國立交通大學

資訊工程系

碩 士 論 文

在壓縮格式下一個以紋路與空間資訊來加強位移向量之創新架構

A Novel Texture & Spatial based Framework for Motion Vectors Enhancement in Compressed Domain

研 究 生：巴夏

指導教授：李素瑛　教授

中 華 民 國 九 十 三 年 六 月

在壓縮格式下一個以紋路與空間資訊來加強位移向量之創新架構

A Novel Texture & Spatial based Framework for Motion Vectors Enhancement in Compressed Domain

研 究 生：巴夏　　Student：Bashar M. A. Ahmad

指導教授：李素瑛 教授　　Advisor：Prof. Suh-Yin Lee

國 立 交 通 大 學

資 訊 工 程 系

碩 士 論 文

A Thesis

Submitted to Department of Computer Science and Information Engineering

College of Electrical Engineering and Computer Science

National Chiao Tung University

in partial Fulfillment of the Requirements

for the Degree of

Master

in

Computer Science and Information Engineering

June 2004

Hsinchu, Taiwan, Republic of China

中華民國九十三年六月

A Novel Texture & Spatial based Framework for Motion Vectors Enhancement in Compressed Domain

Student: Bashar M. Ahmad Advisor: Prof. Suh-Yin Lee

Department of Computer Science and Information Engineering
National Chiao Tung University

Abstract

In this thesis we propose a novel approach for robust motion vector based object detection in MPEG-2 video streams. Rather than processing the extracted motion vector fields that are directly extracted from MPEG-2 video streams in the compressed domain, we perform post processing operation over the extracted motion vector, in order to reduce the noise within the motion vector content, and to obtain more robust object information. We refine this information through our proposed system which composed of a Spatial filter Component, a Temporal filter Component and a Texture filter component. As a result, the object detection algorithm is more capable of accurately detecting objects with more efficient performance in terms of runtime. We compare the performance of our proposed system with other popular and commonly related work and techniques.

Based on the experimental results performed over the MPEG7 testing dataset and measuring performance by using the standard *recall* and *precision* metrics, object detection using our proposed system is remarkably superior to the alternative techniques.

In addition to these results, we describe a user system interface that we developed, where users can maintain the parameters interactively.

在壓縮格式下一個以紋路與空間資訊來加強位移向量之創新架構

研究生: 巴夏　　指導教授:李素瑛 教授

國立交通大學資訊工程學研究所

摘　要

在這篇論文裡，我們提出了一個創新的方法，針對MPEG-2 的影像資料串流以健全完整的位移向量為基礎做物件偵測。我們對擷取出的位移向量，使用後處理的方式來取代直接自壓縮格式中取出的位移向量，以減少雜訊影響及獲得更健全完整的物件資訊。透過我們提出包含空間資訊過濾元件、時間資訊過濾元件以及紋路過濾元件之系統，我們可以使這些資訊變得更精確。基於我們提出的架構，物件偵測演算法能更正確地偵測物件，並得到更快的處理效能。

我們利用MPEG7的測試資料作為我們提出的系統與其他受歡迎和廣泛被使用的方法效能比較之實驗輸入。由實驗結果，我們的系統明顯的比其他方法來得好。此外，我們也提出了一個互動式參數調整使用者介面。

Table of Contents

List of figures

Chapter 1

Introduction

In this chapter we are going to introduce our motivations for investigating this topic and the organization of this thesis.

1.1 Motivations

Because digital imaging and video streams and their standards are becoming more prevalent, it is expedient to have effective algorithms and paradigms to process visual contents. As the proliferation of compressed video sequences in MPEG formats continues, the ability to perform video analysis directly in the compressed domain becomes increasingly attractive. To achieve the objective of identifying and selecting desired information, a reliable object detection mechanism is needed as a primary step. Although object detection has been studied for many years, it remains an open research problem. A robust, accurate and high performance approach remains a great challenge.

There are two sources of information in video that can be used to detect objects: visual attributes (such as color, texture and shape) and motion information (such as motion vector). On one hand, motion detection complicates the object detection problem by imposing the additional requirement of tracking an object's temporal position. On the other hand, it also provides an additional information source that can be exploited for the purpose of object detection by algorithms operating over the uncompressed [1] or compressed domains [2,3]. Often when using visual attributes for object detection, we will need to perform processing in the pixel domain, which includes the additional burden of attribute extraction. Performing the object detection only based on the visual attributes

in the pixel domain could be based on shape [4,5], color [6,7], or other visual features. Approaches using certain complex analysis in the pixel level are extremely computationally intensive and have other drawbacks compared with the approaches in the compressed domain.

We concentrate on doing object detection in compressed domain. Although process in uncompressed domain can get accurate result, but the work in compressed domain has the following advantages:

1. Most videos are not provided in the form of image sequences, but rather as compressed formats.
2. Implementation of the same manipulation algorithms in the compressed domain will be much cheaper than that in the uncompressed domain because the data rate is highly reduced in the compressed domain (e.g., a typical 20:1 to 50:1 compression ratio for MPEG).
3. Given most existing images and videos stored in the compressed form, the specific manipulation algorithms can be applied to the compressed streams without full decoding of the compressed images/videols. Lastly, because that full decoding and re-encoding of video are not necessary, we can avoid the extra quality degradation that usually occurs in the re-encoding process.
4. Compressed video data offer us additional information like DC coefficients and motion vectors.
5. Amount of data to work on is significantly reduced.
6. It is convenient to populate the algorithm on large sets of video databases as they are represented in compressed formats.

7. The availability of motion vectors and pixel values in coded forms can indirectly provide motion and intensity information for object analysis, avoiding the need to re-perform motion estimation.

1.2 Organization of the thesis

The thesis is organized as follows. Chapter 2 presents the introduction and related work. Chapter 3 shows an overview of the proposed scheme. Chapter 4 introduces the texture filter component. Chapter 5 describes spatial filter component. Chapter 6 describes the Object detection and extraction. Chapter 7 presents the experimental results. Chapter 8 draws the conclusion and suggests the future direction.

Chapter 2

Background and Related Work

Since we want to make use of motion vectors and DCT Coefficients, we have to understand the MPEG video compression standard to know what is the motion vector, how it can be used, what are DCT coefficients and how can they be used. In this chapter, we will briefly describe the MPEG video format first and then introduce some related work.

2.1 Overview of MPEG Stream

The MPEG compressed video provides one motion vector for each macroblock of size 16x16 pixels which means that the motion vectors quantized to 1 per 16x16 blocks.

To formulate the problem, an MPEG stream is composed of an ordered sequence of three types of frames, I, P and B, as shown in Figure 1. I (intraframe) frames encodes the whole image in a similar way as JPEG does, that is by 8x8 block-wise DCT (Discrete Cosine Transform) and VLC (Variable Length Coding). These kinds of frames allow random access to the sequence and are used as “pivots” for the motion prediction. A P picture is coded using motion-compensated prediction from a previous P or I picture, using what is called forward prediction. A B picture is coded by using both past and/or future pictures as reference, and thus is called bi-directional. Figure 1 shows the principle of B pictures at work. The frames are grouped at a higher level in Group of Pictures (GOPs), which by definition are the frames between two I frames.

Without loss of generality we assume that in any analyzed MPEG stream, the Group of Pictures (GOP) will have the standard structure [IBBPBBPBBPBBPBB]. According to this assumption we design our system after extracting the motion vector.

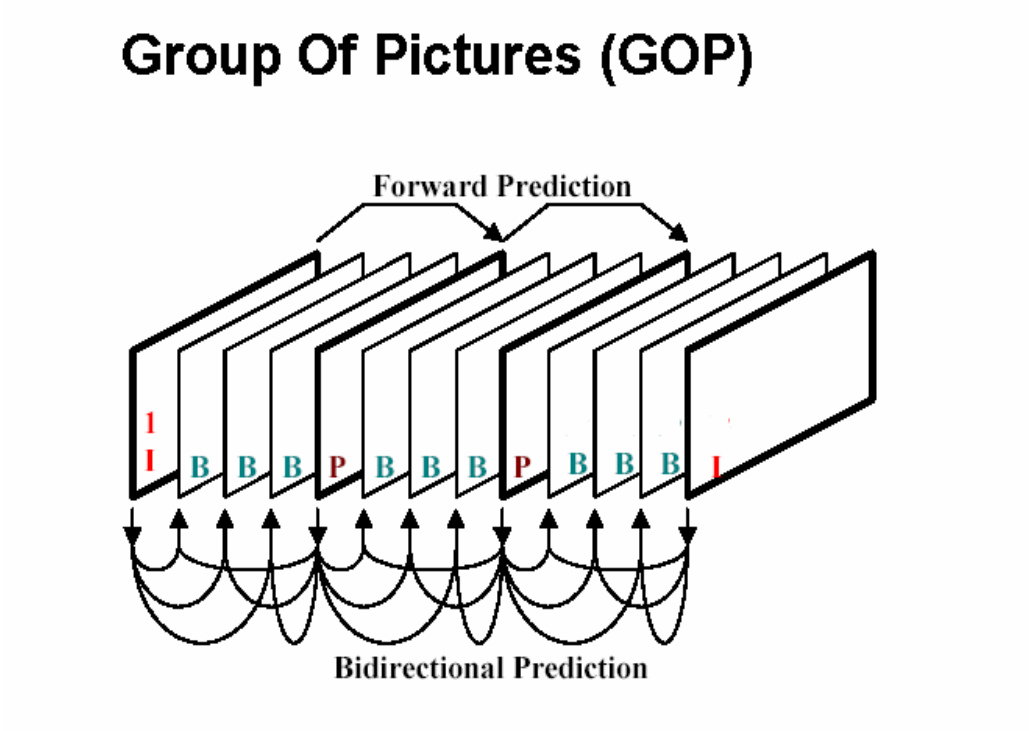

Figure 1: A typical MPEG encoding GOP

2.2 Related work

Much work is being done in the area of motion based video object segmentation in the pixel domain [8, 9, 10, 11] which exploits the visual attributes and motion information. However, very little work has been carried out in the area of compressed domain video object extraction. Pixel domain motion detection is performed based on the motion information at each pixel location such as *optical flow estimation* [12], which is very computationally demanding. On the other hand, the motion information can be

available from the compressed domain. In many cases, especially in the case of well-textured objects, the motion vector values reflect the movement of objects in the stream very well. Some approaches [13,14,15,16,17] utilize these motion vector values directly. Processing digital video directly in the compressed domain has reduced the processing time and enhanced storage efficiency, speed, and video quality. Moreover, in today's ever growing volume of video data provided in compressed formats MPEG1/2, it increasingly makes more sense to perform object detection in the compressed domain. Object detection directly in compressed video without full-frame decompression is clearly advantageous, since it is efficient and can more easily reach real-time processing speeds.

