Video Object Segmentation and Tracking

by

Themesha Murugas

BScEng(Electronic) *Cum Laude*

Submitted in fulfilment of the academic requirements of the Master of Science in Engineering Degree in the School of Electrical, Electronic and Computer Engineering, University of KwaZulu Natal, Durban, South Africa

January 2005

The research described in this dissertation was performed at the University of KwaZulu Natal and the former University of Natal by Ms Themesha Murugas under the supervision of Professor Roger Peplow and Dr Jules-Raymond Tapamo.

This work has been generously sponsored by Thales Advanced Engineering and Armscor.

I hereby declare that all the material incorporated in this thesis is my own original work except where acknowledgement is made by name or in the form of a reference. The work contained herein has not been submitted in part or whole for a degree at any other university.

Signed:

Name: Themesha Murugas

Date: January 2005

As the candidate's supervisor I have approved this dissertation for submission

Signed:		
Name:	R Peplow	J Tapamo
Date:	January 2005	January 2005

In the name of God, Most Gracious, Most Merciful

ACKNOWLEDMENTS

I would like to express my sincere gratitude to my supervisors, Prof Roger Peplow and Dr Jules Raymond Tapamo for their support and guidance during my thesis. They proved to be a source of sound technical advice and helped me acquire key analytical skills that helped me crystallise the vision of this research and uncover the finer details during development.

I would like to express my thanks to our sponsors, Armscor for funding my research and especially Peter Handley and Franzette Vorster for their keen interest in our work.

I would also like to express my gratitude to the NRF who awarded me a scholarship to pursue a master's degree and Karan Naidoo, the UND Scholarships Officer, for his assistance and guidance in securing funding.

To our Head of School, Prof Takawira, I would like to express my gratitude to you for giving me a lighter workload so that I may complete my studies.

To my family, for their constant nagging (which in hindsight I can be grateful for) to ensure that I did actually finish and for their unwavering support and encouragement throughout my studies, I offer my sincere gratitude and appreciation.

To all my friends, throughout this degree, I thank them for keeping me sane (or insane??) through numerous entertaining, sometimes thought-provoking and always fun tea breaks, bottomless coffee runs, late night crunch sessions and cue trips. You have made my experiences during the degree truly memorable and help me shape philosophies that I hope will guide me for the rest of my life.

And, finally, to my best friend, my husband, your belief in me and your encouragement in all my endeavours has helped me surpass hurdles that I did not imagine could be conquered. I am truly grateful to have you by my side.

ABSTRACT

One of the more complex video processing problems currently vexing researchers is that of object segmentation. This involves identifying semantically meaningful objects in a scene and separating them from the background. While the human visual system is capable of performing this task with minimal effort, development and research in machine vision is yet to yield techniques that perform the task as effectively and efficiently. The problem is not only difficult due to the complexity of the mechanisms involved but also because it is an ill-posed problem. No unique segmentation of a scene exists as what is of interest as a segmented object depends very much on the application and the scene content. In most situations *a priori* knowledge of the nature of the problem is required, often depending on the specific application in which the segmentation tool is to be used.

This research presents an automatic method of segmenting objects from a video sequence. The intent is to extract and maintain both the shape and contour information as the object changes dynamically over time in the sequence. *A priori* information is incorporated by requesting the user to tune a set of input parameters prior to execution of the algorithm.

Motion is used as a semantic for video object extraction subject to the assumption that there is only one moving object in the scene and the only motion in the video sequence is that of the object of interest. It is further assumed that there is constant illumination and no occlusion of the object.

A change detection mask is used to detect the moving object followed by morphological operators to refine the result. The change detection mask yields a model of the moving components; this is then compared to a contour map of the frame to extract a more accurate contour of the moving object and this is then used to extract the object of interest itself. Since the video object is moving as the sequence progresses, it is necessary to update the object over time. To accomplish this, an object tracker has been implemented based on the Hausdorff object-matching algorithm.

The dissertation begins with an overview of segmentation techniques and a discussion of the approach used in this research. This is followed by a detailed description of the algorithm covering initial segmentation, object tracking across frames and video object extraction. Finally, the semantic object extraction results for a variety of video sequences are presented and evaluated.

TABLE OF CONTENTS

LIST OF FIGURES AND TABLES

CHAPTER 1: INTRODUCTION

1.1 Motivation

For millenia, we as humans have attempted to capture and record our experiences visually, however until Joseph Niepce produced the first photograph in 1827, such representations were limited to paintings and drawings. Subsequent advancements in camera technology have made video and image cameras commonplace today and visual media has become an essential aspect of communication.

Visually, the simplest to capture is an image as this enables us to view an instant of time, while video is essentially a sequence of images that enables us to view an event spread over a period of time. By first defining an image as a two dimensional spatial representation of light intensity, we may describe video as a three-dimensional light intensity function with the third dimension being time.

Video today has two representations, namely, analog and digital. An analog video is essentially a discretised version in the time dimension and in one of the two spatial dimensions. It remains continuous in the other spatial dimension and in light intensity values. A digital video, on the other hand, is discretised in all the dimensions namely, time, space and light intensity values. Digital video has the advantage over analog video in that it is easily manipulated, easily duplicated without loss of quality and easily stored and transmitted. It is becoming increasingly ubiquitous and is already applied in video conferencing, DVDs, video CD, Video-On-Demand and high definition TV.

With the advent of improved video technology, it is possible to capture video for longer periods of time and at a higher quality. This, however, has resulted in large volumes of video data being produced and subsequently a need for video compression in order to make storage and transmission more feasible. This need for video compression is better appreciated with the following example. Consider storing an uncompressed 2-hour digital colour movie. A digital movie is made up of frames and each frame is subsequently made up of pixels or picture elements. Using a colour transmission standard such as NTSC, which is the American National Television System Committee standard, each pixel would need 3 bytes, 1 byte each for the red, blue and green components. A single frame with a standard resolution of 352x288 pixels, would hence require 304,128 bytes. An PAL format video has 25 frames per second; hence, 7,603,200 bytes are required to store 1 second of the movie. Therefore the 2-hour movie would require 54,743,040,000 bytes (~54 Giga bytes)!

Video coding schemes, like MPEG-1, MPEG-2, H.261 and H.263, were introduced with the intent of standardizing video compression algorithms to meet the bandwidth requirements of the applications that they were designed to support. H.261 and H.263 are International Telecommunications Union (ITU) Telecommunication Standards that were designed for low bit rate applications (64kbps-1920kbps), mainly video-conferencing. The Moving Picture Expert Group (MPEG) standard, MPEG-1 was released as a standard for medium quality and medium bit rate video and audio compression. It allowed video to be compressed by ratios in the range of 50:1 to 100:1, depending on image sequence type and desired quality. The encoded data rate was targeted at 1.5Mb/s, for this is a reasonable transfer rate of a double-speed CD-ROM player. MPEG-2 was later introduced to cover a wider range of applications. The primary application targeted during the MPEG-2 definition process was the all-digital transmission of broadcast TV quality video at coded bit rates between 4 and 9 Mb/sec.

Anticipating the rapid convergence of telecommunications, computer and TV/film industries, the MPEG group officially initiated a new MPEG-4 standardization phase in 1994. The mandate was to standardize algorithms and tools for the coding and flexible representation of audio-visual data to meet the challenges of future multimedia applications and application requirements. Unlike its predecessors, the new MPEG-4 Video Standard was intended to support clusters of functionalities, which may be used by a variety of applications across a broad spectrum of bandwidths.

The main functionalities that are supported are:

- **Transport:** In principle, MPEG-4 does not define transport layers. In a number of cases, adaptation to a specific existing transport layer has been defined.
- **DMIF:** DMIF, or *Delivery Multimedia Integration Framework*, is an interface between the application and the transport that allows the MPEG-4 application developer to avoid addressing issues related to specific transport layers. A single application can run on different transport layers when supported by the right DMIF instantiation.
- **Systems:** MPEG-4 defines a toolbox of advanced compression algorithms for audio and visual information. The data streams that result from the coding process can be transmitted or stored separately, and need to be composed so as to create the actual multimedia presentation at the receiver side. The systems part of MPEG-4 addresses the description of the relationships between the audio-visual components that constitute a scene.
- **Audio:** MPEG-4 audio facilitates a wide variety of applications that could range from intelligible speech to high quality multi-channel audio, and from natural sounds to synthesized sounds.
- **Visual:** The MPEG-4 visual standard allows the hybrid coding of natural images and video together with synthetic scenes. To this end, the visual standard comprises tools and algorithms supporting the coding of natural (pixel based) still images and video sequences as well as tools to support the compression of synthetic 2-D and 3-D graphic geometry parameters (i.e. compression of wire grid parameters, synthetic text, etc). A wide range of bit rates are supported, typically between 5 kbit/s and more than 1 Gbit/s, with resolutions from sub-QCIF(128x96) to 'Studio quality (4k x 4k pixels). For all bit rates addressed, the compression algorithms are very efficient. The standard also allows for content-based functionalities which allow for manipulation and interaction of the video content, scalability of textures, image and video, robustness in error prone environments and face and body animation

To facilitate this new range of capabilities, MPEG-4 has introduced an object-based approach, where the representative architecture supports audio-visual objects. The standard provides a large and rich set of tools for the coding of these objects. In order to allow effective implementations of the standard, subsets of the MPEG-4 Systems, Visual, and Audio tool sets have been identified, that can be used for specific applications. These subsets, called 'profiles', limit the tool set a decoder has to implement. For each of these profiles, one or more levels have been set, restricting the computational complexity. The scope of our research is limited to video and we shall therefore focus on visual profiles and more specifically the representative architecture for these profiles.

To enable the content-based interactive functionalities envisioned, the MPEG-4 standard diverges from the video source format used for the MPEG-1 and MPEG-2 standards, where each frame is processed as a single rectangular region and introduces the concept of Video Object Planes (VOP's). These are objects which are obtained by segmenting each frame of an input video sequence into a number of arbitrarily shaped image regions according to predefined criteria. It should be noted however that the single rectangular region format is considered as a special case of the arbitrarily shaped VOP ideology, where the entire frame is considered as a single VOP.

The concept of video object planes may be further explained using the illustration in Figure 1-1. The frame in the example is segmented into three video objects planes, namely the man, the truck and the background. These three Video Object Planes make up the input to be coded. As may be observed from the diagram, each VOP is an image region of arbitrary shape; however, each contains a single physical object. The shape and location of the region can vary from frame to frame; however successive VOP's belonging to the same physical object in a scene will constitute a single video object (VO) - a sequence of VOP's of possibly arbitrary shape and position. The shape, motion, and texture information of the VOP's belonging to the same VO is encoded and transmitted or coded into a separate video object layer (VOL). In addition, relevant information needed to identify each of the VOL's - and how the various VOL's are composed at the receiver to reconstruct the entire original sequence - is also included in the bit stream. This allows the separate decoding of each VOP and the required flexible manipulation of the video sequence as indicated in Figure 1-1.

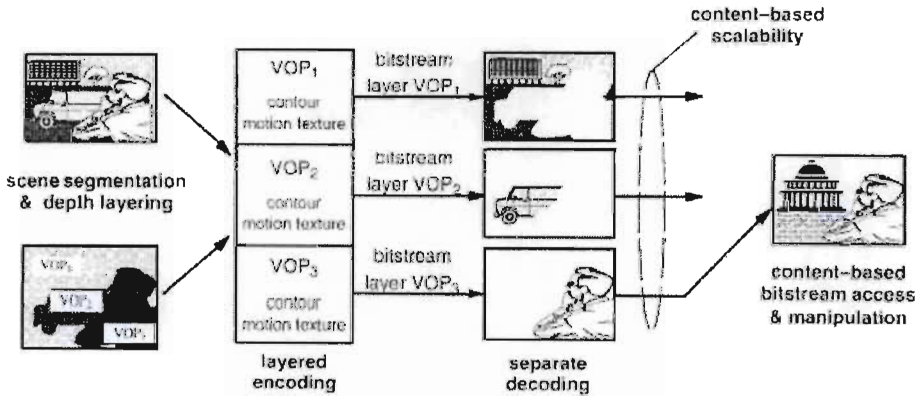


Figure 1-1: Video Object Plane Generation [Siko, 1997]

While MPEG-4 provides a framework for encoding video objects, the actual segmentation of a video sequence is considered as a pre-processing step and is not defined by the standard. The justification for this is that the definition of a video object is very application or user dependent and hence it is not practical to standardise the segmentation of a video sequence. The video source input assumed for the VOL structure therefore needs to either already exist in terms of separate entities (i.e, is generated with chroma-key technology) or be generated by means of on-line or off-line segmentation algorithms.

These segmentation algorithms may either be manual, where the user would have to identify and segment the video object planes before encoding each frame, or automated where computers may be used to perform the segmentation. A manual approach would be incredibly time consuming, tedious and not very practical, hence to maximize the benefits of object-based representation, the MPEG-4 standard needs to be complemented with automatic techniques for extracting video objects from video data.

This need for automated techniques of generating and maintaining video objects has stimulated a wave of research in the field of video object segmentation. It has also raised philosophical questions about the definition of semantically meaningful objects and has uncovered the numerous difficulties of creating a comprehensive theory of segmentation. Since research into video object segmentation has only gained momentum in the last decade, it is still a relatively new field and hence provides fertile ground for philosophical and developmental research.

1.2 Investigated approach

This dissertation deals with the problem of extracting and maintaining video objects. As already mentioned, segmenting video data into video objects for video object plane generation is a difficult task. It is essentially a matter of identifying semantically meaningful objects in a scene and separating them from the background. While, the human visual system is capable of performing this task with minimal effort, development and research in machine vision is yet to yield techniques that perform the task as effectively and efficiently. The problem is not only difficult because of the complexity of the mechanisms involved but also because segmentation itself is an ill-posed problem.

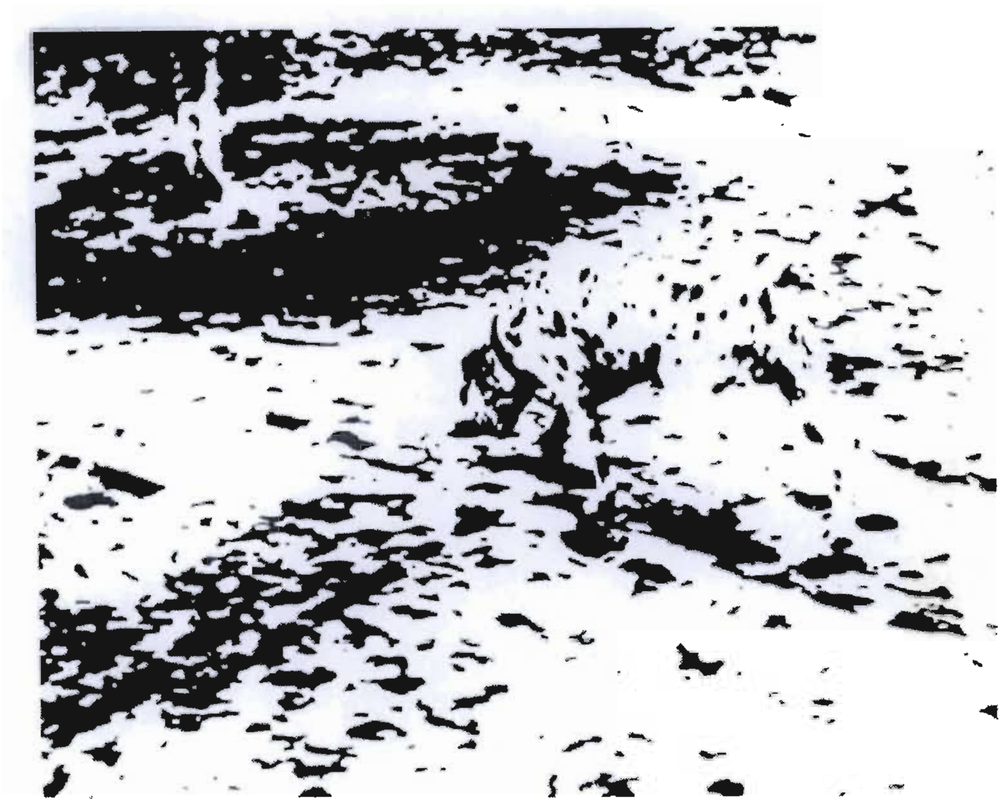


Figure 1-2: The Segmentation Dilemma-Picture of a Dalmatian on Shadowed Background [Fors, 2002]

This may be illustrated by the image in Figure 1-2 of a Dalmatian on a shadowed background. The blobs that form a Dalmatian appear to be assembled as they are, only because they form a Dalmatian. This is hardly a satisfactory explanation for the grouping of image data and is one that raises difficult computational questions. It leads one to conclude that an important aspect of vision is organizing image information into meaningful assemblies and also provided the starting point of this investigation, which was to understand the concept of "meaningful assembly".

This philosophical analysis lead to the conclusion that no unique segmentation of a scene exists as what is of interest as a segmented object depends very much on the application and the scene content. In the light of these inherent difficulties with the segmentation problem, it was also hard to see that there could be a comprehensive theory of segmentation. There were however several classes of segmentation schemes that could be identified, each with their own set of strengths and weaknesses and the choice of segmentation scheme would be determined by the specific problem or application and the constraints and goals associated therewith. Hence, the next step was to review the current hierarchy of image and video segmentation schemes and identify a class or classes that would be suited to video object generation.

The highest level of the hierarchy is composed of three broad categories, manual, semi-automatic and automatic segmentation approaches. While one could appreciate the complexity of a completely automatic segmentation tool for generic images or video sequences, we supported the view that to complement the vision of an audiovisual standard such as MPEG-4, the vision of an automatic, efficient and effective video object extraction scheme was necessary. Hence we focused our research on the class of schemes that facilitate automatic segmentation.

More specifically we focused on the class of automatic schemes that incorporates *a priori* knowledge of the nature of the problem, that is, information about the nature of the video objects or the application domain that is known prior to performing the segmentation and that may be used within the segmentation scheme. In addition to providing information about the spatial domain, video also provides information in the time domain. Hence, it is known prior to viewing or processing a video that it will contain motion (if it does not contain motion the video reduces to being a single static image). It is also known, that motion may be used to identify semantic objects. Hence we focused on segmentation schemes that manipulate motion information.

This, however, first required a review of motion and how one projects the motion in the 3-dimensional real world onto a 2-dimensional video sequence. It was also necessary to review how one may estimate and characterise the motion of objects within a sequence. After understanding the notion of motion and motion estimation techniques, it was then possible to review the different classes of segmentation schemes based on motion. This culminated in the selection of a video object segmentation strategy that had the potential to evolve into a fully automatic generic video object generation tool. We selected a scheme that had the potential to operate in real time and which could be enhanced to extract physical, real world objects.

The analysis of the state of the art in the field of segmentation provided a horizontal perspective of the solution domain. As part of this research work, we have implemented the selected approach to obtain a vertical or more in depth perspective of the segmentation problem and the solution domain.

The selected strategy attempts to increase the *a priori* knowledge by requiring the user to tune a set of input parameters prior to execution of the algorithm. To further reduce the complexity of the problem we have constrained the types of video sequences that can be processed.

We have applied the constraints that there is only one moving object in the scene and that the only motion in the video sequence is that of the object of interest. We also assume that there is constant illumination and no occlusion of the object with parts of the background.

To detect the moving object we use a change detection mask and then apply morphological operators to refine the result. The change detection mask yields a model of the moving components. This model is then compared to a contour map of the frame to extract a more accurate contour of the moving object, which is then used to extract the object of interest itself. Since the video object is moving as the sequence progresses, it is necessary to update the object over time. To accomplish this we have implemented an object tracker based on the Hausdorff object-matching algorithm and update the object contour accordingly.

As mentioned the aim of implementing a segmentation strategy was firstly to obtain a more in-depth understanding of the problem of segmentation and secondly to investigate practically whether a particular scheme which was selected after a literary review and theoretical analysis could be evolved into a fully automatic video object generation tool.

Hence, this research was concluded with an analysis of the implemented strategy and recommendations for driving this research forward towards realising the goal of automatic video object segmentation.

1.3 Organisation of dissertation

The dissertation is organized as follows. Chapter 2 describes the review of the field of segmentation. It begins with an analysis of the concept of semantic objects, presents the hierarchy of segmentation classes and then focuses specifically on motion segmentation. The notion of motion and motion estimation techniques are reviewed and then specific motion segmentation techniques are discussed. The chapter concludes with a motivation for the approach applied to video object segmentation in this study. In subsequent chapters, the selected approach is described in detail. The algorithm may be structurally decomposed into three parts. The first part of the algorithm manipulates both temporal and spatial information to obtain an initial model of the moving object. This is described in chapter 3. The second part is the tracking of the object in subsequent frames and updating it as it changes. This is described in chapter 4. In chapter 5, the method of extracting the video object is presented. Each of these chapters includes the results of the algorithm obtained at the relevant level of development. Finally the thesis is concluded in chapter 6 where a summary of the work, important conclusions and suggestions for the way forward are presented.

CHAPTER 2: EXTRACTING SEMANTIC OBJECTS

In this chapter, we expand on the notion of semantic objects and object segmentation. We first try to formalise what we mean when we refer to semantic objects and then present a hierarchy of segmentation approaches. Research into video object segmentation merges several different fields; these include, pattern recognition, artificial intelligence, general signal processing, image processing and statistics. Many different approaches to segmentation were investigated independently and sometimes within the same time frame in each of these fields. Hence in presenting an overview of the literature, an attempt is made to present the scope of object segmentation schemes by classifying the approaches into three broad categories: automatic, semi-automatic and manual. We then focus on the class of approaches which this research work falls into, namely automatic motion based segmentation schemes. Before delving into motion-based segmentation, it is necessary to present the fundamentals of video motion information. Hence the projection of 3D motion onto the 2D video plane and motion estimation techniques are discussed. Thereafter the subclasses of segmentation schemes, which incorporate motion information, are presented. The chapter is concluded with a discussion of the context of our research in the scope of the current literature.

2.1 Concept of semantic video objects

In the past, a video sequence was characterised by a sequence of frames and each frame was made up of pixels. However, research into bridging the gap between the human visual system and machine processing of images and video has prompted a shift towards representing a video sequence as consisting of meaningful objects. As human beings are able to interpret what they see as meaningful objects, for example, a house, a car or a tree, the current trend in video and image processing is to develop techniques that enable the computer to perform similar meaningful classifications of visual data.

This new approach to video representation increases the interaction capability, allowing the user to manipulate entities as if they were physical objects. It also has benefits in a large number of applications including video coding, video surveillance, virtual reality and video editing.

A semantic video object may be regarded as a collection of pixels that correspond to a real object in successive video object planes of a video sequence. What one may consider a semantic object usually depends on the application. For example, in a room surveillance system, the people would be regarded as video objects, while in a military application that tracks planes; the plane itself is the video object of interest. One may thus conclude that defining a semantic object is usually a complex and delicate task.

The extraction of semantic video objects is usually a two-fold task. It is first necessary to identify the collection of pixels that constitute the video object and then track its evolution as the sequence progresses. When extracting a video object, there are two main requirements. The first is ensuring the spatial accuracy of the object; that is, the precise definition of the object boundary - pixel-wise spatial accuracy is usually the targeted goal; the second requirement is temporal coherence, that is, ensuring the spatial accuracy is maintained over time.

2.2 Semantic video object extraction strategies

In the previous section, it was made apparent that what constitutes a video object depends on a human interpretation of what is meaningful. This interpretation needs to be translated into a set of rules that may be applied through an algorithm or human operator. Depending on the extent of human intervention, the extraction process may be defined as manual, automatic or semi-automatic.

2.2.1 Manual segmentation

Manual segmentation is where a human operator demarcates the boundary of the video object via a user interface. As a result this method is very accurate but also very time consuming and is therefore not suitable for typical applications like video coding or surveillance. The approach does have merit however where the spatial accuracy of the extraction is of a high priority and the segmentation is performed for a limited number of frames. For example, in Marichal and Villegas [Mari, 2000], manual segmentation is used to generate a reference mask, which is used as a benchmark for evaluating the results of an automated technique in terms of spatial accuracy and temporal coherence.

2.2.2 Semi-automatic segmentation

Semi-automatic segmentation techniques allow for a more synergetic relationship between the human user and the computer. The user interaction is still necessary for defining the semantic object but most of the processing is performed by the computer, thereby increasing the speed of extracting a complete and accurately defined video object.

Three main approaches for introducing user interaction in segmentation algorithms can be found in the literature:

- Feature-based: The user is asked to select sets of object pixels that have homogenous colour or texture. The feature information of these pixels is then used as a basis for object extraction where the remaining pixels are classified into one of the object types defined. A typical example of feature based segmentation is demonstrated by Chalom and Bove [Chal,1996]. In their research effort, several features, namely, motion, texture, position and colour are used. The user determines the number of objects the image will be segmented into and selects sample pixels from each region. These sample points are used to model a probability density function for each feature in each of the regions. Using this probability density function, the remaining points in the frame are assigned to one of the regions using hypothesis testing. The advantage of this method is that it is not necessary for the user to precisely select the objects. However, in this approach, the user is not assured that the pixels of an object will remain connected and further interaction might be necessary. The number of objects is also predetermined and the approach cannot cope with new objects entering the scene.

- Contour-based: The user roughly marks the position of object boundaries as either a set of control points or as a sketch of the object and the algorithm subsequently adjusts the boundary to the actual contour. In Gu and Lee [Gu, 1998] contour based semiautomatic segmentation is implemented by allowing the user to use a mouse to trace the outer contour of the object of interest in the initial frame. From this outline of the object and further user input by means of parameter selection, an initial estimate of the outer and inner boundaries of the object are obtained. The object would be accurately extracted when the outer and inner boundaries converge onto the same contour. To achieve this convergence a morphological classification scheme is used to classify the pixels between the inner and outer boundaries into one of the two regions. This yields a segmentation of the initial frame. Motion estimation is then used to track the object as the sequence progresses. Since tracking introduces boundary errors, the object is intermittently updated by further user interaction. These methods yield smooth accurate contours but tend to be more time consuming.

- Region-based: In region-based schemes, an initial partition of the image into regions is performed. The user is then allowed to merge regions from the preliminary segmentation to obtain a semantic object. In Castagno, Ebrahimi, and Kunt [Cast, 1998] region based segmentation is performed by an automatic segmentation algorithm that extracts homogeneous regions based on an adaptive combination of multiple features. The relative weights of motion and spatial information are evaluated locally based on their level of reliability. The obtained regions are then grouped into semantically meaningful objects through user interaction. This approach has the advantage that the computer does not have to solve the ill-posed problem of determining exactly what constitutes a semantic object and handles instead the low level computing.

2.2.3 Automatic segmentation

The integration of automatic and interactive tools leads to an extraction process that allows the segmentation of more generic image sequences. The approach is more efficient than manual approaches, but is still not as robust as a fully automatic segmentation. While semiautomatic schemes have established their own niche, there is still a need for fully automatic segmentation techniques, that can cope with the appearance of new objects, the evolution and even disappearance of existing objects and which does not require a human operator's interaction during the segmentation process.

Fully automatic extraction techniques are still in their infancy, mainly because translating the properties of a semantic object as perceived by a human observer into extraction criteria that can be implemented by a computer, is a very difficult task. These methods algorithmically apply rules that define a semantic video object according to special characteristics of the scene or on *a priori* information. Such methods tend to be used for a specific application or class of applications.

A typical example of methods based on a specific set-up of the scene is the chroma-keying approach. Chroma-keying is a method of blending multiple video streams that is commonly used in the broadcast and film industries. The video streams are blended by removing the portion of the foreground image that matches a designated colour (usually blue) and then mapping the remaining image onto the background video image. This concept has been applied to object segmentation by filming real world objects in front of a known uniformly coloured background. The background is then eliminated by discarding pixels with the background colour. The method gives good spatial accuracy and temporal coherence. However, if the real world objects contain pixels that are the same colour as the background, these pixels are discarded as well. In addition, chroma-key technology relies on the strong assumption that the screen behind the subject of interest is known. It is therefore not generically applicable.

A more viable approach to automatic segmentation is to incorporate some knowledge of the video objects prior to extraction. Methods based on *a priori* information include template matching, face detection and moving object segmentation.

Lin [Lin, 2002] presents an approach that uses *a priori* information to achieve real-time object extraction with an active camera. Prior to performing segmentation, a number of background images with equally spaced pan and tilt angles are captured and analysed to create a panoramic mosaic image of the background. This image is then used as the reference background model and is compared to the live video feed to detect the foreground object. The result is a crude extraction of the object that is used to control the camera to focus the object onto the center of the image plane and then further process it for more accurate extraction and to track the object. In the post-processing, filtering is first performed to mitigate the effects of noise, and then the background model is updated using the background information from the current frame of the video sequence. To refine the segmentation and track the object, template matching is used. The template is generally a specific geometric feature or set of features of the object that is known *a priori.* In this instance, the object extracted from the previous frame is used as the template and is matched to the object in the current frame to track the object.

Another method of incorporating *a priori* information is to tailor the segmentation for a specific class of objects. For example, if it is known prior to object extraction that the objects of interest are human faces then the extraction method can be tailored to specifically detect and extract faces. Garcia and Tziritas [Garc, 1999] applied colour based segmentation where the face detection task consisted of finding the pixels whose spectral characteristics lay in a specific region in the chromaticity diagram. The set of homogenous skin colour regions were then iteratively merged to provide a set of candidate face areas. Feature information, namely, face shape and texture is then incorporated by performing wavelet packet decomposition on each candidate face area and subsequently extracting simple statistical features. These features were used as the basis for face detection.

For extracting moving objects, motion information can be used as a semantic. The motion of a moving object usually differs from that of the background and of other objects. Hence there are many automatic extraction strategies that apply motion information. Automatic motion based segmentation is the interest area of this research work, hence motion-based segmentation schemes are described in more detail in section 2.3.3.

2.3 Motion as a semantic for segmentation

In video sequences, the temporal dimension provides information about the motion of foreground objects and the background. Each moving object usually has a coherent motion that is distinct from that of the background. This makes motion a very useful feature for segmenting video sequences. It may be used to complement other features such as colour, intensity, or edges that are commonly used for segmentation of still images or be used as the sole criterion for segmentation. In this section the notion of motion and techniques for estimating the motion in video sequences is presented. This is followed by an overview of motion based segmentation techniques.

2.3.1 The notion of motion

Each frame of a digital video sequence is a 2-D spatio-temporal sample of a 3-D scene. The unit of representation may be denoted by *I(x,y,k)*. This corresponds to the intensity value of each pixel at coordinates, (x, y), in frame, k. Since the captured sequence represents a 3-D to 2-D projection, the relationship between the observable changes in image intensities, referred to as apparent motion, and the motion of 3-D objects in the scene is a complex one. One of the main reasons for this complexity is that the projection from 3-D onto the 2-D image plane results in loss of information. Hence several points in the 3-D scene with potentially different motions may project to the same 2-D point in the image as shown in Figure 2-1.

The relationship between apparent motion and the actual scene also depends on the properties of motion itself. The simplest scenario is a static scene containing a single moving object being filmed by a stationery camera. However, this can be made more complex by having multiple moving objects, a moving camera and or a dynamic scene. Another influential factor is whether the moving objects are rigid or change shape as a result of the motion, as is the case for a person walking.

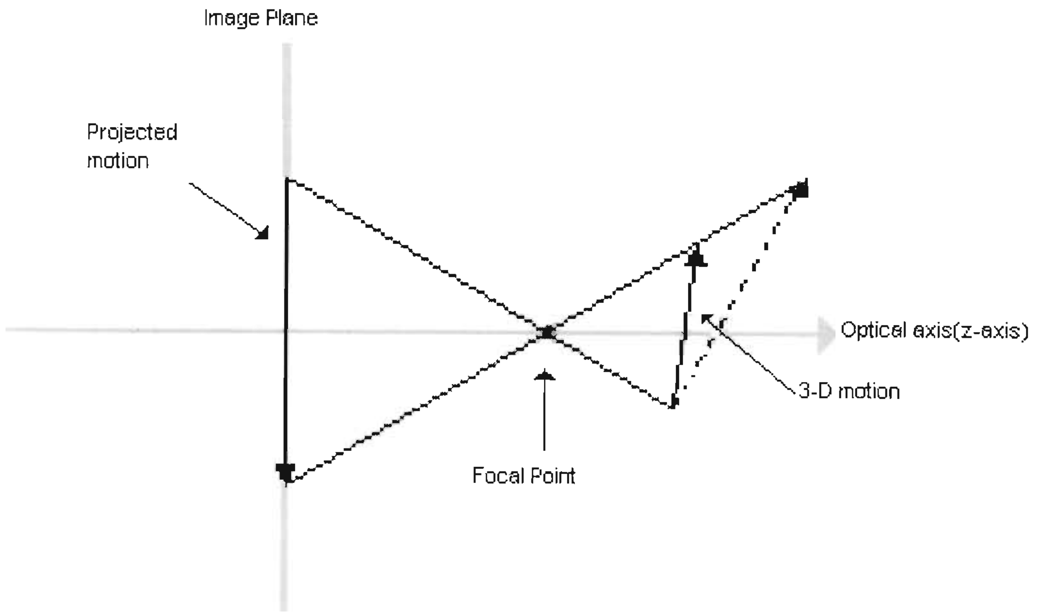


Figure 2-1: The geometry of perspective projection. All light rays reflected off objects in the 3-D scene pass through the focal point before hitting the projected plane. The inherent ambiguity where multiple different 3-D vectors will project to the same 2-D vector, is indicated. [Krug, 1998]

Furthermore, the apparent motion is not necessarily equivalent to the projected 2-D motion, since it is also dependent on factors such as global illumination. As an example consider a static scene with varying illumination. The 2-D motion is zero because there is no 3-D motion but as a result of changing illumination, the apparent motion is non-zero. Moving objects must also contain sufficient texture to generate apparent motion. For example, a smooth, featureless rotating sphere exhibits zero apparent motion despite the existence of 3-D motion. This implies that for motion estimation, the 3-D moving object projection needs to exhibit a sufficient degree of intensity variation.

