DUBLIN CITY UNIVERSITY

SCHOOL OF ELECTRONIC ENGINEERING

Object-Based Video Representations: Shape Compression and Object Segmentation

A thesis submitted as a requirement for the degree of Ph.D in Electronic Engineering

September 1998

Supervisor: Dr. Noel Murphy

Noel Brady B.Eng

Declaration

I hereby certify that this material, which I now submit for assessment on the programme of study leading to the award of Ph.D in Electronic Engineering, is entirely my own work and has not been taken from the work of others save and to the extent that such work has been cited and acknowledged within the text of my work.

Signed: Noel Brady **ID No.:** 91700035

Date: 23-9-98

ACKNOWLEDGEMENTS

I wish to extend my kind gratitude to the many people and organisations who have worked with me in completing this thesis. These include my academic supervisors, Dr. Noel Murphy and Dr. Sean 'Pele' Marlow and my close working colleagues in the Video Coding Group, Mr. Liam Ward and Mr. Noel Edward 'Baby-face' O' Connor, all of whom have been supportive from both the technical and social standpoint. I would also like to thank Dr. Thomas Curran for providing, through Teltec Ireland, enough monetary support to sustain me in the last seven years. Much inspiration has been gained by working with other international researchers and no small thanks must be extended to the hard-working members of the MoMuSys project and also, to the many members of the MPEG committee which I have had the pleasure to meet. In particular, many thanks go to Mr. Frank Bossen, whose contributions on the issue of shape compression were essential. Last but not least, it must be said that I could not have mustered up the determination to complete this thesis without the constant encouragement of Ms. Fiona Harte, a person who knows very little about video compression but who, more than most, understood, and even nurtured, my inner need to be called 'Doctor'.

TABLE OF CONTENTS

Object-based Video Representations: Shape Compression and Object Segmentation

Noel J. Brady

ABSTRACT

Object-based video representations are considered to be useful for easing the process of multimedia content production and enhancing user interactivity in multimedia productions. Object-based video presents several new technical challenges, however.

Firstly, as with conventional video representations, compression of the video data is a requirement. For object-based representations, it is necessary to compress the shape of each video object as it moves in time. This amounts to the compression of moving binary images. This is achieved by the use of a technique called context-based arithmetic encoding. The technique is utilised by applying it to rectangular pixel blocks and as such it is consistent with the standard tools of video compression. The block-based application also facilitates well the exploitation of temporal redundancy in the sequence of binary shapes. For the first time, context-based arithmetic encoding is used in conjunction with motion compensation to provide inter-frame compression. The method, described in this thesis, has been thoroughly tested throughout the MPEG-4 core experiment process and due to favourable results, it has been adopted as part of the MPEG-4 video standard.

The second challenge lies in the acquisition of the video objects. Under normal conditions, a video sequence is captured as a sequence of frames and there is no inherent information about what objects are in the sequence, not to mention information relating to the shape of each object. Some means for segmenting semantic objects from general video sequences is required. For this purpose, several image analysis tools may be of help and in particular, it is believed that video object tracking algorithms will be important. A new tracking algorithm is developed based on piecewise polynomial motion representations and statistical estimation tools, e.g. the expectation-maximisation method and the minimum description length principle.

GLOSSARY

alpha channel	image channel specifying the degree of transparency for each pixel
arithmetic coding	efficient method of encoding data given the probability distribution of that data
ARQ	automatic repeat request: the ability of a receiver to detect a transmission error and to request the re-transmission of the corrupt data
BAB	binary alpha block: a block of pixels from a binary alpha channel
blue-screening	studio procedure through which a foreground object shape maybe recovered
CAE	context-based arithmetic encoding: efficient method for compressing binary image data
CD-ROM	Compact-Disc/Read Only Memory: a high density storage disk used for removable read-only memory on computer systems
composition	procedure by which a video programme is produced by putting together one or more visual objects in space and time
COST 211ter	a European collaborative group coordinating research on video coding for telecommunications
DCT	Discrete Cosine Transform: a reversible transform used to achieve redundancy removal in image compression systems
DECT	Digital European Cordless Telephone: a system supporting the use of mobile telephony within local areas
DPCM	Differential Pulse Coding Modulation: a coding method using prediction and finite precision quantisation
EM	Expectation-Maximisation: an iterative parameter estimation method for solving problems of incomplete data
GN	Gauss Newton: a simplification on the Newton method, used for function optimisation
gradient	a vector of first-order derivatives for multi-variable functions