Motion vector information is an important cue for humans to perceive video content. Thus, the need for reliable and accurate motion vector information becomes clear for those approaches that are employing the motion information [2,3,13,14,15,16,17,18,19,20]. But motion vector information is sometimes difficult to use due to the lack of effective representation and due to the fact that it introduces large amounts of noise that makes further processing of the data impractical. Besides, it is still far from ideal in performance, as the key motion estimation part is carried out using coarse area-correlation method that has proven its inefficiency in terms of accuracy. Some researchers [21] elaborate on the noise in motion vectors due to camera noise and irregular object motion. However, it is known that the motion vectors in MPEG-1/2 may not represent the true motion of a Macro-block. Also the 1 pixel per 16x16 pixel-block scheme makes detection of small objects difficult. Objects that are too small are simply ignored and object contours are distorted based on object detection [22,23,24]. It is known [25] that the motion fields in MPEG streams are quite prone to quantization

errors, especially in low-textured areas. However typical samples in the motion vector field are usually inaccurate [26,27]. These defects can be combated with robust error recovery schemes that repair motion fields and reduce noise. Consequently we can produce a smoother shape boundary, where the motion vectors are used to determine object boundaries in object detection.

Therefore, in this paper, we introduce a technique that can overcome those defects and produce more reliable motion vector information and smoothed object boundaries for the object detection technique. Initially, we pass the motion vectors and DCT values of last component into texture based filter; this will be described in details. The median filter repairs aberrations introduced by the texture based filter, thus making the resulting filtered data more representatives of the original motion vectors and more reliable for use by further compressed domain object detection algorithms.

Then, we pass the filtered motion Vectors to temporal filter component.. This filter comes from temporal neighborhood of the current macroblock based on the intuition that a 'fine' motion vector should not have its direction altered in an abrupt manner. Finally, we use the spatial filter component, which processes the raw and extracted motion vector fields from P frames using a Gaussian filter. The Gaussian filter removes noise and smoothes the motion vectors.

In this way, many situations that may cause trouble in conventional approaches can be handled properly without using complex operations.

Alternatively, some researchers performed the filtering in the pixel domain. This implies they were actually filtering the visual attributes. After their filtering process was

complete, some [1] performed object detection by processing motion vectors with matched filters. Others used filtered visual attributes as a post-processing step [22]. Still others use a texture filter, but as a preprocessing step [27]. [22,27] Although they claimed in their approaches that the result can be improved significantly, drawbacks remain. These drawbacks include the elimination of small details that could contain an object, increased computational complexity, as well as other already mentioned drawbacks of doing object detection in the pixel domain.

Some [26,28] used only the median filter or modified median filter in the compressed domain; not for the raw motion vector but for the processed motion vector to repair the irregularities introduced by the hard cut of some motion vector values or accumulative process of motion vector. In these approaches, the time consumption is high due to the computational complexity. [26,28] partially need to return to the pixel domain. Moreover, [28] used P,B frames. One [26] applied a median filter for the magnitude only, while we apply it for both magnitude and direction which will result in a more accurate and reliable outcome in terms of object detection. As we mentioned before, [25] used Spatial confident measure which is Mean- Filter like, where authors in [33,34] proved this insufficient and unrealistic in terms of real-time application. In addition that, [25] combined both the texture and spatial measure equally, which was proved insufficient and unrealistic for the real-time applications.

[35,36,37] used the texture measure without regarding the spatial measure which for sure is resulting in less accuracy. On the other hand [33,34] used spatial measure without regarding the texture measure which certainly is resulting in less realistic result.

Chapter 3

Overview of the Proposed System

First of all, we will present the following diagram which states an abstract overview of our proposed system, and then we will describe its components in details.

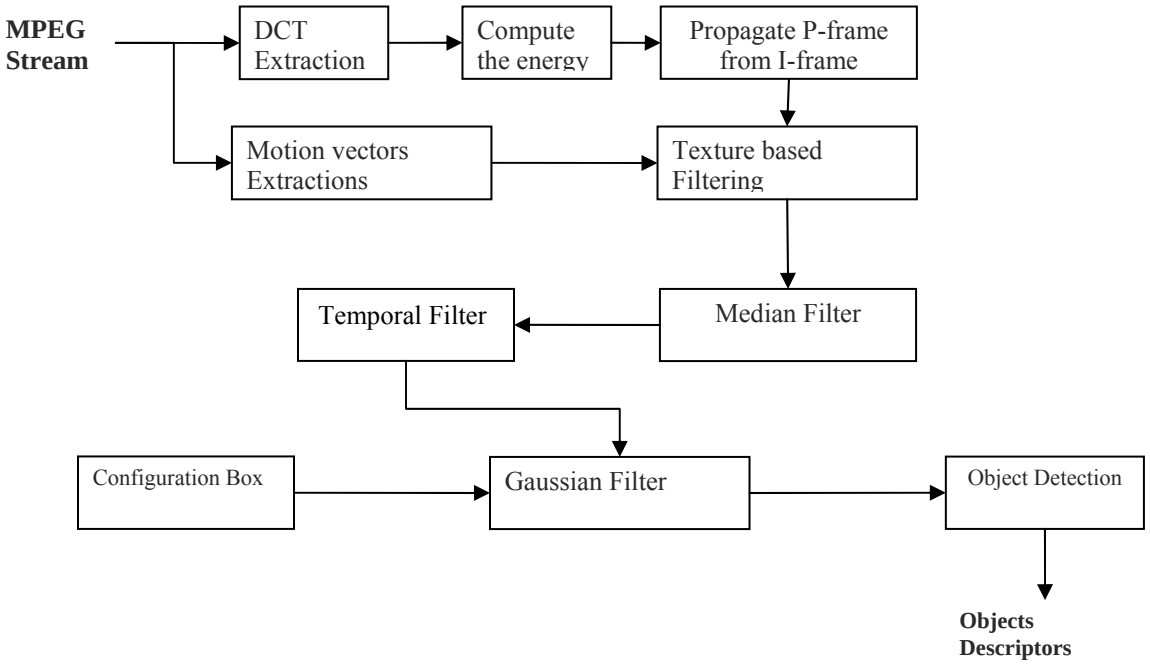

Figure 2: The System overview.

In our proposed approach we first take an MPEG2 video stream with the [IBBPBBPBBPBBPBB] structure. Figure 2 shows the proposed system architecture. Next, we extract the motion vectors from P-frames only in order to reduce the computational complexity. Since in general, in a video with 30 fps, consecutive P-frames separated by two or three B-frames are still similar and would not vary too much.

Besides, it must be noted that B-frames are just "interpolating" frames that hinge on the motion information provided in P-frames and therefore using them for the concatenation of displacements would be redundant. Therefore, it is sufficient to use the motion information of P-frames only to detect the objects.

Meanwhile, we will extract the DCT coefficients from I frames. Those coefficients include the DC coefficient, and the AC components as well. Knowing that, those coefficients are readily available in MPEG stream. Then, we will pass the DCT coefficients into a module to calculate the texture of each frame. Later, we will propagate this texture information into P frame using Inverse Motion Compensation Technique. After that, we will filter each motion vector based on its texture value as what will be described in later chapters.

Furthermore, we use the median filter because it does not alter motion vector values. Rather, it simply rearranges motion vectors, not altering the values contained within any motion vector. Hence the median filter is used to repair potential irregularities introduced by the previous filter processing and in order to straighten up some single motion vector which has been influenced.

Then, we pass the filtered motion vectors to temporal filter component. This filter is derived from the temporal adjacent neighborhood of a macroblock. This Component comes from temporal neighborhood of the current macroblock based on the pervious mentioned intuition that a 'fine' motion vector should not have its direction altered in a drastic manner.

After obtaining the motion vector field's magnitude and direction values, we pass these values through the Gaussian filter. Meanwhile, filter parameters such as standard deviation and kernel size are initialized to obtain the optimal performance. We then pass these filtered motion vectors into our object detection algorithm which first considers the value of the motion vector over a specific threshold, and then the object detection

algorithm is applied to get a set of detected objects in each frame. Steps will be described in detail in the following chapters.

Chapter 4

Texture Filtering

Although there have been some works on modeling the noise of motion vectors for MPEG applications, we figure out that those finesses are not so reliable. In fact, in most cases motion vector are not only inaccurate, but also in some cases they are data of no significance. This would not allow even a sturdy fitting stage to operate reliably.

So, to begin, why do we need a texture filter in object detection content? The reason of the problem can be tracked back to the very nature of prediction-correction coding. Given that the main purpose of MPEG is that to allow a reasonable rendering quality at high compression rates, the motion estimation can afford some errors as long as the errors are small. It is more robust if these low-textured macroblocks are not included in the fitting. For doing so there are several possibilities. The more efficient one would be that of analyzing the AC components of the DCT coefficients, thus staying in the compressed domain. Therefore we propose our scheme inspired by this fact. In order to start our system we should extract the desired features, which is described in the following section.

4.1 Features Extraction from MPEG2

The MPEG compressed video provides one motion vector for each macroblock of size 16x16 pixels, which means that the motion vectors are quantized to 1 vector per 16x16 block. The motion vectors are not the true motion vectors of a particular pixel in the frame. Our object detection algorithm requires motion vectors of each P-frame from

the video streams. Our system takes the sparse motion vectors from the compressed video stream as the only input. For the computational efficiency, only the motion vectors of P-frames are used for object detection algorithm since in general, in a video with 30 fps, consecutive P-frames separated by two or three B-frames, are still similar and would not vary too much. Besides, it must be noted that B-frames are just "interpolating" frames that hinge on the motion information provided in P-frames and therefore using them for the concatenation of displacements would be redundant. Therefore, it is sufficient to use the motion information of P-frames only to detect the objects.

Besides, we need to extract the DCT information from I-frames. Since information are readily available in MPEG streams, thus not too much time is demanded in decoding the MEPG streams. Hence our approach can fit the real time application environment.

