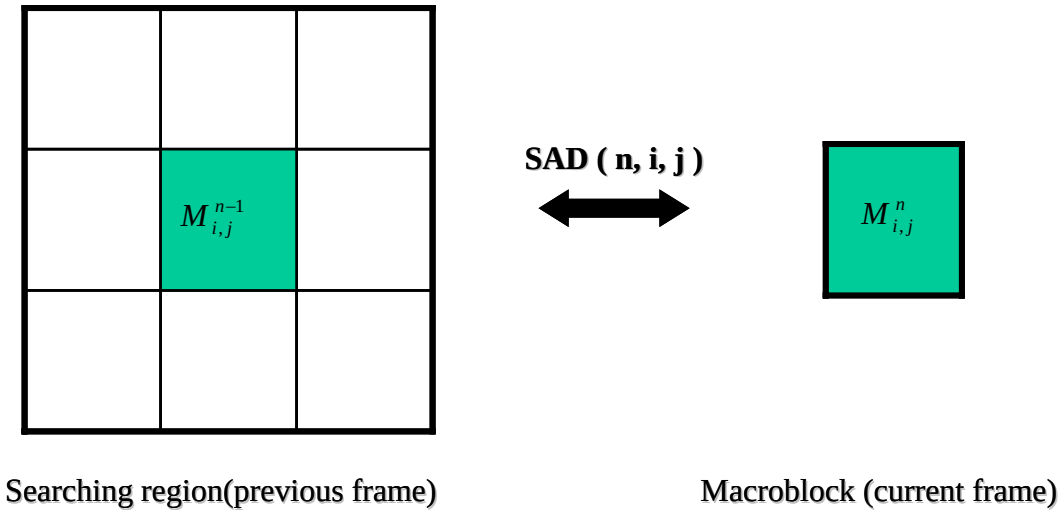


Figure 46: Computation of motion vectors for each macroblock

3. Third step consists of the obtainment of the polar coordinate θ for each pair of motion vectors with equation 10. As a result, orientation of the MV is, then, determined:

$$\theta\,(i,j) = \arctan\left(\left|\frac{MVy(i,j) - \arg_i\left[MB(i,j)\right]}{MVx(i,j) - \arg_j\left[MB(i,j)\right]}\right|\right) \qquad (37)$$

4. Now, we distribute the angles obtained within 16 regions, stated below:

- 1st region: $\left[0, \frac{\pi}{8}\right)$
- 2nd region: $\left[\frac{\pi}{8}, \frac{\pi}{4}\right)$
- 3rd region: $\left[\frac{\pi}{4}, \frac{3\pi}{8}\right)$
- 4th region: $\left[\frac{3\pi}{8}, \frac{\pi}{2}\right)$

- 5^{th} region: $\left[\frac{\pi}{2}, \frac{5\pi}{8}\right)$
- 6^{th} region: $\left[\frac{5\pi}{8}, \frac{3\pi}{4}\right)$
- 7^{th} region: $\left[\frac{3\pi}{4}, \frac{7\pi}{8}\right)$
- 8^{th} region: $\left[\frac{7\pi}{8}, \pi\right)$
- 9^{th} region: $\left[\pi, \frac{9\pi}{8}\right)$
- 10^{th} region: $\left[\frac{9\pi}{8}, \frac{5\pi}{4}\right)$
- 11^{th} region: $\left[\frac{5\pi}{4}, \frac{11\pi}{8}\right)$
- 12^{th} region: $\left[\frac{11\pi}{8}, \frac{3\pi}{2}\right)$
- 13^{th} region: $\left[\frac{3\pi}{2}, \frac{13\pi}{8}\right)$
- 14^{th} region: $\left[\frac{13\pi}{8}, \frac{7\pi}{4}\right)$
- 15^{th} region: $\left[\frac{7\pi}{4}, \frac{15\pi}{8}\right)$
- 16^{th} region: $\left[\frac{15\pi}{8}, 2\pi\right)$