GSM	Global System Mobile: a single system supporting mobile telephony on a worldwide basis
hessian	a matrix of second-order derivatives for multi-variable functions
ICM	iterative conditional modes: a method used to incorporate contextual models into the segmentation process
INTRA compression	the compression of an image by eliminating spatial redundancies within a single video frame
INTER compression	the compression of an image by eliminating spatial and temporal redundancies between video frames
ISDN	Integrated Services Digital Network: a public digital network intended to replace PSTN while also adding new and improved services
ISO	International Standards Organisation
ITU-T	International Telecommunications Union
JBIG	Joint Bilevel Image Group: an ISO/ITU-T group that designed a system for binary image compression
MAD	Mean Absolute Difference
MB	Macro-Block: a defined subset of an image on which compression tools are applied (a macro-block contains 16x16 pixels)
MDL	Minimum Description Length principle: a means of defining an optimisation criterion for model fitting problems where no bound on the complexity of the model is specified
ML	Maximum Likelihood estimation: a method of parameter estimation based on a choice of the most likely parameter values
motion compensation	a method used to achieve INTER compression by using the estimated inter-frame motion
MPEG	Motion Picture Experts Group: an ISO working group dealing with the standardisation of coding methods for multimedia data
MR	Modified Read: a simple coding method used in FAX systems
MRF	Markov Random Fields: a convenient means to impose local contextual constraints on a classification or segmentation

	procedure
MSE	Mean Squared Error
MUX	Abbreviation for multiplexor, a device used to combine several information channels into one
optical flow	a map of 2-D vectors representing the apparent pixel motion between two images of a sequence
outlier	an event which does not belong to any existing group
PDF	Probability Density Function
progressive representation	a representation which facilitates fast transmission of initially low quality pictures, with the quality building up over time as further transmitted data is used to enhance the picture.
PSTN	Public Services Telephone Network: the conventional analogue telephone network
QN	Quasi-Newton: a simplification on the Newton method, used for function optimisation
RGB colour-space	a system used for representing image colour whereby a pixel has a colour combining various degrees of red, green and blue
RLE	Run Length Encoding: a coding method based on assumption that a data sequence contains long consequtive runs of one or more data values
segmentation	the process of grouping image pixels according to some common characteristic
SIMOC	an early object-based video compression algorithm developed within COST 211ter
template	for a given pixel, the template is a definition of the other pixels in the local neighbourhood
UMTS	Universal Mobile TeleCommunications System: a system intended to unify mobile networking standards while providing higher bandwidth
VLC	Variable Length Coding: a coding method by which the more frequent events are represented with the shorter codes.
VM	Verification Model: a in-progress description of a coding

	method as it moves towards standardisation
VOP	Video Object Plane: the raw representation of a video object at any given time instant, any image consisting of Y,U,V and alpha channels
VQ	Vector Quantisation: compression method using tables to store common data combinations
watershed	a means of image segmentation using morphological methods
YUV colour-space	a colour space which is related to the RGB colour space by a linear transformation

1. INTRODUCTION

Digital video compression has been among the most popular research and development fields during the past twenty years. The result of this concentrated effort has been that a number of international standards have been published enabling video communication for various applications. These comprise the ITU-T H-series recommendations and the ISO/IEC MPEG standards. The compression methods published in ITU-T recommendation H.261 **[37]** have enabled video telephony and tele-conferencing at rates ranging from 64 Kbits/s to 2 Mbits/s. The ISO/IEC standard MPEG-1 **[50],[54]** has enabled compressed storage and playback of digital video from hard disks and CD-ROM devices, optimised for rates around 1.5 Mbits/s. These two standards are suitable for progressive video representations. For applications in the digital television domain, methods for interlaced video compression were required. MPEG-2 **[32],[55]** was built on top of MPEG-1 by adding compression tools for interlaced video, thus providing TV and studio quality video at rates between 2 Mbits/s and 16 Mbits/s. By 1994, most envisaged applications had been provided for by the published standards. Even so, the search for increased compression and quality continued. For all applications, network bandwidth was proving scarce and costly. Those networks that were widely available, did not provide an adequate vehicle for video communications with acceptable quality. Additionally, the complexity of the compression algorithms meant that digital video encoders and decoders were relatively expensive. For video telephony, there was a reliance on the deployment of ISDN for the provision of adequate bandwidth communication channels. The slow uptake of ISDN still proves to be a major obstacle to the widespread use of personal video terminals for telephony purposes. For TV applications, broadcasters welcomed any technology that could increase the number of programme channels being broadcast on existing links. Recently, the ITU-T published H.263 **[38]**. The purpose of this standard was to provide increased compression performance so that video applications could be enabled on low bit-rate/low quality channels. Specifically, the networks targeted were the existing PSTN network and the emerging digital mobile networks, e.g. GSM, DECT and UMTS. H.263 succeeded in achieving the same quality as H.261, but, with only half the bandwidth. Also, at this