4.2 Texture energy computation

Object regions are distinguished from background using their distinguishing texture characteristics. Unlike previously published methods [34] which fully decompress the video sequence before extracting the desired object regions. The proposed method helps in locating the candidate object regions directly in the DCT compressed domain using the *intensity variation* information encoded in the DCT domain. Therefore, only a very small amount of decoding is required.

In some application domains[38], some researchers do use the either horizontal intensity variation, or vertical intensity variation. For example in the text detection, it is generally agreed that text regions possess a special texture because text usually consists

of character components which contrast the background and, at the same time, exhibit a periodic horizontal intensity variation due to the horizontal alignment of characters. In addition, character components form text lines with approximately the same spacing between them [38], [39]. As a result, text regions can be segmented using texture features.

But, some of those researchers [40] use just the DC components or the DC components plus two AC components. However, only some objects that abruptly appeared or disappeared can be detected, assuming that changes resulting from other sources can be ignored. Therefore, their method would not be able to handle objects that gradually enter into or disappear from the frames. It is also vulnerable to fast moving objects in a video. As the image resolution is reduced during compression by a factor of 64 (DC sequence only) or 16 (DC+2AC), a considerable amount of information is lost, resulting in a lower accuracy of the method.

In this thesis we propose a texture-based motion vector filter which operates directly in the DCT domain for MPEG video. The DCT coefficients in MPEG video [41], which capture the directionality and periodicity of local image blocks, are used as texture measures to identify high texture regions, rather the non texture regions. Each unit block in the compressed images is classified as ratio of how texture is this unit block based on local horizontal, vertical intensity variations and diagonal intensity. In addition, post processing procedures including morphological operations median filter is applied. The proposed algorithm is extremely fast due to the facts that:

1. It requires very little decoding of the compressed image/video stream.
2. The refinement and post processing proceed on images of reduced sizes.

As mentioned before, a texture-based approach, we propose of using the DCT coefficients directly from compressed images and video as texture features to refine the motion vector values. In DCT compression a two-dimensional image is encoded using the DCT coefficients $\{c_{uv}\}$ of an N x N (N is usually a power of two) image region $\{I_{xy}, 0 \le x < N, 0 \le y < N\}$:

$$c_{uv} = \frac{1}{N} K_u K_v \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} I_{xy} \cos\frac{\pi u(2x+1)}{2N} \cos\frac{\pi v(2y+1)}{2N}, \qquad (1)$$

Where u and v denoted the horizontal and vertical frequencies $(u,v=0,1,\ldots,N-1)$ and $K_w = \frac{1}{\sqrt{2}}, (w \in \{u,v\}),$ for $w=0$ and $K_w = 1,$ otherwise. The AC components $(c_{uv}, u \ne 0$ or $v \ne 0)$ capture the spatial frequency (characterized by u and v values) properties of the N x N image block.

This approach is justified for the following reasons:

1. The DCT coefficient values, which are computed based on the 8 x 8 spatial input, capture local image features.
2. The values of DCT coefficients, which are amplitudes of harmonic waves, denote the relative amount of various 2D spatial frequencies contained in 8 x 8 blocks. Therefore, they can be used as measures of spatial periodicity and directionality, when frequencies in the x and y dimensions are properly tuned.
3. The quantized DCT coefficients can be readily extracted from a video stream and JPEG data. Although they are quantized, the rank information is preserved and we can use them to compute texture features without any decoding procedure.

In summary, values of DCT coefficients in compressed domain images capture the local periodicity and directionality features in the spatial image domain. This is the basis of our approach.

To gain some insight into the DCT spectrum, Figure 3(a) shows an example input image and Figure 3(b) shows the absolute values of the DCT coefficients directly extracted from the compressed domain of the intensity image. Each subimage in Fig. 3(b) represents one DCT channel of the input image. Each pixel in the subimage is the energy in this channel for the corresponding DCT block of the input image, where the magnitude is proportional to the brightness of the pixel. The channels, from top to bottom, indicate horizontal variations, with increasing frequencies; and from left to right, indicate vertical variations, with increasing frequencies.

In particular, the subimage (channel) at the top left corner corresponds to the DC component, which is the averaged and subsampled version of the input image and the subimages on the top row, from left to right, correspond to channels of zero vertical frequency and increasing horizontal frequencies. This figure shows that the top left channels, which represent the low frequency components, contain most of the energy, while the high frequency channels, which are located at the bottom right corner of each subimage, are mostly blank. It also indicates that the channel spectrums capture the directionality and coarseness of the spatial image. For all the vertical edges in the input image, there is a corresponding high frequency component in the horizontal frequencies, and vice versa. Furthermore, diagonal variations are captured by the channel energies around the diagonal line. This example illustrates that the DCT domain features do characterize the texture attributes of an image.

Figure 3(a): Example Input image

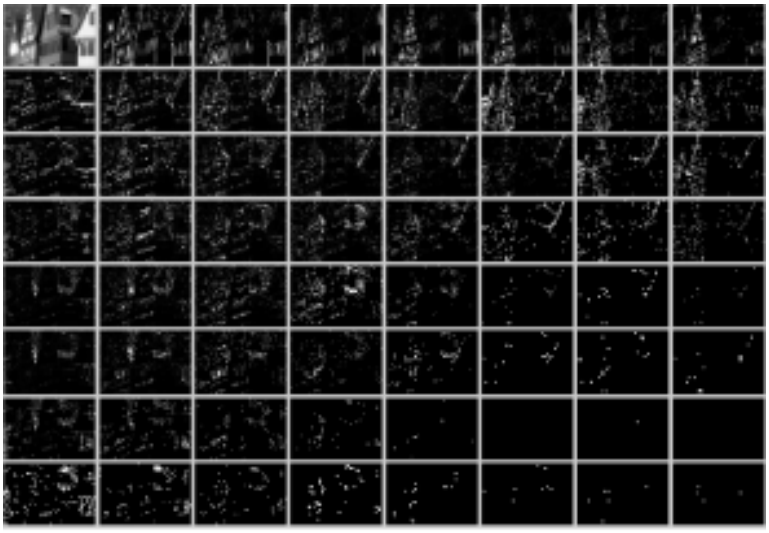

Figure 3(b): The absolute values of the DCT coefficients directly extracted from the compressed domain.

So far, according to the previous discussion, we can design *Directional Texture Energy Map* in DCT domain as shown in Figure 4, by assigning a directional intensity variation indicator for each coefficient in DCT domain either DC or AC components as the following.

H: Horizontal intensity variation.

V: Vertical intensity variation.

D: Diagonal intensity variation.

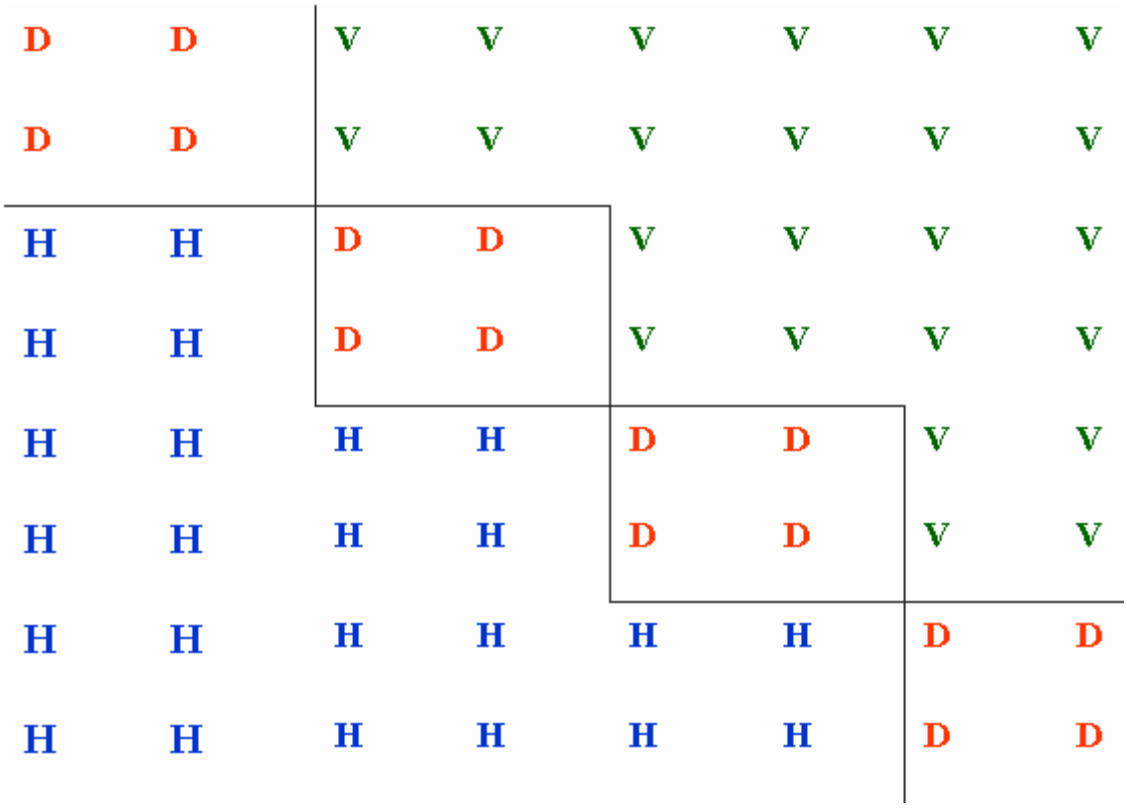

Figure 4: Directional Texture Energy Map in DCT

We are processing in the DCT domain to obtain the directional intensity variation or the so called directional texture energy using only the information in the compressed domain. We perform the following operations. Note that the operating units are the 8X8 blocks in I-frames.

Eh: We compute the Horizontal energy Eh by summing up the absolute amplitudes of the horizontal harmonics of the block, which has been marked as H in the Directional Texture Energy Map.

Ev: We compute the Vertical energy Ev by summing up the absolute amplitudes of the Vertical harmonics of the block, which has been marked as V in the Directional Texture Energy Map.

Ed: We compute the Diagonal energy Ed by summing up the absolute amplitudes of the Diagonal harmonics of the block, which has been marked as D in the Directional Texture Energy Map.

Finally, we will calculate the average energy Ea. for each macroblock, which is the average value of the Vertical energy, Diagonal energy and Horizontal energy.

As in the equation 12

$$E_a = \frac{E_h + E_d + E_v}{3} \qquad (2)$$

After we get the average energy, these average energy values are then thresholded to obtain the blocks of large intensity variations. We will update Motion vector values based on the Ea as described in the following procedure.