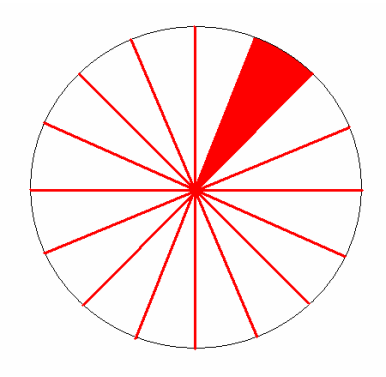


Figure 47: Distribution of angles in regions

5. We count now the largest A_k^1 and second largest A_k^2 amount of angles within a region and the percentages P_k^1 and P_k^2 covered in respect to the whole matrix of angles are computed for each frame:

$$A_k^1 = \max_{all\ A_m\ in\ frame\ k} \{A_m\} \qquad (38) \qquad A_k^2 = \max_{all\ A_m \setminus A_1\ in\ frame\ k} \{A_m\} \qquad (39)$$

$$P_k^1 = A_k^1 \times 100 / M \times N \qquad (40) \qquad P_k^2 = A_k^2 \times 100 / M \times N \qquad (41)$$

with $MxN = \theta$ *matrix size.*

6. Finally, to determine whether there is a pan or a zoom, two fixed thresholds are used so that:

$$\begin{aligned} &if\ (P_k^1 + P_k^2) > threshold1 \Rightarrow Pan\ \det ection \\ &if\ (P_k^1 + P_k^2) < threshold2 \Rightarrow Zoom\ \det ection \end{aligned} \qquad (42)$$

4.6. Results

The video we have used for our research is called ‘futballBetter44-h263_Orig.avi’. It is a high motion football video with a great amount of zooming and panorama shots, as well as fragments where combinations of both techniques are present.

This is the graph obtained with the results of the detections:

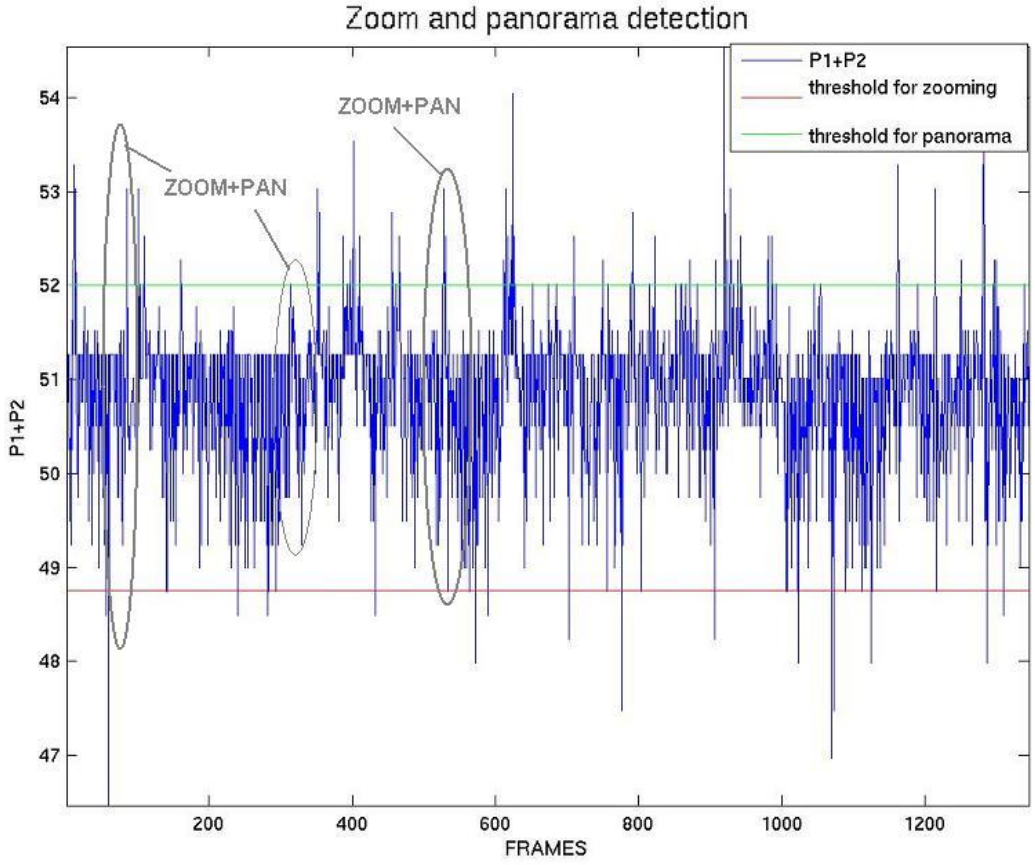


Figure 48: Zoom and pan detection performance for ‘futballBetter44-h263_Orig.avi’