time (in 1996), it was becoming clear that advances in integrated circuit design would soon result in faster general purpose processors capable of bringing video applications to the PC without additional hardware support. This was the situation when a new standardisation effort was launched by the ISO, i.e. MPEG-4.

MPEG-4 initially endeavoured to produce gains in compression efficiency, allied to the provision of new and improved functionality. After an initial round of competitive tests, based on new proposed technology, it became apparent that no considerable gains in compression could be foreseen. Some new techniques were producing 10-20 % improvements over H.263, but it was widely held that gains exceeding 200% would be required in order to justify yet another video standard. Despite this, within the MPEG-4 community the need was strongly felt for new and improved functionalities. Many supported the view that there was scope for much improved error resilient video representations which could better stand the test of highly error-prone network channels, e.g. PSTN and mobile channels. Others sought to provide new representations that were more amenable to editing and post production needs, for applications in multimedia authoring and TV/film production. Others still saw a need for introducing more user-interactivity into multimedia applications, thus merging the concepts of virtual worlds, animation and the more traditional visual media. MPEG-4, now almost completed, is underpinned by these new and improved functionalities, i.e.

- improved error resilience
- object-based video editing
- object-based interactivity.

This is not to say that MPEG-4 ignored the compression problem. Each new or improved functionality is provided, while still endeavouring to maximise compression efficiency. In tackling the object-based functionalities, compressed object-based video representations were naturally called for. The problems and challenges associated with object-based video representations form the subject of this thesis.

Firstly, it is important to define what exactly is meant by an object-based video representation. Traditionally, a video sequence is represented as a sequence of images, with each image represented by a rectangular grid of pixels and each pixel having an associated colour value. In the object-based paradigm, a video sequence consists of one or more video objects, where a video object is represented by a sequence of object images. An object image is, once again, represented by a rectangular grid of pixels. However, the difference is in the fact that each pixel not only has a colour value but also a so-called *alpha* value. This alpha value specifies the degree of transparency for each pixel. Therefore, the alpha component of the object image may be used to define the spatial support or shape of a 2-D object. A pixel with an alpha value of 255 is considered to be part of the object and a pixel with an alpha value of zero is considered to be outside the object. The presence of this alpha component enables the composition of several object images to form a composited image. The process of composition involves blending an object image onto the composited image. The blending is controlled by the alpha value specified at each pixel of the foreground object image. If a pixel has an alpha value of zero, then it is totally transparent and there is no change in the composited image. If the pixel has an alpha value of 255, then it is regarded as totally opaque and the composited image pixel takes the colour value of the object image pixel. In general, the composition at a pixel is specified as:

$$p = q + \left(1 - \frac{\alpha}{255}\right) p$$

where p is the value of the pixel in the composited image and q is the value of the pixel in the object image.

So, an object-based video representation differs from traditional representations in that it contains this additional alpha information used for composition and in the fact that the scene displayed on screen is really due the composition of one or more video objects rather than just a single rectangular image. Since each video object is represented independently of the other video objects, editing the displayed scene becomes relatively easy. This is seen as the single largest advantage of object-based video representations