For every macrblock

$$Motion\_Vector_{new} = \begin{cases} Motion\_Vector_{old} * \dfrac{100 * E}{E_t}\% & , E_a < E_t \\ Motion\_Vector_{old} & , E_a \geq E_t \end{cases}$$

E_t: Threshold Energy it's determined experimentally

We have used an adaptive threshold value which is 1:45 times the average texture energy of the corresponding DCT channel for all the blocks in the Frame.

The processing speed of the proposed method is very fast since it does not require a fully decompressed MPEG video. The DCT information which is used is readily available from the video stream.

Furthermore, the processing unit for the algorithm is a DCT block. So, once the feature vectors are extracted for each block, the segmentation and refinement processes operate on an image which is 1/8th of the original image size in each dimension when 8X8 DCT blocks are used.

The diagram in Figure 5 describes the process for filtering texture in the frame level. The I-frame has no motion values and it stores DCT information of the original frame. Though an I-frame provides no motion information, we still could grasp the textured, and propagate that information to the P frames.

Figure 5 clearly states the operation in algorithmic way. Depending on the frame type we send the frame to its specific module. If this frame is an I-frame, then we know it contains the DC value and AC components. Thus we can calculate the texture energy as stated before according to the previous procedure and texture energy map. After we propagate those values into P-frames then we perform the texture filter on the motion vector values.

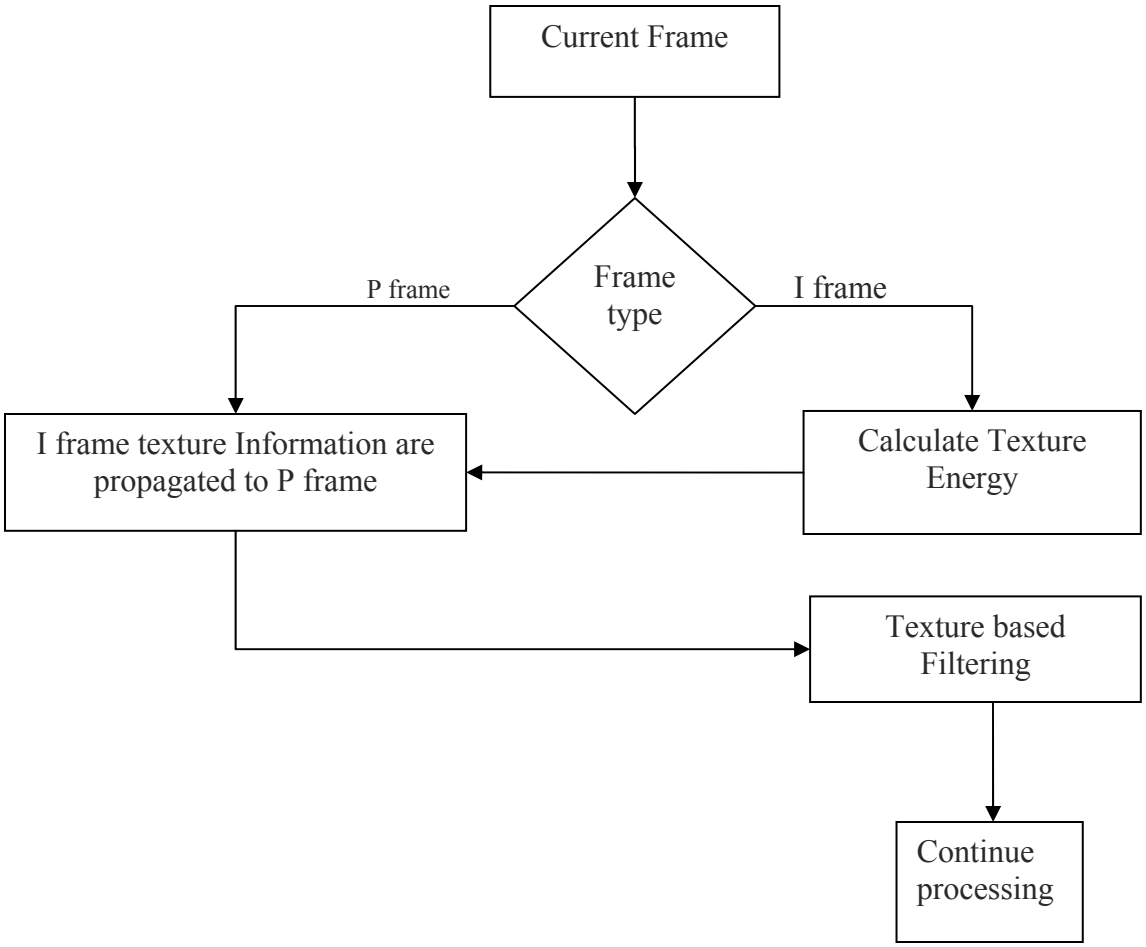

Figure 5: Propagate P-frame texture information from I-frame.

Chapter 5

Temporal and Spatial Filtering

In this chapter, we will describe the Temporal filter, this filter is just derived from the temporal adjacent neighborhood of a macroblock. This Component comes from temporal neighborhood of the current macroblock.

On the other hand spatial filter which is a specific Gaussian filter configured by user interface system, will be described in this chapter. We will also describe the characteristics of this component and the types of noise this filter can overcome.

5.1 Gaussian filter and noise Elimination

In this section we explore the filter types and their characteristics and parameters with deep elaboration on the optimal performance filter which is Gaussian filter.

First of all we should mention about the noise types. We assume that the noise is in the content of the motion vector fields. Noise can generally be grouped into tow types:

- Independent noise.
- Noise which is dependent on the motion vector data.

The independent noise can often be described by an additive noise model, where the extracted motion vector v(i,j) is the sum of the true and reliable motion vector *t(i,j)* and the noise *n(i,j)* as defined in Eq.(3).

$$v(i, j) = t(i, j) + n(i, j) \qquad (3)$$

As it is assumed before, additive noise is evenly distributed over frequency domain, whereas a reliable motion vector is in low frequency area. Hence the noise is dominant for high frequencies and its effect can be reduced using some kind of low pass filter. This can be done either with a frequency filter or with a spatial filter. However, spatial filter is preferable, as it is computationally less expensive than frequency filter

In the second case of data-dependent noise, it is possible to model noise with a multiplicative, or non-linear, model. These models are mathematically more complicated. Hence, if possible, the noise is assumed to be data independent.

In literature, there are many spatial filters and we are using most common and convenient group of spatial filters trying to get the optimal result. The idea of mean filtering is simply to replace each motion vector value in a frame with the mean (`average') value of its neighbors, including itself. This has the effect of eliminating motion vector values which are unrepresentative of their surrounding. Median filter like the mean filter, considers each motion vector in the frame in turn and looks at its nearby neighbors to decide whether or not it is representative of its surroundings. Instead of simply replacing the motion vector value with the mean of neighboring motion vector values, it replaces it with the median of those values. The median is a more robust average than the mean and so a single very unrepresentative motion vector in a neighborhood will not affect the median value significantly. Beside, since the median value must actually be the value of one of the motion vector in the neighborhood, the median filter does not create new unrealistic motion vector values finally Gaussian filter:

A Gaussian filter smoothes MVs by calculating weighted averages in a filter box. The Gaussian Filter is the filter type that results in the most gradual pass band roll-off and

the lowest group delay. The step response of the Gaussian filter never overshoots the steady state value. As the name states, the Gaussian Filter is derived from the same basic equations (4) used to derive the 2-D Gaussian distribution in Eq.(5) as shown in Figure 6.

$$G(x) = e\frac{-x^2}{2\sigma^2} \qquad (4)$$

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{\frac{x^2+y^2}{2\sigma^2}} \qquad (5)$$

The significant characteristic of the Gaussian Filter is that the step response contains no overshoot at all which comes great with what we desire form filtering out and justify the value of motion vector. But it uses a different kernel that represents the shape of a Gaussian (`bell-shaped') hump. This kernel has some special properties which are detailed below.

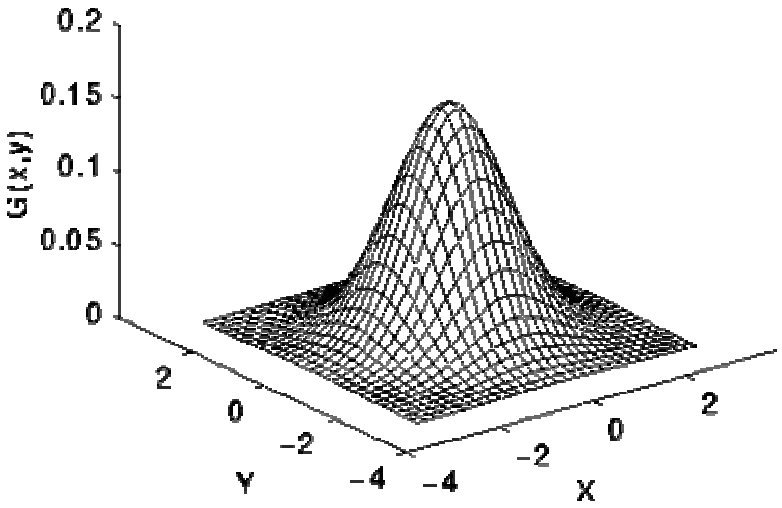

Figure6: 2-D Gaussian distribution with mean (0,0) and σ =1

The idea of Gaussian smoothing is to use this two dimensions distribution as a

`point-spread' function. This is achieved by convolution and we need to produce a discrete approximation to the Gaussian function before we can perform the convolution. In theory, the Gaussian distribution is non-zero everywhere, which would require an infinitely large convolution kernel. However in practice, it is effectively zero more than about three standard deviations from the mean. The degree of smoothing is determined by the standard deviation of the Gaussian. Larger standard deviation Gaussians, of course, require larger convolution kernels in order to be accurately represented.

The Gaussian filter outputs a `weighted average' of each Motion Vector's neighborhood, with the average weighted more towards the value of the central value. This is in contrast to the mean filter's uniformly weighted average in mean filter. Because of this, a Gaussian provides gentler smoothing and preserves the crucial Motion Vector Field value better than a similarly sized mean filter.

Mathematically we can write the convolution as:

$$O(i,j) = \sum_{k=1}^{m}\sum_{l=1}^{n} I(i+k-1, j+l-1)K(k,l) \qquad (6)$$

Where i *runs from* 1 *to* $M-m+1$ *and* j *runs from* 1 *to* $N-n+1$.