Fragments where both pan and zoom are present have been outlined. The reason is that, since our detections are based on the recognition of high peaks as panorama shots and low ones as zoom shots, one can easily think that in shots where the combination of both camera effects is a fact, detection could not be arranged. However, as we can see in the three examples ticked in the figure 48, our method succeeds in recognizing both zoom and pan in the first and third example and fails in the second just for zoom recognition. In order to analyze this case in which zooming remains undetected, the screenshots corresponding to that sequence are hereafter included:

Figure 49: Screenshots of a fragment where detection of zoom is not accomplished from futballBetter44-h263_Orig.avi

As we see, slow panorama and very gradual zooming are techniques applied in this sequence, but just the first of the two is detected. This can be understandable when we look at the shots that conform the sequence, as we see that the fact of being a hybrid sequence (zoom and pan) and also the slowness of the zoom. We do not have to forget that the comparison in our method is made between consecutive frames so the more progressive the zoom is, the more complicated will be the detection.

On the other hand, false detections are also made. This happens in still sequences where the background and mainly the whole picture looks the same. In football video sequences where a great portion of playground is displayed, when analyzed with SAD, a match will be for sure obtained in macroblocks which consist only of green pixels. Therefore, that can lead to the assumption that a zoom is present in the picture. One example where this happens in our video is below included:

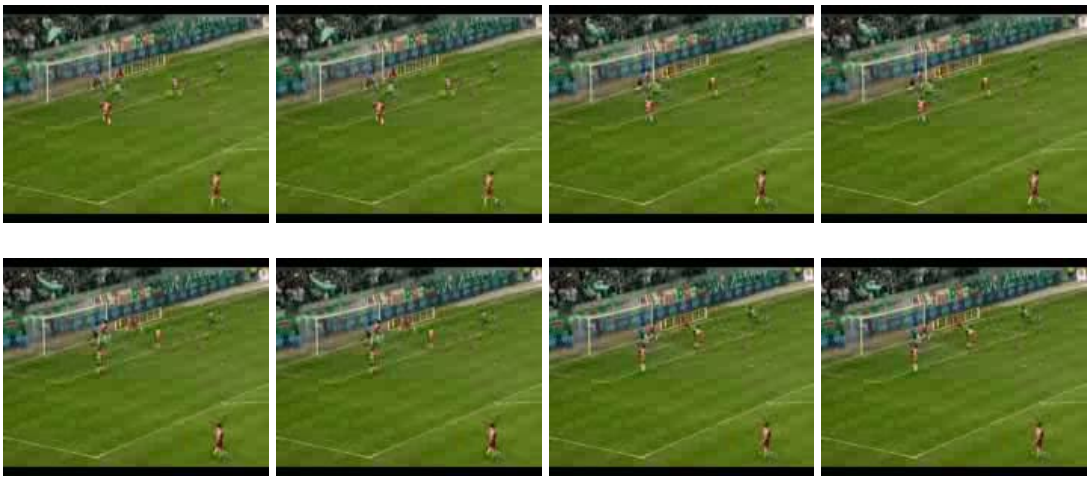


Figure 50: Fragment of a false detection of zooming for futballBetter44-h263_Orig.avi

To end with, statistics of performance (26) (27) are included below:

futballBetter44-h263_Orig.avi	TRUE	FALSE	MISSED	RECALL	PRECISION
Zooming detection	16	3	1	0.94	0.84
Pan detection	16	0	7	1	0.70

We can see great results for zooming are obtained. For panorama shots, though good, statistics are not as optimal. The reason why this happens is that panorama is really difficult to detect in this kind of videos as, mainly, panorama is present in the totality of the frames somehow.