since it allows great flexibility and simplicity in the creation of video content. Since video objects may be stored independently, it also simplifies the re-use of content. Given a database of video objects, a brand new production is possible, simply by being able to configure the composition process, answering questions like, what objects? ... in what positions? ... at what time? Without the object-based representation, compressed video sequences containing the various objects, would have to be decompressed and edited in raw uncompressed format to extract the objects of interest (segmentation). Then, all of these objects would be composited and re-compressed into a new video sequence. The processing power, storage and time requirements for such a procedure are inordinately large. Also, in the event that the same object is required for another production some months later, the processes of de-compression, segmentation, composition and re-compression must all be repeated again. In the light of these difficulties, object-based video representations are very attractive, avoiding the need for transcoding (i.e. decoding and re-encoding) and allowing for the re-use of segmented video objects. However, object-based representations have advantages outside the area of content production. They also present the opportunity for a consumer (viewing a programme or presentation) to interact with the content. For example, it is now feasible for a user to customise his/her TV screen, whereby two programme channels are displayed at once, along with several other dynamic information sources (e.g. sports results, stock prices). While this kind of display is common today, it is not possible for a user to select what content he/she wishes to be displayed. The content make-up is decided by the broadcaster and may not be altered by the consumer. Finally, with object-based representations, it is also becomes feasible to turn multimedia programmes into graphical user interfaces, where the programme contains clickable objects which are linked to some response. For example, the user may click on a bear in the zoo in order to receive further and more detailed information about it in textual format.

MPEG-4, with its object-based video representation, opens many new exciting avenues for the entertainment and multimedia industries and certainly there are many uses which have not been foreseen, as yet. Unfortunately, the above scenarios hide a few important details. Firstly, the flexibility of object-based representations comes with a penalty in terms of compression efficiency. Each video object, as mentioned, must have an alpha

component, i.e. extra information to compress. In order that the overhead associated with the alpha information does not become prohibitive, very efficient alpha compression techniques are required to allow high quality broadcasting and storage of the video objects. Secondly, there must be some means to acquire the alpha information in the first place. In studio productions, it is possible to design the set to allow chroma-keying (blue-screens). An object (always a different colour to the background screen) placed in front of the blue-screen can be easily sensed and the alpha information for the object can be recovered. However, outside of the studio, it is often not feasible to set up the same conditions. As such, some more general means must be provided for acquiring the alpha information from a given sequence. The difficulties associated with the acquisition of alpha information, often referred to as video segmentation, are a major hindrance to the widespread use of object-based video. This thesis addresses the problem of video object segmentation along with the problem of alpha compression (object shape coding).

Object-based video coding was initially investigated because it was believed that it could provide higher compression ratios than those techniques employed in the established standards. Chapter 2 studies some of the reasoning behind these beliefs and presents some of the earliest developments towards highly compressed representations. The review is brought up to date by detailing the motivations of MPEG-4 and briefly presenting the approach taken to compressing video objects.

As discussed, a major issue with object-based video relates to the need to compress the alpha information. Chapter 3 reviews some suitable shape coding techniques and charts the evolution of the solution taken by MPEG-4. The solution developed by the author, i.e. context-based arithmetic encoding (CAE) for the shape compression of moving objects, is introduced and results are presented that demonstrate its efficiency. The CAE approach developed here has been adopted as an essential component of the MPEG-4 standard.

Chapter 4 is devoted to motion estimation. Estimation methods for the family of polynomial motion models are investigated. This family includes models capable of

dealing with arbitrary rigid body motion. While these advanced motion models have been successfully employed in some video compression schemes, the main interest here is in their use for motion segmentation and object tracking purposes. They are later employed as the basis of the segmentation and tracking methods of chapters 5 and chapter 6, respectively. Chapter 4 includes a comparison of various motion estimation methods and the development of fast estimation algorithms for these useful motion models.

Chapter 5 introduces the problem of video segmentation. Through a number of examples, the numerous difficulties of automatic video segmentation are highlighted. A basic iterative framework for segmentation is presented and some promising estimation tools are described, e.g. the Expectation-Maximisation (EM) algorithm, the Minimum Description Length (MDL) principle and mathematical morphology. The chapter concludes that the segmentation of semantic objects cannot be achieved only by automatic means. Instead, a supervised user-controlled approach relying primarily upon automatic tracking algorithms is advocated.

Chapter 6 presents a framework for achieving reliable tracking of a moving video object. The framework is implemented using the previously described estimation tools (polynomial motion models, EM and MDL) and results are presented for a number of sequences. The results suggest that the segmentation of semantic video objects is highly feasible given a supervised approach and a powerful tracking algorithm.