Convolution provides a way of `multiplying together' two arrays of numbers, generally of different sizes, but of the same dimensionality, to produce a third array of numbers of the same dimensionality.

Once a suitable kernel has been calculated, then the Gaussian smoothing can be performed using standard convolution methods. The convolution can in fact be performed fairly quickly since the equation for the two dimension isotropic Gaussian shown above is separable into x and y components. Thus the two-dimension convolution can be performed by first convolving with a one dimension Gaussian in the x direction, and then

convolving with another one dimension Gaussian in the y direction. Thus Gaussian is in fact the only completely circularly symmetric operator which can be decomposed in such a way.

A kernel is a (usually) smallish matrix of numbers that is used in convolutions. Differently sized kernels containing different patterns of numbers give rise to different results under convolution.

Gaussian kernels properties:

- Gaussian filters are a class of smoothing filters where the kernel values have a two-dimension Gaussian shape.
- In two-dimension Gaussian functions are rotationally symmetric. Thus,

The amount of smoothing is independent of the direction. This property implies that no bias is introduced:

- The Gaussian function is uni-modal. As a consequence, the weight given to a neighbor decreases with the distance from the central value.
- The degree of smoothing is parameterized by the standard deviation of the filter.

Gaussian functions are separable. As a result a Gaussian convolution can be implemented by a one dimension horizontal convolution followed by an one dimension vertical convolution. Using this decomposition, the number of operations decreases significantly. The Gaussian formula can be separated, thus the calculations is faster.

5.2 Smoothing of Motion Vector Field

Up to this stage we have the output motion vector from previous component. Then, we will pass the motion vector magnitude and direction values to the Gaussian

filter where using two values instead of only one of them makes our object detection more robust and meaningful. First we need to configure Gaussian filter by setting the parameters such as σ. Such parameters have a crucial effect in the smoothing process. Thus we implement a user interface as demonstrated in Figure 7 where users can interactively change the parameters for filtering until the optimal filter performance is obtained.

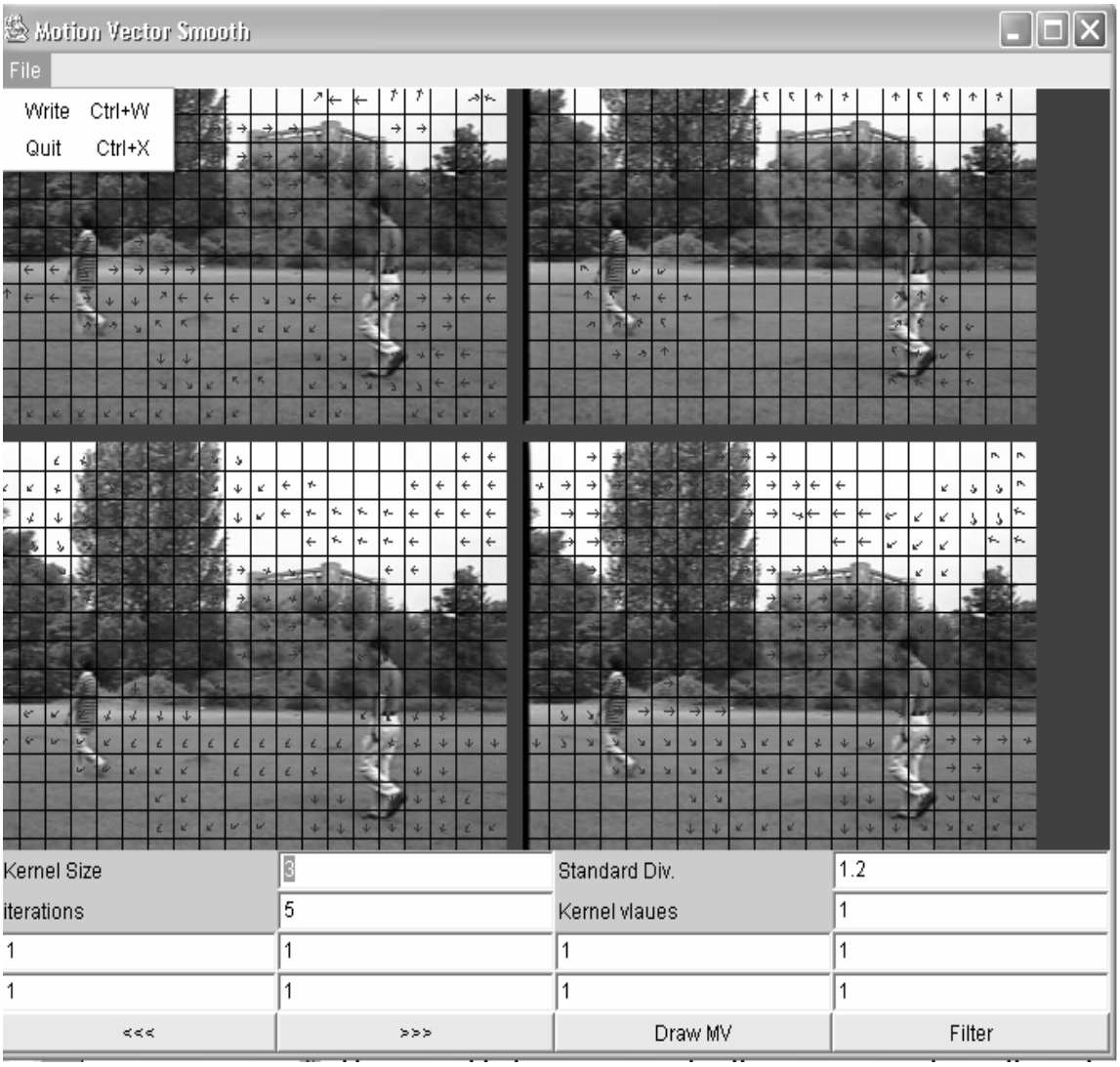

Figure 7: The user interface for motion vector smoothing.

First, the value of σ is chosen to maximize the smoothing. By experiment, the value $\sigma = 1.2$ gives us the best performance for Gaussian filter. Reference [15] explored the use of a Gaussian filter, and excellent results can be achieved with a σ value between 1.0 and 1.5. Also smaller values of σ tend to leave slightly inhomogeneous cluster patterns, while larger values tend to form regularly spaced clusters patterns. This proved to be in agreement with our experiment over the motion vector smoothing. For σ values less than 1.0, the output patterns tend to have disproportionately inhomogeneous clusters as shown in Figure 8. For larger values; clusters tend to form, as shown in Figure 9. Those two figures show the same frame and its corresponding filtered motion vector values.

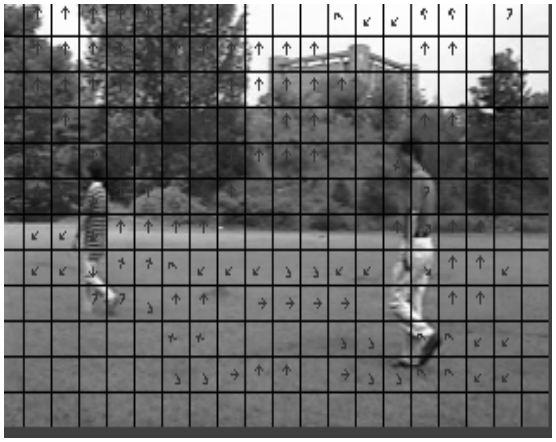

Figure 8: Using small σ value.

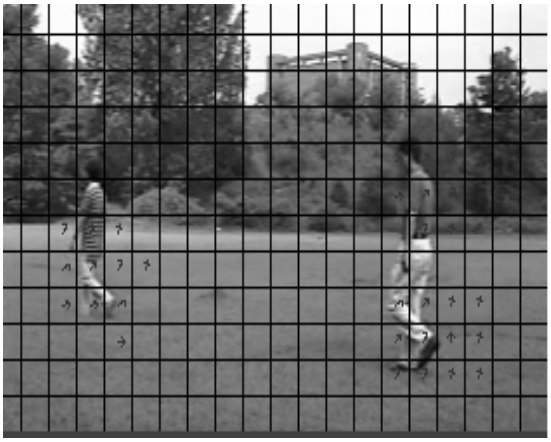

Figure 9: Using large σ value.

Concerning other parameters, the kernel size was chosen to be 3×3 since the window search in our object detection is 3×3 as well. Moreover, the kernel size is recommended to be 3×3 in the interest of reducing the cost of computation. The last parameter to determine is the iteration, the number of times to repeat the convolution step. This will affect the degree of enhancement and the accuracy of the filter. Empirically we find the value 5 to be a suitable value, taking into consideration the

performance and the execution time. Values less than 5 have some slight effect on the performance. Values higher than 5 do not affect performance, but are sure to increase the processing time.

In the following figures we show a result of using filters over motion vectors with a Gaussian filter, median filter, mean filter, and without any filter as well. The value of σ has been chosen to be 1.2 and kernel size to be 3×3. Figure 10 shows the representation of each extracted motion vector over its corresponding frame without any processing, while Figure 11 shows the representation of the filtered motion vector using the Gaussian filter for the same frame. Figure 12 and 13 show the representation of filtered motion vector using mean and median filter respectively for the same frame too.

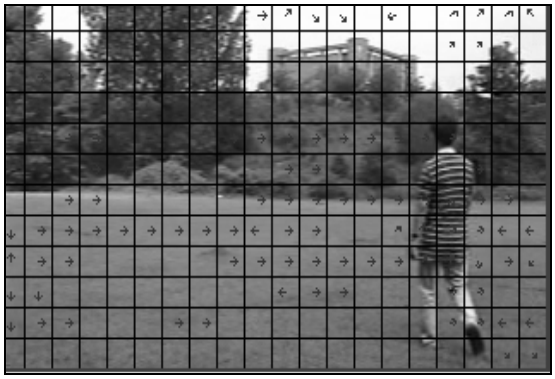

Figure 10: Motion Vector without smoothing.

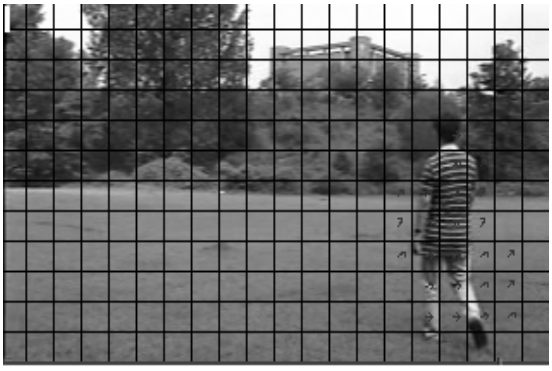

Figure 11: Motion Vector smoothing using Gaussian filter.