Conclusion

This thesis gives a thorough overview of how the segmentation for soccer video sequences can be accomplished. Limitation of bandwidth leads to the usage of the compression techniques. As a result, video quality is reduced and preprocessing techniques should be applied to overcome the problems with annoyed users. Spatial and temporal segmentation play an essential role in this process since they enable the recognition of the scenes relevant for preprocessing. This knowledge together with the ball position can be used for event detection and to the automated parsing of different video scenes.

In terms of spatial segmentation, event detection for three groups of events has been implemented: events where three parallel lines are present (meaning goalmouth area is somewhere in the frame), events where just one straight line is distinguished (centre line is present in the picture) and events where no lines are found (pictures with no playground or close-up shots). For the recognition of straight lines Sobel filtering for edge detection and the Hough transform for ending points lines detection have been used.

Research on temporal segmentation is another of the aims of this diploma thesis. Within this subject, a comparison of scene cut detection methods with different kind of thresholds and its performance regarding the real-time deployment has been contrived. In addition, a zooming and pan detector has also been implemented and evaluated.

The scene change detection technique we have used is the comparison of the similarity between two consecutive frames. Sum of Absolute Differences (SAD) was used as a similarity metric due to its simplicity. The SAD is also implemented in typical encoders to facilitate the motion vector search, thus it could be re-used. The results by means of the precision and recall (referring to the number of false and missed detections respectively) have shown that calculating the SAD over all colors provides additional robustness compared to the case where only the luminance is used. Even more improvement we obtained using the dynamic threshold technique that has also the ability to work for real time videos since the calculation of threshold is only performed based on the previous and current frames.

However, SAD is not sufficient to recognize camera effects like zooming and panning since they can be confused with each other as well as with a fast and sudden movement of the camera. To distinguish between the kinds of the scene change it is necessary to have information about the motion field in the video sequence. A method based on motion vectors has been explained, implemented and tested in this work. Motion vectors provide the information about the direction of the movement, which aloows us to distinguish the uniform movement from the radial one produced by zooming effects. Recall and precision showed excellent results for zooming detection. Good results were achieved also in pan recognition. Sequences in which combination of both take place proved to be the most complicated to detect due to the nature of the method itself.

List of abbreviations

3G: Third Generation.

4G: Fourth Generation.

MMS: Messenger Multimedia Service

CIF: Common Intermediate Format.

QCIF: Quarter CIF (resolution 176x144).

SQCIF : Sub quarter CIF (resolution 128x96).

4CIF : 4 x CIF (resolution 704x576).

16CIF : 16 x CIF (resolution 1408x1152).

BW: Bandwidth.

DPCM: Differencial Pulse Code Modulation

VLC: Variable Length Codes.

ASCII: American Standard Code for Information Interchange.

DCT: Discrete Cosine Transform.

MPEG: Moving Picture Experts Group.

MP3: MPEG Audio Layer III.

HTDV: High Definition Television.

DVD: Digital Versatile Disc.

HD: High Demand.

SAD: Sum of Absolute Differences.

ISO: International Organization for Standardization.

IEC: International Electrotechnical Commission.

JPEG: Joint Photographic Experts Group.

MB: MacroBlock.

DC: Direct current.

3D: Three dimensions.

References

[1] A. Ekin, A.M. Tekalp, Fellow,IEEE , R. Mehrotra, “Automatic Soccer Video Analysis and Summarization”, IEEE transactions on image processing, vol.12, no.7, July 2003.

[2] P.Soille, “Morphological Image Analysis. Principles and Applications ”, 2nd Edition, Springer-Verlag Berlin Heidelberg , 2004 .

[41] Media Cybernetics, Inc., “Technical Support for Media Cybernetics Products Spatial: filtering”, http://support.mediacy.com/answers/showquestion.asp?faq=36&fldAuto=334 .

[3] B. Girod, G. Greiner, H. Niemann, “Principles of 3D Image. Analysis and Synthesis”, Kluwer Academic Publishers, USA, 2002.