2. OBJECT-BASED VIDEO CODING: PAST AND PRESENT

This chapter serves to highlight some of the initial motivations underlying the object-based video coding approach **[72]**. These relate entirely to the desire for more efficient video coders. The motivations are illustrated by pointing out important differences between object-based and the conventional block-based approach of H.261, MPEG-1 and so on. This chapter also serves to highlight the difficulties associated with object-based coding. It presents and analyses some of the earliest technical solutions to the problems of segmentation, shape coding and object-based motion estimation and compensation. Much of the following discussion refers to a very prominent object-based coding method called SIMOC. SIMOC stands for Simulation Model of COST 211ter. The COST 211ter document **[23]** fully describes the encoding system and was drafted during 1994 based on contributions from all the members of COST 211ter. Much of the technical content of this document originated from earlier work conducted by the University of Hannover **[36]**, **[63]**. SIMOC constitutes one of the earliest, most complete and well-specified object-based encoding algorithms. It comprises not only tools for coding shape, motion and texture but also a segmentation approach.

In order to bring the review up to date, the approach of MPEG-4 is briefly discussed and contrasted with that of SIMOC. At the time of writing, the MPEG-4 standard is approaching completion and it is most interesting to see how it relies, as much as possible, on the old established technology of block-based coding, while at the same time achieving object-based video representation.

2.1 Classifying Object-based Coders, Motivations and Characteristics

In the following, an object-based video coder is defined to be any coder which utilises shape information. There are mainly two types of object-based coders. The first type is the *compression-oriented* variety. Examples of compression-oriented object-based coders are SIMOC **[23]** and SESAME [22]. These algorithms have the primary

objective of outperforming the compression performance of their block-based counterparts by exploiting shape information (see the next sub-section for the associated motivations). They can be viewed as comprising an analysis system and a coding system. The analysis system is responsible for producing the shape information, i.e. the segmentation map of the source images. The coding system compresses the YUV data of the source images, using the segmentation map to define the regions on which the compressed representation is based. Usually, the coding system is also required to compress the shape information. The compression efficiency of such algorithms is highly reliant on the analysis system. A good analysis system produces segmentation maps that are highly amenable to efficient compression. Hence, it is natural and wise to make the analysis system an intrinsic part of the overall encoding algorithm rather than to decouple the two sub-systems. An additional characteristic of compression-oriented approaches is that the segmentations do not necessarily correspond with the semantic content of the source scene. The segmentations are intended to facilitate efficient compression, but they do not necessarily facilitate (semantic) content access. Furthermore, the methods of shape and texture coding are not specifically designed to support content access. Instead, these algorithms exploit all forms of redundancy (including inter-object redundancy). As such, access to a single object calls for a full decompression of the bitstream.

Semantic content access is the objective of the second type of object-based coder, i.e. the *content-oriented* coder. The dominant characteristic of this type of coder is related to the fact that the compressed video bitstream is composed of one or more compressed video objects and in particular, to the fact that the representation of a given video object is not dependent on the representation of any other video object. That is, inter-object redundancy is never exploited. As such, content access (e.g. cut-and-paste operations) is possible at the bitstream level without any decompression and re-compression. Just as before, the shape of the video object is provided by a segmentation procedure and compressed along with the YUV data. However, in the case of content-oriented algorithms, the segmentation is not treated as an integral part of the coding algorithm. Instead, the shape of the semantic object is correctly viewed as part of the source data and not something which the video encoder has any influence upon. The most obvious

example of a content-oriented video representation is the MPEG-4 video encoder **[57]**, see sub-section 2.4.

Now, that the distinction has been made between compression and content-oriented object-based video representations, more detail is given on the motivations for compression-oriented object-based coders.

2.2 Compression-oriented Motivations

Firstly, it is important to realise that SIMOC and other early object-based coders were designed with the sole aim of improved coding efficiency. That is, it was never really intended to provide content-based functionality. In the late eighties, work was already well advanced in the area of block-based video encoding. The ITU-T recommendation H.261 **[37]** had been published, enabling videotelephony and videoconferencing at rates of 64Kbits/s and upwards. It was noted, however, that the quality of these video codecs was not acceptable at low bit-rates, i.e. at 64Kbits/s and below. Specifically, low bit-rate video contained very disturbing block[1] and "mosquito"[2] artefacts. These artefacts, as illustrated in Figure 2-1, were entirely due to the block-based motion compensation and the block-based discrete cosine transform (DCT) employed.

[1] Block artefacts are characterised by large transitions in the image intensity. These transitions are found along the block borders and are a direct effect of coarse quantization of the DCT coefficients and/or the block-based motion compensation. In bad cases, the positions of blocks becomes evident in the decoded pictures.

[2] "Mosquito" artefacts are the result of a combination of inaccurate motion compensation and coarse DCT quantization. The effect is very much a temporal one where disturbing changes in the intensity take place over time. Mosquito effects are usually observed close to the edge of a moving object.