2.3.2 Motion estimation

Optical flow techniques

One approach to characterising apparent motion is to use a correspondence vector or optical flow field. A correspondence vector describes the displacement of a pixel between two frames while the optical flow *(u,v)* at pixel *(x,y,k)* refers to the velocity, $\left(\frac{dx}{dt}, \frac{dy}{dt}\right)$. Since velocity is the time derivative of displacement, correspondence vectors and optical flow are related.

The above approach to motion estimation is based on the assumption of optical flow which is described mathematically by the optical flow equation [Horn, 1981] and is shown in equation 2.1.

Given a pixel, $\{x(0), y(0), 0\}$, one can associate a path through time that describes its projected motion:

$$I\left(x(t), y(t), t\right) = c \qquad 2.1$$

where c is a constant and $\{x(t), y(t), t\}$ is the path of the pixel parameterised with respect to time. This is further illustrated in Figure 2-2.

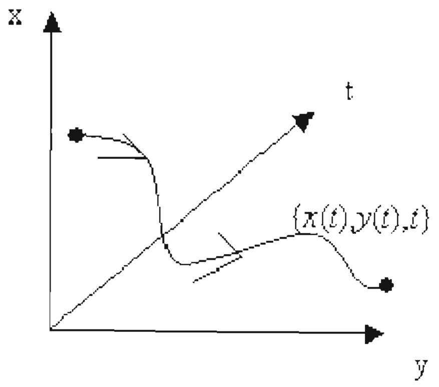


Figure 2-2: Illustration of optical flow: The optical flow constraint assumes that a point that moves along a path in 3-D space of a video sequence has a constant intensity.

Equation 2.1 may be rewritten in the form reflected in equation 2.2 by applying the Taylor's series expansion and the chain rule.

$$\frac{\delta I}{\delta x}\frac{dx}{dt} + \frac{\delta I}{\delta y}\frac{dy}{dt} + \frac{dI}{dt} = 0 \qquad 2.2$$

This equation relates the projected motion of the pixel, $\left(\frac{dx}{dt}, \frac{dy}{dt}\right)$ to the partial derivatives of $I(x,y,t)$ and hence, an alternate representation of the video sequence where pixels are represented and grouped according to their projected motion. Before reviewing motion estimation algorithms based on optical flow, it is necessary to reflect on some of the issues associated with such techniques. Firstly, when using equation 2.2 to find the projected motion, the problem becomes a minimisation problem with two unknowns and only one constraint. Usually another constraint such as motion smoothness is added to equation 2.1. Secondly, the optical flow equation assumes that the intensity of a point is constant over time, however as discussed in section 2.3.1, there are many counterexamples such as a rotating reflective surface, changing lighting conditions and shadows. It is therefore necessary to compensate for such scenarios prior to determining the optical flow.

In addition to these, there are two inherent problems [Truc, 1998] with the optical flow assumption: the aperture problem and the correspondence problem. The aperture problem occurs when there is insufficient greylevel variation in the considered image region to uniquely constrain the problem, that is, more than one candidate motion fits the image equally well. The correspondence problem occurs when there is more than one correspondence vector for a single pixel. It may also occur when a moving object covers background that was previously uncovered or vice versa as in such regions no data correspondence exists and the motion is therefore undefined. As a result of such intrinsic problems with motion analysis, additional assumptions are necessary to obtain a unique solution. These usually impose further smoothness constraints on the optical flow field to achieve continuity.

If the problem is sufficiently constrained to avoid the abovementioned problems, the optical flow field may be used to perform motion segmentation. The dense field would need to be grouped into regions of uniform motion and the motion field in each region would need to be described. Two ways of describing motion fields are possible, namely, parametric and non-parametric methods.

In the nonparametric representation, a dense field is estimated where each pixel is assigned a correspondence or flow vector. An example of a non-parametric representation is the popular block matching approach [Stil, 1999] and variants thereof. Each frame is divided into equally sized blocks and for each block the best match in the next (or previous) frame is computed. The correspondence vector indicates the translation the block and hence all constituent pixels, has undergone. Block-based methods are heavily dependent on the size of the block. A too large block-size might contain more than one motion and cannot accurately locate motion boundaries, whereas small blocks often result in wrong matches within uniform regions in the presence of noise. A weakness of block-matching algorithms is their inability to cope with rotations and deformations. Nevertheless, their simplicity and relative robustness make them a popular technique.

Nonparametric dense field representations are generally not suitable for segmentation because an object moving in the 3-D space generates a spatially varying 2-D motion field even within the same region, except for the simple case of pure translation. That is the reason why parametric models are commonly used in segmentation algorithms. However, dense field estimation is often the first step in calculating the model parameters. Parametric models require a segmentation of the scene and describe the motion of each region by a set of a few parameters. Both the six-parameter affine and eight-parameter perspective models [Stil, 1998] are very popular, but many other transformations exist depending on the assumptions made. The motion vectors can then be synthesized from these model parameters. A parametric representation is more compact than a dense field description and less sensitive to noise, because many pixels are treated jointly to estimate a few parameters. Parametric models describe each region by one set of parameters that is either estimated by fitting a model in the least squares sense to a dense motion field obtained by a nonparametric method or directly from the luminance signal as in [Wang, 1994][Musm, 1995]. Although parametric representations are less noise sensitive, they still suffer from the intrinsic problems of motion estimation. It should be noted that one has to be careful when interpreting an estimated flow field. Most likely, it is necessary to include additional information such as colour or intensity to accurately and reliably detect boundaries of moving objects.

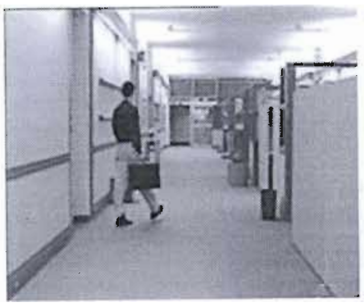
Change detection techniques

An alternate approach to motion estimation is change detection, a simple but powerful type of motion analysis. An example of this is shown in Figure 2-3. The original video sequence is captured with a stationary camera and the change areas, black regions in Figure 2-3(b), indicate the silhouette of a man walking.

From the optical flow equation, discussed in the previous section, if the background is stationary, then $\frac{dx}{dt} = 0$ and $\frac{dy}{dt} = 0$. Furthermore, motion paths associated with the background follow a simplified form of the optical flow equation:

$$\frac{\delta I}{\delta t} = 0 \qquad 2.3$$

Therefore, non-background pixels have non-zero values of $\frac{\delta I}{\delta t}$. This may be used to detect change through the behaviour of $\frac{\delta I}{\delta t}$ as a function of time. Change detection may be applied to sequences with a non-stationary background by first compensating the background motion through offsetting of the frames. In such schemes, however, the reliance on motion compensation may compromise the change detection as the change detection then depends on the accuracy of the motion compensation. Some movement is also not equivalent to change: for instance, in Figure 2-3(b), the homogenous regions within the human figure do not exhibit change properties through temporal correlation. Also change detection is a binary operator that distinguishes only two classes of objects, background and non-background and may therefore incorrectly group objects in scenes of three or more classes.


(a) Original image of video sequence, a person walking through a hall with a stationary camera

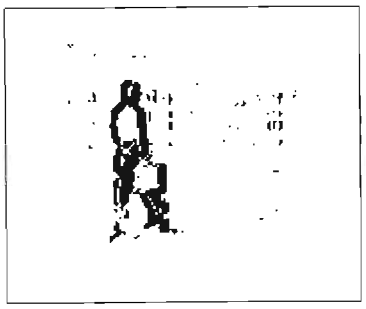

(b) The Change Detection image indicates changing regions.

Figure 2-3: Example of Change Detection

2.3.3 Motion segmentation

Optical flow methods

The classical approach to motion segmentation is to first perform dense field motion estimation and base the segmentation on this motion information. One of the earliest works that applied this technique was by Adiv [Adiv, 1985]. In this approach, the optical flow field is first determined. Thereafter, flow vectors that have similar affine co-ordinates are grouped into clusters using a Hough transform. By assuming that each cluster represents the motion of a planar object, the object's motion may be recovered and clusters with similar motions are then merged. Finally, ungrouped motion vectors are assimilated into one of their neighbouring segments.

Wang and Adelson [Wang, 1994] extended this approach to the problem of video object extraction. They consider a video sequence as being composed of several overlapping layers. Each layer consists of an intensity map, describing the intensity profile of a coherent motion region over many frames, an alpha map describing its relationship with other layers and a parametric motion map describing the motion of the region. As in Adiv's approach, the optic flow field is first estimated. Based on a set of initial conditions, hypotheses of motion are generated. Each hypothesis is defined by an affine parameter set estimated by a least squares method. The parameters are evaluated to determine how reliable they are. After each location in the image is classified to one of the hypotheses, the parameters are recalculated using the results of the initial hypothesis generation. This process is repeated until a reliable segmentation is achieved. The results of the segmentation of the first frame are then used to iteratively segment the next frame and the process continues.

The layered representation provides good results for sequences that contain a few regions undergoing simple motions but more complex motions that are not easily described by the affine motions require special treatment. The layered approach itself can also add artefacts, for example, since the affine approximation of the motion is imperfect, the synthesised motions do not perfectly match the actual motions. Also the layered segmentation is imperfect so bits of an object might be attached to the incorrect layer. Other problems with the algorithm are that it is overly dependent on the motion field estimation and it has difficulty in handling the appearance and segmentation of new objects.

A more elegant approach to segmenting a dense motion field into moving objects is by using Bayesian methods. The main idea is to find the maximum *a posteriori* (MAP) estimate of the segmentation, *X*, for a given observation, *O*, that is, to maximise $P(X|O) \propto P(O|X) * P(X)$.

The application of a Bayesian MAP criterion to motion field segmentation was first performed by Murray and Buxton [Murr, 1987]. The approach was based on the assumption that the objects were roughly planar and that normal components of flow were available. The cost function used consisted of a term measuring the spatial variation in the segmentation labelling, a term measuring the error between the current motion estimate and the normal flow field, a term representing "line processes" (i.e. motion boundaries) and a term measuring the temporal variation in the segmentation labelling. The function was minimized iteratively, using simulated annealing, starting with a "random" guess at the solution.

Bouthemy and Lalande[Bout, 1990] applied this method on real images taken with a static camera, in order to find moving vehicles. A similar cost function to that of Murray and Buxton was used but without line processes and very simple labelling was applied where each image point was classified as either static or moving. François and Bouthemy [Bout, 1991] extended this method to segment images taken from a moving camera; the optical flow constraint equation was used to find normal optic flow. Regions were assumed to have linearly varying flow, and were labelled according to their motion type (i.e. qualitatively estimated as having divergence, rotation or shear).

These approaches to motion segmentation only used optical flow data in the segmentation decision, and hence their performance was limited by the accuracy of the estimated optical flow field and inevitably suffered from the problems discussed in section 2.3.2.

Chang, Tekalp and Sezan [Chan, 1993] improved on these approaches by incorporating intensity information into the observation, *O*. They presented a general MAP framework for segmentation of image sequences using estimated inter-frame displacement vector fields and greylevel information. Using the MAP criterion, they define the desired segmentation x^*, as the one that maximises the *a posteriori* probability of segmentation x, given the observed data, $y = (u, v, g)$, where u and v represent the horizontal and vertical components of the displacement vector field and g the greylevel intensity information.

From Bayes rule, the *a posteriori* probability is expressed as

$$P(x \mid y) \propto P(y \mid x)P(x) = P(u, v, g \mid x)P(x) \qquad (2.4)$$

$P(u,v,g|x)$ denotes the conditional probability density of the observation given the segmentation and $P(x)$ is the *a priori* probability distribution of the segmentation process.

The conditional probability is characterised by a model that regards the data entities, u, v, and g as being statistically independent as in equation 2.5.

$$P(u, v, g \mid x) = P(u|x)\, P(v|x)\, P(g|x) \qquad (2.5)$$

Each of these is modelled by a Gaussian probability distribution function that takes into consideration the pixel site, the segmentation label and white Gaussian noise. Equation 2.6 gives the standard Gaussian distribution function.

A more elegant approach to segmenting a dense motion field into moving objects is by using Bayesian methods. The main idea is to find the maximum *a posteriori* (MAP) estimate of the segmentation, *X*, for a given observation, *O*, that is, to maximise $P(X|O) \propto P(O|X) * P(X)$.

The application of a Bayesian MAP criterion to motion field segmentation was first performed by Murray and Buxton [Murr, 1987]. The approach was based on the assumption that the objects were roughly planar and that normal components of flow were available. The cost function used consisted of a term measuring the spatial variation in the segmentation labelling, a term measuring the error between the current motion estimate and the normal flow field, a term representing "line processes" (i.e. motion boundaries) and a term measuring the temporal variation in the segmentation labelling. The function was minimized iteratively, using simulated annealing, starting with a "random" guess at the solution.

Bouthemy and Lalande[Bout, 1990] applied this method on real images taken with a static camera, in order to find moving vehicles. A similar cost function to that of Murray and Buxton was used but without line processes and very simple labelling was applied where each image point was classified as either static or moving. François and Bouthemy [Bout, 1991] extended this method to segment images taken from a moving camera; the optical flow constraint equation was used to find normal optic flow. Regions were assumed to have linearly varying flow, and were labelled according to their motion type (i.e. qualitatively estimated as having divergence, rotation or shear).

These approaches to motion segmentation only used optical flow data in the segmentation decision, and hence their performance was limited by the accuracy of the estimated optical flow field and inevitably suffered from the problems discussed in section 2.3.2.

Chang, Tekalp and Sezan [Chan, 1993] improved on these approaches by incorporating intensity information into the observation, *O*. They presented a general MAP framework for segmentation of image sequences using estimated inter-frame displacement vector fields and greylevel information. Using the MAP criterion, they define the desired segmentation x^*, as the one that maximises the *a posteriori* probability of segmentation x, given the observed data, $y = (u, v, g)$, where u and v represent the horizontal and vertical components of the displacement vector field and g the greylevel intensity information.

From Bayes rule, the *a posteriori* probability is expressed as

$$P(x \mid y) \propto P(y \mid x)P(x) = P(u, v, g \mid x)P(x) \tag{2.4}$$

$P(u,v,g|x)$ denotes the conditional probability density of the observation given the segmentation and $P(x)$ is the *a priori* probability distribution of the segmentation process.

The conditional probability is characterised by a model that regards the data entities, u, v, and g as being statistically independent as in equation 2.5.

$$P(u, v, g \mid x) = P(u|x)\, P(v|x)\, P(g|x) \tag{2.5}$$

Each of these is modelled by a Gaussian probability distribution function that takes into consideration the pixel site, the segmentation label and white Gaussian noise. Equation 2.6 gives the standard Gaussian distribution function.

$$p(b) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(b-\mu)^2}{2\sigma^2}} \tag{2.6}$$

μ = mean of distribution (also at the same place as mode and median)

σ^2 = variance of distribution

b is a continuous variable ($-\infty \leq b \leq \infty$)

The mean of the conditional probability distribution for greylevel information is modelled by using the sample mean of greylevels belonging to the same segmentation label.

The distribution mean associated with the displacement data is determined by one of two methods. The first method assumes that the horizontal component information and the vertical component information are slowly varying functions of the spatial co-ordinates and defines them to be equal to the mean value of the displacement vectors associated with a particular segment. In the second method, all sites in a frame that share the same segment label are assumed to be part of the same object. The object's motion is characterised by an 8-parameter model that is derived from the displacement field vectors within the object. The horizontal and vertical component means are then obtained from the model parameters.

The variance for both the greylevel and motion information is defined as a function of the displacement field accuracy. The displaced frame difference is used as a measure of this accuracy and if the motion information is reliable, it is more heavily weighted in determining the conditional probability ($\sigma^2_{\text{motion}} = 1$, $\sigma^2_{\text{graylevel}} >> 1$) else the greyscale information is given a higher confidence weighting ($\sigma^2_{\text{motion}} >> 1$, $\sigma^2_{\text{graylevel}} = 1$).

The *a priori* probability density is determined on the basis of an *a priori* probabilistic model for the desired segmentation process, *x*. It is assumed that *P(x)* has a Gibbs distribution, and is thus given by,

$$P(x) = \frac{e^{-U(x)}}{Z} \tag{2.7}$$

where Z is the partition function and $U(x)$ is the Gibbs potential function associated with the sum of all clique potentials in the lattice [Li, 1995]. The potential function is defined as the sum of both spatial and motion-compensated-temporal cliques and the clique potentials are chosen so as to encourage spatio-temporal connectivity.

Having thus derived a method for computing the conditional and *a priori* probabilities, the *a posteriori* probability is finally maximised by iterative conditional mode [Besa, 1986] to obtain the segmentation.

The Bayesian framework may also be used to treat motion estimation and segmentation jointly. In further work by Chang *et al* [Chang, 1998] the observation is reduced to only the grey values of the current frame, g_k, and search frame, g_{k-1}, and both the motion field and segmentation have to be estimated as in equation 2.8.

$$P(u,v,x|g_k,g_{k-1}) = \frac{P(g_k|u,v,x,g_{k-1})P(u,v|x,g_{k-1})P(x|g_{k-1})}{P(g_k|g_{k-1})} \tag{2.8}$$

The denominator is constant with respect to the unknowns and is therefore not considered in the MAP estimates. The conditional probability density function, which is the first term in the numerator, quantifies how well the motion and segmentation estimates fit the given frames. It is modelled by a Gibbs distribution where the displaced frame difference is used in the potential function to give an indication of the accuracy of the optical flow estimates. The second term in the numerator is the conditional probability of the motion field given the segmentation and the search frame. This is also modelled by a Gibbs distribution. The potential function is a weighted sum of two estimates; the deviation of the motion field from the parametric motion field estimate of the segmentation and a piecewise smoothness constraint imposed on motion vectors that share the same segmentation label. Lastly, the third term in the numerator, the *a priori* probability of the segmentation is also modelled as a Gibbs function with the potential function defined to encourage the formation of contiguous regions. The MAP estimate is maximised by iterated conditional mode and highest confidence first [Chou, 1990].

Bayesian methods provide a more elegant framework for solving the segmentation of a scene than most ad hoc approaches; however most of the Bayesian methods have been proposed within a coding framework and hence do not yield as good results for semantic object extraction. The main drawback of Bayesian methods is the high computational burden. Many of the approaches also require the number of segments to be input by the user.

Change detection methods

Rather than estimating and characterising motion as a measure for segmentation, another approach is to detect changes in a video sequence and use this as a criterion for segmentation. One of the earliest works using this approach was by Musmann, Hotter and Osterman [Musm,1989]. A change detector, based on frame differencing, is used to divide the current frame into changed and unchanged regions. Each distinct changed image region is interpreted as a separate object. A hierarchically structured approach is applied which starts with the analysis of the largest object in the hierarchy to estimate its shape, motion and colour parameters, and analyse the smaller objects in successive steps of the hierarchy. An eight-parameter motion model is used to estimate the motion of the object. The parameters are obtained directly from the frame difference of two successive frames. A Taylor series expansion is used to describe the image luminance in the object's neighbourhood as a function of the unknown mapping parameters. An estimate of the frame difference is then expressed as a function of the spatial intensity gradients and the motion parameters. Frame differences and gradients are easy to compute and the model parameters are obtained by linear regression. If the prediction error after motion compensation is too large, the object is further subdivided and analyzed in subsequent levels of hierarchy. The algorithm sequentially refines the segmentation and motion estimation until all changed regions are accurately compensated.

It is considered by many that Musmann, Hotter and Osterman pioneered research in the field of change detection object-based video segmentation and coding. Many researchers have since based their work on this approach.

Mech and Wollborn [Mech, 1998] incorporate change detection in a four-step approach to estimating the 2D shape of moving objects. First, an eight-parameter motion model is used to estimate and compensate for apparent camera motion. In the next step, a possible apparent scene cut is detected and if necessary the algorithm is reset. The actual segmentation takes place in the third and fourth steps. A change detection mask is estimated by a local thresholding technique and then smoothed by a relaxation technique. A memory is applied to the change detection masks to improve the temporal coherency of the resulting objects. Next, the change detection mask is subdivided into parts belonging to the moving object and the uncovered background using an estimated displacement vector field. Finally, the resultant object mask is adapted to luminance edges in the current frame to improve the accuracy of the estimated object.

Neri *et al* [Neri, 1998] perform moving object segmentation by applying statistical methods to frame differencing and motion estimation. In their operation, foreground objects are discriminated from the background on the basis of the estimated motion field. The motion field evaluation however is restrained to those areas that present significant inter-frame changes only. Object movement contributions appear in the inter-frame differences as highly structured signals whose statistical behavior strongly deviates from a Gaussian, while random variations due to noise, illumination changes, and slow variations of the background textures are likely to be Gaussian. They manipulate this characteristic by applying partially modeled stochastic signal detection and estimation to perform coarse detection and motion estimation. Consequently, they resorted to the Higher Order Statistics of the inter-frame difference.

In particular, potential foreground regions are identified by comparing the locally estimated fourth-order moment of the inter-frame difference with a threshold proportional to the estimated background activity. The choice of the fourth-order moment of the inter-frame difference is suggested by the trade-off between noise rejection capabilities and variance of the sample statistics. This method reveals areas belonging to moving objects, as well as uncovered background. Motion analysis is then performed on the coarse detection results to more accurately detect the boundary. The motion estimation is performed by analyzing the temporal evolution of the coarse detection results, or more, specifically the thresholded fourth-order moment of the inter-frame differences. Finally, a cascade of morphological filters is applied to remove irregularities and smooth the contour.

Instead of trying to extract moving objects in the foreground, a complimentary approach to moving object segmentation, is to identify and remove the background. Chien, Ma and Chen [Chien, 2002] present a segmentation algorithm using a background registration technique. The basic idea of their approach is change detection. However, unlike the change detection methods discussed, the motion information is not extrapolated from the frame difference of two consecutive frames. Instead, an up-to-date background information buffer is constructed and maintained and each frame is compared with the background. An object region is any region that is significantly different from the background region. Their approach, however, necessitates a static background or global motion compensation prior to executing the algorithm.

Their algorithm is divided into five major steps. In the first step, the frame difference mask is calculated by thresholding the difference between two consecutive input frames. Then, according to the frame difference mask of the past several frames, pixels that are not moving for a long time are considered as reliable background in the background registration step. This step maintains an up-to-date background buffer as well as a background registration mask indicating whether the background information of a pixel is available or not. In the third step, the current input image and the background image are compared to generate the background difference mask. This background difference mask is the primary information for object shape generation. In the fourth step, an initial object mask is constructed from the background difference mask and the frame difference mask. If the background registration mask indicates that the background information of a pixel is available, the background difference mask is used as the initial object mask. Otherwise, the value in the frame difference mask is copied to the object mask. The initial object mask generated in the fourth step has some noise regions because of irregular object motion and camera noise. Also, the boundary region may not be very smooth. In the last step, these noise regions are removed and the initial object mask is filtered to obtain the final object mask.

Mathematical morphology methods

Mathematical morphology is the manipulation of shapes using mathematical principles [Hara,1987]. It is very attractive for the purpose of video segmentation as it efficiently deals with geometrical features such as size, shape contrast and connectivity, which are considered as segmentation orientated features.

One of the earlier works that applied mathematical morphology to video segmentation was by Salembier and Pardas [Salem, 1994] who incorporated morphology into a hierarchical segmentation scheme. The first level of the segmentation is coarse with only a few regions; however the contours are perfectly localised. Subsequent levels of the hierarchy successively refine the segmentation by introducing new regions without modifying previous segmentation results.

All levels of the hierarchy basically perform segmentation and therefore involve the same elementary steps. The first step is simplification as this facilitates segmentation. The simplified sequence and the current segmentation are then used to extract markers that indicate the presence of 3D homogenous areas with time being the third dimension. In the third step, the extracted markers and the input signal to the current level of the hierarchy are used in a decision step that outputs the segmentation result. The last step is a quality estimation step that indicates to the next level in the hierarchy where the segmentation needs to be refined or improved.

A large number of morphological tools rely on two basic sets of transformations known as erosions and dilations. In their work, Salembier and Pardas use two sets of greyscale erosions and dilations. The first deals with erosion and dilation with a flat structuring element. If *f(x)* denotes an input signal and M_n, a window or flat structuring element of size n, then the erosion and dilation by the flat structuring element M_n is as represented in equation 2.9.

$$\text{Erosion:}\quad \varepsilon_n(f)(x) = \text{Min}\{f(x+y), y \in M_n\}$$
$$\text{Dilation:}\quad \delta_n(f)(x) = \text{Max}\{f(x-y), y \in M_n\} \qquad (2.9)$$

The second, geodesic dilation of size one, is defined as the minimum between the dilation of size one of the original signal *f(x)* and the reference function, *r*. The geodesic erosion is then defined by duality. Both are represented in equation 2.10

$$\begin{aligned}&\text{Geodesic Dilation of size one:} \quad \delta^{(1)}(f,r) = \text{Min}\{\delta_1(f),r\}\\ &\text{Geodesic Erosion of size one:} \quad \varepsilon^{(1)}(f,r) = -\delta^{(1)}(-f,-r)\end{aligned} \qquad (2.10)$$

Salembier and Pardas use geodesic dilation and erosion of infinite size, which is also called reconstruction by dilation (by erosion) [Salem, 1995] and is given by equation 2.11. As may be observed from the equation, geodesic dilations and erosions of arbitrary size are defined by iterations.

$$\begin{aligned}&\text{Reconstruction by dilation: } \gamma^{(\text{rec})}(f,r) = \delta^{(\infty)}(f,r) = \ldots\delta^{(1)}(\ldots\delta^{(1)}(f,r)\ldots,r)\\ &\text{Reconstruction by erosion: } \varphi^{(\text{rec})}(f,r) = \varepsilon^{(\infty)}(f,r) = \ldots\varepsilon^{(1)}(\ldots\varepsilon^{(1)}(f,r)\ldots,r)\end{aligned} \qquad (2.11)$$

Elementary erosions and dilations allow the definition of morphological filters such as the morphological opening and closing indicated in equation 2.12.

$$\begin{aligned}&\text{Morphological opening: } \gamma_n(f) = \delta_n(\varepsilon_n(f))\\ &\text{Morphological closing: } \varphi_n(f) = \varepsilon_n(\delta_n(f))\end{aligned} \qquad (2.12)$$

Morphological opening simplifies the original image by removing bright elements that do not fit within the structuring element and morphological closing deals similarly with dark elements. If the simplification has to deal with both dark and light elements, an open-close or close-open filter has to be used. Salembier and Pardas used morphological filters in the simplification step of their algorithm, but favoured morphological filters by reconstruction as these perform better at contour preservation. An opening by partial reconstruction and closing by partial reconstruction, as indicated by equation 2.14, was used. Partial reconstruction filters differ from reconstruction filters (equation 2.13) by performing a smoothing of the reference signal by either a small opening or closing. This approach is better for 3D signals as it allows a partial reconstruction of the contour which is sometimes necessary, especially in the time domain where covered and uncovered areas may be connected during the reconstruction process.

$$\begin{aligned}&\text{Opening by reconstruction of erosion: } \gamma^{(\text{rec})}(\varepsilon_n(f),f)\\ &\text{Closing by reconstruction of dilation: } \varphi^{(\text{rec})}(\delta_n(f),f)\end{aligned} \qquad (2.13)$$

$$\begin{aligned}&\text{Opening by partial reconstruction: } \gamma^{(\text{rec})}(\varepsilon_n(f),\gamma_k(f))\\ &\text{Closing by partial reconstruction: } \varphi^{(\text{rec})}(\delta_n(f),\varphi_k(f))\end{aligned} \qquad (2.14)$$

A second popular morphological tool, namely watersheds, is used in the decision step of the algorithm. The watershed algorithm as its name suggests derives from topographic works. In image processing, this notion has been introduced by considering the greylevel values of a picture as the altitude of an imaginary relief map. As indicated in Figure 2-4, watershed lines are used to partition the space by associating a region called a catchment basin to each local minimum. One of the more efficient computations of the watershed is based on immersion simulations [Vinc, 1991], which consists of flooding the surfaces from its local minimum. Starting from the minima of lowest altitude, the water progressively fills up the catchment basins. When the water level reaches that of other minima, these also become active and the flooding process also originate from them. When the waters from two different minima merge, an imaginary dam is created to prevent any mixing. This process is continued until the water is higher than the absolute maximum. Hence, a segmentation is obtained that is based on signal minima. The classical watershed algorithm uses the morphological gradient of the signal to segment, however Salembier and Pardas use a modified approach that works directly on the signal. The second modification is that instead of defining the priority of a pixel by its greylevel, the priority is defined by the degree of certainty with which a pixel belongs to a given region.

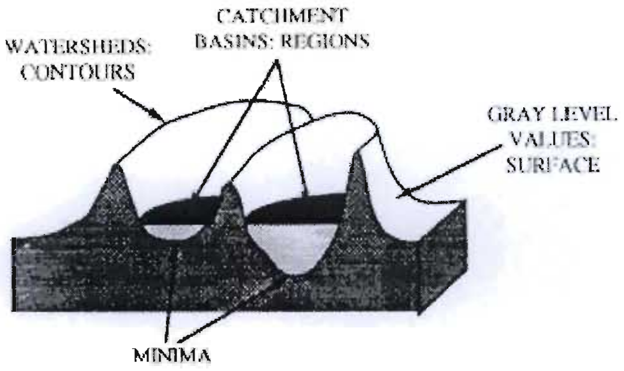


Figure 2-4: Graphical illustration of watershed analogy in image processing [Salem, 1994]

Salembier *et al* [Salem, 1996] extended this work by incorporating an additional projection step that warps the previous partition onto the current frame. The projection is computed by the watershed algorithm and ensures temporal continuity and segmentation linking.

The approach by Salembier is an expansion of 2-D methods into a 3-D framework. Choi, Lee and Kim [Choi, 1997], on the other hand, presented a framework for specifically manipulating the 3D or video framework. Their approach consists of a three-step algorithm that performs spatio-temporal segmentation using morphological tools and a joint similarity measure.

The first step is joint marker extraction. A joint marker is a region homogenous in motion and luminance within an object that is used to identify the object. The image is first simplified by morphological open-close by reconstruction filters. As mentioned previously these filters remove regions that are smaller than a given size but preserve the contours of the remaining objects. The luminance or intensity markers are then identified by labelling flat zones, that is, by labelling the connected components of the space where the function is of constant grey-level. Motion markers within the intensity markers are then identified by an iterative method. With each iteration, the aim is to identify the largest homogenous motion region among the motion vectors of the intensity marker that do not yet belong to any of the motion markers created in previous iterations. If this region is larger than the size threshold, it is added as a joint motion-luminance marker.

The joint marker extraction step determines the number and interior of the regions to be segmented. There are however, still a number of pixels that have not been assigned to a marker. These pixels are usually situated close to the boundaries of the object. The next step is to more accurately determine the boundaries of each object. The watershed algorithm is used to achieve this. The pixels within the joint marker definitely belong to the object they identify and are used to initialize each object. The flooding process starts from these markers and grows the object regions by successively adding pixels from the uncertainty area to the nearest similar region. A joint similarity measure, defined below, is used as a similarity criterion.

$$S(x,y;R) = \alpha S_m(x,y;R) + (1-\alpha)S_i(x,y;R) \tag{2.15}$$

The motion similarity, $S_m(x, y; R)$, between the pixel under consideration at coordinates (x, y), and the region, R, is determined by the displaced frame difference. It is calculated using the intensity value of the pixel and the motion vector at the pixel as generated by the affine motion parameters of the region, R. The intensity similarity, $S_i(x, y; R)$, is calculated as the absolute difference between the intensity value of the pixel under consideration and the intensity value at the pixel as generated by first order polynomial coefficients of the region, R. The joint similarity measure, $S(x, y; R)$, is the weighted sum of the motion and luminance similarities where α is used as a weighting factor. Once a new pixel has been assigned to a region, the motion and intensity models are updated so that the joint similarity may be accurately computed for new pixels.

The final step of the algorithm is motion based region fusion. The segmentation to this point has produced regions that are homogenous in both motion and luminance. Regions may correspond to the same object, that is, having homogenous motion, but different luminance polynomials. Motion based merging of the regions is performed to reduce the over-segmentation and thus simplify the segmentation. Adjacent regions with the same affine motion are merged together. This is done by computing an optimal affine transformation given a set of flow vectors for a pair of adjacent regions. If the standard deviation of the actual flow vectors from those predicted by the optimal affine parameters is less than a given threshold, the candidate regions are merged. The process is repeated until no more merging is possible.

As already mentioned, segmentation algorithms incorporating mathematical morphology are attractive because they allow the exploitation of object-oriented characteristics and facilitate more meaningful segmentation results. They are also computationally efficient in comparison to the previous methods discussed. For these reasons, "morphology-based" spatio-temporal segmentation algorithms are gaining increasing popularity.

2.4 Investigated approach

The analysis of the state of the art of semantic object extraction and more specifically motion segmentation of video objects, as discussed in the previous sections, was used to guide the approach to segmentation as implemented in this research effort.

From the investigations we concluded that it is advantageous to incorporate motion information in video object segmentation strategies as it supplements the two-dimensional spatial information with information from the time domain. The two primary classes of motion estimation are optical flow techniques and change detection methods. Both approaches are popular in the literature but the latter is generally computationally less intensive, and allows one to extract object information with simpler mathematical models. It was also observed from the investigations that morphological tools are well suited to object segmentation because they efficiently compute geometric feature information such as size, shape and connectivity.