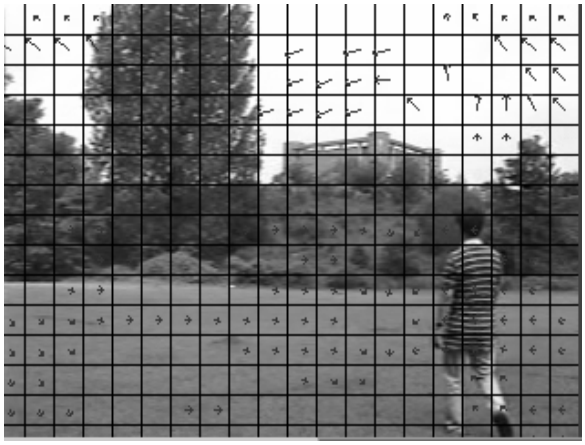

Figure 12: Motion Vector smoothing using Mean filter.

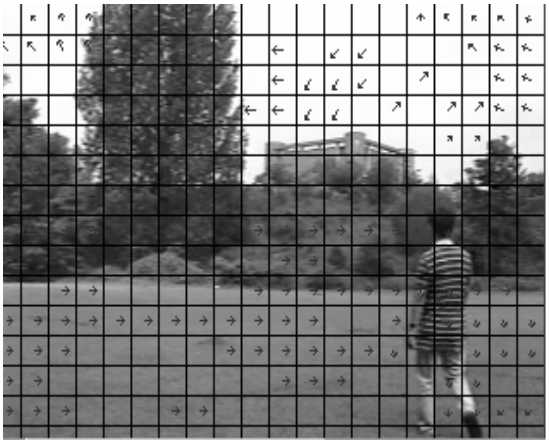

Figure 13: Motion Vector smoothing using Median filter.

Chapter 6

Object Detection and Extraction

A desirable object detection extraction scheme as [16] illustrated that real-time object-based applications should meet the following criteria:

1. A segmented object should conform to human perception, that is, semantically meaningful objects should be segmented.
2. A segmentation algorithm should be efficient and achieve fast speed.
3. Initialization should be simple and easy for users to operate (human intervention should be minimized).

We emphisze that our propsed system is applicable for any motion vector based object detection and extraction alogorithim.

6.1 Object Detection preprocessing

So far, we have obtained the fine motion vectors. We started by eliminating undesired motion vectors before the process of detection in order to achieve more robust performance. Motion vectors with magnitude equal to or approaching zero are recognized as undesirable and hence are not taken into consideration. On the contrary, motion vectors with larger magnitude are considered.

6.2 Object Detection Algorithm

We emphasize that our proposed system is applicable for any motion vector based object detection and extraction algorithm. In this context we use motion vector based

algorithm proposed by [33],[34] as test bed in our experimental result. Following are the description of the used algorithm.

An object detection algorithm is used to detect potential objects in video shots. Initially, undesired motion vectors are eliminated. Subsequently, motion vectors that have similar magnitude and direction are clustered together and this group of associated macroblocks of similar motion vectors is regarded as a potential object. Details are presented in the object detection algorithm. Figure 14 and 17 show the detection results of using the filtered motion vectors and without using filtered motion vectors respectively. Figure 15 shows the result of object detection using directly extracted motion vectors, Figure 14 shows the result using filtered motion vectors through the Gaussian filter. Figure 15 and 16 show the results using filtered motion vector through single median and mean filters respectively. All of these figures use the same frames with the same extracted motion vectors. The detected objects are marked over with thicker and darker line in the Figure 14 through 17.

The Object Detection Algorithm

Input: P-frames of a video clip

Output: object sets $\{Obj_1 , Obj_2 , \ldots Obj_N \}$ where N is total number of regions in P-frame and Obj_N means the N^{th} object of the P-frame. Each object size is measured in terms of number of macroblocks.

1. Cluster motion vectors that are of similar magnitude and direction into the same group with region growing approach.

   1.1 Set search windows (W) size 3×3 macroblocks.

1.2. Search all macroblocks (*MB*) within *W*, and compute the difference ($diffMag_k$ and $diffAng_k$) of motion vector (MV) magnitude ($|MV|$) and direction ($\angle MV$) between center MV_{center} and its neighboring eight motion vectors MV_k within *W*.

$$diffMag_k = abs(|MV_{center}| - |MV_k|)$$

$$diffAng_k = abs(\angle MV_{center} - \angle MV_k)$$

, *where* $k \in [1,8]$ and MV_{center} *is the MV in the center position of W*

$MV_k \in$ *MVs within W except* MV_{center}

For all $1 \leqq k \leqq 8$, *flag*

$$F_k = \begin{cases} 1, & diffMag_k < T_{Mag} \text{ and } diffAng_k < T_{Ang} \\ 0, & \text{otherwise} \end{cases}$$

, *where* T_{Mag} *is the predefined threshold for MV magnitude and* T_{Ang} *is the threshold for MV direction*

If $\sum_{k=1}^{8} F_k \geq 6$, *mark* F_{center} *of* MV_{center} *as 1, where* F_{center} *is the flag of the center MV within W.*

Otherwise, set all flags within W to 0.

1.3 Go to step 1.2 until all macroblocks are processed.

1.4 Group macroblocks that are marked as 1 into the same cluster.

1.5 Compute each object center and record its associated macroblocks.

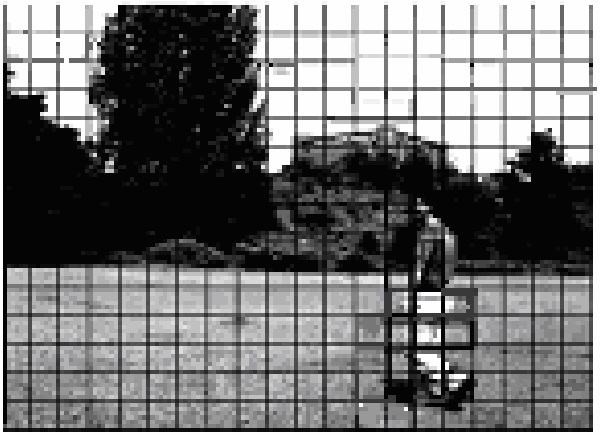

Figure 14: Object Detection using Gaussian filter

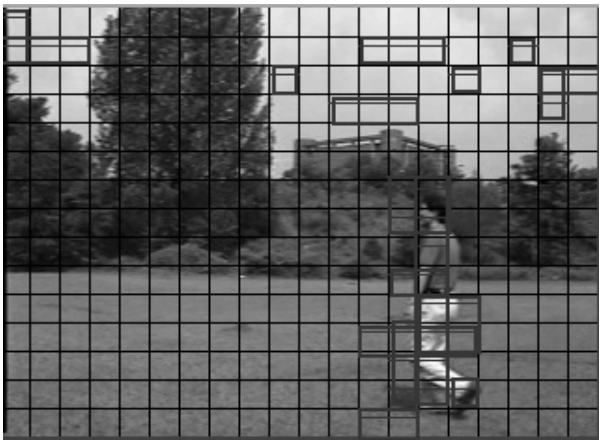

Figure 15: Object detection without filter.

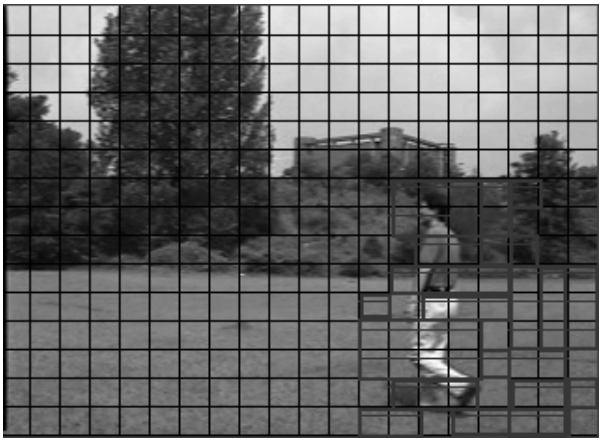

Figure 16: Object detection using Median filter

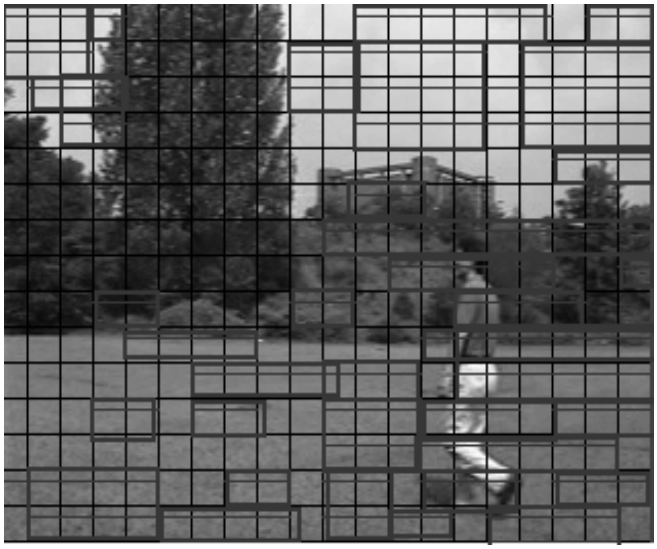

Figure 17: Object detection using Mean filter

Chapter 7

Experimental Results and Discussion

We have designed an experiment in order to verify optimal performance, and to test the noise model assumptions. The experiment has been designed to test the proposed scheme on three video clips. These video clips are in MPEG format and are part of the MPEG7 testing dataset. Testing is performed using four types of other's related work, Group **A** using texture filter only [35], group **B** using spatial filter only mainly Gaussian filter [33], Group **C** using texture and spatial filer as equally important[25], and group **D** our system. Finally without any kind of post processing is a base for comparison. In order to compare the performance among those four system results. The frame size is 320×240 which implies that we have 20×15 macroblocks in each P frame. Our testing dataset presents walking persons in different positions, speed, and can vary slightly in object size.