Figure 2-1: Artefacts in low bit-rate block-based video. On the left are high quality video frames and on the right are the same frames coded at very low bit-rate. Distortion takes the form of blur, blockiness (see top-right) and "mosquito" effects (see the noise close to contours of the ball and the train in the bottom-right image).

Block-based motion compensation, in particular, was considered to be sub-optimal since it was extremely limited in the kinds of motion it could synthesise. Protagonists of object-based representations ventured that representing video in terms of objects, instead of blocks, would lead to an elimination of these troublesome artefacts and higher coding gains as a result. Several object-based representations were proposed. The most natural of these was the 3-D object model. For these models, 3-D information about the scene was required. Considering the difficulty of 3-D analysis and the tight constraints placed on the implementation complexity of video codecs, 3-D object-based coding was immediately viewed as impractical given the technology of the time. As a result, most attention was devoted to investigating 2-D object-based coding. In this case, it was required to know only 2-D scene information, i.e. the 2-D shape and location of each projected 3-D object was needed. This was deemed to be more feasible since it was not so different from the block-based techniques. The outstanding difference was that the image would now be partitioned into arbitrarily shaped regions rather than fixed-sized blocks. Just as motion prediction and error encoding were used to code each block, so

motion prediction and error encoding would be used to code each arbitrarily shaped region. However, to faciliate 2-D object-based video coding, the shape of each region had to be coded and transmitted to the receiver. This was an undesirable overhead. Nevertheless, it was hoped that the envisaged advantages of coding in an object-based manner would justify this extra bit-rate. Mainly, it was believed that object-based motion compensation would significantly improve upon the block-based equivalent. By this (hopefully) improved motion compensation, it was proposed that a significant bit-rate saving would be made in encoding the prediction error *and* that this saving would exceed the cost of shape transmission. This was the initial justification for 2-D object-based video coding.

To fully understand this argument, it is necessary to further analyse the deficiencies in block-based motion compensation. Consider a scene containing a single moving object translating over a static background. See Figure 2-2 for a simple example. This scene contains two motions, a zero motion for the background and a non-zero motion for the foreground. In the most basic block-based coder, the motion within the scene is represented by a 2-D translational vector for each 16x16 block. Generally, the block-based partition will not be "in phase" with the moving foreground object. That is, many blocks will contain pixels from the static background, which have zero motion, and pixels from the foreground object, which are moving, as is the case in Figure 2-2. For those blocks that reside on the occluding edges of the moving object, it is not possible to represent the two motions present. The motion vectors for these edge blocks must, therefore, be a compromise between representing the zero motion and representing the non-zero motion and as such the prediction error for these blocks is expected to be large. Consequently, the prediction error encoding is costly and at low bit-rates the quantisation noise in the DCT domain is responsible for the aforementioned mosquito artefacts.

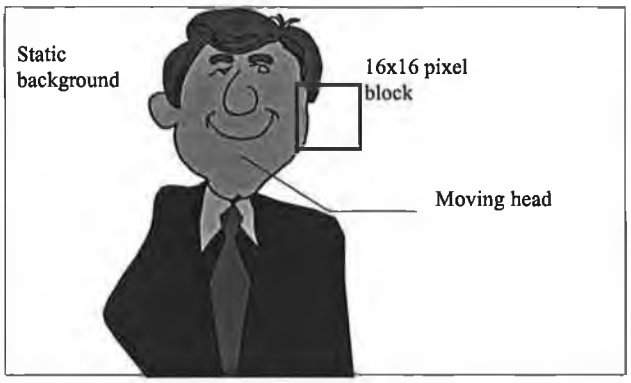


Figure 2-2: A simple illustration of the deficiency in the block-based model for motion compensation.

Now, imagine that the encoder is still block-based but, the shape and location of the moving object is known. Then, for these blocks which overlap a moving edge, two motions could be estimated and transmitted. The prediction error and associated coding cost would naturally be expected to be smaller. By expending bits to transmit the object shape, improved predictions are possible and prediction error encoding is less costly. This is one of the main philosophies underlying the object-based approach and it was a motivation for Orchard's block-based approach **[68]**, which attempted to find and segment image blocks for which a single motion vector was insufficient.