Hence we chose an object extraction approach that incorporates mathematical morphology and change detection. It is based on work by Meier and Ngan [Meier, 1998] whose research in this area is in turn based on that by Huttenlocher *et al* [Hutt, 1990] who originated their work as a theoretical analysis of efficiently computing the Hausdorff distance as a function of translation and rigid motion. In subsequent work [Hutt, 1993a], they proved the theoretical algorithms by showing that the Hausdorff distance could be efficiently computed in practice as well. This result stimulated research into applications of the Hausdorff distance in image and video processing, which culminated in a third publication by Huttenlocher *et al* [Hutt, 1993b] which showed that it could be used for tracking non-rigid objects in video sequences. While the Hausdorff tracker was robust, the difficulty of such an application lay in obtaining an initial model of the object to commence the tracking. Meier and Ngan [Meier, 1998] advanced their research by incorporating change detection and morphological tools into an algorithm for obtaining an initial model and thus lay the foundation for a video object extraction and tracking system.

This research work is an analysis of the system proposed by Meier and Ngan and proposes a further step towards automatic segmentation by automating the threshold selection in the change detection algorithm.

CHAPTER 3: MODEL INITIALISATION

The aim of the segmentation strategy is to extract arbitrarily shaped moving objects from a video sequence. The extraction is performed using 2D image information and is based on the criteria of motion. The design and development of a completely automatic system for extracting semantically meaningful video objects from generic video sequences is a very difficult task and to the best of our knowledge is still an open research problem. To simplify and constrain the focus of our research, a few assumptions were made. The first is that the only movement in the video sequence is by the video object of interest. This implies that the background and camera is stationary. The second is that there is no occlusion of the object either with the background or with the other objects. Given this assumption, the algorithm can be designed for a single object and easily extended to multiple objects. The third constraint is that there is constant illumination and the fourth that there is only a single camera.

Our approach is based on the work of Meier and Ngan [Meier, 1998], which applies pattern recognition, morphological and object tracking principles to moving object segmentation. The core of the technique is an object tracker that maintains the temporal correspondence of objects throughout the video sequence. An edge image is first used to derive a binary model of the object. The tracker then matches the model against subsequent frames updating the model each frame to accommodate for changes in shape and rotation. The main difficulties in the approach are obtaining an initial model and updating the model for changes in shape and rotation. The initial model is obtained by performing change detection that incorporates morphological tools. This step is the focus of this chapter. In the next chapter, the tracking algorithm is presented and in chapter 5 the object extraction is discussed.

Section 3.1 describes the first part of the model initialisation step which is to extract a model of moving components by using frame differencing and morphological tools. The result of this stage is a change detection mask that uncovers the occlusion zones between the object and the background. This mask tends to coincide with the boundary of the object but does not match the contour exactly. To yield a more accurate extraction of the contour of the object, spatial information is incorporated by comparing the change detection mask to the luminance edge map of the original frame and selecting the subset of the edge map pixels that coincide with the moving components as the model of the moving object. These steps are described in section 3.2.

3.1 Change detection mask

The algorithm for the estimation of the moving components model can be subdivided into the steps illustrated in Figure 3-1. In the following subsections, these steps will be described.

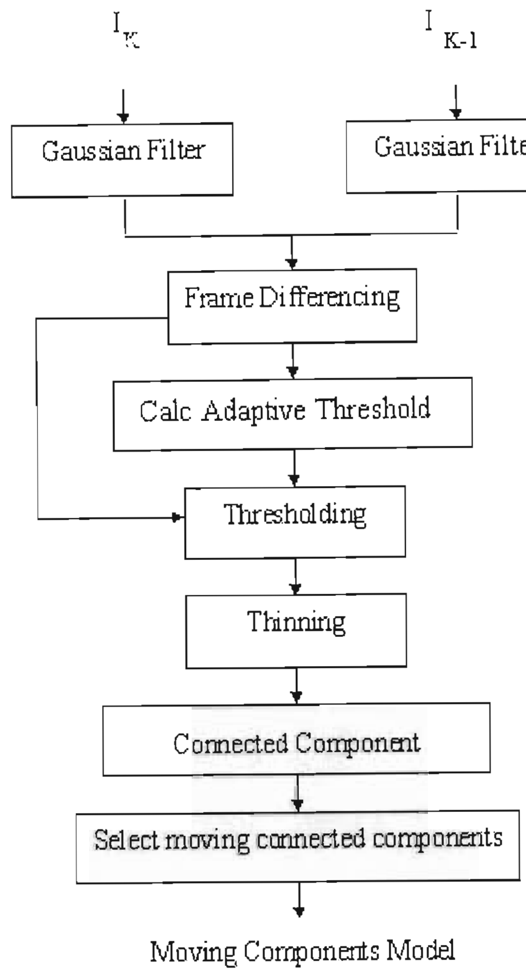


Figure 3-1: Block Diagram of Change Detection Model

3.1.1 Gaussian filtering

In image processing, noise may refer to any entity in the image, data or intermediate results that is not relevant for the purposes of the main computation. Furthermore different types of noise may be countered by different techniques depending on its nature and characteristics. The first step in this work is to compensate for inherent noise in the image by applying smoothing to the original frames, I_k and I_{k+1}, by classical filter operations. It is assumed that the noise is additive and random, that is, a spurious, random signal, $n(i,j)$, is added to the true pixel values, $\ddot{I}(i,j)$:

$$I(i,j) = \ddot{I}(i,j) + n(i,j) \tag{3.1}$$

There are two primary classes of filters, linear and non-linear. Linear filters are linear shift invariant, that is, they can be implemented by convolving the filter kernel with the image. They have well defined properties and their output is easier to predict than the latter. Non-linear filters, on the other hand, are more difficult to characterize and more computationally intensive. For our work, we have chosen a linear filter, for its computational simplicity.

Mathematically, linear filtering may be defined as follows.

Let I be an NxM image and A, the kernel of a linear filter, that is, a WxW mask. The filtered image, I_A, is then given by the discrete convolution in equation 3.2.

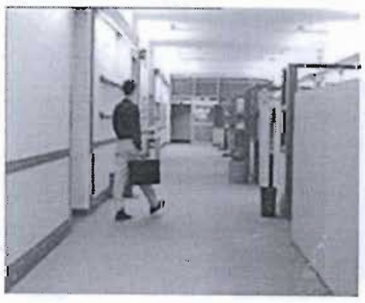
$$I_A = I*A = \sum_{h=-\frac{W}{2}}^{\frac{W}{2}} \sum_{k=-\frac{W}{2}}^{\frac{W}{2}} A(h,k)I(i-h,j-k) \qquad (3.2)$$

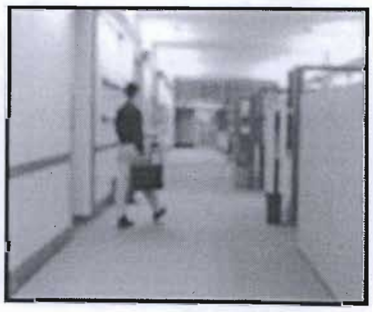
A simple choice for the kernel would be one that replaces each pixel with the mean or average of its neighbourhood. In our work, we have used a Gaussian filter which is a particular case of averaging in which the kernel is a 2-D Gaussian. This filter is a better low-pass filter than the simple mean filter because the Fourier transform of a Gaussian is still a Gaussian and therefore it has no secondary lobes. In addition, the kernel is separable; hence the Gaussian filter can be efficiently implemented.

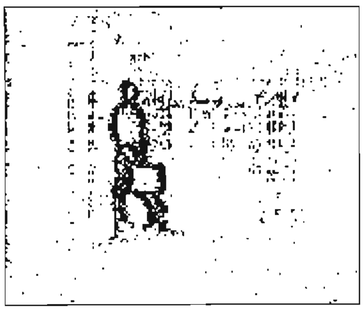
The coefficients of the mask are computed using the definition of a Gaussian distribution. It is first necessary to determine a relationship between the mask width, W, and the variance of the continuous Gaussian, σ. This relationship may be established by imposing the criterion that W should subtend most of the area under the Gaussian distribution. An adequate choice is given in equation 3.3 below, which ensures that W subtends 98.76% of the area.

$$\sigma = \frac{W}{5} \qquad (3.3)$$

The Gaussian filter was applied to each original frame before frame differencing and thresholding, which are discussed in the following sections. Figure 3-2(a) shows the original frame of the MPEG-4 video test sequence, *Hall Monitor*. This sequence is a corridor scene where the man in the foreground is the moving object walking down the corridor. Figure 3-2(c) shows the results after frame differencing and thresholding without any noise smoothing, while Figure 3-2(d) shows the results after smoothing the original image with a 5x5 Gaussian kernel. It is apparent from results that a more accurate identification of the moving area is obtained with Gaussian filtering. One of the shortcomings of linear filtering, however, is that it leads to blurring of the original image as shown in Figure 3-2(b).


(a)


(b)


(c)

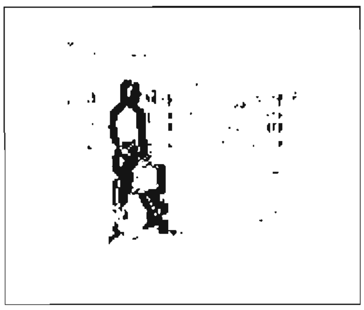
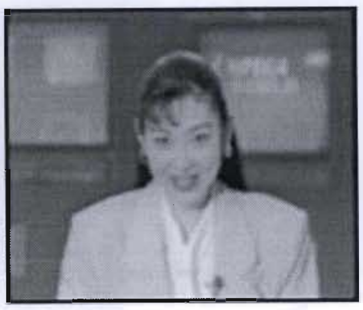
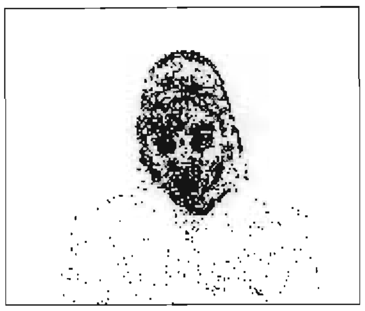
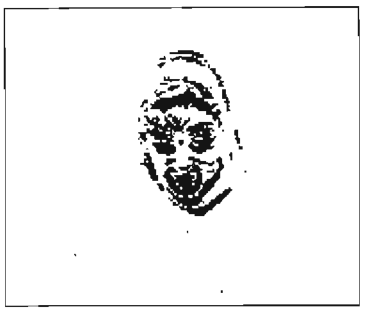

(d)

Figure 3-2: (a) Frame 40 of MPEG4 Test Sequence, *Hall Monitor*, (b) Frame 40 after Gaussian filtering (5x5 kernel), (c) Image after frame differencing and thresholding (Threshold = 5, Input Frames 40 and 41), (d) Image after frame differencing and thresholding (Threshold = 4, Input Frames 40 and 41). 5x5 Gaussian filter first applied to original frames

It should be noted that the test sequence used, namely, *Hall Monitor*, is characterised as a noisy video sequence. In the case where the video sequence is of a better quality with regards to noise, the effect of the filtering step is less significant. An example of a sequence with very little noise is the MPEG 4 test sequence, *Akiyo*, which is a scene of a news presenter reading a broadcast. Figure 3-3(a) shows the original frame and Figure 3-3(b) the filtered image. As may be observed from the binary difference frames in Figure 3-3(c), with a threshold of two there is almost no noise even without Gaussian filtering. Adding the filter prior to frame differencing provides a similar result. This is shown in Figure 3-3(d).



(a) (b)

(c) (d)

Figure 3-3: (a) Frame 252 of MPEG4 Test Sequence, *Akiyo*, (b) Frame 252 after Gaussian filtering (5x5 kernel), (c) Image after frame differencing and thresholding (Threshold = 2, Input Frames 252 and 251), (d) Image after frame differencing and thresholding (Threshold =2, Input Frames 252 and 251). 5x5 Gaussian filter first applied to original frames

Although in this work the Gaussian filter is applied before frame differencing, it may instead be applied after frame differencing but before thresholding to obtain similar results. This is illustrated in Figure 3-4. In this work Canny edge detection is performed on each original frame later in the processing. Since the first step of Canny edge detection is Gaussian filtering, the computing may be reduced by performing Gaussian filtering once for both the change detection and Canny edge detection steps. Hence the Gaussian filter was applied to the original frames.

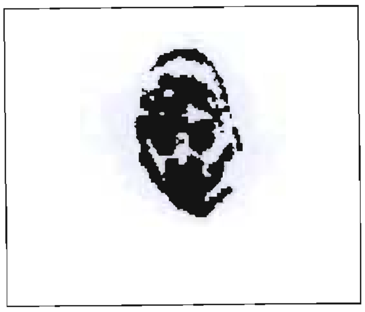


Figure 3-4: Image after frame differencing and thresholding (Threshold =3, Input Frames 252 and 251). 5x5 Gaussian filter applied after frame differencing but before thresholding.

3.1.2 Frame differencing

After noise filtering, the next step is to identify regions of change in the image. This is achieved by finding the absolute difference between consecutive frames. If one considers an image sequence consisting of n frames, denoted by, I_0, I_1,...I_{n-1}, to be the input to the algorithm. Then, for each image pair, I_k and I_{k-1}, the absolute point-wise image difference is computed as in equation 3.4.

$$\Delta_k(i,j) = \left|I_k(i,j) - I_{k-1}(i,j)\right| \tag{3.4}$$

where:

$1 \leq i < NR$ and $1 \leq j < NC$

Variables NC and NR refer to the number of rows and number of columns of the 2D image respectively. The output, Δ_k, is a greyscale image where the greylevel of each pixel gives an indication of the amount of change at that point, that is, the higher the greylevel, the greater the point-wise change.

Although it is easy to compute, frame differencing has two major shortcomings. Firstly, unless the moving object is strongly textured, only occlusion areas will be marked as changed. This drawback does not adversely affect the algorithm as in the model initialisation phase we are interested in obtaining the outline or silhouette of the moving object, not its entire surface. The second shortcoming is that if a part of the object becomes stationary for a short period, it will be lost for the period that it is stationary. Frame differencing is being used to obtain an initial model which will be subsequently tracked. Hence, if it is initially moving and then stops, the tracking algorithm will compensate for this. The problem arises however in the event that the object is initially stationary as there is then no initial criteria for segmentation. The algorithm is only effective once an initial motion is detected.

3.1.3 Automatic thresholding

In the previous section, a difference map, Δ_k, representative of image changes was obtained. The map reflects changes both due to noise and motion. After frame differencing, a global threshold is applied to the output to minimise the changes due to noise and emphasise the changes due to movement.

The first step is to choose a threshold value, which in Meier and Ngan, is input manually by the user. This method of threshold selection is disadvantageous, as the threshold needs to be tuned by the user. The system may be automated in this respect by incorporating a method of adaptively determining the threshold for a pair of frames in the sequence. For the approach to be feasible, however, it should be at a minimal computational cost. As will be discussed in subsequent chapters, change detection is also performed on each pair of frames in the object update step. If the threshold is manually selected, it is fixed for the entire video sequence; hence, it is not always the most optimal threshold for each pair of frames that are processed. By automating the process better results may be obtained in the change detection step, as the threshold would be adaptively selected to optimise the processing of each pair of frames. A brief overview of adaptive thresholding techniques and the approach implemented in this work is presented in section 3.1.3.1. Having selected a threshold τ it is applied to the difference map as illustrated in equation 3.5.

$$B_k(i,j) = \begin{cases} MovingObjectPixel & \text{if } \Delta_k(i,j) > \tau \\ BackgroundPixel & otherwise \end{cases} \tag{3.5}$$

The output, B_k is a binary map indicating stationary or background regions and change or moving object regions.

3.1.3.1 Automating the threshold

The aim of automatic thresholding is to implement an algorithm that adaptively selects the most optimal threshold. By optimal, one desires a threshold value that preserves change regions and removes as much unwanted noise as possible.

Thresholding techniques in the literature apply a statistical approach. Aach, Kaup and Mester [Aach,1993] formulate change detection as a Bayesian estimation problem. Prior knowledge is incorporated by formulating the *a priori* probabilities using 2-D Gibbs random fields. The threshold is then determined by a boundary relaxation technique. The method has the advantage in that it is non-iterative as it requires only a single raster scan. However, due to the Bayesian approach, it is computationally intensive. Neri *et al* [Neri,1998] apply higher order statistics to the adaptive thresholding problem. Each frame is modelled as the sum of four signals, the static background, the covered or uncovered background, the moving object and random noise. They then formulate the problem of detecting inter-frame variations produced by the moving objects as an instance of the detection of a partially modelled stochastic non-Gaussian signal in Gaussian noise. The moving object or non-Gaussian signal is then detected by computing the fourth-order moment pixel by pixel and comparing it to a threshold that is approximated independent of the signal characteristics. This method, however, gives only a coarse estimation of the moving object and has to be refined by motion analysis.

The algorithm applied is based on the work by Rosin [Rosin, 1997] who analysed the four primary classes of global thresholding strategies. These are modelling the noise intensity, modelling the signal intensity, modelling the spatial distribution of the signal and modelling the spatial distribution of the noise. Experimental results indicated that the most promising results are obtained from spatial methods. In this work we selected the approach that models the spatial distribution of the noise, as it was less computationally intensive. It models the noise as a random signal and uses statistical techniques to select a threshold that minimises the randomness of the data.

More specifically, the noise is assumed to be white and observations of noise pixels follow a Poisson probability distribution [Owen, 1990]. This implies that the intensity and spatial distribution of noise pixels are random. The spatial distribution of the noise is analysed to estimate a measure of its randomness.

A Poisson distribution is used to model the number of events that occur in an interval of time or region of space. In this particular application, the random variable is the number of pixels in the difference map that are above the threshold. A Poisson distribution is characterised by the following equation:

$$P[X=x]=\frac{\mu^{x}e^{\mu}}{x!};x=0,1,2,... \qquad (3.6)$$

where:

- X is the number of observations
- $P[X=x]$ is the probability that the number of observations is equal to some value, x
- μ is the average or expected number of observations in the test region

With a Poisson distribution, the mean and variance are equal, that is:

$$\sigma^2=\mu \qquad (3.7)$$

This implies that the ratio of the sample mean to the sample variance is a natural test for a Poisson distribution, and is called the relative variance, V_r.

The input to the algorithm for adaptively selecting a threshold is the difference map, D_k. Using the difference map, the histogram of grey levels, $h(g)$, is first computed using the expression in equation 3.8.

$$h(g)=h(g)+1 \Leftrightarrow (\Delta_k(i,j)=g) \qquad (3.8)$$

where $\Delta_k(i,j) \in D_k$ and $g \in G$.

G is the range of greylevels. For an 8-bit image, this is 256. The histogram is then used to compute a range of possible threshold values, T. T is the subset of greylevel values for which the corresponding histogram values are greater than zero, that is:

$$T=\{g \in G \mid h(g)<0\} \qquad (3.9)$$

For each possible threshold value, we compute the relative variance. This is done by sliding a WxW window across the difference map and counting the number of observations per window, x_n, as illustrated in equation 3.10.

$$x_n{}^t = \sum_{\delta_i=0}^{w-1} \sum_{\delta_j=0}^{w-1} 1_B\left(\Delta_k(i-\delta_i, j-\delta_j) > t\right) \tag{3.10}$$

where $0 < n \leq N$, N is the number of samples, B = {true, false} and $t \in T$ is the given threshold. The set characteristic function is defined by:

$$1_B(x) = \begin{cases} 1 & \text{if } x = \text{ true} \\ 0 & \text{otherwise} \end{cases} \tag{3.11}$$

Using this data, the sample mean, sample variance and relative variance may be calculated as illustrated in equations 3.12, 3.13 and 3.14 respectively.

$$\bar{x}_t = \frac{1}{N} \sum_{n=0}^{N-1} x_n \tag{3.12}$$

$$s_t{}^2 = \frac{1}{N-1} \sum_{n=0}^{N-1} (x_n - \bar{x}_t)^2 \tag{3.13}$$

$$V_{r_t} = \frac{s_t{}^2}{\bar{x}_t} \tag{3.14}$$

As already mentioned, our aim is not to detect spatially random noise but to avoid it in the threshold image. We therefore select the threshold which maximises the relative variance, that is, the global threshold τ is such that:

$$V_{r_t} \leq V_{r_\tau}, \ \forall\, t \in T \tag{3.15}$$

3.1.3.2 Results

To obtain the first set of results of the automatic thresholding algorithm, we used frames 30 and 31 of Hall Monitor, performed Gaussian smoothing on each, computed the difference map of the two frames and used the result as an input.

Figure 3-5 (a) shows the original frame and (b) to (i) show the binary difference frames for a range of manually selected thresholds. As may be observed for thresholds of one and two, the result is very noisy. As the threshold is increased the amount of noise decreases but so too does the meaningful information. For a threshold of ten, portions of the object are removed. Subjectively, an optimal threshold would be six or seven as this retains only the man in the foreground. However, this result is optimal because we know specifically that we are looking for a man in the foreground. The algorithm however does not have this high-level information. It is required to detect regions of change due to motion. This includes the man in the foreground and the uncovered background to the left of him. Hence an optimal threshold would be four or five.

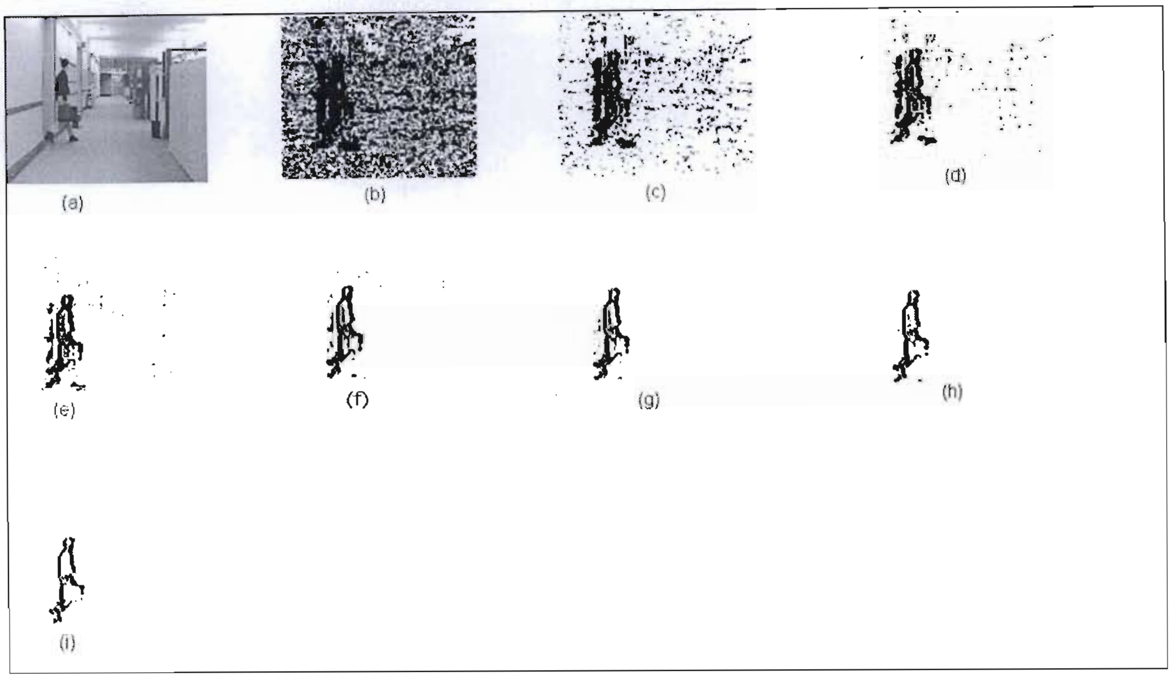


Figure 3-5: Original frame 30 of Hall Monitor and manually thresholded frames for a range of threshold values. (a) Original Frame, (b) Threshold = 1, (c) Threshold = 2, (d) Threshold = 3, (e) Threshold = 4, (f) Threshold = 5, (g) Threshold = 6, (h) Threshold = 7, (i) Threshold = 10.

When running the algorithm, it was first necessary to tune the window size parameter, W, and the number of pixels, s, by which the window was shifted. Since we are examining the spatial distribution of the pixels, if the window size were too small, then there would be too few pixels within the window to determine the spatial correlation of the data. To illustrate, the difference frame generated from *Hall Monitor*, frames 30 and 31, was used as an input and the relative variance for a range of threshold values was computed.

Figure 3-6 illustrates the relative variances for a range of threshold values. The window size was chosen as the extreme case of 1. As would be expected from equation 3.12 and 3.13, the signal resembles a random signal across the range of threshold values.

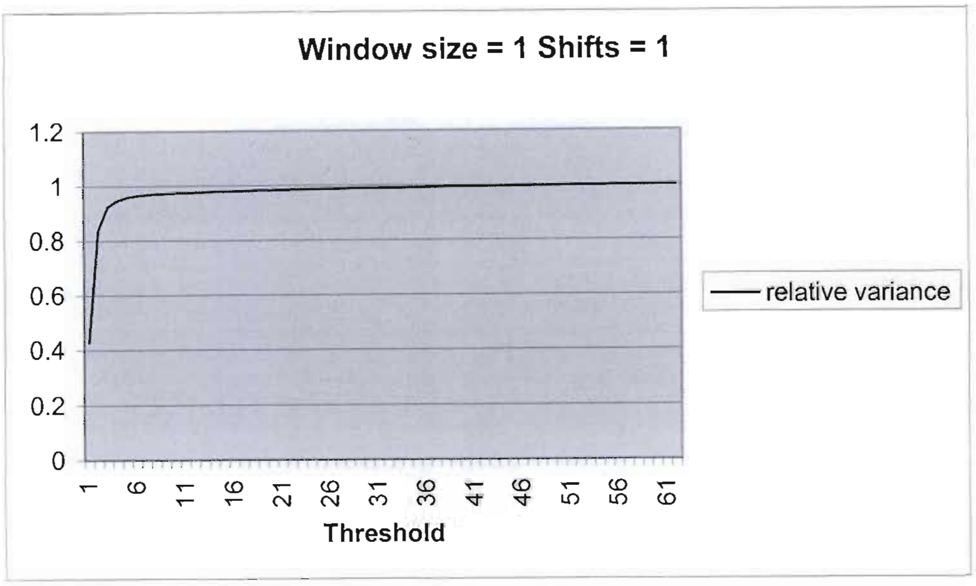


Figure 3-6: Graph of relative variance versus threshold. The difference frame generated from *Hall Monitor*, frames 30 and 31 was used as the input.

At the other extreme if the window size is too large relative to the frame resolution then there are too few samples. Hence the window size and number of shifts are chosen according to the image resolution.

The window size was increased from 1 and the number of pixels above threshold for each window was examined. Using the same input as in the previous test, the window size was set to 64 by 64 and the number of shifts to 32 and the pixels above threshold per window for a range of threshold values were analysed. From the graph illustrated in Figure 3-7, we observed that in windows which contain mainly noise, the number of pixels above the threshold becomes zero at approximately the same threshold value and have very similar distributions, while in windows which contain motion, the number of pixels above the threshold become zero at higher threshold values and the distributions differ as the motion is not necessarily uniform. As mentioned previously, the optimal threshold would be the minimum threshold that retains the motion information and removes the noise windows. From the graph, we may extrapolate that a threshold between 3 and 6 would suitably accomplish this task. This is accordance with the subjective selection of an optimal threshold of four or five.

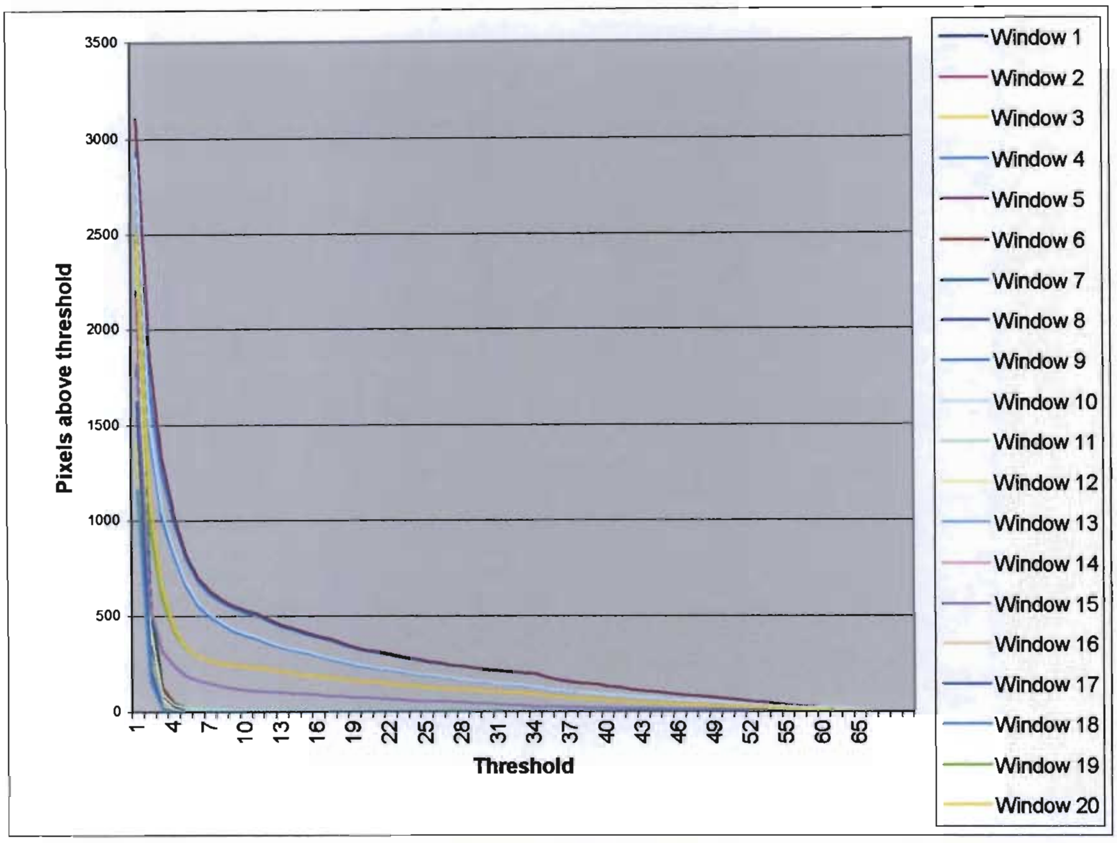


Figure 3-7: Graph illustrating pixels above threshold per window for a range of threshold values (window size = 64, shifts = 32). The difference frame generated from *Hall Monitor*, frame 30 and 31 was used as the input

Similar tests were performed on a range of window sizes and shifts as illustrated in Table 3-1. From the results it is observed that if the window size is not at an extreme for a given resolution, the automatically selected threshold is relatively independent of the input parameters. Computationally however, for a smaller window size there are more samples to process, hence a greater computational burden.

Table 3-1: Table of selected threshold for a range of window sizes and shifts. The difference frame generated from *Hall Monitor*, frames 30 and 31 was used as the input

Window size	Shifts	Process Time(ms)	Selected Threshold
1	1	121	66
4	4	191	6
8	4	420	4
8	8	100	4
16	8	240	4
16	16	70	4
32	16	201	3
32	32	50	3
64	32	160	3
64	64	40	3
128	64	120	3
128	128	40	1

For further tests, we chose window size of 16 and shifts of 16 as an optimal trade-off between number of samples and computational cost for QCIF resolution. Figure 3-8 illustrates the graph of relative variance versus threshold for the given window size and number of shifts. As may be extrapolated, the optimal threshold is obtained where the relative variance is a maximum. This is a value of 100 at a threshold of 4.

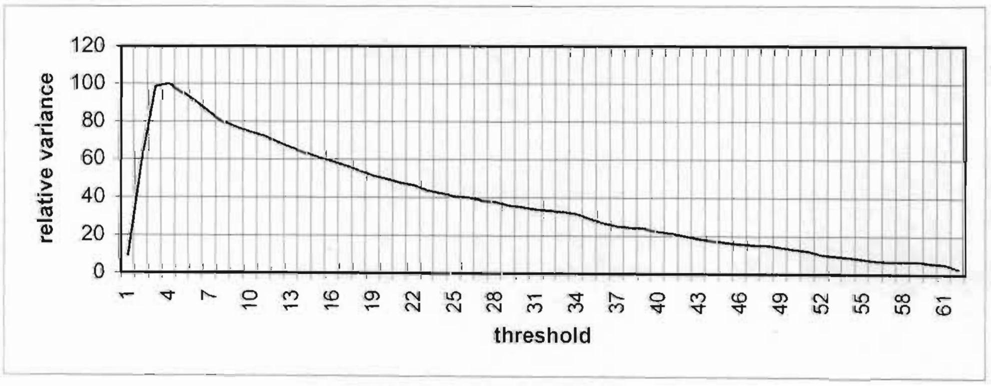


Figure 3-8: Graph of relative variance versus threshold. The difference frame generated from *Hall Monitor*, frames 30 and 31 was used as the input

Figure 3-9 illustrates the number of pixels above the threshold per window for a threshold of four. The result clearly illustrates that this is a correlated signal. Figure 3-10 shows a similar graph for threshold of 1. As expected from Figure 3-7, the signal for a low threshold more closely approximates random noise.