The following settings were used. For the Gaussian filter, the σ value was set to 1.2, the kernel size to 3×3, and iteration to 5. For the median filter, the mean filter kernel size was set to 3×3 since its computational burden is less and in accordance with the researcher's recommendation, where they considered a 3×3 kernel size to be convenient and sometimes optimum. Moreover, it fits our object detection search window whose size is also 3×3. Besides, the kernel content was set to be 1 for all of the cells, which implies enabling all the elements in the kernel to take part in the neighboring box at the convolution step. Finally the iteration number is set to the optimal value 5.

We choose the *recall* and *precision metrics* because they are most commonly used to evaluate object detection system performance [15,17,30,31]. In Eq. (3) Eq. (4), we can see the definition of recall and precision. In each frame, the *number of hits* is the number of macroblocks that contained an object and this object is correctly detected. The *number of false alarms* is the number of macroblocks that contained no object yet are falsely identified as containing objects. The *number of misses* is the number of macroblocks that contain an object but yet the detection algorithm failed to detect it. We use the macroblock as the unit of measurement because we are doing the object detection in the compressed domain.

$$\text{precision} = \frac{\text{number of hits}}{\text{number of hits} + \text{number of false alarms}} \tag{7}$$

$$\text{recall} = \frac{\text{number of hits}}{\text{number of hits} + \text{number of misses}} \tag{8}$$

Figure 15-18` show the results of object detection performance over the second video clip among the MPEG testing dataset. We show the precision metric and recall metric of our object detection scheme for this video clip both with and without the filter being used, and we construct manually the ground truth of the video clip. Figure 18 and 19 illustrate the values of the recall and precision metrics for each frame in the video clip. We note that the performance of our system is consistently superior to performance using others schemes. We show the average recall metric and average precision metric for the

whole clip in Figure 20 and 21. Again, our system topped them all. Through our experiment we noticed that there is a weakness in the single Gaussian filter performance when the object location is in the frame border. This can be explained due to the lack of information in the neighborhood near the border. We can infer in the single Gaussian filter case, when the person in the video clips is just coming in or just going out, the performance will not be so good as before.

In summary, the proposed system boosts the performance. In addition, the computational complexity is low. Both the Gaussian and median filters are available as a readily implemented component in both hardware and software. Besides, the DCT coefficient and AC component are readily available in MPEG stream. Because we refine the motion vectors resulting in vectors that are easy to process, execution time of the object detection algorithm after using the filter will be reduced significantly compared to that without using a filter. Although we add another block for filtering, the efficiency is almost the same or even better in terms of execution time for the entire object detection process.

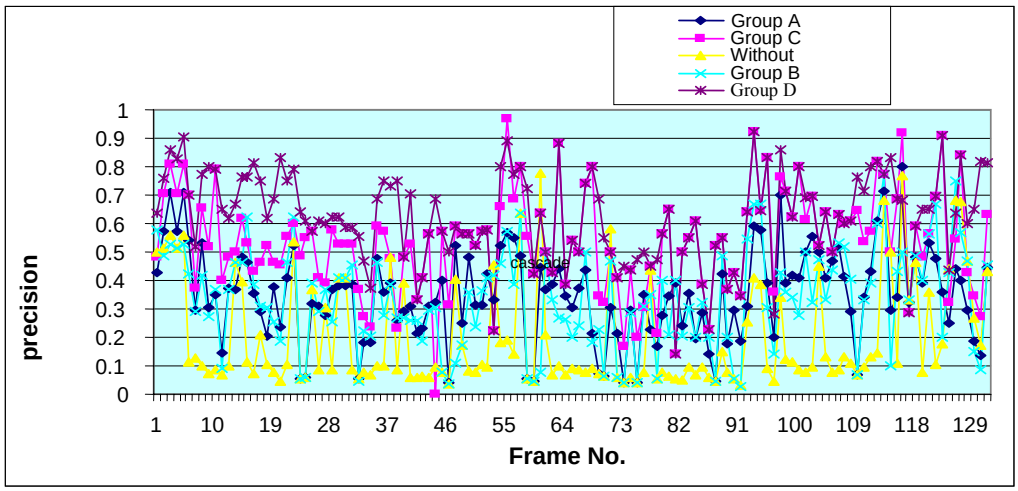

Figure 18: Precision for Object detection in P frames

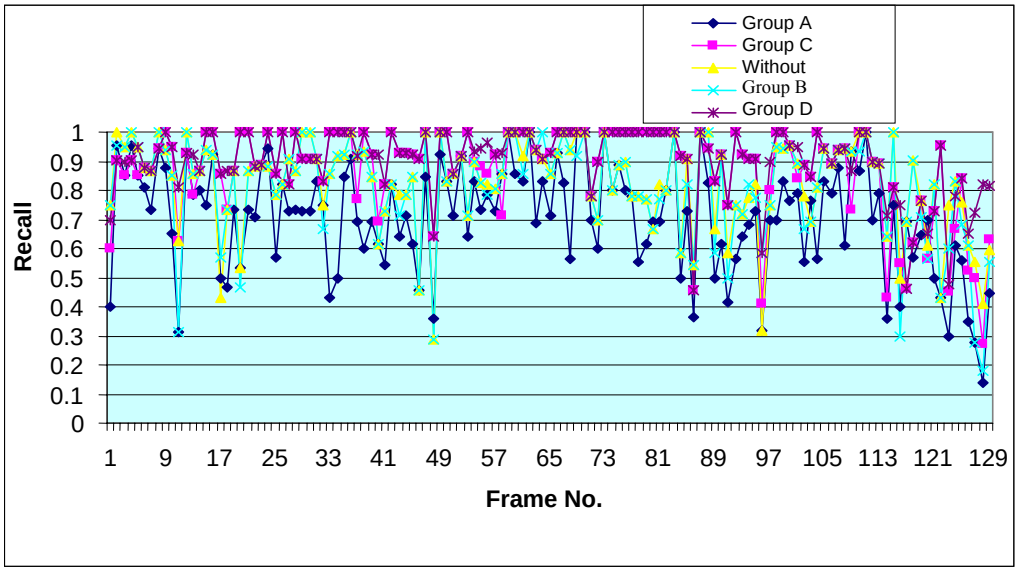

Figure 19: Recall for Object detection in P frames

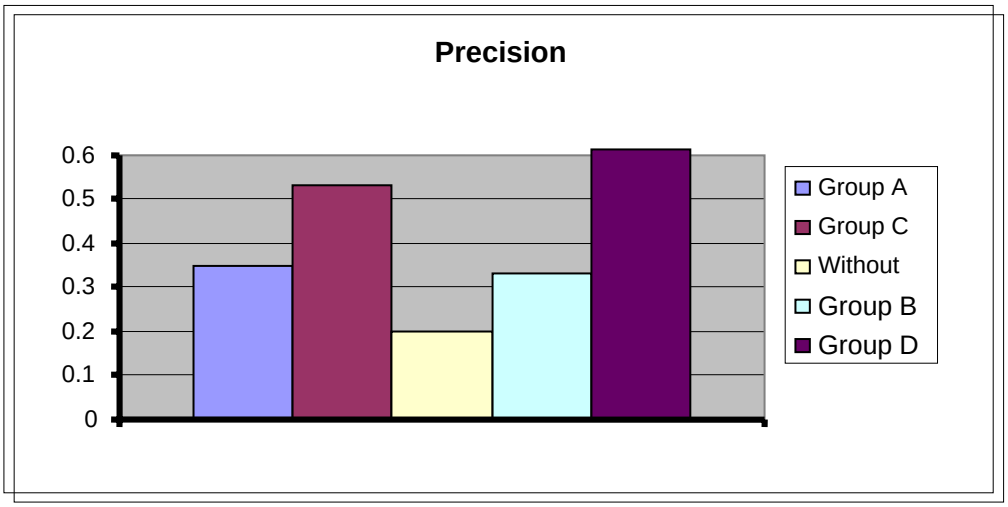

Figure 20: Average precision of object detection for 2nd Video Clip

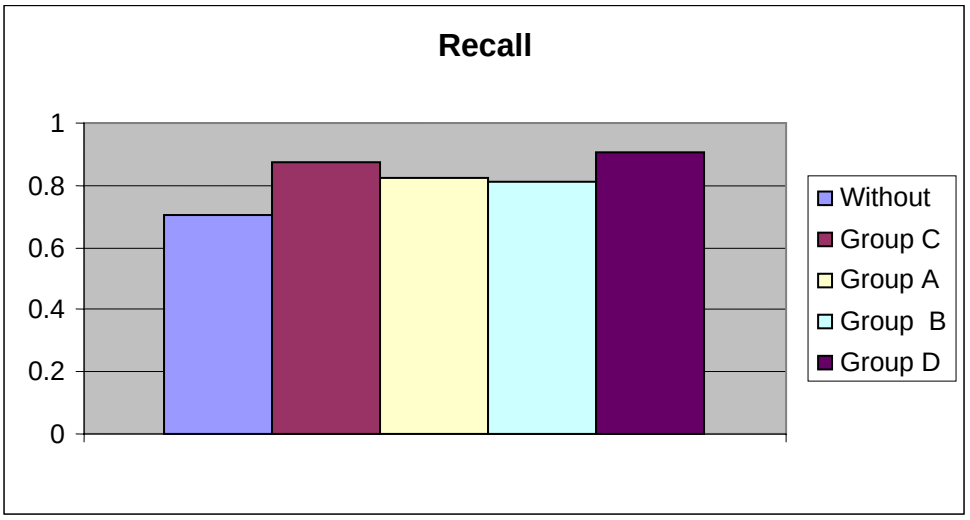

Figure 21: Average Recall of object detection for 2nd Video

Chapter 8

Conclusion and Future Work

Along with the increasing popularity of video over internet and versatility of video applications such as video surveillance, vision-based control, human-computer interfaces, medical imaging, robotics and so on, the availability and efficiency of videos will heavily rely on object detection and other related object tracking capabilities. Hence, we present an effective, efficient, and reliable scheme for automatically extracting independently moving video objects using motion vectors fields. Embedding our system with a specific configuration as a primary step before starting our object detection algorithm makes the performance much better and reduces the computation time for the object detection system as a whole. This has been verified by examining the results of our experiments. Furthermore, we believe our scheme can satisfy the requirements mentioned in Chapter 7, where we emphasize the importance of accurate motion vectors to human perception. It is a well-known fact that motion information is an important cue for humans to perceive video content. We are achieving the additional advantages of efficiency and speed by staying fully in the compressed domain, using only the P frame, and using a simple approach to filter implementation. Moreover, initialization and operation of our system is simple as well.