[4] J. S. Boreczky, L. A.Rowe, “Comparison of shot boundary detection techniques”, in Storage and Retrieval for Still Image and Video Databases IV, I. K. Sethi and R. C. Jain, eds., *Proc. SPIE* 2670, 1996.

[5] A. Ekin, A. M. Tekalp, “Sports Video Processing for Description, Summarization, and Search”, Ph.D Thesis, Department of Electrical and Computer Engineering, University of Rochester, Rochester NY, July 2003.

[6] H.J Zhang, A. Kankanhalli, S.W. Smoliar, “Automatic Partitioning of Full-motion Video”, Multimedia Systems,vol. 1, no. 1, pp. 10-28, 1993

[7] B. Shahraray, "Scene Change Detection and Content-Based Sampling of Video Sequences", in Digital Video Compression: Algorithms and Technologies, A. Rodriguez, R. Safranek, E. Delp, Editors, Proc. SPIE 2419, pp. 2-13, February 1995.

[8] A. Hampapur ,R. Jain, T. Weymouth, "Digital Video Segmentation", Proc. ACM Multimedia 94, San Francisco CA, pp. 357-364, October 1994.

[9] A. M. Alattar, “Detecting and compressing dissolve regions in video sequences with dVI multimedia image compression algorithm” , in Proc. IEEE ISCAS, vol. IV, pp. 13-16, 1993.

[10] J. Meng, Y. Juan, S.-F. Chang, “Scene change detection in an MPEG-compressed video sequence”, in Symp. Electronic Imaging: Science and Technology: Digital Video Compression, Algorithms and Technologies, pp. 14-25, Feb. 1995.

[11] A. M. Alattar, “Wipe scene change detector for segmenting uncompressed video sequences" , in Proc. of IEEE ISCAS, vol. IV, pp. 249-252, 1998.

[12] C. Taskiran, E. J. Delp, “Video scene change detection using the generalized sequence trace”, in Proc. IEEE Int. Conf. on Acoustics, Speech and Signal Processing, pp. 2961-2964, 1998.

[13] R. Kasturi, R. Jain, "Dynamic Vision", in Computer Vision: Principles, Kasturi R., Jain R., Editors, IEEE Computer Society Press, Washington, 1991.

[14] U. Gargi, R. Kasturi, S. H. Strayer, “Performance characterization of video-shot-change detection methods”, in IEEE Trans. on Circuits and Systems for Video Tech., vol. 10, pp. 1-13, February 2000.

[15] T. Kikukawa, S. Kawafuchi, “Development of an automatic summary editing system for audio-visual resources”, in Trans. Electron. Inform., pp. 204-212, 1992.

[16] B. Shahraray, “Scene change detection and content-based sampling of video sequences”, in Symp. Electronic Imaging: Science and Technology: Digital Video Compression, Algorithms and Technologies, pp. 2-13, Feb. 1995.

[18] B-L. Yeo and B. Liu, “Rapid scene analysis on compressed video”, in IEEE Trans. on Circuits and Systems for Video Tech., vol. 5, pp. 533-544, Dec. 1995.

[19] A. Dimou, O. Nemethova, M. Rupp, “Scene change detection for H.264 using dynamic threshold techniques”, in Proceedings of EURASIP EC-SIP-M 2005, Smolenice, Slovakia, June 2005

[20] R. Dahyot , “Image Processing : a DSP application”, http://www.mee.tcd.ie/~dahyot/pdf/TCD2003_edge%20detection_lecture.pdf

[21] J. Bresenham, “Algorithm for computer control of a digital plotter”, in IBM Systems Journal, vol. 4(1), pp. 25–30, 1965.

[22] J.D. Foley, A. Van Dam, S. Feiner, and J.F. Hughes, “Computer Graphics Principles and Practice”, Addison-Welsey, 2nd edition, 1996 .

[23] U. Gargi, R. Kasturi, S. H. Strayer, “Performance characterization of video-shot-change detection methods”, in IEEE Trans. on Circuits and Systems for Video Tech., vol. 10, pp. 1-13, Feb. 2000.

[24] F. Arman, A. Hsu, M.-Y. Chiu, “Image processing on compressed data for large video databases”, in Proc. of the First ACM Int. Conf. on Multimedia, pp. 267-272, 1993.