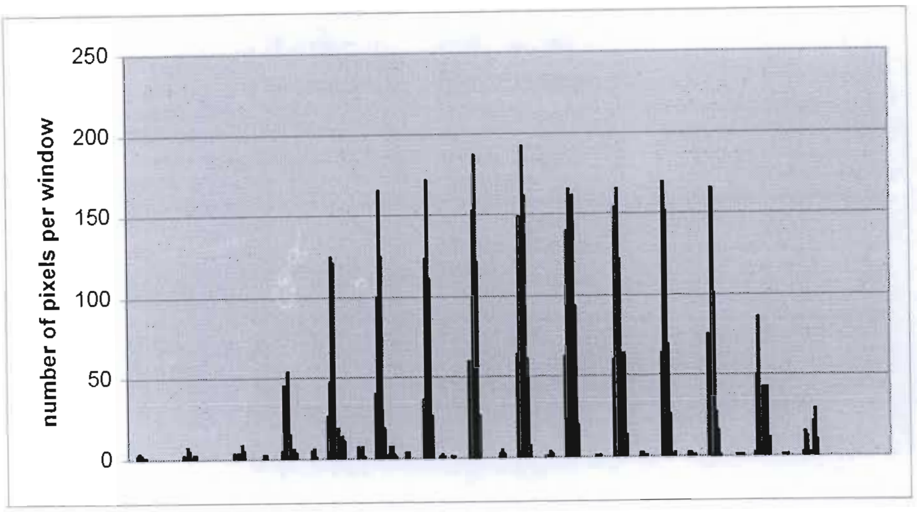


Figure 3-9: Column Graph illustrating number of pixels per window for threshold = 4. The difference frame generated from *Hall Monitor*, frame 30 and 31 was used as the input

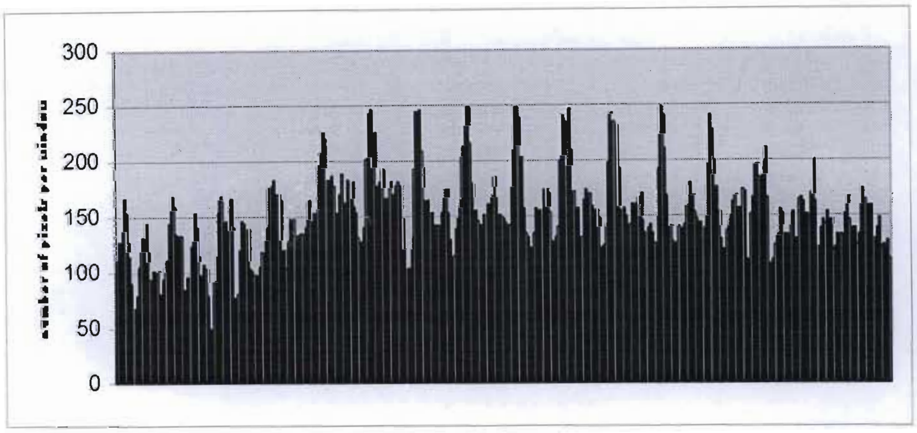


Figure 3-10: Column Graph illustrating number of pixels per window for threshold = 1. The difference frame generated from *Hall Monitor*, frames 30 and 31 was used as the input

The above set of results was generated from the *Hall Monitor* sequence, which is a noisy video sequence. It has fast motion and the object of interest occupies a small portion of the total frame.

The performance of the algorithm was assessed for a contrasting video sequence. The *Akiyo* video sequence has low noise and a large object which has slow motion. Frames 30 and 31 were used as a test input and Gaussian smoothing and frame differencing was performed before running the threshold algorithm. A window size of 16 and number of shifts of 16 was used since the frames were in QCIF format.

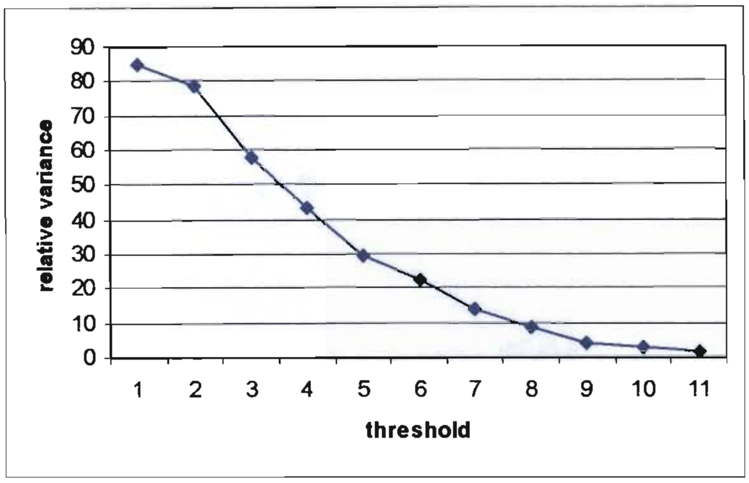


Figure 3-11: Graph of relative variance versus threshold for difference frame from Akiyo video sequence

Figure 3-11 illustrates the relative variances for a range of threshold values for the difference frame input. From the graph the maximum relative variance and hence optimal threshold is at 1. Figure 3-12 shows the number of pixels above the threshold per window for a range of threshold values. At a threshold of 1, the signal is already correlated and as the threshold is increased, the amplitude of the signal decreases indicating loss of information. From this information, we may therefore extrapolate that the optimal threshold ought to be 1. Figure 3-13 (a) shows the original frame 30 of the Akiyo sequence and (b) the binary difference frame. From the result we may observe that at a threshold of 1, the object of interest is correctly recognised, as there are virtually no noise pixels.

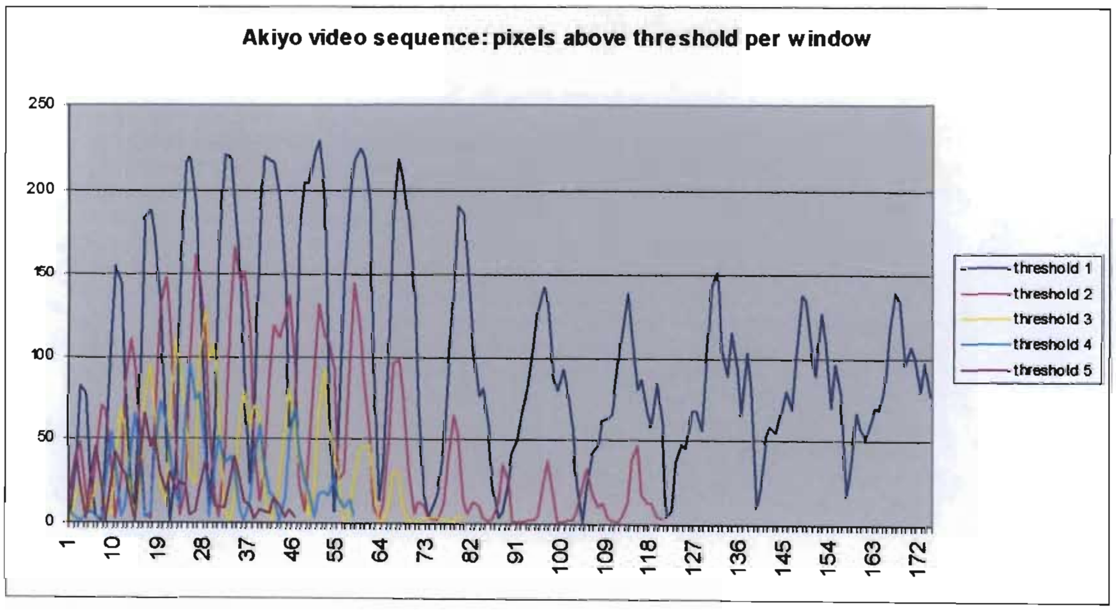
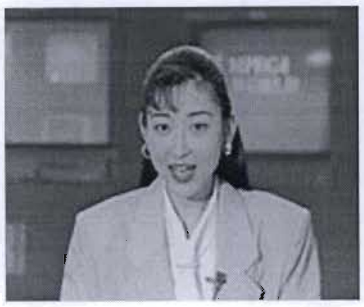
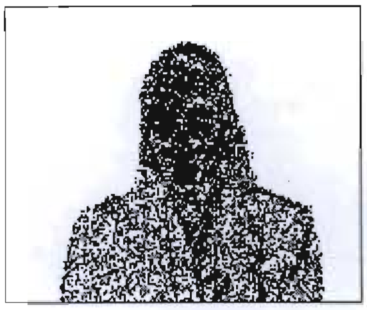


Figure 3-12: Graphs of pixels above threshold per window for a range of threshold values



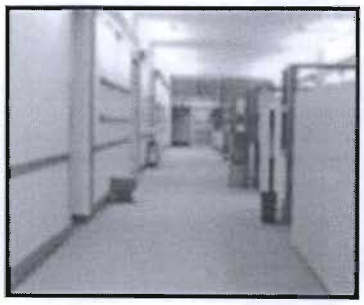
(a)



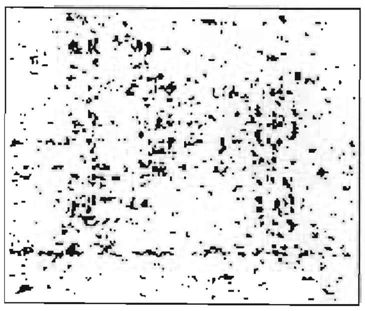
(b)

Figure 3-13: (a) Original frame 30 of Akiyo (b) Binary Difference frame, threshold = 1

Having presented the results on two contrasting video sequences, the effect of no movement between consecutive frames is now discussed. Figure 3-14(a) shows an original frame from the beginning of the *Hall Monitor* sequence in which there are no moving objects. In the next frame in the sequence there are also no moving objects, hence the results of frame differencing and automatic thresholding yields a binary difference frame that contains only noise. The threshold algorithm makes the assumption that there are changes due to both noise and motion in the difference frame, hence the threshold that maximises relative variance would yield the most correlated signal, which ideally removes most of the noise and retains the motion information. In this particular instance however, the assumption does not hold. Hence, the threshold that maximises the relative variance yields the most correlated signal relative to the other possible threshold values but converges on noise. In such cases, manual thresholding might be necessary.



(a)



(b)

Figure 3-14: (a) Original frame 1 of *Hall Monitor*, after Gaussian filtering. (b) Binary difference frame (Input frame 1 and 2) included to illustrate the effect of automatic thresholding when there is no movement between consecutive frames (Threshold=2)

Another "special" case that was examined was the case where the object of interest is moving with different motions, for example, the MPEG 4 video test sequence, *Silent*. The lady signing in the foreground moves her hands and head rapidly while her torso moves relatively slowly. The thresholding algorithm converges on the faster motions as the slower motions are comparable to the noise in the image. Figure 3-15(a) shows a sample of an original frame of the sequence and Figure 3-15(b) shows the binary difference frame. As may be observed the head and the hand and the shadows cast by each is detected as change areas but the rest of the torso is not detected. This is compensated for in the tracking stages of the algorithm which tracks slow motion and fast motion separately.

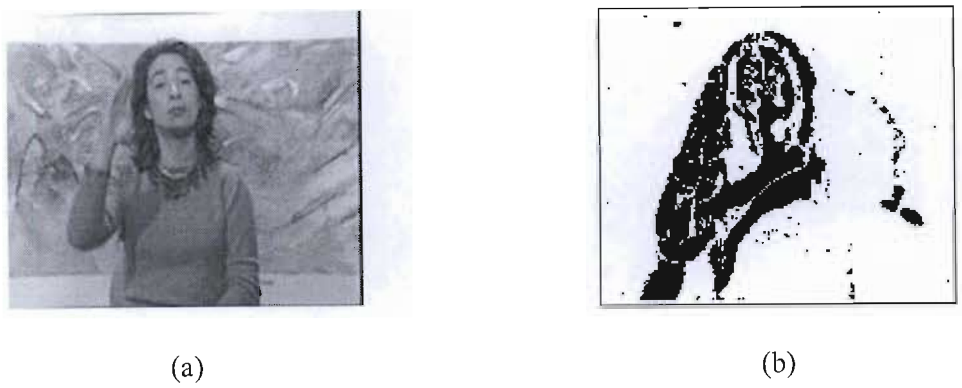


Figure 3-15: (a) Original frame 14 of MPEG 4 Test Sequence, *Silent*. (b) Binary difference frame (Input frame 14 and 15) included to illustrate the effect of automatic thresholding when there are both fast and slow motions between consecutive frames (Threshold=2)

In Meier and Ngan, the user selects the threshold manually. We have replaced this with a reliable adaptive thresholding algorithm at a low computational cost. The trade-off however, is that the window size must be selected by the user. This however is dependent on the frame resolution and is not content dependent. Hence, for a particular video sequence the optimal value for this parameter would be constant, unlike the optimal threshold which changes for each pair of difference frames.

For both high and low noise signals, slow and fast motion and different object sizes, the automated algorithm performs reliably. In scenes where there is a combination of slow and fast changes, the algorithm converges on the faster motion, but this loss of slow motion is compensated for later in the tracking stage. The main shortfall of the algorithm is in cases where there is no motion between consecutive frames as the threshold detection converges on noise. This might yield poorer results than manual thresholding.

3.1.4 Thinning

The aim of the change detection step is to identify regions of change between consecutive frames and to extrapolate from these change areas a model of moving components. In sections 3.1.1, 3.1.2 and 3.1.3 Gaussian filtering, frame differencing and thresholding were discussed. These steps yield a binary difference map, B_k, which represents change areas with some noise. The next step is to remove as much of the remaining noise as possible and to obtain a model of the moving components from the change areas by applying morphological operators to the binary map, B_k.

Firstly, the occlusion areas revealed in the binary map, B_k are normally more than one pixel wide and may be thinned or eroded to a single pixel width. This minimises the data without losing any pixel information. An important criterion for the thinning approach used, is that connectivity must be preserved. Two thinning approaches were applied and tested. The first was based on tracking the maxima of Euclidean distances [Shih, 1995] and the second was Pavlidis thinning algorithm [Pavli, 1980], which alternately performs erosion from northeast, east, southeast, south, southwest, west, northwest and north. The output from this process is a binary image, T_k, of the skeleton of the moving object and noise areas. In sections 3.1.4.1 to 3.1.4.3 these thinning algorithms are discussed in more detail.

3.1.4.1 Pavlidis thinning

Pavlidis's [Pavli, 1980] approach to object thinning is one of the most widely used and accepted approaches. The algorithm traces the contour of regions and repeatedly peels them off leaving only a set of skeletal pixels. A pixel is regarded as a skeletal pixel if it meets the definition of a multiple pixel. With reference to Figure 3-16, multiple pixels may be intuitively defined as pixels where two disjoint arcs of the boundary are mapped or pixels where the contour folds on itself or pairs of adjacent pixels where each pixel contains a different arc. To ensure connectivity, a pixel which has a neighbour that has been identified as a skeletal pixel during an earlier tracing of the contour, is also added to the set of skeletal pixels.

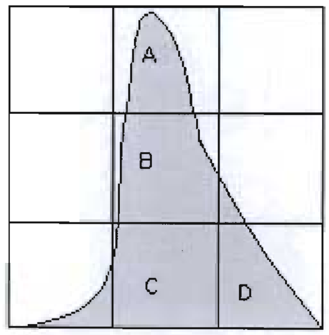


Figure 3-16: Intuitive illustration of multiple pixels. A corresponds to a contour folding, B corresponds to two disjoint arcs mapped on the same pixel and C and D are adjacent pixels where disjoint arcs of the boundary are mapped. [Pavli, 1980]

Implementation of Pavlidas Thinning

Pavlidas thinning may be implemented using mathematical morphology. More specifically, the Hit or Miss transform [Sera, 1982, Gonz, 1992] is applied in object thinning.

Mathematically the transform may be defined as in equation 3.16. For an image set, A, the transform is computed by eroding A by a suitable structuring element B_1 and eroding the complement of A, A_c, by a suitable structuring element B_2 and then finding the intersection of the two results.

$$HitOrMiss(A) = (A \odot B_1) \cap (A^c \odot B_2) \tag{3.16}$$

Erosion is a basic morphological operation and is defined in equation 3.17. The erosion of A by B is the set of all points z such that B, translated by z, is contained in A. $(B)_z$ denotes the translation of B by point $z = (z_1,z_2)$ and is expressed in equation 3.18.

$$A \odot B = \{z \mid (B)_z \subseteq A\} \tag{3.17}$$

$$(B)_z = \{c \mid c = b + z, \text{for } b \in B\} \tag{3.18}$$

The structuring elements in equation 3.16, namely, B_1 and B_2, are determined by the application. B_1 relates to the foreground pixels and B_2 to background. Hence the transform returns the points at which B_1 has successfully found a match in A and B_2 found a match in A_c.

The thinning operation is defined by equation 3.19.

$$thin(A) = A - HitOrMiss(A) \tag{3.19}$$

For thinning, pattern matching is performed hence there is no background operation, that is only B_1 is used, not B_2.

For symmetric thinning, a sequence of structuring elements is used. A is first thinned by one pass of B^1, then by one pass of B^2, until it is thinned by one pass of B^8. The process is repeated until convergence is reached. Each individual thinning pass is performed using equation 3.19.

There are essentially eight structuring elements, each being a one-pixel rotation of the previous, as represented below.

$$B^1 = \begin{bmatrix} 0 & 0 & 0 \\ X & 1 & X \\ 1 & 1 & 1 \end{bmatrix} \quad B^2 = \begin{bmatrix} X & 0 & 0 \\ 1 & 1 & 0 \\ 1 & 1 & X \end{bmatrix} \quad B^3 = \begin{bmatrix} 1 & X & 0 \\ 1 & 1 & 0 \\ 1 & X & 0 \end{bmatrix} \quad B^4 = \begin{bmatrix} 1 & 1 & X \\ 1 & 1 & 0 \\ X & 0 & 0 \end{bmatrix}$$

$$B^5 = \begin{bmatrix} 1 & 1 & 1 \\ X & 1 & X \\ 0 & 0 & 0 \end{bmatrix} \quad B^6 = \begin{bmatrix} X & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & X \end{bmatrix} \quad B^7 = \begin{bmatrix} 0 & X & 1 \\ 0 & 1 & 1 \\ 0 & X & 1 \end{bmatrix} \quad B^8 = \begin{bmatrix} 0 & 0 & X \\ 0 & 1 & 1 \\ X & 1 & 1 \end{bmatrix}$$

Figure 3-17: Structure elements for recursive thinning

Intuitively, the algorithm repeatedly peels off foreground pixels from each of the eight cardinal directions, until no more pixels can be removed without losing connectivity.

3.1.4.2 Thinning by tracking the maxima of the Euclidean distance

The thinning algorithm based on tracking the maxima of Euclidean distances is another morphological approach to thinning. In our work we have implemented and tested the algorithm proposed by Shih and Pu [Shih, 1995].

Obtain the Euclidean Distance transform of the image

The Euclidean distance transform replaces each object pixel with the Euclidean distance of the pixel to the nearest background pixel. The distance transform is obtained by adopting the mathematical morphology approach [Borg, 1986]. The morphological tool, grey-scale erosion is used.

Grey-scale erosion is calculated by:

$$(f \odot k) = \min_{(x+m,y+n)\varepsilon F,(m,n)\varepsilon K} \{f(x+m,y+n)-k(m,n)\} \tag{3.20}$$

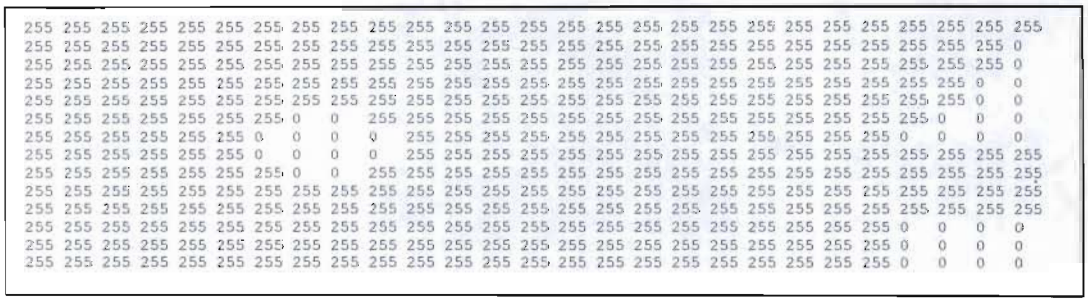
where f is the binary image consisting of object and background pixels and k is the structuring element.

To compute the Euclidean distance, k is the nxn distance matrix. The larger the value of n, the more accurate the distance calculation is. The trade-off however is that the computational burden is increased. A 5x5 matrix is illustrated below. The actual distance is scaled by a multiple of 10 to accommodate a higher accuracy, ie, a distance of 10 is equivalent to a Euclidean distance of one.

$$\begin{bmatrix} 28 & 22 & 20 & 22 & 28 \\ 22 & 14 & 10 & 14 & 22 \\ 20 & 10 & 0 & 10 & 20 \\ 22 & 14 & 10 & 14 & 22 \\ 28 & 22 & 20 & 22 & 22 \end{bmatrix}$$

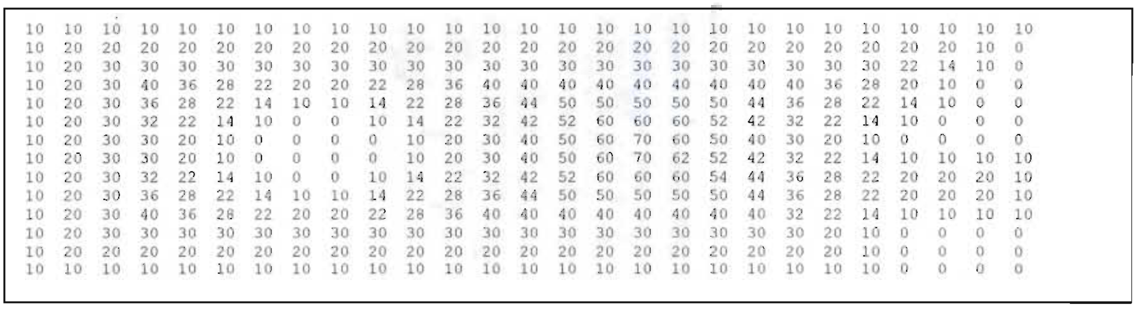
To obtain the distance transform, the grey scale erosion is repeated until all foreground pixels are replaced by distances.

Figure 3-18 illustrates the greylevel matrix of a binary test image with object pixels represented by 255 and background pixels by 0 and Figure 3-19 the corresponding Euclidean distance matrix. By comparing the test image to the distance transform of the image, it may be observed that object pixels closed to the background pixels along a horizontal or vertical have a Euclidean distance of ten, that is, they are one pixel distance away from the nearest background pixel. Pixels that are further away have progressively higher distances.



255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	0
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	0
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	0	0
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	0	0
255	255	255	255	255	255	255	0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	0	0	0
255	255	255	255	255	255	0	0	0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	0	0	0	0
255	255	255	255	255	255	0	0	0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	0	0	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	0	0	0	0
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	0	0	0	0
255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	255	0	0	0	0

Figure 3-18 Test image to illustrate thinning algorithm



```
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10
10 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 10 0
10 20 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 22 14 10 0
10 20 30 40 36 28 22 20 20 22 28 36 40 40 40 40 40 40 40 40 40 36 28 20 10 0  0
10 20 30 36 28 22 14 10 10 14 22 28 36 44 50 50 50 50 50 44 36 28 22 14 10 0  0
10 20 30 32 22 14 10 0  0  10 14 22 32 42 52 60 60 60 52 42 32 22 14 10 0  0  0
10 20 30 30 20 10 0  0  0  0  10 20 30 40 50 60 70 60 50 40 30 20 10 0  0  0  0
10 20 30 30 20 10 0  0  0  0  10 20 30 40 50 60 70 62 52 42 32 22 14 10 10 10 10
10 20 30 32 22 14 10 0  0  10 14 22 32 42 52 60 60 60 54 44 36 28 22 20 20 20 10
10 20 30 36 28 22 14 10 10 14 22 28 36 44 50 50 50 50 50 44 36 28 22 20 20 20 10
10 20 30 40 36 28 22 20 20 22 28 36 40 40 40 40 40 40 40 40 32 22 14 10 10 10 10
10 20 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 30 20 10 0  0  0  0
10 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 10 0  0  0  0
10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 0  0  0  0
```

Figure 3-19: Euclidean distance transform of test image

Detect the base and apex points

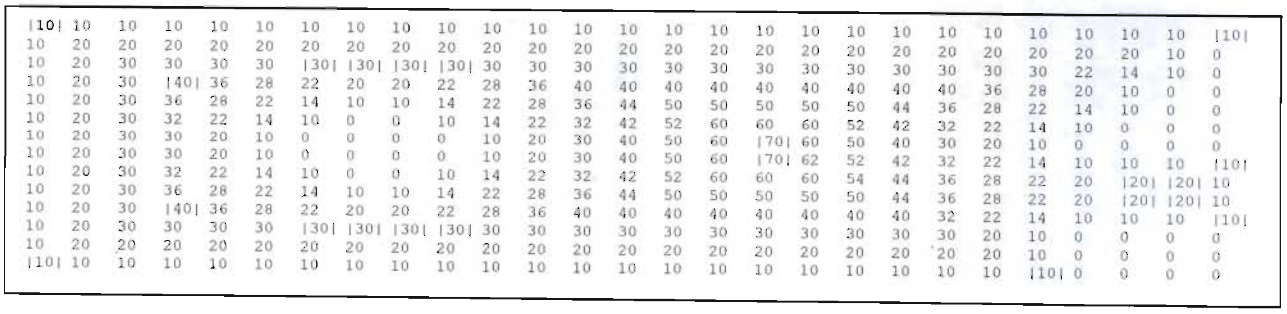
The skeleton is initialised by detecting base or corner points and apex points. Apex points are defined as local maxima in a 3x3 neighbourhood, that is, pixel $a(i,j)$ is an apex point if it meets the following criterion:

$$a(i,j) = \max_{x=[-1;1],y=[-1;1]} \{a(i+x, j+y)\} \qquad (3.21)$$

The base point is defined as a corner point which is surrounded by a majority of background pixels. The 45°, 90°, and 135° configurations respectively are represented below. A base point belongs to either one of these or variations of these up to eight 45° rotations.

$$\begin{matrix} 1 & 0 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}, \qquad \begin{matrix} 1 & 1 & 0 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{matrix} \qquad \text{and} \qquad \begin{matrix} 1 & 1 & 1 \\ 1 & 1 & 0 \\ 0 & 0 & 0 \end{matrix}$$

Shih and Pu, however, only consider sharp convexities as significant and therefore only consider the 45° and 90° base points. Figure 3-20 illustrates the base and apex points of the test image. These are the points enclosed in square brackets.



```
[10] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 [10]
10 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 10 0
10 20 30 30 30 30 [30] [30] [30] [30] 30 30 30 30 30 30 30 30 30 30 30 30 30 22 14 10 0
10 20 30 [40] 36 28 22 20 20 22 28 36 40 40 40 40 40 40 40 40 40 36 28 20 10 0 0
10 20 30 36 28 22 14 10 10 14 22 28 36 44 50 50 50 50 50 44 36 28 22 14 10 0 0
10 20 30 32 22 14 10 0 0 10 14 22 32 42 52 60 60 60 52 42 32 22 14 10 0 0 0
10 20 30 30 20 10 0 0 0 0 10 20 30 40 50 60 [70] 60 50 40 30 20 10 0 0 0 0
10 20 30 30 20 10 0 0 0 0 10 20 30 40 50 60 [70] 62 52 42 32 22 14 10 10 10 [10]
10 20 30 32 22 14 10 0 0 10 14 22 32 42 52 60 60 60 54 44 36 28 22 20 [20] [20] 10
10 20 30 36 28 22 14 10 10 14 22 28 36 44 50 50 50 50 50 44 36 28 22 20 [20] [20] 10
10 20 30 [40] 36 28 22 20 20 22 28 36 40 40 40 40 40 40 40 40 32 22 14 10 10 10 [10]
10 20 30 30 30 30 [30] [30] [30] [30] 30 30 30 30 30 30 30 30 30 30 30 20 10 0 0 0 0
10 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 20 10 0 0 0 0
[10] 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 10 [10] 0 0 0 0
```

Figure 3-20: Base and apex points reflected on Euclidean distance transform of test image

The base and apex points only form a small portion of a disconnected skeleton. They are considered as sources of the skeleton that is grown up from them by tracing the maxima of the Euclidean distance.

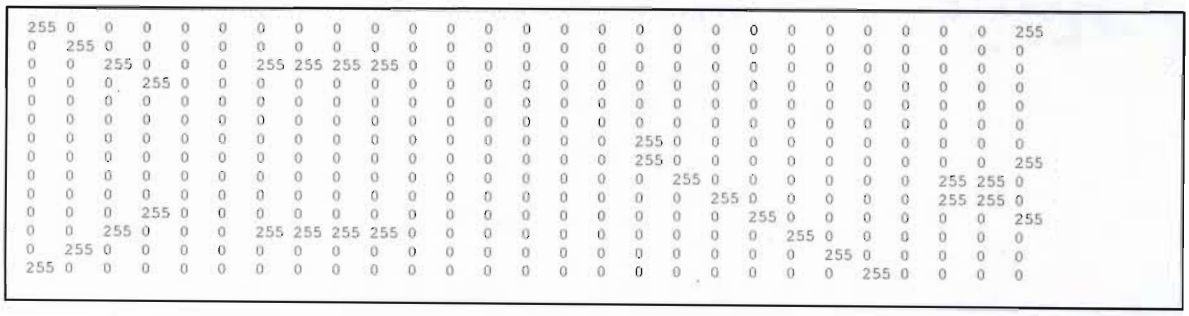
Perform directional uphill generation

As mentioned, the skeleton is grown from the apex and corner points by tracking the maxima. First directional uphill generation is performed. Starting from a corner or apex point, A, the maximum of the neighbourhood of A, N_0, is found. A and N_0 are then used to find the directional neighbours of N_0. A directional neighbour is a pixel in the eight neighbourhoods of N_0 and located within a ±45°-slope change of the current medial axis, as defined by A and N_0. Applying this rule to the configuration below - $\{N_3, N_2, N_4\}$ forms the directional neighbourhood of N_0, denoted by, D_N. N_3 is along the current medial axis of A and N_0 while N_4 is at a -45° slope change and N_2 is at a +45° slope change.

$$\begin{array}{ccc} N_4 & N_3 & N_2 \\ . & N_0 & . \\ . & A & . \end{array}$$

In directional uphill generation, the maximum of the N_0 and its directional neighbours is added as the next skeletal point, that is:

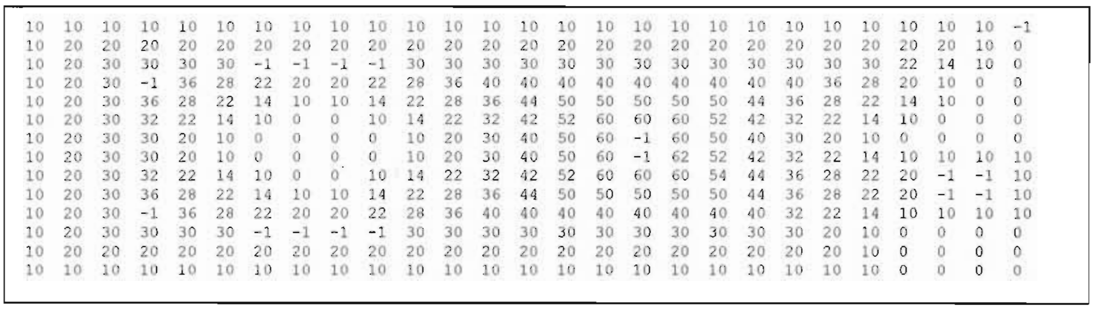
$$N^U_{next} = \max_{N_i \in D_N \cup \{N\}} \{N_i\} \qquad (3.22)$$



255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255
0	255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	255	0	0	0	255	255	255	255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0	0	0	255
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0	0	0	255	255	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0	0	255	255	0
0	0	0	255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0	0	0	255
0	0	255	0	0	0	255	255	255	255	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0	0	0
0	255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0	0
255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0

Figure 3-21: Skeletal pixels of test image after directional uphill generation

Figure 3-21 shows the skeletal point after uphill generation. As is observed, the skeleton is not yet connected. Downhill generation is performed after uphill generation to obtain a connected skeleton. Directional downhill generation is initialised from base and apex points which cannot be further tracked. In Figure 3-22 pixels marked with distance –1 are initial pixels for directional downhill generation.



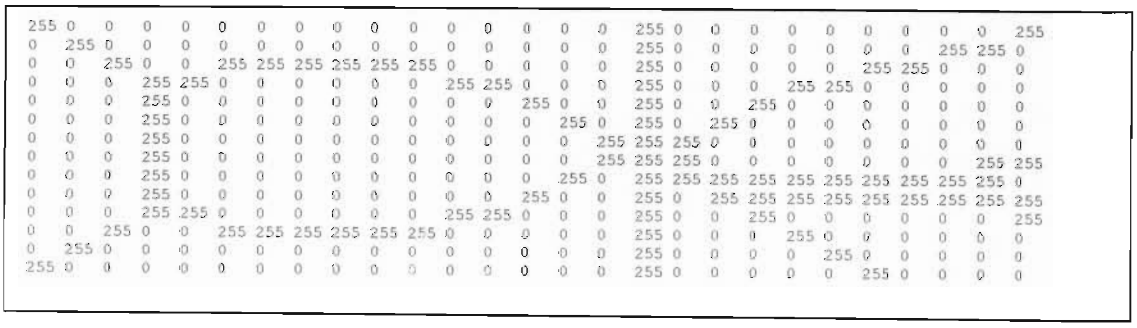
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	-1
10	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	10	0
10	20	30	30	30	30	-1	-1	-1	-1	30	30	30	30	30	30	30	30	30	30	30	30	30	22	14	10	0
10	20	30	-1	36	28	22	20	20	22	28	36	40	40	40	40	40	40	40	40	40	36	28	20	10	0	0
10	20	30	36	28	22	14	10	10	14	22	28	36	44	50	50	50	50	50	44	36	28	22	14	10	0	0
10	20	30	32	22	14	10	0	0	10	14	22	32	42	52	60	60	60	52	42	32	22	14	10	0	0	0
10	20	30	30	20	10	0	0	0	0	10	20	30	40	50	60	-1	60	50	40	30	20	10	0	0	0	0
10	20	30	30	20	10	0	0	0	0	10	20	30	40	50	60	-1	62	52	42	32	22	14	10	10	10	10
10	20	30	32	22	14	10	0	0	10	14	22	32	42	52	60	60	60	54	44	36	28	22	20	-1	-1	10
10	20	30	36	28	22	14	10	10	14	22	28	36	44	50	50	50	50	50	44	36	28	22	20	-1	-1	10
10	20	30	-1	36	28	22	20	20	22	28	36	40	40	40	40	40	40	40	40	32	22	14	10	10	10	10
10	20	30	30	30	30	-1	-1	-1	-1	30	30	30	30	30	30	30	30	30	30	30	20	10	0	0	0	0
10	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	20	10	0	0	0	0
10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	10	0	0	0	0

Figure 3-22: Marking points for directional downhill generation

Perform directional downhill generation

Downhill generation is used to obtain a connected skeleton from the skeletal pixels generated by uphill generation. From each marked pixel, the maximum of the directional neighbours, as illustrated by equation 3.23, is recursively found until a skeleton point or image boundary is reached.

$$N^D{}_{next} = \max_{N_i \in D_N} \{N_i\} \tag{3.23}$$



255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0	0	0	255
0	255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0	0	0	0	255	255	0
0	0	255	0	0	255	255	255	255	255	255	0	0	0	0	0	255	0	0	0	0	0	255	255	0	0	0
0	0	0	255	255	0	0	0	0	0	0	255	255	0	0	0	255	0	0	0	255	255	0	0	0	0	0
0	0	0	255	0	0	0	0	0	0	0	0	0	255	0	0	255	0	0	255	0	0	0	0	0	0	0
0	0	0	255	0	0	0	0	0	0	0	0	0	0	255	0	255	0	255	0	0	0	0	0	0	0	0
0	0	0	255	0	0	0	0	0	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	0	0	0
0	0	0	255	0	0	0	0	0	0	0	0	0	0	0	255	255	255	0	0	0	0	0	0	0	255	255
0	0	0	255	0	0	0	0	0	0	0	0	0	0	255	0	255	255	255	255	255	255	255	255	255	255	0
0	0	0	255	0	0	0	0	0	0	0	0	0	255	0	0	255	0	255	255	255	255	255	255	255	255	255
0	0	0	255	255	0	0	0	0	0	0	255	255	0	0	0	255	0	0	255	0	0	0	0	0	0	255
0	0	255	0	0	255	255	255	255	255	255	0	0	0	0	0	255	0	0	0	255	0	0	0	0	0	0
0	255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0	255	0	0	0	0	0
255	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	255	0	0	0	0	0	255	0	0	0	0

Figure 3-23: Final skeleton after directional downhill generation

From the result in Figure 3-23 it is observed that the skeleton is reduced to single pixel width completely except for the line projecting inwards from the right border. This result is acceptable though because the pixels in consideration have the same distance value.