For the texture filter, AC coefficients in a DCT transformed macroblock can indirectly provide information on how textured the area of the image is. Low-textured region tends to cause poor encoding matching errors. As B and P frames are residue coded, their texture measures are propagated from the I-frames by inverse motion

compensation. The resultant macroblock from performing inverse motion compensation could overlap with four other 8x8 DCT blocks in I-frames. We measure the average of the four neighboring blocks' energy as an approximation to true block energy. Texture measure is then based on AC energy computed by grouping AC DCT coefficient into Horizontal, Vertical and diagonal energy groups then computing the average of the three energy.

In the future, we will use our proposed scheme in adopting the method which convert motion vectors in the MPEG coded domain to a uniform set, which is independent of the frame type and the direction of prediction. By utilizing these normalized motion vectors in our system, we expect to achieve better performance in object detection.

References

[1] N. Haering, R. J. Qian, and M. I. Sezan, “A Semantic Event-Detection Approach and Its Application to Detecting Hunts in Wildlife Video,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 6, pp. 857-868, 2000.

[2] H. L. Eng, and K. K. Ma, “Bidirectional Motion Tracking for Video Indexing,” Proc .Third IEEE Workshop on Multimedia Signal Processing, pp. 153-158, 1999.

[3] L. Favalli, A. Mecocci, and F. Moschetti, “Object Tracking for Retrieval Applications in MPEG-2,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 3, pp. 427-432, 2000.

[4] C. Faloutsos, R. Barber, M. Flickner, J. Hafner, W. Niblack, D. Petkovic, and W. Equitz, “Efficient and Effective Querying by Image Content,” Journal Intelligent Information Systems, Vol. 3, No. 1, pp. 231-262, 1994.

[5] A. Pentland, R. Picard, and S. Sclaroff, “Tools for Content-Based Manipulation of Image Databases.Storage and Retrieval of Image and Video Databases II,” Proc. SPIE, 34-47, 1994.

[6] V. V. Vinod and H. Murase, “Video Shot Analysis using Efficient Multiple Object Tracking,” Proceeding of the International Conference on Multimedia Computing and Systems, pp. 501-508, 1997.

[7] P. Fieguth, “Color-Based Tracking of Heads and Other Mobile Objects at Video Frame Rates,” Proceeding of IEEE Conference on Computer Vision and Pattern Recognition, pp. 21-27, 1997.

[8] N. Brady and N. O'Connor, "Object detection and tracking using an EM-based motion estimation and segmentation framework," Proceeding of IEEE International Conference on Image Processing, pp. 925–928, 1996.

[9] David P. Elias, The motion Based Segmentation of Image Sequences: Ph.D. thesis (Trinity College, Department of Engineering, University of Cambridge, Aug. 1998).

[10] N. Vasconcelos and A. Lippman, "Empirical Bayesian EM-based motion segmentation," Proc. of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 527–532, 1997.

[11] P. H. S. Torr, R. Szeliski, and P. Anandan, "An integrated bayesian approach to layer extraction from image sequences," IEEE Trans. Pattern Analysis and Machine Intelligence, Vol. 23, No. 3, pp. 297–303, 2001.

[12] R. Wang, and T. Huang, "Fast Camera Motion Analysis in MPEG domain," Proceeding of IEEE International Conference on Image Processing, pp. 691-694, 1999.

[13] R. C. Jones, D. DeMenthon and D. S. Doermann, "Building mosaics from video using MPEG Motion Vectors," Proceeding of ACM Multimedia Conference, 1999, pp. 29-32.

[14] J. I. Khan, Z. Guo and W. Oh, "Motion based object tracking in MPEG-2 stream for perceptual region discriminating rate transcoding," Proceeding of ACM Multimedia Conference, pp. 572-576, 2001.

[15] D.-Y. Chen, S.-J. Lin and S.-Y. Lee, "Motion Activity Based Shot Identification and Closed Caption Detection for Video Structuring," Proc. Visual Information Systems, 5th International Conference, pp. 288-301, 2002.

[16] T. R. M. King, Efficient and Effective Methods for Object Segmentation in Video Images: Final report (Fall Semester the Johns Hopkins University, http://www.apl.jhu.edu/Notes/Beser /525759/kingfinalreport.pdf, 2002).

[17] D.-Y. Chen, and S.-Y. Lee, “Motion-Based Semantic Event Detection for Video Content Description in MPEG-7,” Proceeding of IEEE Pacific-Rim Conference on Multimedia, pp. 110-117, 2001.

[18] Y. P. Tan, D. D. Saur, S. R. Kulkarni, and P. J. Ramadge, “Rapid Estimation of Camera Motion from Compressed Video with Application to Video Annotation”, IEEE Transactions on Circuits and Systems for Video Technology, Vol. 10, No. 1, pp. 133-146 , 2000.

[19] S.-F. Chang, “Compressed-Domain Techniques for Image/ Video Indexing and Manipulation,” IEEE International Conference on Image Processing, pp. 314-317, 1995.

[20] S.-F. Chang, Compositing and Manipulation of Video Signals for Multimedia Network Video Service: Ph.D. Dissertation (U.C. Berkeley, 1993).

[21] S. Chien, S. Ma, L. Chen, “Efficient Moving Object Segmentation Algorithm Using Background Registration Technique,” IEEE Transactions on Circuits and Systems for Video Technology, Vol. 12, No. 7, pp. 577-586, 2002.

[22] G. Kühne, S. Richter, and M. Beier, “Motion-based segmentation and contour-based classification of video objects,” Proc. ACM Multimedia Conference, pp. 41-50,2001.

[23] R. Curwen and A. Blake, Dynamic Contours Real-Time Active Splines (Active Vision MIT Press, 1992).

[24] R. Goldenberg, R. Kimmel, E. Rivlin, and M. Rudzsky, “Fast Geodesic Active Contours,” Proc. Int'l Conf. Scale-Space Theories in Computer Vision, pp. 34-45, 1999.

[25] R. Wang, H.-J. Zhang and Y.-Q. Zhang, “A Confidence Measure Based Moving Object Extraction System Built for Compressed Domain,” Proceeding of IEEE International Symposium on Circuits and Systems, pp. 21-24, 2000.

[26] Y. Ma, and H.-J. Zhang, “A New Perceived Motion based Shot Content Representation,” IEEE International Conference on Image Processing, pp. 426-429, 2001.

[27] M. Pilu, On Using Raw MPEG Motion Vectors To Determine Global Camera Motion (HPL-97-102, http://www.hpl.hp.com/techreports/97/HPL-97-102.pdf, 1997).

[28] R. V. .Babu, and K. R. Ramakrishnan , “Compressed Domain Motion Segmentation for Video Object Extraction”, Proceeding of IEEE International Conference on Acoustics, Speech, and Signal Processing, pp. 3788-3791, 2002.

[29] R. Ulichney, “Filter Design for void and cluster dither arrays,” Proc. SID Int. Symposium, pp. 809-812,1994.

[30] D.-Y. Chen, S.-Y. Lee, and H.-T. Chen, “Motion Activity Based Semantic Video Similarity Retrieval,” Proc. IEEE Pacific-Rim Conference on Multimedia, pp. 319-327, 2002.

[31] S.-C. Chen, M.-L. Shyu, C. Zhang, and R.L. Kashyap, “Video Scene Change Detection Method Using Unsupervised Segmentation and Object Tracking,” Proc. International Conference on Multimedia and Expo, pp. 57-60, 2001.

[32] N. W. Kim, T. Y. Kim, and J. S. Choi, “Motion analysis using the normalization of Motion Vectors on MPEG compressed domain,” Proceeding of The 2002 International Technical Conference On Circuits/Systems, Computers and Communications, pp. 1408-1411, 2002.

[33] Ashraf M.A. Ahmad; Duan-Yu Chen and Suh-Yin Lee “ROBUST COMPRESSED DOMAIN OBJECT DETECTION IN MPEG VIDEOS” Proc. 0f the 7th IASTED International Conference Internet and Multimedia System Applications, pp. 706-712, 2003.

[34] Ashraf M.A. Ahmad; Duan-Yu Chen and Suh-Yin Lee “Robust Object Detection Using Cascade Filter in MPEG Videos” Multimedia Software Engineering, 2003. Proc. Fifth International Symposium, pp. 196 – 203, 2003.

[35] Yu Zhong, Hongjiang, and Anil K. Jain “Automatic Caption Localization in Compressed Video”, IEEE Transactions on Pattern Analysis and Machine Intelligence, Vol. 22, No. 4, pp. 385-392, 2000.

[36] Jianhao Meng, Yujen Juan, Shih-Fu Chang “Scene Change Detection in a MPEG Compressed Video Sequence”, in IS&T SPIE Proceedings:Digital Video Compression Algorithm and Technology, Vol. 2419, pp. 14-25, (San Jose) , 1995.

[37] Yulin Wang and Ebroul Izquierdo “High-Capacity Data Hiding in MPEG-2 Compressed Video”, 9th Int. Workshop on Systems, Signals and Image Processing, Manchester, 2002.

[38] A. K. Jain and S. Bhattacharjee, “Text Segmentation Using Gabor Filters for Automatic Document Processing”, Machine Vision and Applications, Vol. 5, No. 3, pp. 169-184, 1992.

[39] A.K. Jain and Y. Zhong, “Page Segmentation Using Texture Analysis”, Pattern Recognition, Vol. 29, No. 5, pp. 743-770, 1996.

[40] B.L. Yeo and B. Liu, “Visual Content Highlighting via Automatic Extraction of Embedded Captions on MPEG Compressed Video”, Proc. SPIE Digital Video Compression: Algorithms and Technologies, Vol. 2668, pp. 38-47,1995.

[41] D. LeGall, “MPEG: A Video Compression Standard for Multimedia Applications”, Comm. ACM, Vol. 34, No. 4, pp. 46-58, 1991.

[42] Roberto Castagno, Touradj Ebrahimi, Murat Kunt “Video segmentation based on multiple features for interactive multimedia applications” IEEE Transaction on circuits and systems for video technology, Vol 8. No 5, pp 111-122, 1999.

[43] Hualu Wang, Shi-Fu Chang “ A highly efficient system for automatic face region detection in MPEG video, IEEETransaction on circuits and systems for video technology, Vol 7 No 4, pp. 89-99,1997.