[25] K. Shen, E. J. Delp, “A fast algorithm for video parsing using MPEG compressed sequences”, in Proc. IEEE Conf. on Image Processing, 1996.

[26] B-L. Yeo, B. Liu, “Rapid scene analysis on compressed video”, in IEEE Trans. on Circuits and Systems for Video Tech., vol. 5, pp. 533-544, Dec. 1995.

[27] I. H. Witten , R. M. Neal , J. G. Cleary, “Arithmetic coding for data compression”, Communications of the ACM, v.30 n.6, p.520-540, June 1987

[28] J. Meng, Y. Juan, S.-F. Chang, “Scene change detection in an MPEG-compressed video sequence”, in Symp. Electronic Imaging: Science and Technology, Digital Video Compression, Algorithms and Technologies, pp. 14-25, Feb. 1995.

[29] S-C. Pei, Y-Z. Chou, “Effective wipe detection in MPEG compressed video using macroblock type information”, IEEE Trans. on Multimedia, vol. 4, pp. 309-319, Sept. 2002.

[30] T.D. Little, C, Ahanger, G., Folz, R.J., Gibbon, J.F., Reeve, F.W., Schelleng, D.H., and Venkatesh, D, “A Digital On- Demand Video Service Supporting Content-Based Queries”, in Proc. ACM Multimedia 93, Anaheim CA, pp. 427-436, August 1993.

[31] Zabih, R., Miller, J., Mai, K., “A Feature-Based Algorithm for Detecting and Classifying Scene Breaks”, in Proc. ACM Multimedia 95, San Francisco CA, pp. 189-200, November 1993.

[32] C.L.Huang, B.Y.Liao, “A Robust Scene-Change Detection Method for Video Segmentation”, IEEE Transactions on Circuits and Systems for Video Technology, vol. 11, no.12, pp.1281-1288, Dec. 2001.

[33] H. J. Zhang, A. Kankanhalli, S. W. Smoliar, “Automatic partitioning of full- motion video”, Multimedia Systems, vol. 1, pp. 10-28, 1993.

[34] P.K.Sahoo, S.Soltani, A.K.C. Wong ,Y.C.Chen, “A survey of thresholding techniques”, in CVGIP, vol.41, pp.233-260,1988.

[35] K.W.Sze, K.M.Lam, G.Qiu, “Scene Cut Detection Using the Colored Pattern Appearence Model”, in ICIP, vol. 2, pp.1017-1020, 2003.

[36] X.Wang, Z.Weng, “Scene Abrupt Change Detection”, in Canadian Conference on Electrical and Computer Engineering, vol. 2, pp 880-883, 2000.

[37] H.C.Liu, G.Zick, “Automatic Determination of Scene Changes in MPEG Compressed Video”, in ISCAS, vol.1, pp.764-767, 1995.

[38] R. Kasturi, R. Jain, “Dynamic vision in computer vision: principles”, in IEEE Computer Society Press, Washington DC, 1991.

[39] A.Dumitras, B.Haskell, “A look-ahead method for pan and zoom detection in video sequences using block-based motion vectors in polar coordinates”, in Proceedings of IEEE International Symposium on Circuits and Systems (ISCAS), Vancouver (Canada), May 2004.

[40] I.Koprinska, S.Carrato, “Video Segmentation: A Survey, Signal Processing”, Image Communication, 16(5), Elsevier Science, pp. 477 – 500, 2001.

[41] D.A. Huffman, “A method for the construction of minimum redundancy codes”, in Proc. IRE 40, p 1098 - 1101, 1952.

[42] D. A. Lelewer, D. S. Hirschberg,“Data compression”, *ACM Comput. Surv.* 19, 3, 261-296.

[43] S.S. Beauchemin, J.L. Barron, “The computation of optical flow”, ACM Computing Surveys, 27(3), pp.433-467, September 1995 .

[44] O. Nemethova, M. Zahumensky, M. Rupp: “*Preprocessing of Ball Game Video Sequences for Robust Transmission over Mobile Networks*”, in: "*Proceedings of the CIC 2004 The 9th CDMA International Conference*", 2004.