3.1.4.3 Results

Both thinning algorithms were tested on a series of natural and synthetic images. It was observed from the results that when the objects are solid, that is, little or no background points within the foreground, both algorithms perform well. Connectivity is preserved in both and the thinning is to single pixel width. However, in the case where the background is interspersed between the foreground pixels, Pavlidis algorithm performs better at maintaining connectivity.

Figure 3-24(a) presents a synthetic image in which the white portions are background and the black portions are foreground. The results of thinning by maxima tracking and Pavlidas thinning are presented in Figure 3-24 (b) and (c) respectively. The breaks in connectivity are apparent in the Euclidean distance tracking approach. The Pavlidas approach has many more side nodes developing from the main skeleton however the important criterion of connectivity is preserved. As a result, the Pavlidis thinning was the preferred approach in this work.

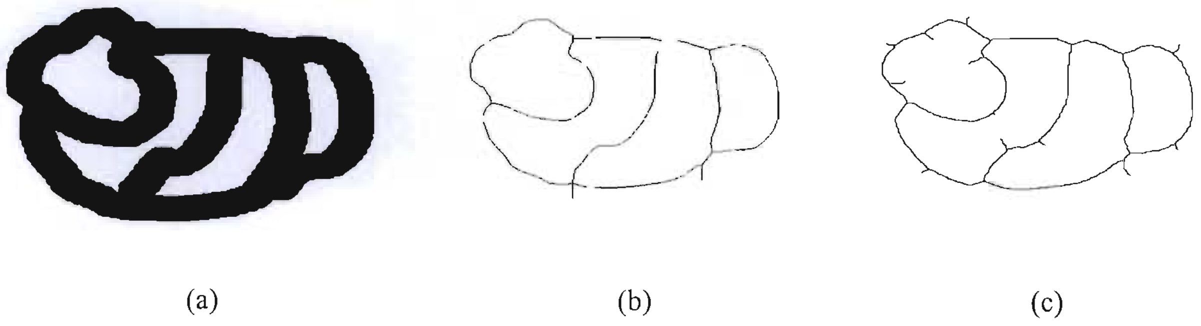


Figure 3-24: (a) Synthetic Test Image (b) Thinned image using maxima tracking of Euclidean distances (c) Thinned image using Pavlidas thinning algorithm

3.1.5 Connected component labelling

After thinning, the pixels belonging to the moving object tend to be connected, while pixels due to noise form smaller clusters. A connected component labelling algorithm [Rose, 1996] is implemented to find connected components. The aim is to then use the connected component information to separate the moving components from the noise.

Given the thinned input image T_k, the greylevel of the image is given by:

$$T_k(i,j) = \begin{cases} 255 & \text{for object pix el} \\ 0 & \text{for background pixel} \end{cases} \qquad (3.24)$$

We can define two pixels, $p,q \in T_k$ as equivalent, $p \sim q$, if there exists a path from p to q and both are object pixels.

A path is the sequence, $p_1, p_2, .. p_n$ such that the chessboard distance [Gonz, 1992], $d(p_i,p_{i+1}) = 1$ where $i = 1..n$.

To find the connected components of image T_k, is to define the equivalence classes.

If C is an equivalent class then $\forall p, q \in C, p \sim q$

Each class, C_z, is labelled by a unique greylevel, hence for n classes, the output image, CCL_k, would be a greylevel image of $n+1$ greylevels. The one additional greylevel is for the background.

$$CCL_k = \left(\bigcup_{z=1}^{n} C_z \right) \bigcup B \tag{3.25}$$

where n is the number of equivalent classes and B is the set of background pixels.

Furthermore, $C_y \cap C_z = \varnothing$ such that $y,z = 1..n$.

The connected component-labelling algorithm was implemented by a fast, simple and efficient two-pass connected component-labelling operator [Gonz, 1992]. The operator scans the image by moving along a row until it comes to an object pixel p, where p denotes the pixel to be labelled at any stage in the scanning process. The four neighbours of p, which have already been encountered in the scan, are then examined. As indicated by N_1 to N_4 in the matrix below, these are the neighbours to the left, above and along each of the upper diagonals.

$$\begin{pmatrix} N_2 & N_3 & N_4 \\ N_1 & p & \cdot \\ \cdot & \cdot & \cdot \end{pmatrix}$$

Based on the neighbouring pixels, p is labelled as follows:

If all four neighbours are 0, a new label is assigned to p, else

if only one neighbour has an object pixel, its label is assigned to p, else

if more than one of the neighbours are object pixels, one of the labels are assigned to p and a note is made of the equivalences.

After completing the scan, the equivalent label pairs are sorted into equivalence classes and a unique label is assigned to each class. As a final step, a second scan is made through the image, during which each label is replaced by the label assigned to its equivalence classes.

Figure 3-25(a) was used as a test image to illustrate connected component labelling and Figure 3-25(b) displays the resultant image. Each connected component is assigned a unique value to illustrate the individual components.

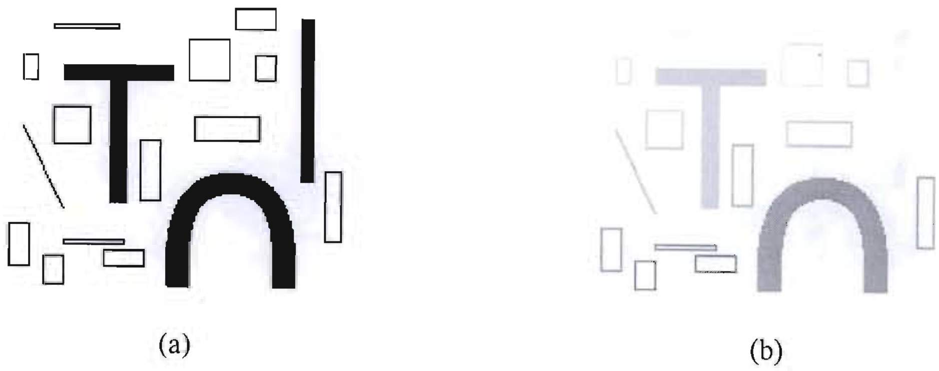


Figure 3-25: (a) Test image for connected component labelling. (b) Result of connected component labelling

The algorithm works very reliably for synthetic images. For binary images obtained from natural images, as is the case in this research work, the algorithm is as reliable. However due to noise, the number of connected components may increase considerably. Since equivalent classes are only determined on the second pass, in very noisy images with a large number of components, overflow errors may result. This problem may be circumvented by ensuring that the input parameters for the previous stages, especially filtering and thresholding, are properly tuned to minimise the noise.

3.1.6 Selecting the moving components

Our aim at this point is to use the connected component information to select the moving components. The set of moving components, MCC_k, is a subset of the connected components, that is:

$$MCC_k = \bigcup_{y=1}^{m} MC_y \subseteq \bigcup_{z=1}^{n} C_z \tag{3.26}$$

Given the constraint that there is only one moving object, it is logical to assume that the largest connected component, C_{max} belongs to the object. Given that the size is determined by the number of pixels making up the component, C_{max} may be determined as follows:

if size($C_{\max}$) $\geq$ size(C_z) $\forall\ 1 \leq z \leq n$
Then $C_{\max} \in MCC_k$

To select smaller components, which are likely to be part of the moving object, we looked at the spatial distribution of the components instead of their size. This was done to ensure that even small components that might be below a size threshold, but are likely to be part of the moving object would also be selected. Smaller components were included as moving components if they were within a distance α of existing moving components - α being an input parameter to the system.

Given pixel, $p \in MC_y$, and pixel, $q \in C_z$,

$$C_z \subset MCC_k \Leftrightarrow d(p,q) \leq \alpha \tag{3.27}$$

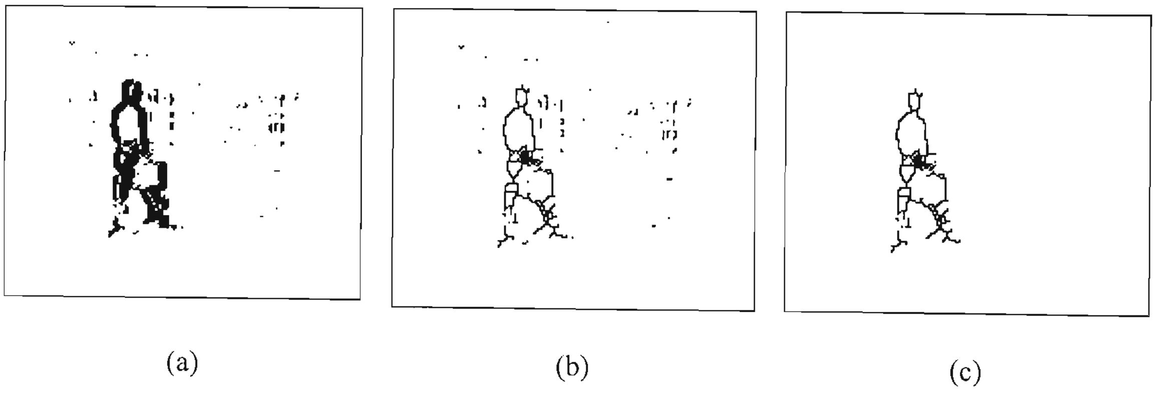

(a) (b) (c)

Figure 3-26: (a) Difference of frames 40 and 41 of Hall Monitor after thresholding. (b) Thinned difference frame. (c) Binary frame after connected component labelling and MCC selection

Figure 3-26(a) shows the binary frame result after frame differencing and thresholding a pair of consecutive frames from the Hall Monitor sequence. This image was then used as an input to the thinning stage and the result is presented in Figure 3-26(b). As may be noticed, the image consists of a large connected component that corresponds to the object of interest and several small components that are mainly due to noise. Figure 3-26(c) shows the result after selecting only the moving components according to the method presented. As may be observed, the small noise components have been completely removed and only the components within spatial range of the largest component are retained. Figure 3-27 is a zoom on the left leg of the man in the picture to focus in on the area where smaller components are within the spatial threshold. From the image it is evident a more complete extraction of the man's leg was possible by including smaller components close to the main large component.

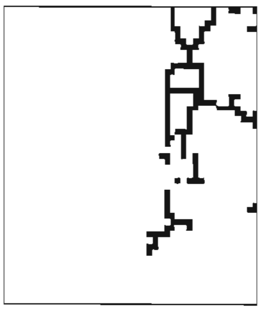


Figure 3-27: Zoom in on left leg of man

In Meier and Ngan, the moving components were based on the size of the components, that is, they select the largest components as moving components. While this seems a plausible approach, they do not present the criteria for selection and the problem of defining what is a large component is not readily solvable. By examining instead the spatial distribution of the components to choose the moving components, we increase the likelihood that the selected components would be part of the moving object as connected components that belong to the same object would tend to be close together. However, if the constraint of one moving object were relieved, then it would be necessary to impose a size threshold to identify the main component of each object and then look at the spatial distribution around each of these components to identify the remainder of each moving component.

3.2 Applying spatial information

In the previous section, the time changing or temporal information was used to obtain a model of the moving components. In this section spatial information is incorporated to refine the model, specifically to improve the contour extraction of the moving object.

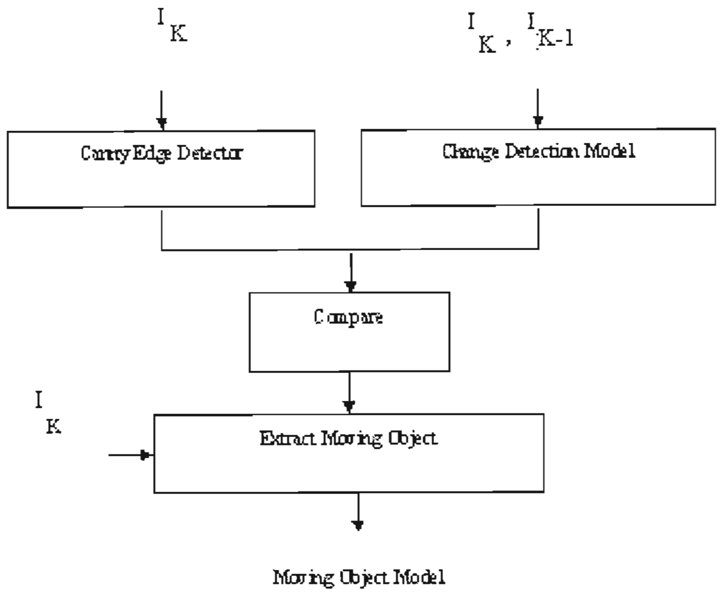


Figure 3-28: Block Diagram Illustrating Object Segmentation Method

With reference to Figure 3-28, a Canny edge detector [Canny, 1986] is applied to the original frame to generate an edge map, which is then compared to the model of moving connected components. All edge map pixels within a distance η of the moving connected components are selected as the moving object.

Given, E_k, as the edge map and MCC_k as the moving component model, the object model, O_k, may be generated as illustrated:

If $d(p,q) \leq \eta$
Then $p \in O_k$

where pixel, $p \in E_k$ and pixel, $q \in MCC_k$ and η is inputted as a parameter to the system.

3.2.1 Canny edge detection

Edge detection is a popular feature detection tool in object recognition. Sharifi, Fathy and Mahmoudi [Shari, 2002] present a survey of the most commonly used edge detection methods. Their survey incorporates five categories of edge detection operators, namely:

- Gradient edge Detectors (first derivative or classical)
- Zero crossing (second derivative)
- Laplacian of Gaussian (LoG)
- Gaussian edge detectors
- Colored edge detectors

A total of seven operators, namely, Canny, Sobel, Kirsch, Shen-Castan, Marr-Hildreth and two variations of the Laplace operator were chosen for evaluation. In their work, they examine how effectively each of the operators performs edge detection by a qualitative analysis of the resultant edge maps and quantitatively by using average risk [Spree, 1992]. The edge detection algorithm is viewed as a complex decision-making process and the average risk of these decisions is used as a performance measure of the detector. The average risk is computed by estimating the probability of a number of different types of errors, for example, edge thickness, missed edge pixels and then computing the weighted sum of these probabilities.

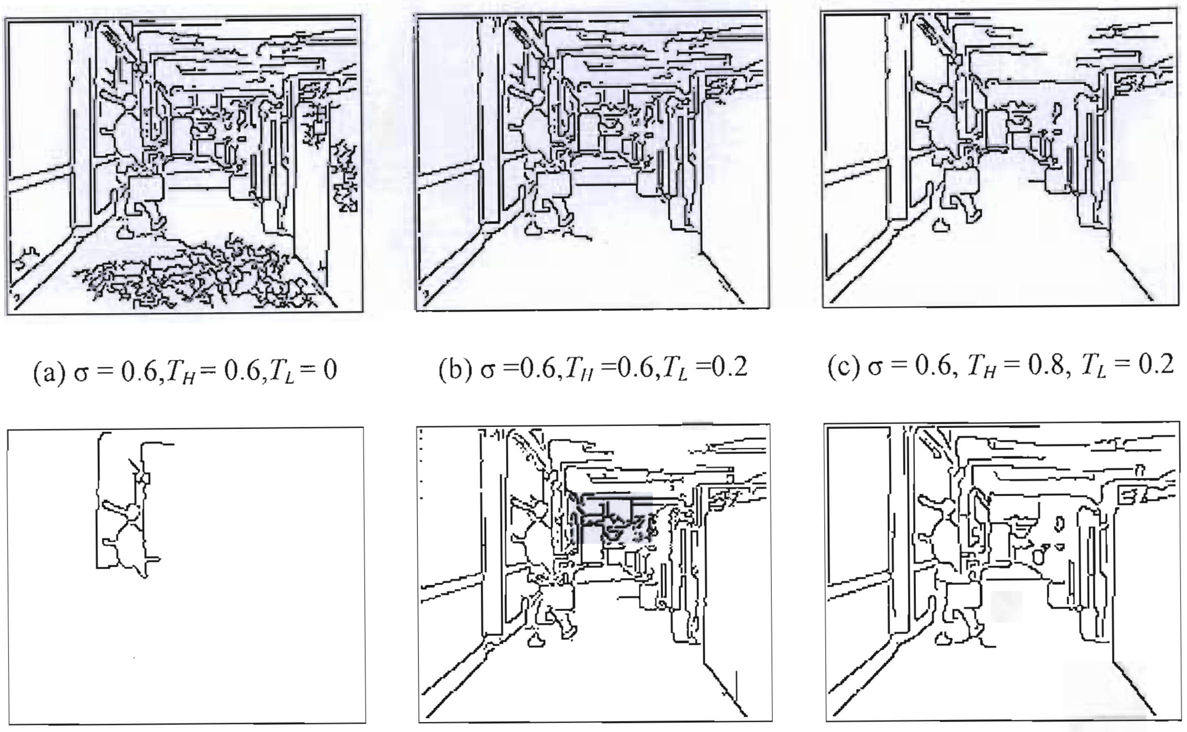
Both the quantitative and qualitative results indicated that the best performing edge detectors were the Canny and Shen-Castan detectors - both of which are Gaussian-based operators. The key advantage of such operators is that they perform well in noisy conditions; however, the trade-off is their computational complexity. Since noise considerations are important in our application, we chose a Gaussian based edge detector, namely the Canny operator.

The Canny operator was designed to be an optimal edge detector based on particular criteria. It takes as input a greyscale image, and produces as output a binary image showing the positions of tracked intensity discontinuities.

The Canny operator works in a multi-stage process. The image is first smoothed by Gaussian filtering as described in 3.1.1. Then a 2-D first derivative operator is applied to the smoothed image to highlight regions of the image with high spatial first derivatives. Edges give rise to ridges in the gradient magnitude image. The algorithm then tracks along the top of these ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output. This is known as *non-maximal suppression*. Finally, the ridge points are tracked to yield an edge map. Hysteresis is incorporated into the tracking by two thresholds: T_H and T_L, with $T_H > T_L$. Tracking can only begin at a point on a ridge higher than T_H. Tracking then continues in both directions along the ridge peak out from that point until the height of the ridge falls below T_L. This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments.

The Canny detector has three input parameters which need to be tuned, namely the variance, σ, in Gaussian smoothing and the high and low thresholds, T_H and T_L. As verified quantitatively in Heath *et al* [Heath,1997] and qualitatively in Figure 3-29, the choice of input parameters does have an impact and optimally choosing the parameters provides a significant difference in the quality of the edge image.

With reference to Figure 3-29(a) and (b), the lower threshold, T_L, affects the amount of detail around the strong edges that are incorporated. By comparing Figure 3-29 (b), (c) and (d), the high threshold, T_H, affects the strong edges. If this value is too high only the highest maxima are included. Finally, the variance, σ, affects the amount of smoothing. This is discussed in greater detail in 3.1.1, however it should be noted that too low a variance will not remove noise effectively, while too high a variance would blur the edges which we are trying to preserve. In the application, these parameters are tuned prior to executing the algorithm.



(a) $\sigma = 0.6, T_H = 0.6, T_L = 0$ (b) $\sigma = 0.6, T_H = 0.6, T_L = 0.2$ (c) $\sigma = 0.6, T_H = 0.8, T_L = 0.2$

(d) $\sigma = 0.6, T_H = 1, T_L = 0.2$ (e) $\sigma = 0.4, T_H = 0.7, T_L = 0.2$ (f) $\sigma = 1, T_H = 0.7, T_L = 0.2$

Figure 3-29: Edge map of Figure 3-2(a) for different input parameter sets

3.3 Results

Figure 3-30 shows the resultant image after each step of the model initialisation phase when the first two frames of the *Hall Monitor* sequence are used as an input. Since these frames do not contain any motion, the thresholding algorithm converges on noise as discussed in section 3.1 and again represented in Figure 3-30(a). Since the noise pixels are randomly distributed across the image, after thinning the resultant image contains small, unconnected components. Hence the connected component labelling and moving components selection steps yield a single connected component which is essentially a small cluster of noise pixels. Since the noise signal does not correspond to the actual objects in the image, when it is compared to the edge map of the original frame, Figure 3-30(d), the resultant model is a blank frame, Figure 3-30(e). The algorithm however does not always give as good results. If the noise pixels form larger clusters or if they converge on the image information, then the comparison of the moving component model to the edge map would return portions of the static background as the moving object model.

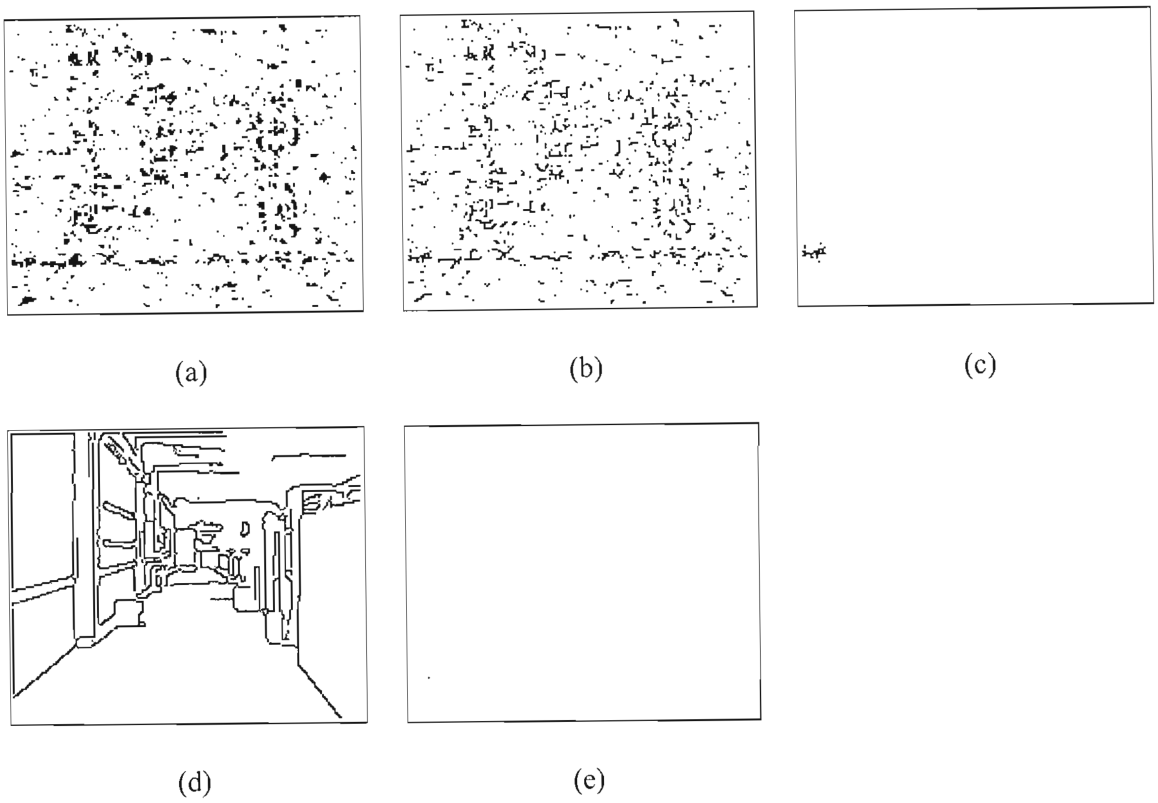


Figure 3-30: (a) Binary difference frame, threshold = 2, generated from frame 1 and 2 of *Hall Monitor*. (b) Thinned binary difference frame, Pavlidis thinning. (c) Binary difference frame after connected component labelling and thresholding. (d) Canny edge map of original frame, 1. $\sigma = 0.8$, $T_H = 0.8$, $T_L = 0.2$ (e) Model after performing change detection and incorporating spatial information.

Figure 3-31 shows the results of the model initialisation process for the same sequence, however in this instance there is a moving object in the foreground. Figure 3-31(c) shows the effectiveness of removing noise pixels by identifying moving connected components and Figure 3-31(e) shows the effectiveness of incorporating spatial information to obtain a more accurate contour and in this instance to remove portions of the uncovered background, which had been detected in the change detection stage. An inherent shortcoming of using change detection techniques to identify meaningful objects is also observed in the figure as one of the man's legs does not move much between consecutive frames and is therefore not included in the model in Figure 3-31(e). The leg and other slow moving components would need to be registered during the tracking stage.

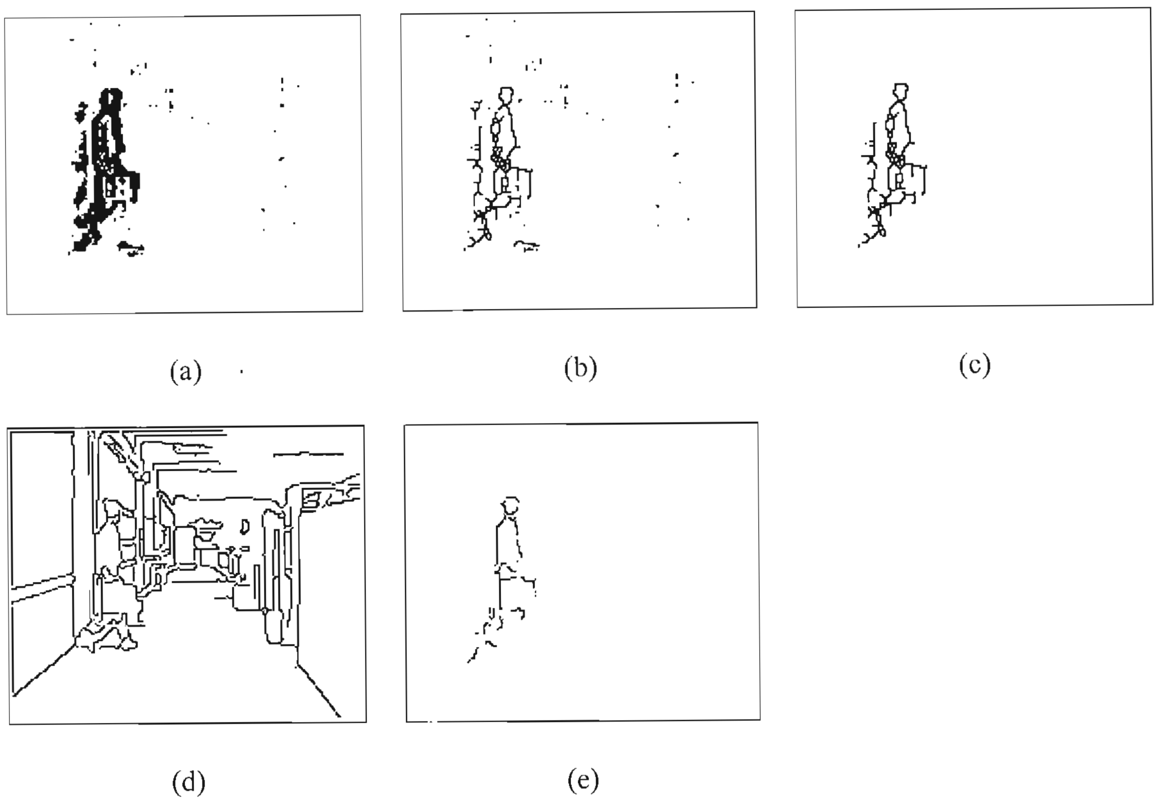


Figure 3-31(a) Binary difference frame, threshold = 4, generated from frame 30 and 31 of Hall Monitor. (b) Thinned binary difference frame, Pavlidis thinning. (c) Binary difference frame after connected component labelling and thresholding. (d) Canny edge map of original frame, 30. $\sigma = 0.8$, $T_H = 0.8$, $T_L = 0.2$ (e) Model after performing change detection and incorporating spatial information.

Figure 3-32 shows the results from first two frames of the sequence, *Akiyo*. Since the scene is that of a newsreader reading a bulletin, there is very little movement and the frame differencing and thresholding, Figure 3-32(a), yields a textured silhouette of the moving object, which after thinning yields the same silhouette composed of several connected components that are spatially close to each other. The resulting image after moving connected component selection shows the effectiveness of the spatially dependent thresholding strategy used to select moving connected components. By selecting all components within the spatial neighbourhood of the largest connected component, a reasonably accurate final model is obtained, Figure 3-32(e).

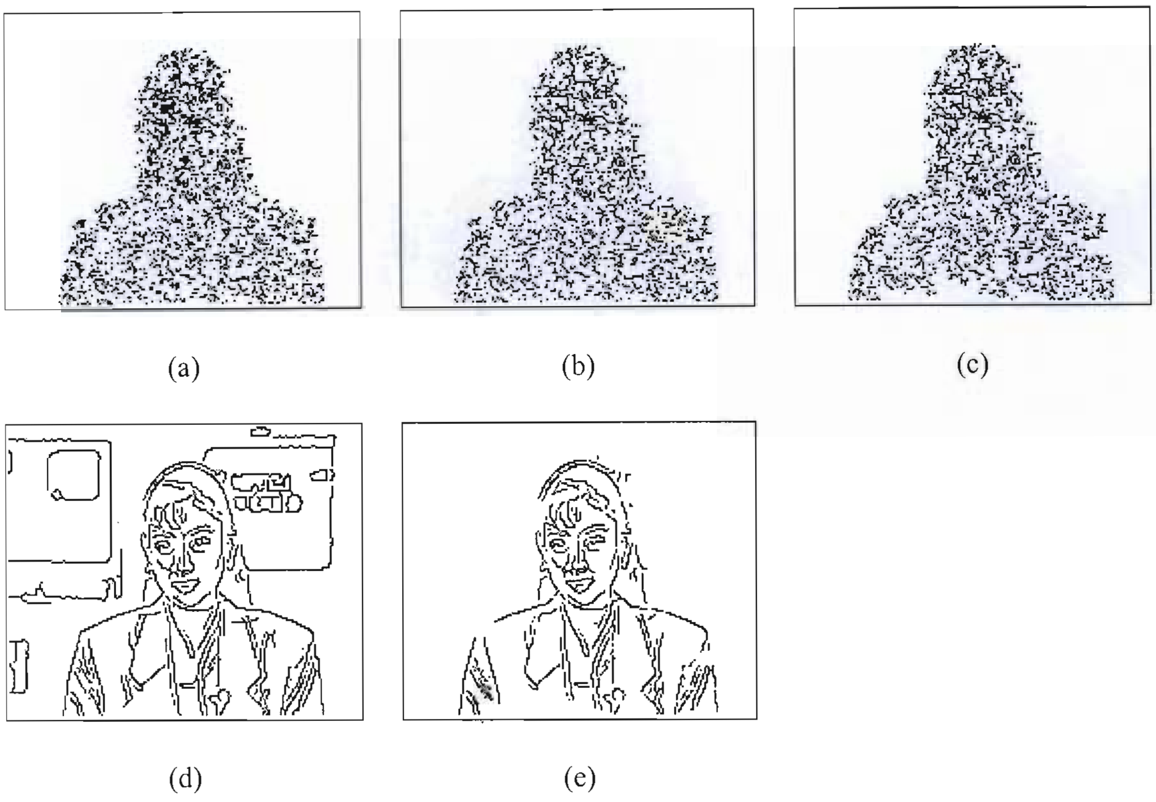


Figure 3-32: Binary difference frame, threshold = 1, generated from frame 1 and 2 of *Akiyo*. (b) Thinned binary difference frame, Pavlidis thinning. (c) Binary difference frame after connected component labelling and thresholding. (d) Canny edge map of original frame, 1. $\sigma = 0.6$, $T_H = 0.7$, $T_L = 0.2$ (e) Model after performing change detection and incorporating spatial information.

The model initialisation technique investigated is based on change detection and therefore requires movement of the object between consecutive frames to be successful. Hence if there were no movement in the first two frames of the sequence, the model initialisation would have to be repeated until motion is present. In addition, when there is movement of the object between consecutive frames but not all portions of the object are moving, the slower moving portions of the object would only be detected when the movement of these portions speed up. These two shortcomings of the model initialisation stage necessitate the incorporation of model updating in the tracking process. Tracking and model updating are discussed in the next chapter.

CHAPTER 4: OBJECT TRACKING

Moving objects in a video sequence change and evolve with time. When extracting moving objects, it is therefore necessary to track the object over time and compensate for changes to the object. In the literature there are two popular approaches to video object tracking, namely template matching and statistical tracking.

With regard to statistical tracking, Kalman filtering [Kalm, 1960] is probably one of the most commonly used approaches. The Kalman filter is essentially a mathematical model that implements a predictor-corrector type estimator that is optimal in the sense that it minimizes the estimated error covariance — when some presumed conditions are met. In image processing, Kalman filtering is used to track a set of object features. Since the Kalman filter is a robust optimal estimator, an attempt was made to include it as the tracking mechanism in the video object extraction system. Having obtained the model as in the previous section, the centroid of the object model is computed and used as the feature to be tracked. However, since the centroid itself changes as the object moves and the object's motion is non-linear, the Kalman filter design became more tedious than was previously anticipated and would have itself involved a sizable research effort.

In the literature however, numerous research works have applied Kalman filters. Foresti [Fore, 1999] applies Kalman filtering in a common framework for real-time object recognition and tracking for remote video surveillance. He uses a shape descriptor, namely, the statistical morphological skeleton, for object recognition and tracking. Recognition is obtained by comparing an analytical approximation of the skeleton function extracted from the analyzed image with that obtained from model objects stored in a database. Tracking is then performed by applying an extended Kalman filter to a set of observable quantities derived from the detected skeleton and other geometric characteristics of the moving object.

As mentioned, template matching is another common approach to tracking. This extends from simple block matching as applied in MPEG-1 and MPEG-2 to more complicated sequential and hierarchical template-based algorithms. Schoepflin *et al* [Schoe, 2001] use a sequence of template deformations to model and track the contour of generic video objects. The deformed template from the previous frame is used as the input to the next frame. Firstly, global affine deformations are performed to accommodate for changes in scale and rotation. Piecewise affine deformations are then used to improve the contour locally by accommodating for changes in parts of the object that move independently in relation to the entire object and finally active contours or snakes are used to refine the entire contour. The deformable template approach enables the algorithm to track objects whose pose and shape change in time relative to the template. The experimental results presented demonstrated that their method was able to track a variety of video objects, including those undergoing rapid changes. However the results also showed that more accurate contour extraction is possible. The approach is especially susceptible to occlusion, uncovered background areas and strong background edges.

To perform object tracking we used a template matching approach because it may easily be applied to the model that was initialised in the previous section. The model of the object is used as a template which is compared to the original frame for a match. Instead of using the original grey scale image which is very sensitive to changes in illumination, the original frame is first converted to a binary edge image; this also results in fewer computations later. As in the previous chapter, the edge map is obtained by the Canny operator [Canny, 1980]. A robust matching method is then required to be able to detect objects that are undergoing translation, rotation, and deformation. Affine deformations would not work well in our algorithm, as the motion is not necessarily rigid, thus the Hausdorff object tracker [Hutt, 1993b] as used by Meier and Ngan was applied in this work as well. In this approach, the model of the video object of interest is matched against subsequent frames by minimizing the Hausdorff distance. The approach is computationally efficient and robust in relation to noise and deformation. Since it is a non-parametric approach, it is also capable of tracking both rigid and non-rigid motion.

The Hausdorff tracker is used to track the global motion of the object, however, in semantic object extraction, parts of the object may have motion which is independent of this global trajectory, for example, a man may be walking down a corridor but his hands and head would move independently of this motion. While the Hausdorff tracker would be efficient at tracking the global motion, the model needs to be updated after every frame to compensate for the deformations due to local motion. The changes due to local motion may be identified and compensated by performing change detection after every frame. The composite model of the semantic object may then be obtained by juxtaposing the results from change detection and Hausdorff tracking. The change detection algorithm is the same as for model initialisation. Figure 4-1 represents a flow chart of the entire system. The flow chart has been colour coded to represent each stage of the sequential algorithm. The purple blocks pertain to the model initialisation stage described in the previous chapter. The output of this stage, $Model\_F_k$ is incorporated into the tracking stage, grey blocks, to compensate for local motion. In the tracking stage, the model obtained from Hausdorff tracking, $Model\_S_k$ is juxtaposed with $Model\_F_k$ to generate the final model, $FinalModel_k$. The final model is then used to segment the semantic object from the original frame, orange block. This stage is described in detail in the next chapter. The process is looped for each frame, k, of the sequence. To facilitate tracking, background filtering is performed on the edge map prior to Hausdorff matching.

In this chapter, the Hausdorff object tracking algorithm is first explained. Then the background filtering technique is discussed and finally results from the tracking stage are presented.

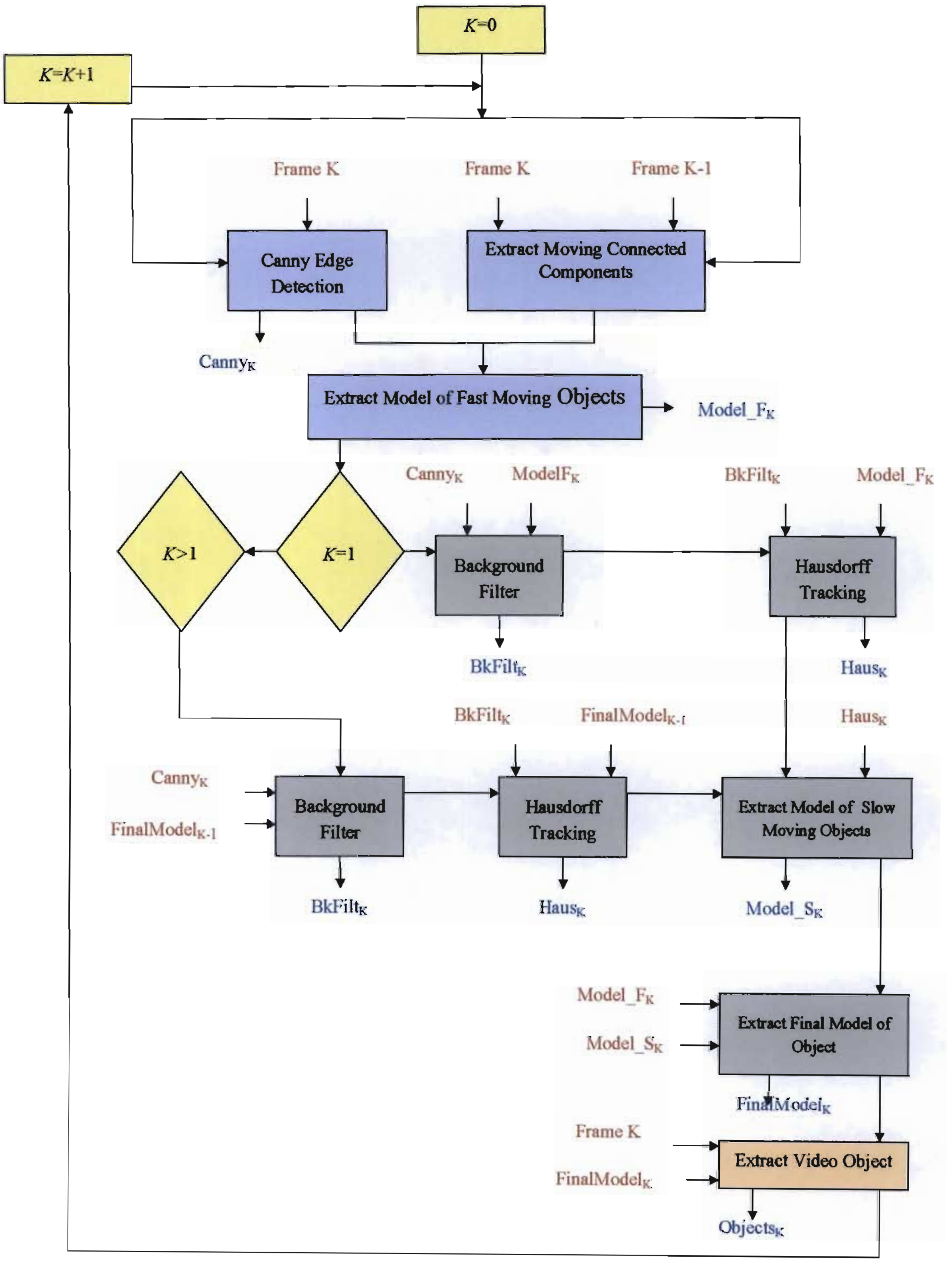


Figure 4-1: Overview of Segmentation Algorithm

4.1 Hausdorff object matching

4.1.1 The Hausdorff distance as applied to object matching

The Hausdorff distance was proposed by Huttenlocher *et al* [Hutt, 1990] as a measure to compare binary images or portions thereof. In the case of object tracking, there are two sets of feature points that are compared to each other, these are:

1. $O_k = \{o_1, o_2 \ldots, o_n\}$ which is the subset of edge pixels from frame k that forms the model of the object to track in the next frame, $k+1$,
2. $E_{k+1} = \{e_1, e_2 \ldots, e_p\}$ which is the set of edge pixels obtained from the next frame, $k+1$, in which we have to search for the object.

Then, the Hausdorff distance is defined as

$$H(O_k, E_{k+1}) = \max\{\, h(O_k, E_{k+1}), h(E_{k+1}, O_k)\} \tag{4.1}$$

where

$$h(O_k, E_{k+1}) = \max_{o \in O_k} \min_{e \in E_{k+1}} \|o - e\| \tag{4.2}$$

and

$$h(E_{k+1}, O_k) = \max_{e \in \mathrm{E}_{k+1}} \min_{o \in \mathrm{O}_k} \|e - o\| \tag{4.3}$$

From equation 4.2, for every model pixel, the distance to the nearest edge pixel is calculated and the maximum of these values is assigned to $h(O_k, E_{k+1})$. Similarly, in equation 4.3, for every edge pixel, the distance to the nearest object pixel is calculated and the maximum of these values is assigned to $h(E_{k+1}, O_k)$. Then the Hausdorff distance, as defined in equation 4.1, is the maximum of these two partial Hausdorff distances.

To find a best match for the model in the edge map, a set of translations, T, is specified. The model is then translated by each element of T, $t = (t_x, t_y)$, as represented by equation 4.4 and the Hausdorff distance for the translated model, H_t, is computed using equation 4.1.

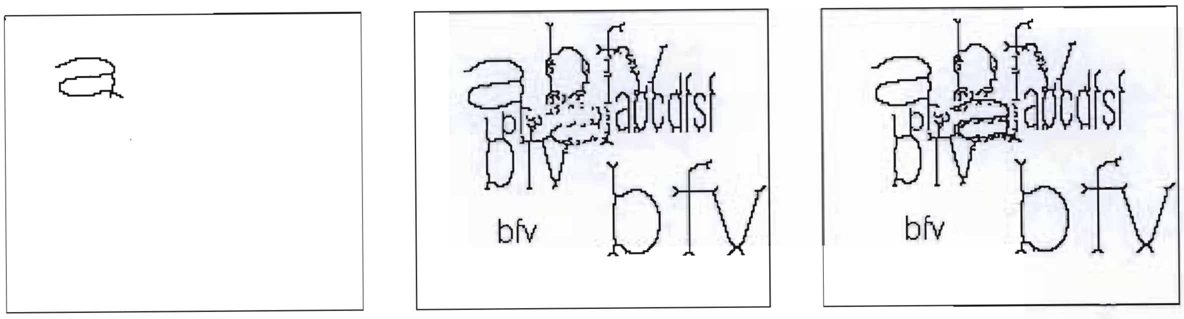
$$O_{k,t} = [o + t]_{o \in O_k} \tag{4.4}$$

The best match corresponds to the translation that yields the minimum Hausdorff distance, as represented in equation 4.5.

$$H_{match} = \min_{t \in T} H_t \tag{4.5}$$

Using $O_{k,match}$, the updated model, $Model\_S_k$ may be obtained from the Canny edge map as in section 3.2.

Figure 4-2 illustrates the Hausdorff matching. Figure 4-2(a) is the object image and Figure 4-2(b) is the edge image of interest. Figure 4-2(c) shows the translated object image after Hausdorff matching, superimposed on the edge image. In the edge image there are two possible matches for the object of interest, the "a" in the top left corner of the edge which is slightly larger than the object and the incomplete "a" in the centre of the edge image which is the same size as the object. As the resultant image reflects the object matches correctly to the "a" in the centre of the image. The results indicate that the method is effective in finding the optimal match even when there is more than one possible match.



(a) (b) (c)

Figure 4-2: (a) Object image (b) Edge image (c) Result of Hausdorff matching

4.1.2 Implementing the Hausdorff object tracker

In Meier and Ngan and Huttenlocher *et al* [Hutt, 1993a] the Hausdorff distance is computed by using the Euclidean distance transformation [Borg, 1986]. First, the distance transform is calculated for the edge image so that from each pixel position the distance to the nearest edge pixel is known. The model is then translated in all directions and for each translation the Euclidean distance transform at the location of the model point directly gives the distance between the model and the nearest edge pixel. This information is then used to compute the minimum Hausdorff distance as described above.

Since the model does not necessarily match perfectly with the image, even in the case of an optimal match, there may be a few outlier pixels which would set the partial distance higher than other partial distances for the particular set of translations, resulting in a sub-optimal match. This problem is especially grave in the case of a rapidly changing object or when there is partial occlusion of the object.

To minimise the effects of this problem, the generalised Hausdorff distance as expressed in equations 4.6 and 4.7 is applied. As may be observed, instead of using the maximum value as in equation 4.2 and 4.3, the distances are sorted in ascending order and the z^{th} and a^{th} values are chosen respectively where z and a are user input parameters. For the choice of $z = n$, equation 4.6 is equivalent to equation 4.2, however, for $z<n$, it implies that n-z points are probably outliers and are discarded.

$$h_{t,z}(O_{k,t}, E_{k+1}) = \underset{o \in O_{k,t}}{z^{th}} \min_{e \in E_{k+1}} \|o - e\| \tag{4.6}$$

$$h_{t,a}(E_{k+1}, O_{k,t}) = \underset{e \in E_{k+1}}{a^{th}} \min_{o \in O_{k,t}} \|e - o\| \tag{4.7}$$

The matching process can become computationally intensive, depending on the number of translations that are performed. To limit this problem, early scan termination is performed [Hutt, 1993a]. In early scan termination, a maximum partial Hausdorff distance, *maxHausdistance*, is specified as a user input parameter and only translations for which the partial Hausdorff distances are less than this threshold, that is, $h_{t,z}(O_{k,t}, E_{k+1}) < maxHausdistance$ and $h_{t,a}(E_{k+1}, O_{k,t}) < maxHausdistance$, are considered. To further reduce computational intensity, the set of translations, *T*, is limited to a user specified number of translations in each direction.

Furthermore, in the case of cluttered background, the majority of the pixels in the edge image would not pertain to the object of interest. Hence, the partial Hausdorff distance, $h_{t,a}(E_{k+1},O_{k,t})$ would not contribute towards finding an optimal match as the edge pixels would be distributed independent of the object. With reference to the example in Figure 4-3, the edge map represents a cluttered background, hence translating the model in the vicinity of the object of interest will not give much indication of the optimal match as the partial edge Hausdorff distance would be more greatly influenced by the background pixels. In this instance it is preferable to use only $h_{t,z}(O_{k,t}, E_{k+1})$, the partial object Hausdorff distance as represented in equation 4.8.

$$H_t(O_k, E_{k+1}) = h_{t,z}(O_{k,t}, E_{k+1}) \tag{4.8}$$

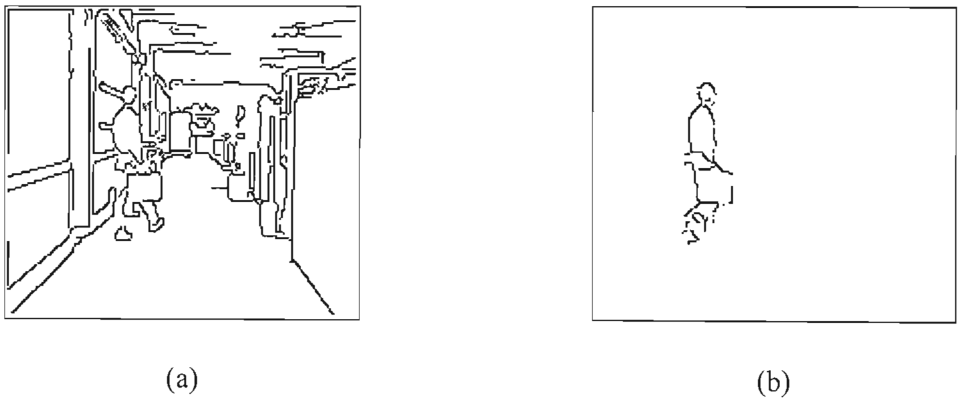

(a) (b)

Figure 4-3:(a) Edge map of frame *k*+1 (b) Object extracted from frame *k*

4.2 Background filtering

In the previous section the theoretical and practical aspects of the **Hausdorff track**er were discussed. At the end of the section, it was noted that a cluttered background could affect and impede obj**ect** matching. Even if the partial Hausdorff distance, $h_{t,z}(O_{k,t}, E_{k+1})$, is only used, unnecessary edge pixels due to a cluttered background might still affect the result as the optimal match is found by choosing "the best worst case" instead of correlating individual pixels. There is therefore justification for incorporating a background filter to limit the effects of background pixels on the object matching.

A simple background filter was implemented by incorporating a temporal buffer for each edge pixel. The buffer stores the number of consecutive times the pixel was an edge pixel. If this value exceeds a user input threshold, *Bkstat*, it is probable that the pixel belongs to the set of static background pixels rather than the moving object and can therefore be discarded. If however the object remains stationary for an extended period then object pixels would be regarded as background pixels and also be removed. To alleviate this problem the constraint that an edge pixel may only be classified as a static background pixel if it is not a subset of the set of object pixels was included. Using B_k as the set of static background pixels for frame, *k*. The above filtering process may be expressed in set notation as in equation 4.9.

$$B_k = \bigcap_{i=k-Bkstat}^{k} E_i \bigcap (O_{k-1})^c \qquad (4.9)$$

4.3 Results

In the previous two sections the theory and implementation issues regarding Hausdorff object tracking and background filtering were presented. In this section, results of the entire object tracking algorithm are presented, a sequence of original video frames is used as an input, and the binary images of the tracked model are generated as output. First the Hausdorff tracker is evaluated, both with and without background filtering. Thereafter, the local motion tracking is incorporated and the results compared to the previous set.

Prior to assessing the performance of the object tracker, it was necessary to tune the input parameters, *maxhausdistance*, *z*, and *T*. *z* is the percentage of pixels that will be considered and ought to be set to a percentage very close to 100% if the entire object is to be located in the image. For our tests we usually set it to 99.9%. The optimal set of translations, *T*, and the maximum Hausdorff distance, *maxhausdistance*, are dependent on the speed of the object's motion relative to the frame rate and are therefore dependent on the specific video sequence. It can be set arbitrarily large to not be sensitive to the motion, however this is at the expense of increased processing time.

The effect that each of these parameters had on the processing time of the entire algorithm has been evaluated and the results indicated that the parameter *T*, had the greatest impact. This was as expected as increasing *T* by one effectively increases the set of translations by eight. Figure 4-4 shows the relationship between *T* and processing time. The graph reflects a quadratic relationship between processing time and *T*.

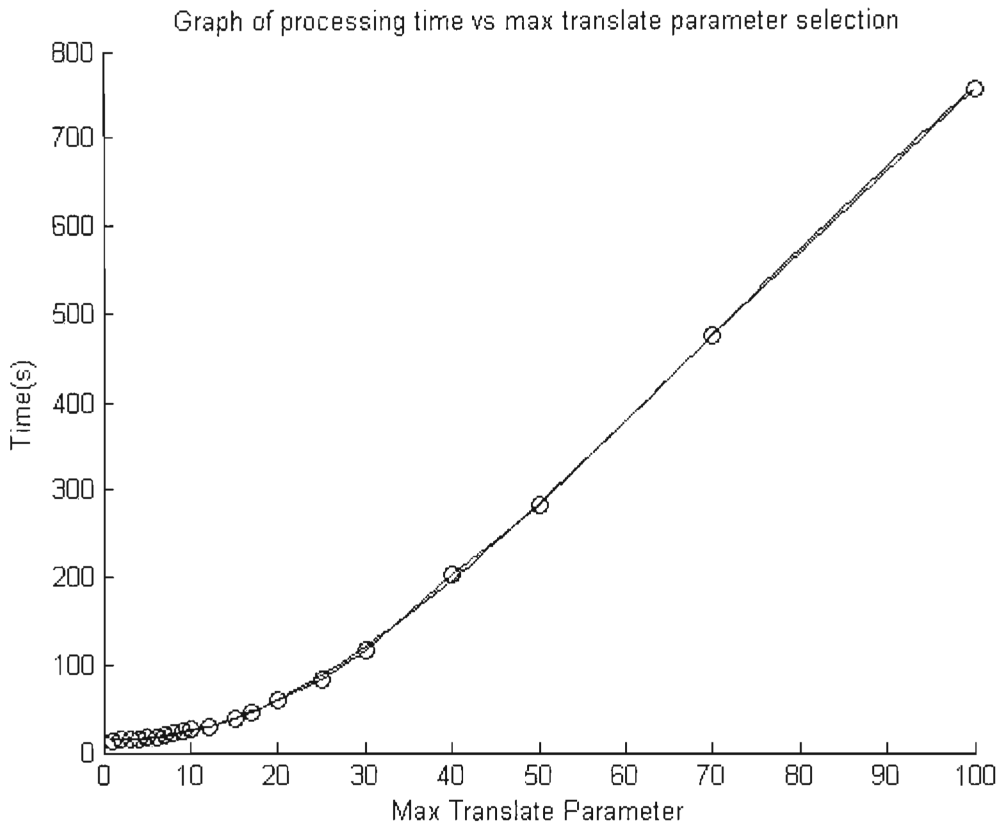
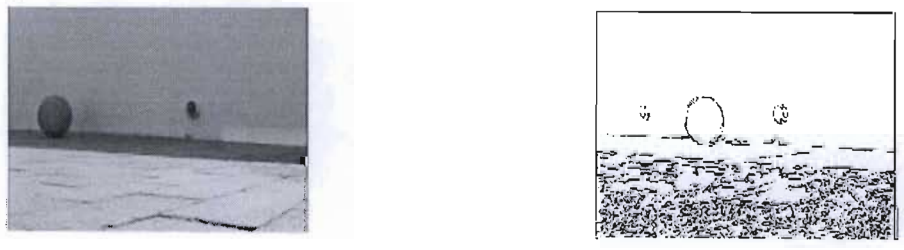


Figure 4-4: Effect of maximum translation parameter, *T*, on processing time

Figure 4-4 reflects the relationship between processing time and the maximum translation threshold, *T*. The parameter was varied from 1 to 100 and the processing time for the entire algorithm was recorded. Frames 30 to 45 of *Hall Monitor* were used for the input sequence and a rank of 99.9% and *maxhausdistance* of 500 was used.



(a) (b)

Figure 4-5: Ball rolling sequence (a) Example of an original frame (b) Edge Map of an original frame

Figure 4-5 indicates an original frame and an edge map from an outdoor video sequence of a ball rolling along a wall. As may be observed, the scene has an uncluttered background and little noise. The sequence was captured using a digital camera at 25 frames per second and 384 by 288 pixel resolution.

Figure 4-6 shows the results of the updated model after Hausdorff tracking. Although the tracking correctly locates the new position of the ball in subsequent frames, as the scene progresses, less of the contour information is captured. This occurs because the shape or contour of the 2-D projection of the ball changes as it rolls. Hence it is not sufficient to superimpose the model for frame$_{k-1}$ on to the optimal match position for frame$_k$ and simply extract matching pixels. It is necessary to update the contour first. This is done using change detection and the results are presented in Figure 4-7. As is indicated in the figure, a more accurate contour is extracted when both fast and slow moving changes are tracked.

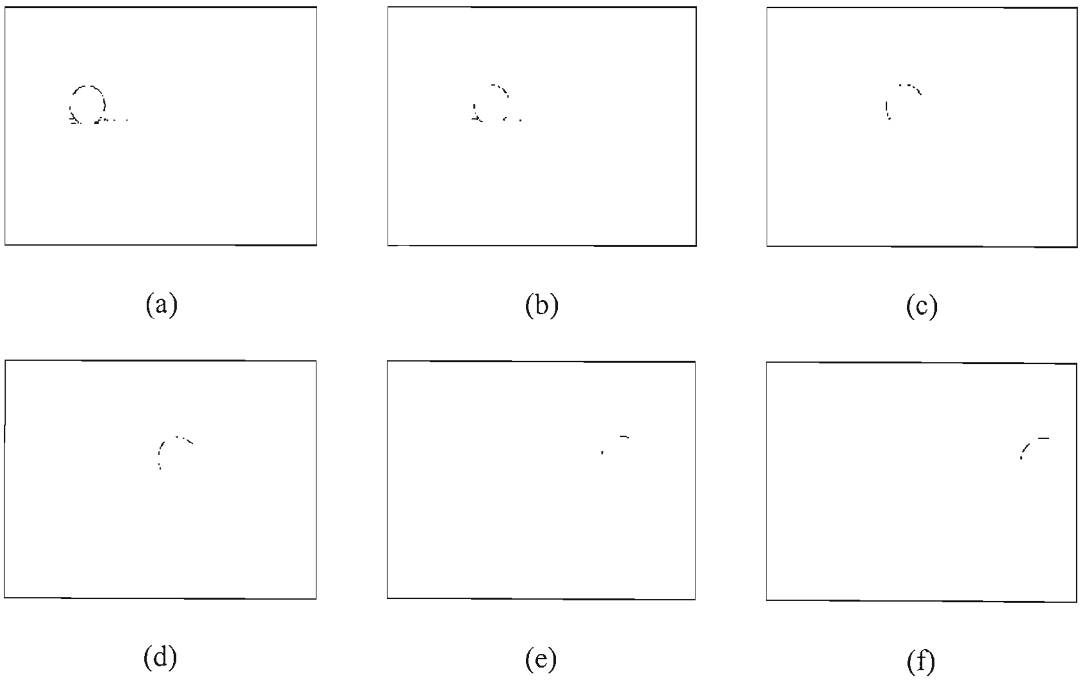

(a) (b) (c) (d) (e) (f)

Figure 4-6: (a)-(f) indicate updated model of ball after Hausdorff tracking

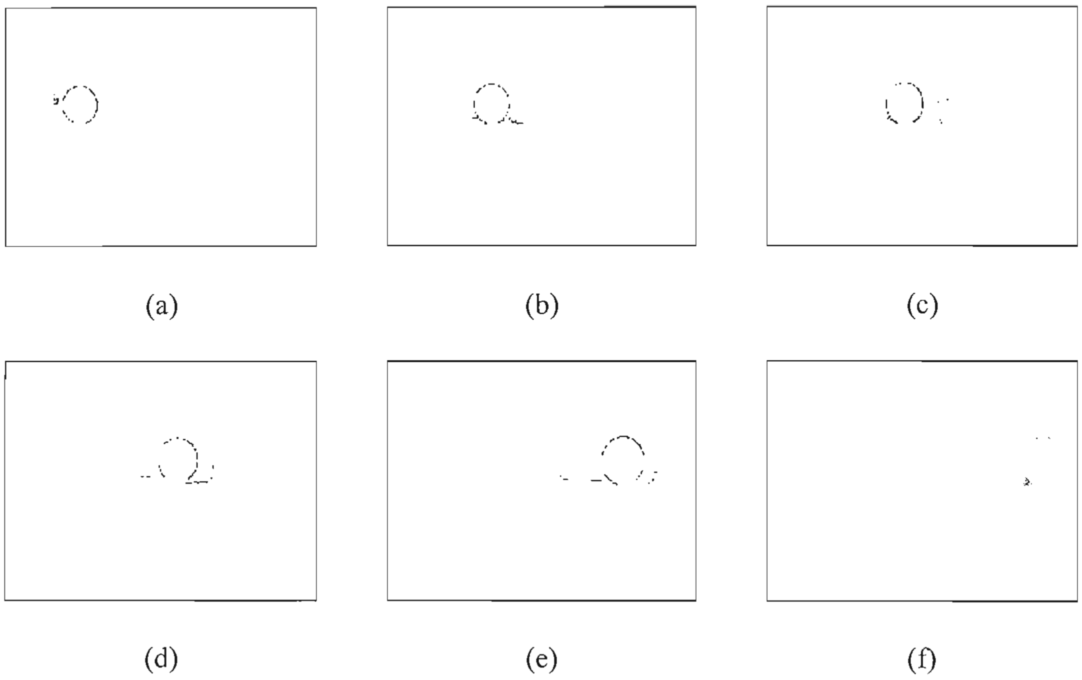


Figure 4-7: (a)-(f) indicate updated model of ball after Hausdorff tracking and change detection

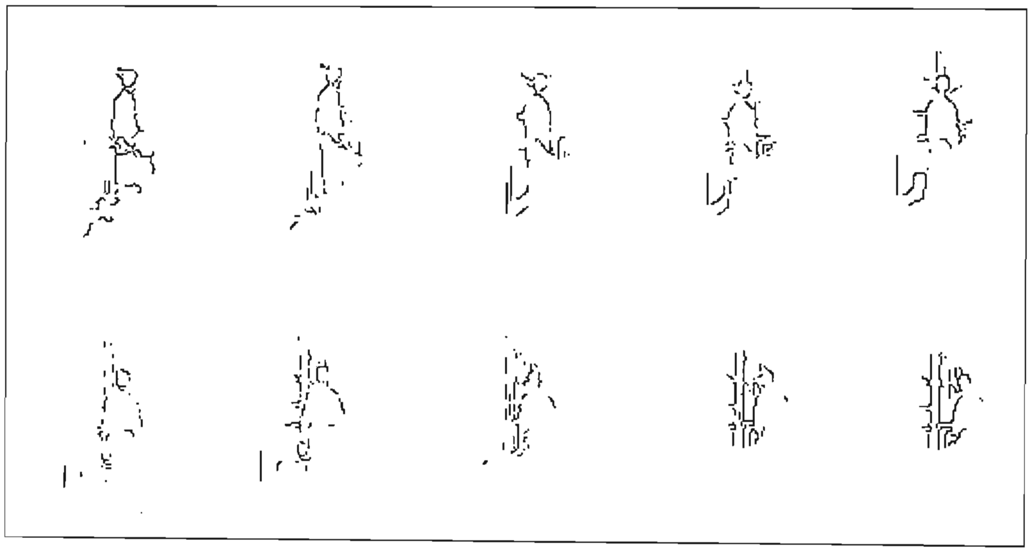


Figure 4-8: Results of Hausdorff tracking as applied to MPEG 4 video sequence *Hall Monitor*: frames 31, 32, 36, 40, 44, 50, 54, 61, 69 and 74

Figure 4-8 shows the results of Hausdorff object tracking for several frames from the sequence *Hall Monitor*. This sequence is more complex as the object of interest, namely the man in the foreground is completely non-rigid and each limb moves differently relative to his torso. The scene also has a cluttered background and due to the indoor lighting shadow effects come into play. Frame 31 shows the initial model generated for tracking and the subsequent frames show the binary model extracted from the edge map after tracking. Since the video sequence is one of a man walking down a corridor, the object's shape is deforming over time, and as may be observed from the result, the accuracy of the model is reduced as the sequence progresses. As mentioned previously, the Hausdorff tracker keeps track of the slow changes of the object, but is unable to track the fast changes. This is evident in the above result, where the man's upper torso is moving slower and with more rigid motion than the legs and is therefore more accurately tracked.

In Figure 4-9, the same set of frames is presented however in this set both slow and fast changes are tracked and the juxtaposition of the two forms the tracked object model. As is evident from the result a more semantically meaningful extraction of the entire moving object is obtained. One may note especially the effectiveness of the change detection algorithm at tracking fast non-rigid motion.

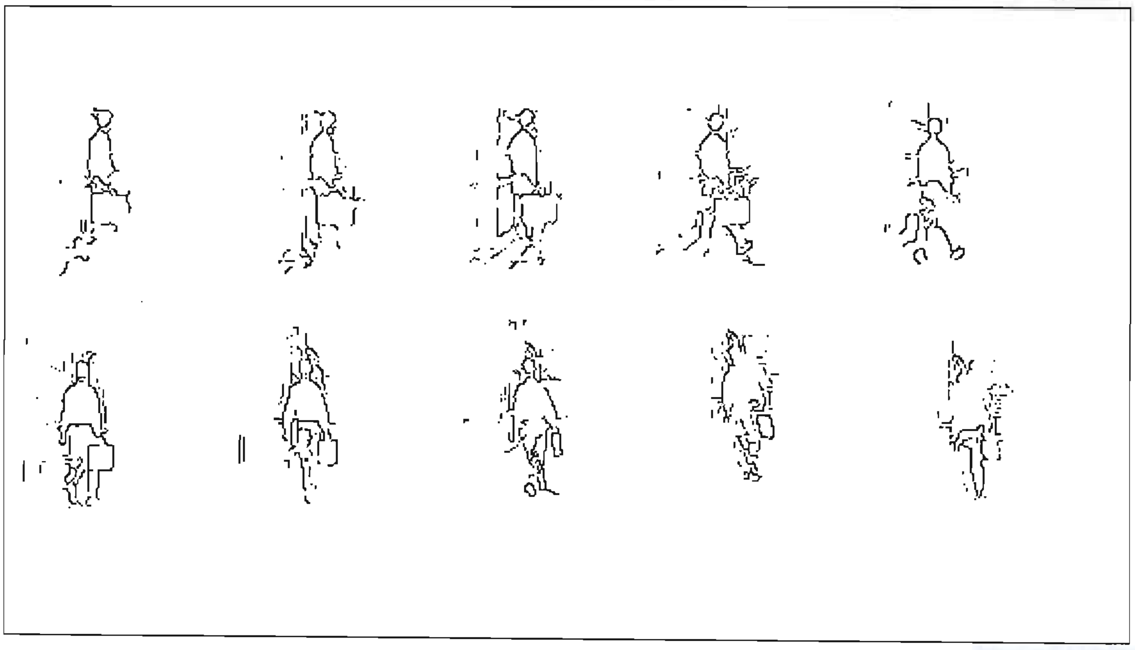


Figure 4-9: Results of Hausdorff tracking and change detection as applied to MPEG 4 video sequence *Hall Monitor*: frames 31, 32, 36, 40, 44, 50, 54, 61, 69 and 74

The above result may be further improved if less of the background was included in the object model. Figure 4-10 shows the result with the inclusion of the temporal filter. A memory length of five frames was used. From the result, one may observe that for frames 32 and 36, more clutter is included as the filter has not yet initialise, while for the remaining frames clutter that is very close to the object is included, but a significant amount has been removed. Figure 4-11 and Figure 4-12 show the results of the fast and slow moving models respectively which were combined to obtain the result in Figure 4-10. By comparing Figure 4-8 to Figure 4-12, one may observe that the inclusion of change detection and background filtering significantly improves the result. In Figure 4-11 the amount of clutter is much less than in the final model, Figure 4-10, however the algorithm on its own fails in frame 74, when most of the object remains stationary.

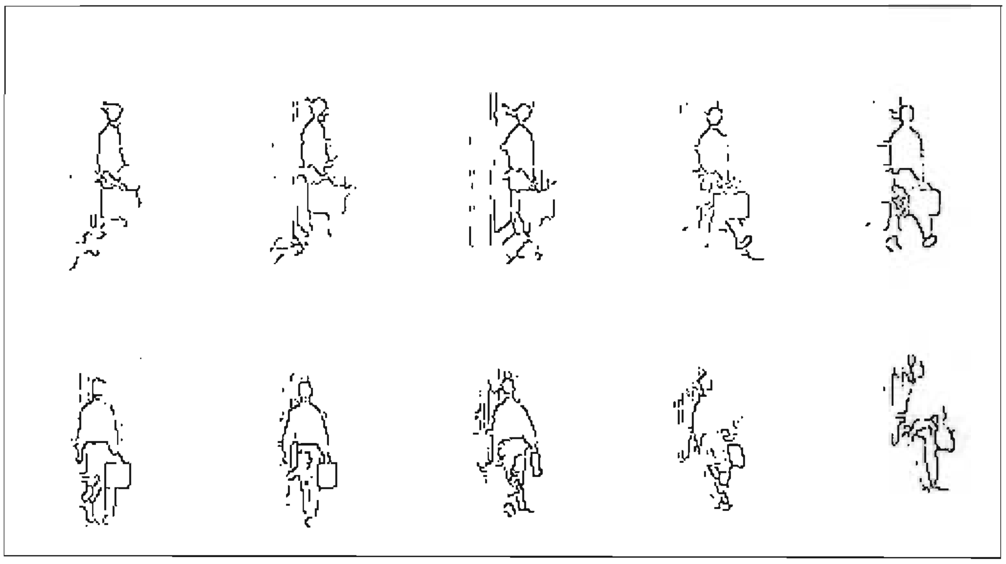


Figure 4-10: Results of Hausdorff tracking and change detection after background filtering as applied to MPEG 4 video sequence *Hall Monitor*: frames 31, 32, 36, 40, 44, 50, 54, 61, 69 and 74

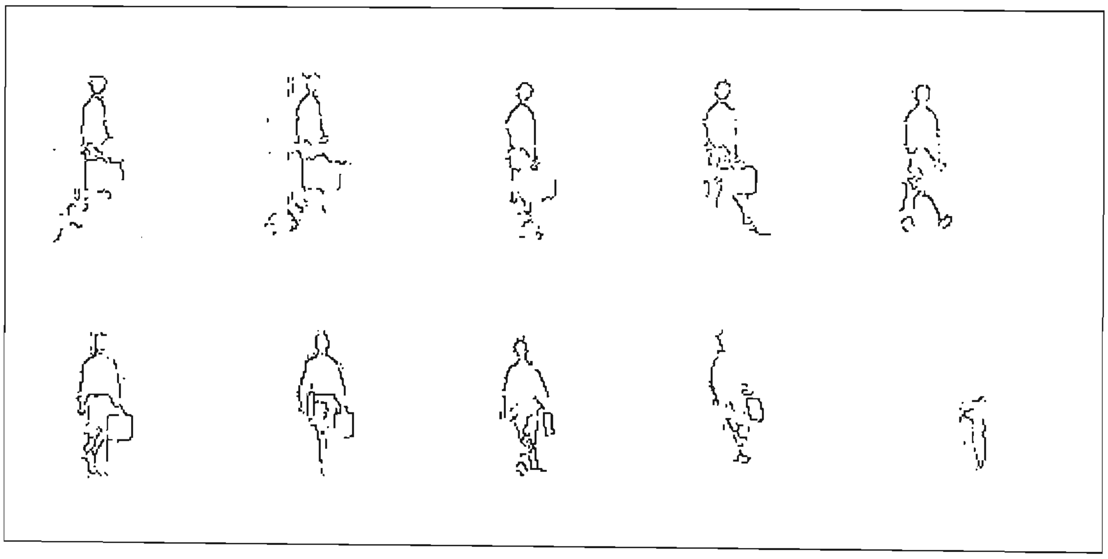


Figure 4-11: Results of fast moving model from complete tracking algorithm as applied to MPEG 4 video sequence *Hall Monitor*: frames 31, 32, 36, 40, 44, 50, 54, 61, 69 and 74

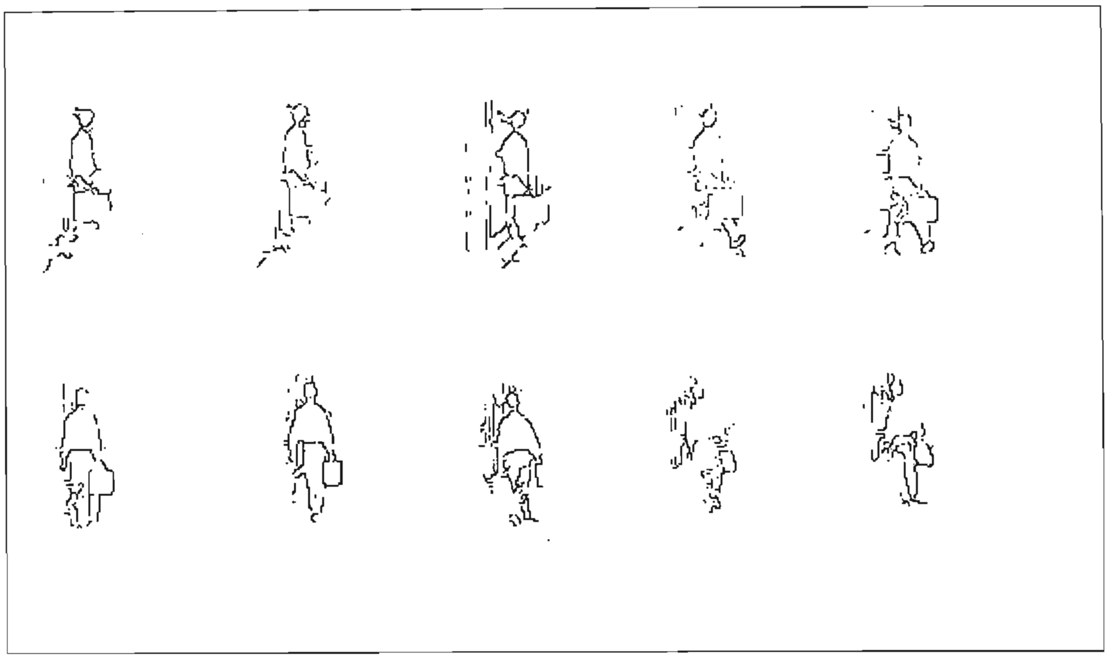
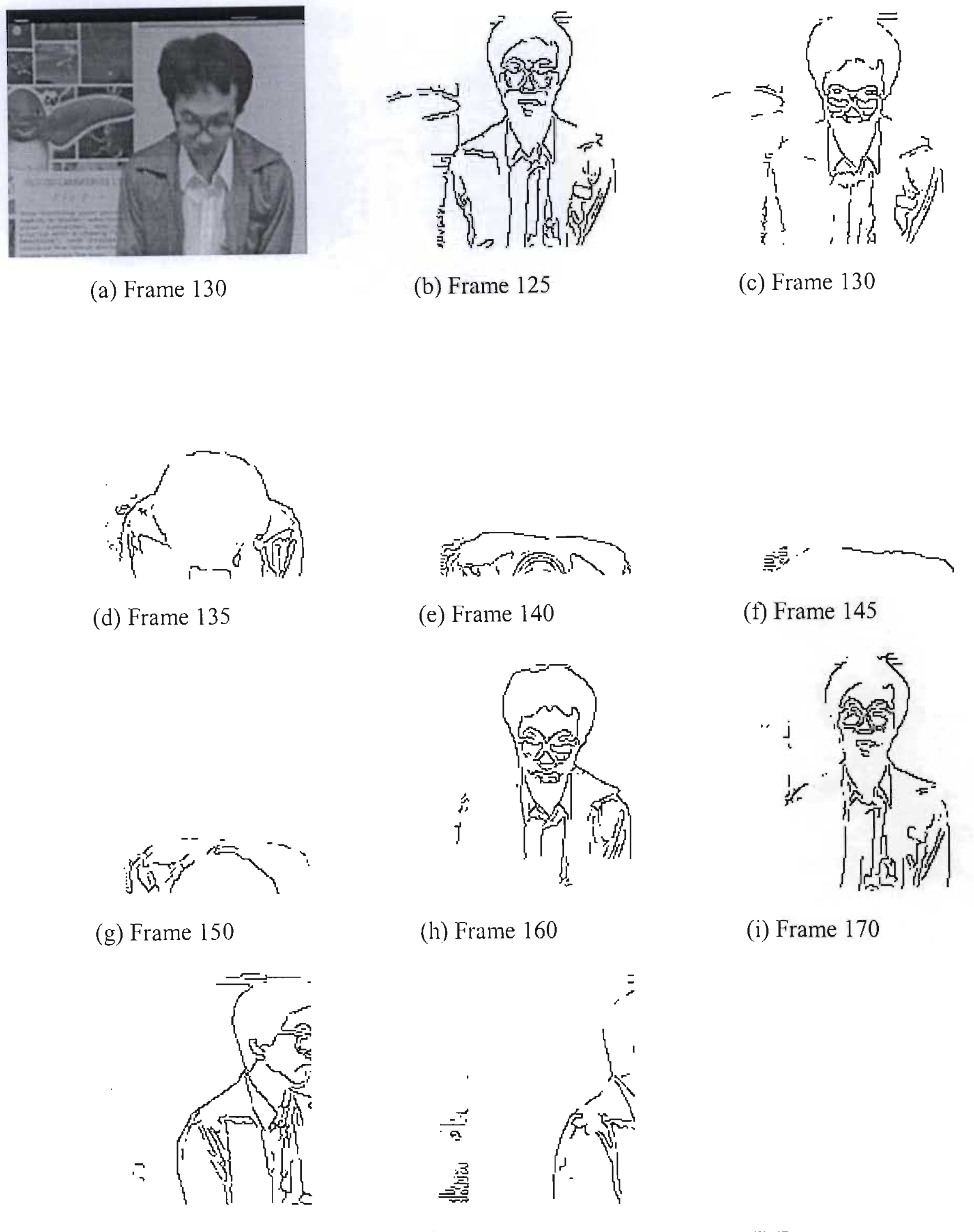


Figure 4-12: Results of the slow moving model from complete tracking algorithm as applied to MPEG 4 video sequence *Hall Monitor*: frames 31, 32, 36, 40, 44, 50, 54, 61, 69 and 74

Figure 4-13 shows the results of the final model after Hausdorff tracking, change detection and background filtering (*BkStat* = 5) for the video sequence *Bowing*. As observed in Figure 4-13 (a), the object of interest is that of a man who starts standing straight in front of the camera, bows forward, then turns to his left and walks out of the view of the camera. The bowing motion is fast and causes rapid changes in the shape of the object of interest. Results in Figure 4-13 (b) – (i) show the updated model keeping track of these changes. In the latter part of the sequence, when the mans walks out of the scene, only the stationary background is within the view of the camera and the tracking technique correctly identifies no objects to track, as indicated in Figure 4-13 (l). One of the shortcomings of the tracking mechanism is also highlighted in this set of results and this is the inability to compensate for shadows. This is indicated by the texture regions to the man's right in Figure 4-13 (b), (c), (i), (j) and (k).


(a) Frame 130 (b) Frame 125 (c) Frame 130

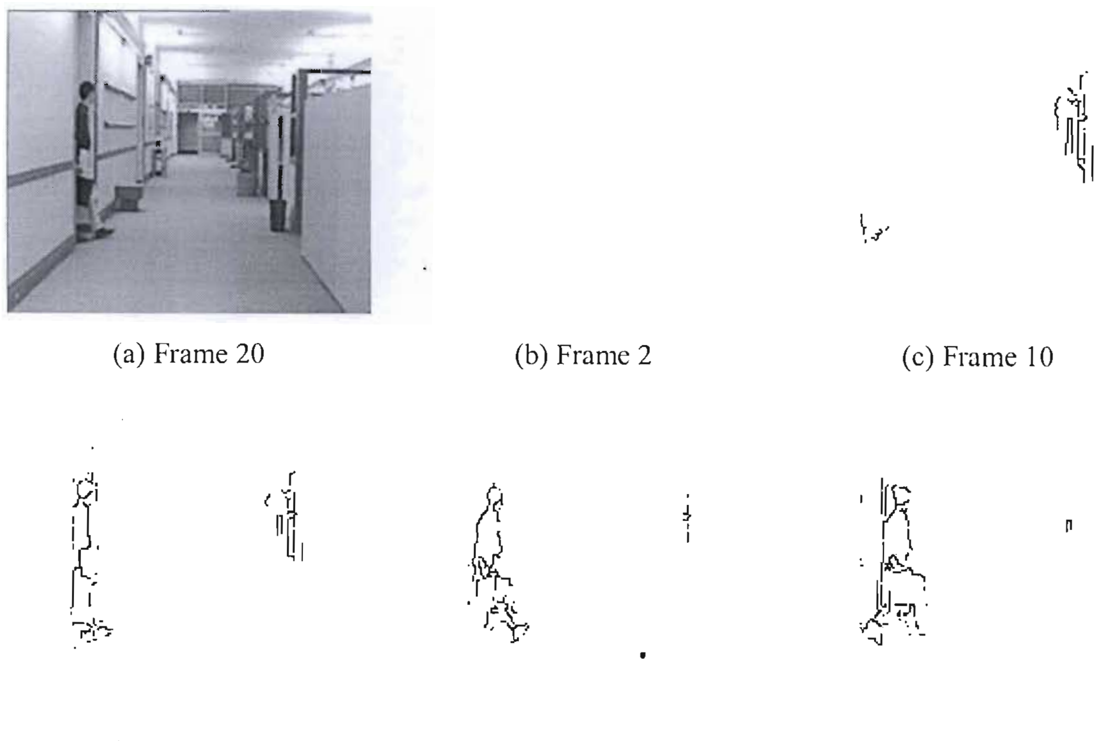
(d) Frame 135 (e) Frame 140 (f) Frame 145

(g) Frame 150 (h) Frame 160 (i) Frame 170

(j) Frame 200 (k) Frame 205 (l) Frame 215

Figure 4-13: MPEG 4 Video Test Sequence *Bowing* (a) Example of original frame (b-l) Results of model from complete tracking algorithm as applied to the sequence

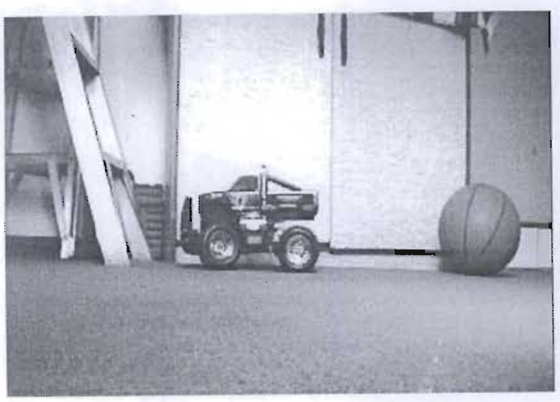
Figure 4-14 shows another set of results from the video sequence *Hall Monitor*, to illustrate the ability of the algorithm to identify a moving object from a video sequence that begins with only the stationary background and where the moving object enters the sequence as it progresses. Figure 4-14 (b) shows that no object is identified when there is no moving object in the sequence. However, *Hall Monitor* is a noisy sequence, which causes changes in the image. Hence in Figure 4-14 (c) the tracking algorithm has converged on noise. However, when the object appears in successive frames, the algorithm correctly identifies the man as the object of interest and gradually stops tracking the noise and tracks the man instead, as in Figure 4-14(d-f). It should be noted that without the change detection, the Hausdorff tracking alone would not be able to converge on a moving object in a scene if it enters the scene after model initialisation.


(a) Frame 20 (b) Frame 2 (c) Frame 10

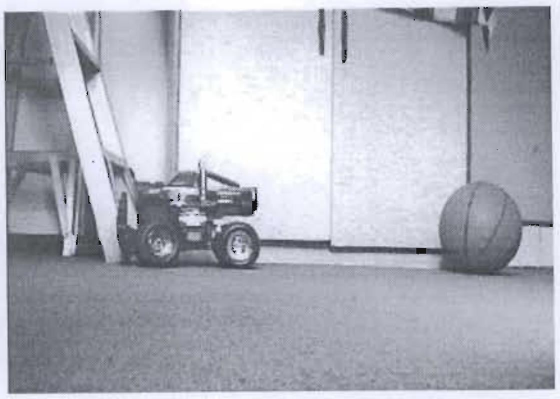
(d) Frame 20 (e) Frame 25 (f) Frame 30

Figure 4-14 (a) Original frame from MPEG 4 sequence *Hall Monitor* (b-f) Results of final model after Hausdorff tracking, change detection & background filtering(*BkStat*=5)

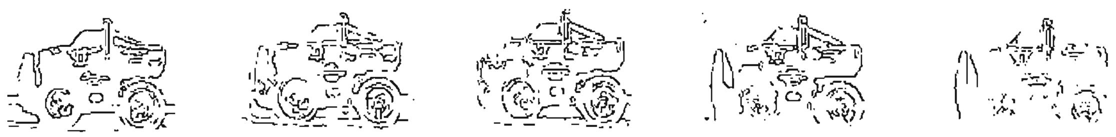
Figure 4-15(a) and (b) show original frames from the Toy Truck video sequence. Like the ball rolling sequence, the video was captured using a digital video camera at 384 by 288 resolution. In the sequence, a remote controlled truck moves across the view of the camera towards the ladder where it stops. As a result the truck is stationary from frames 58 to 80. Figure 4-15(c) shows the results of the final model for frames 30, 40 and 50, where the truck is moving and frames 60 and 70 where it is stationary. When the truck is moving the final model is the combination of the results from both Hausdorff tracking and change detection, however when the truck stops valid results are only obtained from Hausdorff tracking. Since there is no movement between consecutive frames, the change detection algorithm converges on noise in the frame. As may be observed from the model for frame 70, the model does degrade over time. This is the result of the edge detection; although the frames appear identical, the canny edge maps are not identical. Hence, while most of the stationary object is tracked, parts of it are lost. It was also observed that when the object stops, the results from change detection sometimes lead to an invalid final result. This occurs if the adaptive threshold converges on random noise and returns most of the original frame after change detection. To avoid this problem, a manual threshold may be used.



(a)



(b)



(c)

Figure 4-15: (a) Original Frame 40 Toy Truck Sequence. (b) Original Frame 70 of Toy Truck Sequence. (c) Final model of Toy Truck as extracted from frames 30, 40, 50, 60 and 70 in sequence.

From the analysis, we may conclude that the combination of both slow and fast motion tracking mechanisms facilitate the tracking of an object with non-rigid motion and which has components with different motions. The inclusion of the temporal filter helps to remove the background clutter but can be counterproductive if the object of interest remains stationary for several frames. It is therefore advisable to use it as an optional parameter. Even without the background filter, when a moving object stops in the scene, the contour degrades over time. Although the original consecutive frames appear identical, as a result of lighting and reflections the corresponding canny edge maps are not always the same and hence the same object in consecutive frames would not map perfectly onto each other. One possible method of alleviating this problem is to tune the Canny parameters to be less sensitive to weak edges, however this would also greatly lessen the contour detail of the object. The algorithm is able to successfully identify and track an object that enters the scene during the sequence and also responds correctly to an object that has left the scene. However when there are no objects in the scene, the adaptive thresholding method may converge on noise, yielding invalid results. While this was not a critical shortcoming in the model initialisation phase, the implications for tracking are more severe. Further work on the thresholding strategy is necessary to alleviate this problem. One possible solution is to examine the relative variance and the change in the relative variance for the range of threshold values and use this as an additional criterion for selecting the optimal threshold. The more distinct the peak in the relative variance versus threshold graph, the greater the likelihood of a correlated signal, hence if the graph is relatively flat, the user may be requested to input a threshold, or a higher threshold may be selected. With regard to processing time, the Hausdorff tracker is the most computationally intensive component of the system, and careful tuning of the tracking parameters is critical to ensure improved processing performance. The overall performance of the tracker may be regarded as satisfactory and with further investigations the shortcomings mentioned might be overcome.

CHAPTER 5: EXTRACTING THE OBJECT

In Chapter 3, the model initialization using a change detection mask and morphological operators was discussed. The change detection mask yields a model of the moving components. This is then compared to a contour map of the frame to extract a more accurate contour of the moving object. This model, based on the contour of the object, is then used for tracking.

Tracking is necessary as the video object is moving as the sequence progresses and it needs to be updated over time. To accomplish this, an object tracker based on the Hausdorff object-matching algorithm was implemented. This was discussed in Chapter 4. After tracking, the object contour is updated accordingly and the video object is extracted. In this chapter, the approach to extracting the video object is discussed.

5.1 VOP extraction algorithm

The output from the object tracking module for the current frame, k, is a binary map of the updated contour of the moving object. The contour pixels are defined by the set of pixels, O_k. Figure 5-1 shows the original frame 41 of MPEG 4 video test sequence, *Hall Monitor* and Figure 5-2 shows the binary contour map of the moving object that is obtained from this frame after tracking and updating.

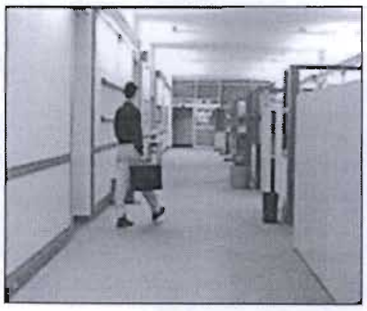


Figure 5-1: Frame 41 of MPEG4 Test Sequence, *Hall Monitor*

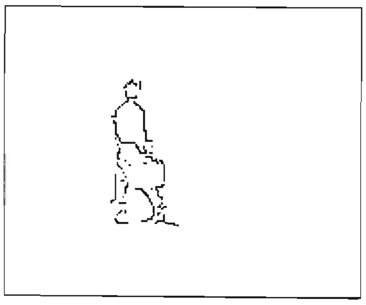


Figure 5-2: Binary moving object contour map of frame 41 of Hall Monitor

The object tracking and updating phases identify the moving object and the next step is to extract the object from the original input frame. One of the shortcomings however, is that the object contour is not closed; the object extraction is therefore not a trivial exercise. Before object extraction, it is necessary to first ensure that the contour forms a closed boundary of the object. Hence the first step is to trace the outer contour of the video object and "fill" any gaps with pixels.

The algorithm employed traces the outer contour in either a clockwise or anti-clockwise direction and compensates for breaks by connecting a straight line from the unconnected pixel to the next closest object pixel in the path. Since the preceding stages are expected to yield a reasonably accurate model of the contour, the breaks in the contour are not expected to be large and therefore it is adequate to use a straight line to compensate. Thereafter, a raster scan approach can be applied to extracting the pixels within the bounds of the contour as the object itself. In 5.1.1 the algorithm for closing the contour is presented and in 5.1.2 the object extraction is discussed.

5.1.1 Closing the contour

Given $O_k = \{o_1, o_2, \ldots, o_n\}$, as the set of object pixels with cardinality, n, we may define C_k as the set of outer boundary pixels such that:

$$C_k = \{c_1, c_2, .., c_m\} \subseteq O_k$$
$$\text{with } |C_k| = m$$

Furthermore,

$$c_i \in C_k \text{ iff}$$
$$\text{if } \forall o \in O_k,$$
$$\left[\begin{array}{l}\left[\begin{array}{l}(\mathrm{col}(o) = \mathrm{col}(c_i)) \text{ and} \\ \left[(\mathrm{row}(c_i) \geq \mathrm{row}(o)) \text{ or} (\mathrm{row}(c_i) \leq \mathrm{row}(o))\right]\end{array}\right] \\ \text{or} \\ \left[\begin{array}{l}(\mathrm{row}(o) = \mathrm{row}(c_i)) \text{ and} \\ \left[(\mathrm{col}(c_i) \geq \mathrm{col}(o)) \text{ or} (\mathrm{col}(c_i) \leq \mathrm{col}(o))\right]\end{array}\right]\end{array}\right]$$

where col(o) is the column of o and row(o) is the row of o.

We can now define Cl_k as the set of pixels which form the closed outer contour of the object as follows.

$$Cl_k = \{c_1, c_2, .., c_p\} = (SC_k \cup I_k)$$
where
$$SC_k \subseteq C_k$$

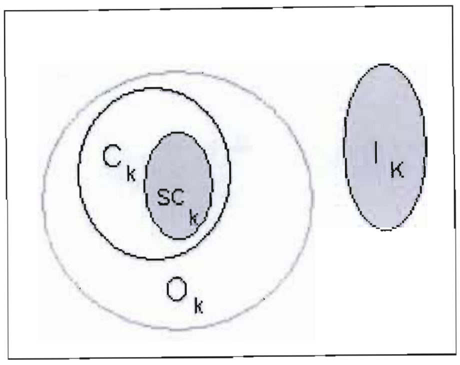


Figure 5-3: Venn Diagram illustrating the relationship between the pixel sets. The shaded portion is Cl_k.

The Venn diagram in Figure 5-3 illustrates the set relations. The cardinality of Cl_k is p and I_k is the set of pixels that are added to interpolate gaps in the contour. I_k is constructed at the same time that Cl_k is constructed.

The set Cl_k is initialised as follows:

Given $i = 1$,
$cl_i \in Cl_k$
such that
$\text{row}(cl_i) = \min\limits_{c \in C_k} \text{row}(c)$
and
$\left[\left(\text{col}(cl_i) \le \text{col}(c)\right)\right]$

The above initialisation sets the left topmost pixel as the starting point for tracing and interpolating the contour. This was an arbitrary choice so selected as a convenient point to commence tracing the contour. Having initialised Cl_k, the remainder of the set may be constructed as follows:

If there exists a $c \in C_k$ such that $d(c,cl_i) = 1$

Then $cl_{i+1} = c$ and $i = i+1$

where $d(p,q)$ is the chessboard distance between two pixels.

If no such pixel exists however, then it is necessary to interpolate the contour at that point. To interpolate the contour, we first search for a pixel in C_k which meets the criteria in equation (5.1) and if it is found, it is added to the set of closed contour pixels.

$$cl_{i+n} = c \text{ such that } \min_{c \in C_k} d(cl_i, c) \tag{5.1}$$

Since the newly added element, cl_{i+n} is unconnected to the last element cl_i, it is necessary to interpolate this break in the contour. This is done by first determining the gradient of the straight line connecting cl_{i+n} and cl_i as in equation (5.2).

$$\text{grad}(cl_{i+n}, cl_i) = \frac{\Delta y}{\Delta x} \tag{5.2}$$

where $\Delta y = \text{row}(cl_{i+n}) - \text{row}(cl_i)$ and $\Delta x = \text{col}(cl_{i+n}) - \text{col}(cl_i)$

We then use the gradient information to generate the interpolating pixels, $cl_{i+1} \ldots cl_{i+n-1}$, as follows:

for $a = [n-1, 1]$

$$\text{row}(cl_{i+n-a}) = \begin{cases} \text{row}(cl_i) + (|\Delta y| - 1) * \text{sign}(\Delta y) & \text{for } \Delta y \neq 0 \\ 0 & \text{for } \Delta y = 0 \end{cases}$$

If such an element is defined then, Δy is updated if non-zero.

$$\Delta y = (|\Delta y| - 1) * \text{sign}(\Delta y) \quad \text{for } \Delta y \neq 0$$

Similarly,

$$\text{col}(cl_{i+n-a}) = \begin{cases} (|\Delta x| - 1) * \text{sign}(\Delta x) & \text{for } \Delta x \neq 0 \\ 0 & \text{for } \Delta x = 0 \end{cases}$$

If such an element is defined then, Δx is updated if non-zero.

$$\Delta x = (|\Delta x| - 1) * \text{sign}(\Delta x) \quad \text{for } \Delta x \neq 0$$

The algorithm described above, yields a set of pixels, Cl_k, which forms the closed outer boundary of the object. It is now possible to extract the pixels within the boundary as the object. This is discussed in the next section.

5.1.2 Extracting the object

The object is extracted by determining the left and rightmost boundaries of the object for each row of the binary closed contour map that contains contour pixels. The pixels between these bounds in the original image are then extracted as the object.

To accomplish this, for each row of the original frame, I_k, we first find the leftmost boundary pixel, l, of the object for that row, if one exists.

If there exists an $l \in I_k$ and $cl \in Cl_k$,

such that

$\text{col}(l) \leq \text{col}(cl)$ and $\text{row}(cl) = \text{row}(l)$,

then l is the leftmost boundary pixel for row(l).

Similarly, if there exists an $r \in I_k$ and $cl \in Cl_k$,

such that

$\text{col}(r) \geq \text{col}(cl)$ and $\text{row}(cl) = \text{row}(l)$,

then r is the rightmost boundary pixel for row(r).

Logically, for every valid leftmost boundary pixel, there is a corresponding rightmost one, such that:

col(l) $\leq$ col(r) and row(r) = row(l).

Hence for each row where this condition exists, it is possible to extract the pixels between them as the object.

Figure 5-4 (a) represents the binary closed contour map of an arbitrary object and Figure 5-4(b) shows the object extracted by the method described above. It may be observed that portions of the background, that is, areas outside the contour, are also extracted as part of the object. These are the darker shaded portions in the figure. This extraction may be improved by examining the top and bottom boundaries and removing the pixels between the contour and these boundaries, Figure 5-4(c).

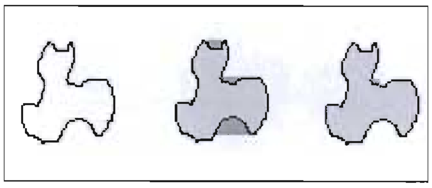


Figure 5-4: (a) Arbitrary object with closed contour; (b) Extract Object: Shaded portions indicate the object; (c) Improved object extraction

It is first necessary to identify the boundary pixels for each column. The upper boundary pixel, t, can be found as follows:

If there exists a $t \in F_k$ and $cl \in Cl_k$,

such that

row(t) $\leq$ row(cl) and col(cl) = col(t),

then t is the topmost boundary pixel for col(t).

The bottommost boundary pixel for each column is then determined as such:

If there exists an $b \in F_k$ and $cl \in Cl_k$,,

such that

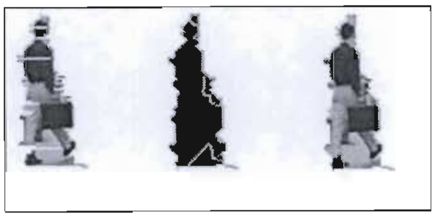
row(b) $\geq$ row(cl) and col(cl) = col(b),

then b is the bottommost boundary pixel for col(b).

Figure 5-4(c) shows the improved object extraction, after removing pixels between the top and bottom boundaries of the extracted object and the object contour.

5.2 Results

In this section, results of the object extraction method discussed in the previous section, are presented. A sequence of original video frames is used as an input, and the monochrome extracted object are generated as an output.



(a) (b) (c)

Figure 5-5: (a): Object Extracted from Hall Monitor Frame 41 without first closing the contour; (b) Closed Outer Boundary of Object in Hall Monitor Frame 41; (c) Object Extracted from Hall Monitor Frame 41 after contour closing.

Figure 5-5(a) shows the object extracted using the method in section 5.1.2 but without first closing the contour. Figure 5-5 (b) shows the closed outer contour of the binary object model (Figure 5-2) obtained from Frame 41 of *Hall Monitor* and Figure 5-5 (c) the subsequent object extraction. An inherent problem with the system is that it does not compensate for shadows hence the darkened portion of background between the man's two feet are also included as part of the object as can be observed in Figure 5-5 (c). The method extracts the outermost contour of the object and cannot compensate for holes within the object, hence in Figure 5-5(c), the lighter portions of the background, between the man's legs and above the shadow region are also included. The extraction is also not as smooth as would be desired. The *Hall Monitor* sequence has a very cluttered background which results in portions of the background being included. Nevertheless, by comparing Figure 5-5 (a) and Figure 5-5 (c) it is apparent that a more complete and semantically meaningful extraction of the object is obtained with the algorithm used. Furthermore the method is performed at a low computational cost.

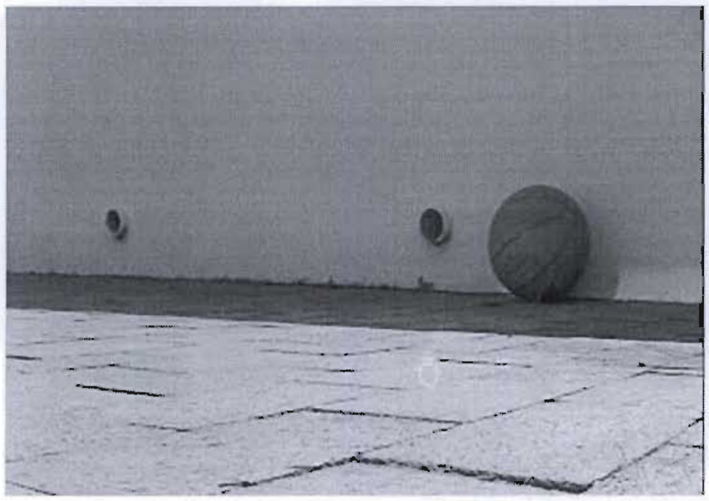


Figure 5-6: Sample frame to illustrate sequence, ball rolling

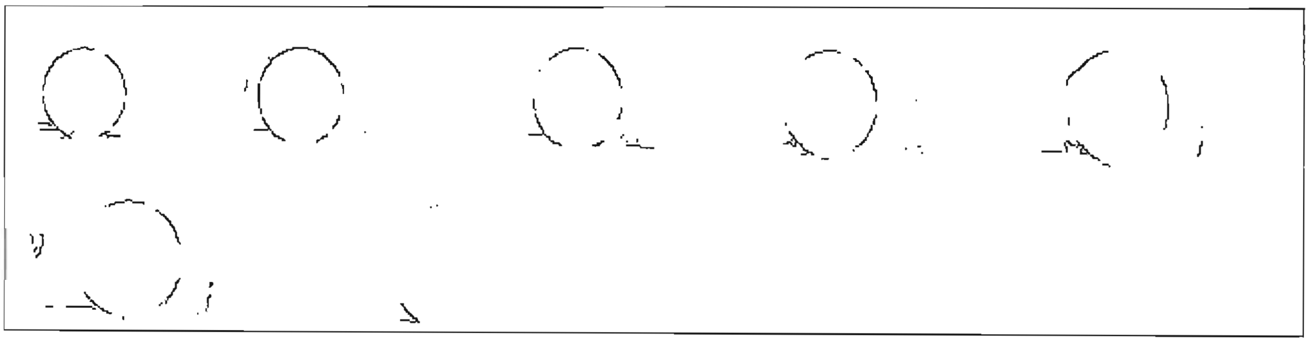


Figure 5-7: Final models as generated for a selection of frames from sequence, ball rolling

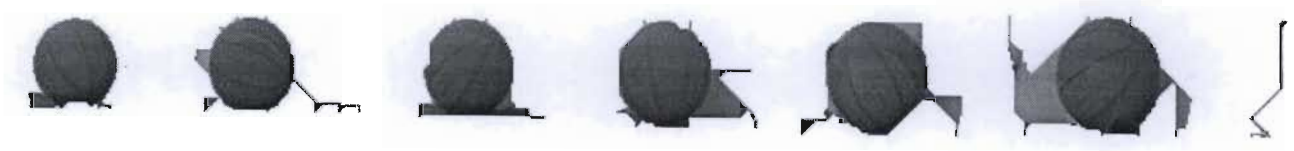


Figure 5-8: Corresponding extracted objects for the same sequence.

Figure 5-6 illustrates a sample frame from the sequence ball rolling which was used in the previous chapter. The video scene is that of a ball rolling across the view of the stationary camera. Figure 5-7 shows the final model for a series of frames and Figure 5-8 the corresponding extracted objects. The results show inherent problems of the algorithm regarding shadows, revealed background and objects moving out of the view of the camera object. The first four frames show the effect of shadows on the object extraction. Since the shadows are not compensated for they are considered to be part of the object and hence the parts of the background on which the shadows fall are also included as part of the object. In Chien et al [Chien, 2002], a morphological gradient filter was applied to reduce the effect of shadows. This approach does prove effective when the background is smooth as the shadow results in a gradual change of the luminance values of the background. Hence they are "removed" by the gradient filter. The approach however fails in the presence of a textured or cluttered background as the effect of changing illumination is more prominent and comparable to the change effect of motion itself and hence not effectively "removed" by the gradient filter. The fifth and sixth frames show the distortion of the object by uncovered background objects. The uncovered object in question is the gutter outlet to the left of the ball. Without further *a priori* knowledge regarding the contour of the object, one cannot confidently exclude or remove the uncovered background from the extracted object. The last frame in the sequence shows the failure of the object extraction as the object moves out of the view of the camera. Insufficient contour information is available in the scene for accurate object extraction.

The problems highlighted in the previous paragraph provide motivation for more *a priori* information for reliable semantic object extraction. For example, if it was known *a priori* that the moving object was a rigid object and a subset of possible shapes were available, then a best fit algorithm would yield more accurate object extraction than the results presented.

Similar problems were observed in other sequences tested.

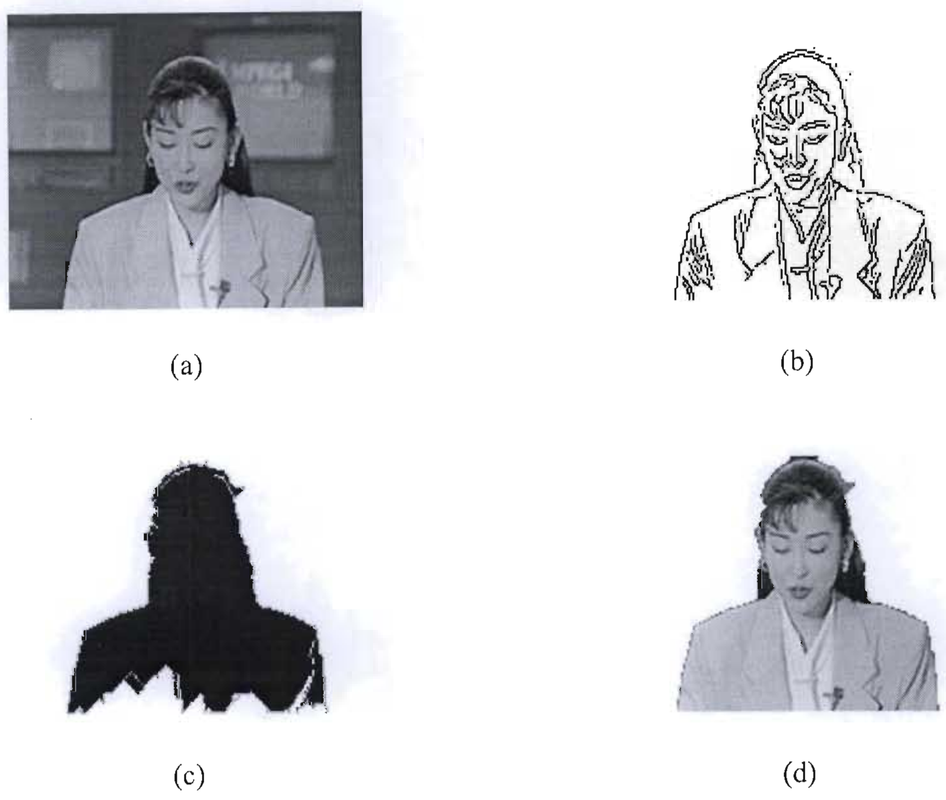


Figure 5-9: (a) Sample frame of MPEG 4 Sequence, *Akiyo*. (b) Final model of moving object. (c) Closed contour of final object model. (d) Extracted object

The results for a sample frame from the sequence, *Akiyo* are presented in Figure 5-9. As may be observed from the extracted object in Figure 5-9(d), the bottom boundary of the frame forms the bottommost boundary of the object. Hence, "hollowing" out the bottom boundary would provide worse results instead of better. To overcome this problem, the choice to remove these pixels was amended to be a user input parameter as it depends on scene content. The *Akiyo* sequence is not affected by shadows and the background is neither textured nor very cluttered. The movement of the object is also within a limited range with very little background being uncovered. Hence, the results for the object extraction of sequence, *Akiyo* is good. A sample of the extracted objects from the sequence is presented in Figure 5-10. Small portions of the background above her head are included as this area is continuously occluded and revealed as she moves her head in the sequence.

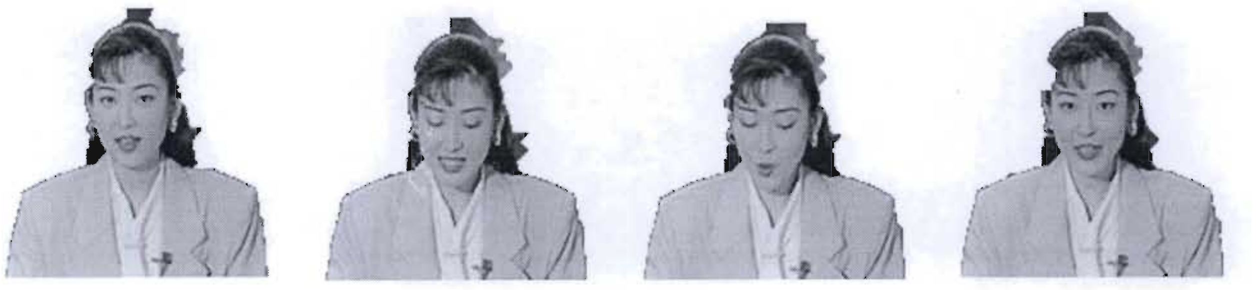


Figure 5-10: Sample of extracted objects from MPEG 4 video test sequence, *Akiyo*

Figure 5-11 shows the final model for a sample of frames from the sequence, *Hall Monitor* and the corresponding extracted objects are presented in Figure 5-12. In this sequence the effect of a cluttered background is more prominent. As the man moves down the corridor, the background becomes covered and then uncovered. While the temporal filter discussed in the previous chapter does reduce the amount of background clutter, the portions of the uncovered background in the immediate vicinity of the moving object are extracted as part of the object. These portions of the background may only be removed with further knowledge of the object of interest.

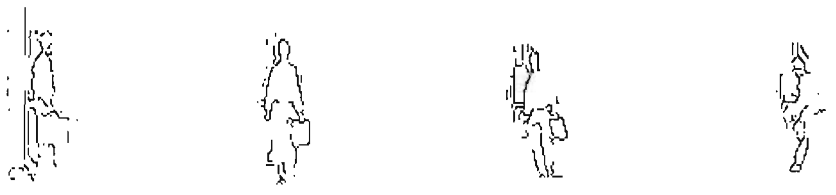


Figure 5-11: Final models as generated for a selection of frames from MPEG 4 video test sequence, *Hall Monitor*

Figure 5-12: Corresponding extracted objects for the same sequence

In this chapter a novel, intuitive method of tracing the outer contour of the object as obtained from the tracked model and compensating for breaks in the contour was presented. After contour closing, the object extraction becomes trivial and the results of object extraction for different sequences were presented. One may conclude from testing that the problem of object extraction is certainly an ill-posed one. While good results are obtained under constrained conditions such as sequences where the object casts no shadows on the background and minimal occlusion and uncovering of the background occurs, the majority of sequences yield unsatisfactory results. Gradient filters have been used in the literature to eliminate the effect of shadows, however their effectiveness is limited. It is difficult to envision an object extraction algorithm that is able to discern the exact contour of the object of interest without further knowledge of the object's features. Such functionality would require a shift from object detection to object recognition. The human visual system performs such segmentation by effectively having an enormous, high speed database of semantically meaningful objects and it relates low level features like motion, shape and texture to the objects in the database. A possible way forward at this point, would be to constrain the scope of the problem to be application specific and then form a database of possible objects that may be detected and perform object extraction as the human visual system does, such an approach however, would be a divergence from the original aim of the research.

CHAPTER 6: CONCLUSION

6.1 Summary of work

The vision that inspired this research effort was a fully automated video object extraction tool for generic image sequences that may be used to complement the MPEG4 standard. This research effort embraced this vision by first reviewing the state of the art in the field of video segmentation.

A key deduction that was made was that the very nature of the segmentation problem is ill-posed and hence a comprehensive theory of segmentation is difficult to envisage. The idea of semantically meaningful is very context dependent and hence the video object of interest is dependent on the application or set of applications that the system will be used in. This allows a hierarchy of segmentation approaches to be formed and the class of approaches that may be applied for a specific application would depend on the video object of interest.

This realisation already starts forming cracks in the vision of an automatic video object extraction tool for generic image sequences but also leads to another key deduction; an effective segmentation tool for generic images requires *a priori* information. This may be by having prior knowledge of the nature of the problem or by user inputs via a set of parameters that need to be tuned prior to execution of the algorithm. With the vision of a segmentation tool for generic image sequences, the only prior knowledge that one has available is that by the very nature of video; there will be motion of objects in the scene. Hence motion may be used as a criterion for segmentation.

We selected an existing approach to motion-based segmentation for video object generation. The approach, proposed by Meier and Ngan [Meier, 1998], was selected because it applied change detection instead of a dense optical field approach to manipulating motion information. This was advantageous because change detection approaches are considerably less computationally intensive and the approach therefore had the potential to be executed in real time. It also incorporated mathematical morphology which was observed from the literature to be well suited to object segmentation because it may be used to efficiently compute geometric feature information such as size, shape and connectivity.

The aim of the approach selected was to extract arbitrarily shaped moving objects from a video sequence by using 2D image information and motion as criteria for extraction. The following assumptions were made to simplify and constrain the focus of the approach.

The first assumption or constraint is that the only movement in the video sequence is assumed to be that of the video object of interest. This implies that the background and camera are stationary.

The second assumption is that there is no occlusion of the object either by the background or other objects. By using this assumption, the algorithm was developed and tested for a single object but may be extended to multiple objects.

The third constraint is that there is constant illumination and the fourth that there is only a single camera.

The approach applies pattern recognition, morphological transforms and object tracking principles to moving object segmentation. The algorithm was structurally decomposed into three phases. The first phase, model initialisation, uses both temporal and spatial information to obtain a model of the moving components. The next phase applies Hausdorff object matching to tracking and updating of the object as it moves and evolves in the video sequence and the third phase is the extraction of the object from each frame of the sequence.

The main contribution of this developmental effort, to that by Meier and Ngan, was the implementation of an adaptive global thresholding algorithm in the change detection stage of the model initialisation phase. In Meier and Ngan, the user manually selects the threshold. The automated algorithm is computationally efficient and performs reliably for both noisy and low noise signals, slow and fast motion and different object sizes. The trade-off however, is that the window size must be selected by the user. This however is dependent on the frame resolution and is not content dependent. Hence, for a particular video sequence the optimal value for this parameter would be constant, unlike the optimal threshold, which changes for each pair of difference frames. The main shortfall of the algorithm however is in cases where there is no motion between consecutive frames as the threshold detection converges on noise. This might yield poorer results than manual thresholding.

The system currently developed to perform the segmentation was not optimised for performance and does not run in real time. On a 3GHz, Pentium 4 processor with 256MB of RAM, it processes a sub-QCIF video clip at approximately 2 frames per second. The main bottleneck in the processing is the Hausdorff matching. When this component of the algorithm is removed, the output is processed at 10 frames per second. With optimised coding and a state of the art computer, increasing the frame rate by an order of magnitude would be a realisable goal.

With regard to subjective quality of segmented objects, the algorithm performs satisfactorily in comparison to most segmentation schemes in its class. However, it should be noted that further investigations subsequent to this research effort have yielded better results. In Mirza and Murugas [Mirz, 2003] a moving object segmentation approach based on the work by Chien et al [Chien, 2002] is investigated. As in this research effort, the assumption of a stationary camera and background is also made and based on this assumption the background instead of the moving foreground is detected and registered. The moving object is then the difference of the actual frame and the registered background frame. This approach produces better results, as an unchanging background is more suited to detection than a moving, changing foreground object. Furthermore, it processes objects as blobs instead of contours and therefore the ill-posed problem of completing the contour is avoided. The main advantage, however, of the approach by Meier and Ngan over the approach by Chien, is that their approach allows for the object to stop moving and still be tracked, while in the latter approach, if the object stops for several frames, it will eventually be classified as a static background pixel.

6.2 Review of possible enhancements to current investigated segmentation approach

In chapter 5, it was noted that the investigated approach to segmentation does not compensate for the effect of shadows or "holes" within the object. Hence portions of the background may be extracted as part of the segmented object. To extract only the semantic object of interest, more *a priori* information is necessary to improve object extraction. This is possible if the application is made more specific or the rules governing the segmentation made more rigid; for example, if the extraction is tailored specifically for video surveillance applications then the classes of objects and the description of each class of interest may be more rigidly defined and hence better segmentation possible. Further work in this direction, may require a shift towards artificial intelligence topics such as theory of semantics and classifiers.

The current approach was constrained to monochrome images. An alternative approach to compensating for the effect of shadows might be to use colour images. The spatial and temporal information increases, as tones with similar values in grayscale may have very distinctive values in a colour scale and this would facilitate more accurate contour extraction. The tradeoff to working with colour however, is the increase in input data and processing, hence further research may be conducted into incorporating colour to improve segmentation results but should include an evaluation of the impact colour images have on the processing time

Alternatively, the research may be streamlined towards object based coding, and the focus would then shift towards describing the contour, motion and texture of the extracted object in a coding framework such as MPEG 4. The MPEG 4 concept of video objects requires more accurate semantic object extraction than has been achieved with the current segmentation approach and would require further research as described in the above two paragraphs. However, in the context of the current segmentation approach, further investigations may be conducted into efficiently encoding the contour, motion and texture of the extracted object and into motion analysis of the object as it is tracked along its trajectory.

Further work may also be conducted into easing the constraints imposed on the system. If the constraint on occlusion were maintained, then minor modifications to the current approach would allow the extraction of multiple objects. However, it would be more practical to expand the scope of the work to segment out multiple objects that may occlude one another. This may be achieved by analysing the trajectory obtained from Hausdorff tracking and maintaining a memory of the texture and contour information of each object, so that each may be discerned when many objects occlude one another. The constraint on a stationary background may also be relaxed, by incorporating a global motion compensation algorithm as a preprocessing step to the current approach. The illumination constraint holds for a wide variety of video sequences, hence it is not a critical constraint that needs to be eased.

6.3 Context of research

As mentioned in section 6.1, for the specific case of a static background, the segmentation approach investigated is outperformed by the approach by Chien *et al* [Chien, 2002]. The latter approach also has the advantage that it can handle multiple objects, however when multiple objects occlude, they are extracted as a single object. Hence, in the context of realising our vision of a fully automated approach to video object generation, the approach by Chien provides a more solid foundation for further work.

The next step in the research would be to handle the occlusion problem. A possible way forward in this respect, would be, to incorporate a tracking mechanism, possibly the Hausdorff tracker, and as suggested in section 6.2, maintain a history of the trajectory, shape and texture information to discern each object.

In parallel with this research, it would be necessary to investigate relaxing the constraint of a static background. To compensate for camera panning, it may be possible to use a global motion compensation strategy, as suggested in section 6.2, as a pre-processing step to the static background segmentation algorithm. However, in addition to this it would be necessary to investigate motion segmentation schemes that directly address the moving camera problem and compare the results.

If the direct approach offers considerably better results, then it might be necessary to incorporate different segmentation strategies within a single framework and the choice of segmentation algorithm would depend on a user input indicating whether the camera moves or not. In a subsequent publication, Meier and Ngan [Meier, 1999], proposed a similar combination of segmentation strategies approach. In their work, however, the choice was based on the presence of camera motion and the type of motion, ie fast or slow relative to the frame rate.

The next possible extension to the problem is to shift from a monochrome to a colour framework. The inclusion of colour would introduce its own peculiar set of complexities as working with colour implies more spatial and temporal information but at the expense of a higher computational burden. The use of colour has not been investigated at all at this point and it is thus not possible to predict the complexities involved. As suggested in section 6.2, the incorporation of colour might also help resolve the problem with shadows.

Up to this point the proposals for a video object generation tools have been related to lower level processing. To fully realise a semantic video object generation system, with accurate shape extraction, it might be necessary, to mimic the human visual system and use an object database to recognise and more accurately extract the video object. This would shift the research from the field of video processing to that of semantics, classifiers and possibly artificial intelligence. The MPEG7 standard is an emerging standard that addresses issues of video database indexing and retrieval and it would be viable to investigate incorporating the toolsets from the standard in such a strategy.

It is clear from an analysis of the path ahead that realising the vision of an automatic video object generation tool, would probably culminate in a complex, computationally intensive system. While such a system may not be feasible at present, the rapid technological advancements in the computing industry might certainly make such a system viable within the next five years.

REFERENCES

[Aach, 1993]	Aach, T., Kaup, A., Mester, R., October 1993, "Change detection in image sequences using Gibbs random fields: a Bayesian approach", *IEEE International Workshop on Intelligent Signal Processing and Communications Systems*, Japan, pp 56-61
[Adiv, 1985]	Adiv, G., 1985, "Determining three-dimensional motion and structure from optical flow generated by several moving objects", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 7, No. 4 pp.384—401
[Besa, 1986]	Besag, J., 1986, "On the statistical analysis of dirty pictures", *Journal of the Royal Statistical Society*, Vol. 8, No. 3
[Borg, 1986]	Borgefors, G., 1986, "Distance transformations in digital images," *Computer vision, graphics and image processing*, Vol. 34, pp. 344-371
[Bout, 1990]	Bouthemy, P., Lalande P., 1990, "Detection and tracking of moving objects based on a statistical regularization method in space and time." In O. Faugeras, Editor, *Proc. 1st European Conf. on Computer Vision*, pages 307--311. Springer-Verlag
[Bout, 1991]	Bouthemy P., François, E., 1991, "Multiframe-based identification of mobile components of a scene with a moving camera.", *Proc. Conf. Computer Vision and Pattern Recognition*
[Bout, 1993]	Bouthemy, P., Francois, E., 1993, "Motion Segmentation And Qualitative Dynamic Scene Analysis From An Image Sequence", *International Journal in Computer Vision*, Vol. 10, No. 2, pp 157-182
[Canny, 1986]	Canny J., November 1986, "A Computational Approach to Edge Detection", *IEEE Trans. Pattern Anal. & Machine Intelligence*, Vol. PAMI 8(6), pp. 679-698
[Cast, 1998]	Castagno, R., Ebrahimi, T., Kunt, M., September 1998, "Video Segmentation Based on Multiple Features for Interactive Multimedia Applications", *IEEE Trans. On Circuits and Systems for Video Technology*, Vol 8, No 5, pp 562-571

[Cava, 2002] Cavallaro, A., 2002, *From Visual Information to Knowledge: Semantic Video Object Segmentation, Tracking and Description.* Ph.D thesis, Ecole Polytechnique Federale de Lausanne.

[Chal, 1996] Chalom, E., Bove, M., September 1996, "Segmentation of an image sequence using multidimensional image attributes", in *Proceedings of the IEEE International Conference on Image Processing, ICIP-1996*, Lausanne

[Chan, 2001] Chan, T. F., Vese, L. A., February 2001, "Active Contours Without Edges", *IEEE Transactions on Image Processing*, Vol. 10, No. 2, pp 266-277

[Chang, 1993] Chang, M.M., Sezan, M.I., Tekalp A. M., 1993, "Motion-field segmentation using an adaptive MAP criterion," in *IEEE Int. Conf. Acoust., Speech, Signal Processing, ICASSP '93*, vol. 5, pp. 33-36.

[Chang, 1998] Chang, M.M., Sezan, M.I., Tekalp A. M., 1997, "Simultaneous Motion Estimation and Segmentation," in *IEEE Transactions on Image Processing*, vol. 6, no. 9, pp. 1326-1333.

[Chien, 2002] Chien, S., Ma S., Chen, L., July 2002, "Efficient moving object segmentation algorithm using background registration technique", *IEEE Trans. On Circuits and Systems for Video Technology*, Vol 12, No 7, pp 577-586

[Choi, 1997] Choi, J.G., Lee, S., Kim, S., April 1997, "Spatio-temporal video segmentation using a joint similarity measure", *IEEE Trans. On Circuits and Systems for Video Technology*, Vol 7, No 2, pp 279-285

[Chou, 1990] Chou, P., Brown, C., 1990, "The theory and practice of Bayesian image labeling", *IJCV*, Vol. 4, No. 3, pp. 185–210

[Fore, 1999] Foresti. G., December 1999, "Object recognition and tracking for remote video surveillance", *IEEE Transactions on Cir and sys for Video Technology*, Vol 9, No. 7, pp 1045-1062

[Fors, 2002] Forsyth, D.A., Ponce, J., *Computer vision: a modern approach*, Prentice Hall

[Garc, 1999] Garcia, C., Tziritas, G., September 1999, "Face Detection Using Quantized Skin Color Regions Merging and Wavelet Packet Analysis", *IEEE Transactions on Multimedia*, Vol. MM-1, No. 3, pp. 264-277.

[Gonz, 1992] Gonzalez, R. C., Wood R. E., 1992, *Digital Image Processing*, Addison Wesley

[Grin, 2001] Grinias, I., Mavrikakis, Y., Tziritas, G., October 2001,"Region growing color image segmentation applied to face detection", International Workshop on Very Low Bitrate Video Coding (VLBV 01), Athens, Greece.

[Gu, 1998] Gu, C., Lee, M., September 1998, "Semiautomatic Segmentation and Tracking of Semantic Video Objects", *IEEE Trans. On Circuits and Systems for Video Technology*, Vol 8, No 5, pp 572-584

[Hara, 1987] Haralick, R.M. *et al*, July 1987, "Image Analysis Using Mathematical Morphology", *IEEE Trans. Pattern Anal. Machine Intelligence*, Vol. 9, pp. 532–550

[Heath, 1997] Heath M., Sarkar, S., Sanocki, T., Bowyer, K.W., Dec. 1997 , "A robust visual method for assessing the relative performance of edge detection algorithms", in *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 19, no. 12, pp. 1338—1359.

[Horn, 1981] Horn, B., Shunck, B., 1981, "Determining Optical Flow", *Artificial Intelligence* 17, pp. 185-203

[Hutt, 1990] Huttenlocher D. P., Kedem K., 1990, "Computing the minimum Hausdorff distance for point sets under translation", *Proc. of 6th Annual ACM Symp. on Comp. Geom. (SCG'90, Berkeley, CA)*, pp. 340-349.

[Hutt, 1993a] Huttenlocher D.P, Klauderman G.A, Rucklidge W.J., September 1993, "Comparing images using the Hausdorff distance," *IEEE Trans. Pattern Anal. Machine Intelligence*, Vol. 15, pp. 850–863.

[Hutt, 1993b] Huttenlocher D.P., Noh J. J., Rucklidge W. J., May 1993, "Tracking nonrigid objects in complex scenes," in 4th Int. Conf. Comput. Vision, Berlin,Germany, pp. 93–10

[Jeso,2001] Jesorsky, O., Kirchberg, K., Frischholz, R., June 2001,"Robust face detection using the Hausdorff Distance", *Audio- and Video-Based Biometric Person Authentication, 3rd International Conference, AVBPA 2001*, Halmstad, Sweden, Proceedings, Springer (ISBN 3-540-42216-1).

[Kalm, 1960] Kalman, R. E., 1960, "A New Approach to Linear Filtering and Prediction Problems", *Transaction of the ASME—Journal of Basic Engineering*, 82(Series D), pp. 35-45.

[Krug, 1998] Kruger, S., 1998, *Motion analysis and estimation using multiresolution affine models*, PhD Thesis, University of Bristol

[Li, 1995] Li, S., Z., 1995, *Markov Random Field Modeling in Computer Vision*, Springer-Verlag

[Lin, 2002] Lin, C. *et al*, Dec 2002, "Real time object extraction and tracking with an active camera using image mosaics" in *Proc. IEEE Int. Workshop on Multimedia Signal Processing*, Virgin Islands, USA.

[Marco, 1999] Marcotegui, B., *et al*, 1999, "A visual object generation tool allowing friendly user interaction", in *Proceedings of International Conference on Image Processing*, pp 391-395, 1999

[Mari, 2000] Marichal, X., Villegas, P., 2000, "Objective evaluation of segmentation masks in video sequences" in *Proceedings of X European Signal Processing Conference*, pp. 2193-2196.

[Mech, 1998] Mech, R., Wollborn, M., April 1998, "A Noise Robust Method for 2D Shape Estimation of Moving Objects in Video Sequences Considering a Moving Camera", *Signal Processing*, Vol. 66, No. 2, pp. 203-217

[Meier, 1998] Meier, T., Ngan, N.K., Sept 1998, "Automatic Segmentation of Moving Objects for Video Object Plane Generation", *IEEE Trans. On Circuits and Systems for Video Technology*, Vol 8, No 5, pp 525-538

[Meier, 1999] Meier, T., Ngan, N.K., Nov 1999, "Video Segmentation for Content Based Coding", *IEEE Trans. On Circuits and Systems for Video Technology*, Vol 9, No 8, pp 1190-1203

[Mich, 2000] Micheletti, J.D., Wurpts, M.J., April 2000, "Applying Chroma-Keying Techniques in a Virtual Environment", *Proceedings SPIE Aerospace*, http://www.tss.swri.edu/pub/2000AEROSENSE_HMD.htm

[Mirz, 2003] Mirza, M, Murugas, T, November 2003, "Automatic moving object segmentation using adaptive thresholding and static background detection", *Proceedings Annual Symposium of the Pattern Recognition Association of South Africa*, pp. 113-118

[Muru, 2002] Murugas, T., Tapamo, J.R., Peplow, R., 2002, "Extraction of an Object Model for Video Tracking", *Proceedings IEEE Africon 2002*, Vol 2, pp.317-322

[Musm, 1989] Musmann, H. G., Hotter, M., Osterman, J., Oct 1989, "Object-oriented analysis-synthesis coding of moving objects", *Signal Processing: Image Communication*, Vol. 1, pp. 117-38

[Musm, 1995] Musmann H.G., November 1995, "A layered coding system for very low bit rate video coding", *Signal Processing: Image Communication*, Vol. 7, No.4-6, pp.267-278.

[Neri, 1998] Neri et al, April 1998, "Automatic Moving Object and Background Separation", *Signal Processing*, Vol. 66, No. 2, pp. 219-232

[Owen, 1990] Owen, F., 1990, *Statistics*, Pitman, London

[Pavli, 1980] Pavlidis, T., 1980, "A Thinning Algorithm For Discrete Binary Images", *Computer Graphics and Image Processing*, Vol. 13, pp. 142-157

[Rosen, 1996] Rosenfeld, A., Pfaltz J.L., October 1996, "Sequential operations in digital picture processing", *J. ACM*, Vol. 13, Oct 1966, No. 4, pp. 471-494

[Rosin, 1997]	Rosin, P., 1997, "Thresholding for Change Detection", *Electronic Proceedings: Eighth British Machine Vision Conference*, http://www.bmva.ac.uk/bmvc/1997/papers/007/thresh2.html
[Salem, 1994]	Salembier, P., Pardas, M., September 1994, "Hierarchical Morphological Segmentation for Image Sequence Coding", *IEEE Transactions on Image Processing*, Vol.3, pp 639-651
[Salem, 1995]	Salembier, P., Serra, J., August 1995, "Flat zones Filtering, Connected Operators and Filters by Reconstruction", *IEEE Transactions on Image Processing*, Vol.4, pp 1153-1160
[Salem, 1996]	Salembier *et al*, June 1996, "Morphological operators for image and video compression", *IEEE Transactions on Image Processing*, Vol. 5, pp 881-898
[Schoe, 2001]	Schoepflin *et al*, 2001, "Video Object Tracking With a Sequential Hierarchy of Template Deformations", *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 11, No. 11, pp 1171-1182
[Sera, 1982]	Sera, J., 1982, *Image analysis and mathematical morphology*, Academic Press
[Shari, 2002]	Sharifi, M., Fathy, M., Mahmoudi, M.T., 2002, "A Classified and Comparative Study of Edge Detection Algorithms", *Proceedings IEEE ITCC Conference*, Nevada
[Shih, 1995]	Shih F., Pu C., 1995, "A Skeletonization Algorithm By Maxima Tracking On Euclidean Distance Transform ", *Elsevier Science Ltd: Pattern Recognition*, Vol. 28, No. 3, 1995, pp331-341
[Siko, 1997]	Sikora T., Feb 1997, "MPEG4 Video Verification Model", *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 7, pp 19-31
[Spree, 1992]	Spreeuwers, L.J, van der Heijden, F., 1992, " An EdgeDetector Evaluation Method Based On Average Risk", *11th IAPR International Conference on pattern recognition*, pages 771-774, The Hague, Netherlands

[Stil, 1999] Stiller, C., Konrad, J., July 1999, "Estimating motion in image sequences- a tutorial on modelling and computation of 2D motion", *IEEE Signal Processing Magazine*, pp. 70-91

[Truc, 1998] Trucco, E., Verri, A., 1998, *Introductory Techniques For Computer Vision*, Prentice Hall, USA

[Vinc, 1991] Vincent, L., Soille, P., June 1991, "Watersheds in digital spaces: An efficient algorithm based on immersion simulations", *IEEE Transactions on Pattern Recognition and Machine Intelligence*, Vol.13, pp 583-598

[Wang, 1994] Wang J.Y.A., Adelson E.H., February 1994, "Spatio-temporal Segmentation of Video Data", *Proceedings of SPIE on Image and Video Processing II*, Vol. 2182, pp. 120-13

APPENDIX A: TABLE OF USER INPUT PARAMETERS FOR SEGMENTATION AND TRACKING ALGORITHM

Section	Parameter	Chapter
Resolution	Number of Rows	
	Number of Columns	
Frame Information	Start frame index	
	Number of frames to process	
Gaussian Filtering	Mask width	3
Change Detection Threshold	Window size	3
	Number of shifts	
Connected Component Labelling	MCC threshold	3
Canny Edge Detection	High threshold	3
	Low threshold	
	Variance	
Find Model in Edge Map	Proximity threshold	3
Background Filtering	Stationery pixel threshold	4
Hausdorff tracking	Rank	4
	Max translate distance	
	Max Hausdorff distance	
Object Extraction	Hollow out top	5
	Hollow out bottom	