Another deficiency of the block-based approach is also illustrated in the above example. Despite the content of the scene, a block-based encoder must explicitly or implicitly transmit motion information for every block in the image. In the given example, there are only two motions in the scene. The background is static and the transmission of a single zero-valued motion vector would suffice. The foreground object has a translational motion and a single motion vector is all that is required here, even if the object covers many blocks. The transmission of the object shape means that a very compact motion description can be transmitted for each object. Object-based coders, therefore, have the potential to produce improved motion compensated predictions using very compact motion representations. Note that, by this argument, it should be expected that on higher resolution video, the object-based approach should show more

improvements over the block-based approach, since at high resolution, block-based algorithms must code more block motion vectors **[73]**.

In summary, with respect to the goals of improved coding efficiency, object-based coding is promising due to the advanced interframe predictions which are possible. Also, very compact object-based motion representations are possible. However, efficient shape codes are essential if a net coding gain is to result. While the arguments in favour of object-based video coding seem sound, no object-based coder yet exists which is generically applicable to scenes of all types *and* which is superior to the state-of-the-art in low bit-rate block-based coding, i.e. H.263. There are thought to be a number of reasons for this. Firstly, the prediction gains achieved with object-based representations are often minimal. This point has been illustrated by Wuyts *et al* [96] albeit in a very limited test scenario. Secondly, the use of compact motion codes is not generically applicable. Most objects in real scenes do not possess rigid motion. They may be constructed of many rigidly moving parts, e.g. articulated objects, but the automatic decomposition of an object into its rigidly moving parts is a very difficult task, akin to motion segmentation, and it results in additional shape information to be coded. Rigid body object models and associated coding systems have been implemented **[22]**, but it is unclear yet if a superior coding performance is attained. For example, the SESAME coding scheme of Corset *et al* [22] did not perform as well as expected in the November 1995 MPEG-4 subjective tests. Thirdly, shape encoding algorithms have not had the time to mature sufficiently and the algorithms used until now were perhaps not optimal. Fourthly, the verification of object-based coding schemes has always been hindered by the lack of suitable automatic segmentation methods. Many segmentation approaches such as change detection **[35]** almost totally ignore coding constraints, resulting in segmentations which are very unsuitable for coding purposes. Fifthly, many people working in the field of object-based coding have ignored much of the knowledge and experience gained in the area of block-based coding and have chosen instead to implement content specific coding schemes, which show improvement only on a small subset of sequence types.

The failure of past attempts in the area of object-based coding has been, in many ways, due to the immaturity of the technology. More recently, a technique has been described by Karczewicz *et al* **[43]** which has compared well with the latest block-based technology. This technique has been extensively tested over a large set of test material within the MPEG-4 experiment process. The current conclusion is that the small gains that can be attained come with an unacceptable cost in terms of implementation complexity. The question remains if the undoubted extra complexity associated with compression-oriented object-based schemes will ever be justified. The next sub-section describes one of the first and most prominent approaches to object-based coding, i.e. SIMOC, and many of the criticisms already spoken of above are illustrated by way of example.

2.3 The SIMOC Object-based Coder

Figure 2-3 shows a simplified block diagram of the SIMOC encoder. A segmentation algorithm based on change detection methods is used to produce a ternary segmentation mask identifying three classes of pixel, i.e. static background, moving area and background uncovered. Apart from the first frame of the source sequence, only the moving areas and the uncovered background areas are encoded. The shape of the moving area is coded using a vertex-based interframe coding method. The motion of the moving area is estimated based on a uniform grid of motion vectors. The grid vectors within the moving area are encoded and transmitted. Motion-compensated prediction of the moving area is performed and so-called model failure regions within it are identified. Model failure regions are those that are not well predicted by the motion compensation. These regions require the prediction error to be encoded and transmitted. Furthermore, the shapes of these regions are also required for transmission. Pixels comprising the uncovered background area are also encoded.

SIMOC is significant because it was really the first 2-D object-based video encoder and its principles contributed to the MPEG-4 initiative. Unfortunately, it was plagued by the fact that assumptions underlying the method imposed too many limitations on the nature of the sequences that it could efficiently code. The main criticism of SIMOC was that it was not a generic coder and coding performance was optimised over a very

unrepresentative class of sequences. As will be explained, several underlying assumptions are imposed upon source content and this has much to do with its lack of genericity. Despite this, some of the individual coding tools proposed within the framework of SIMOC are still considered to be very useful and many have been tested during the MPEG-4 standardisation process. The key areas of segmentation, shape coding, motion compensation and prediction error coding are now explored in more detail.

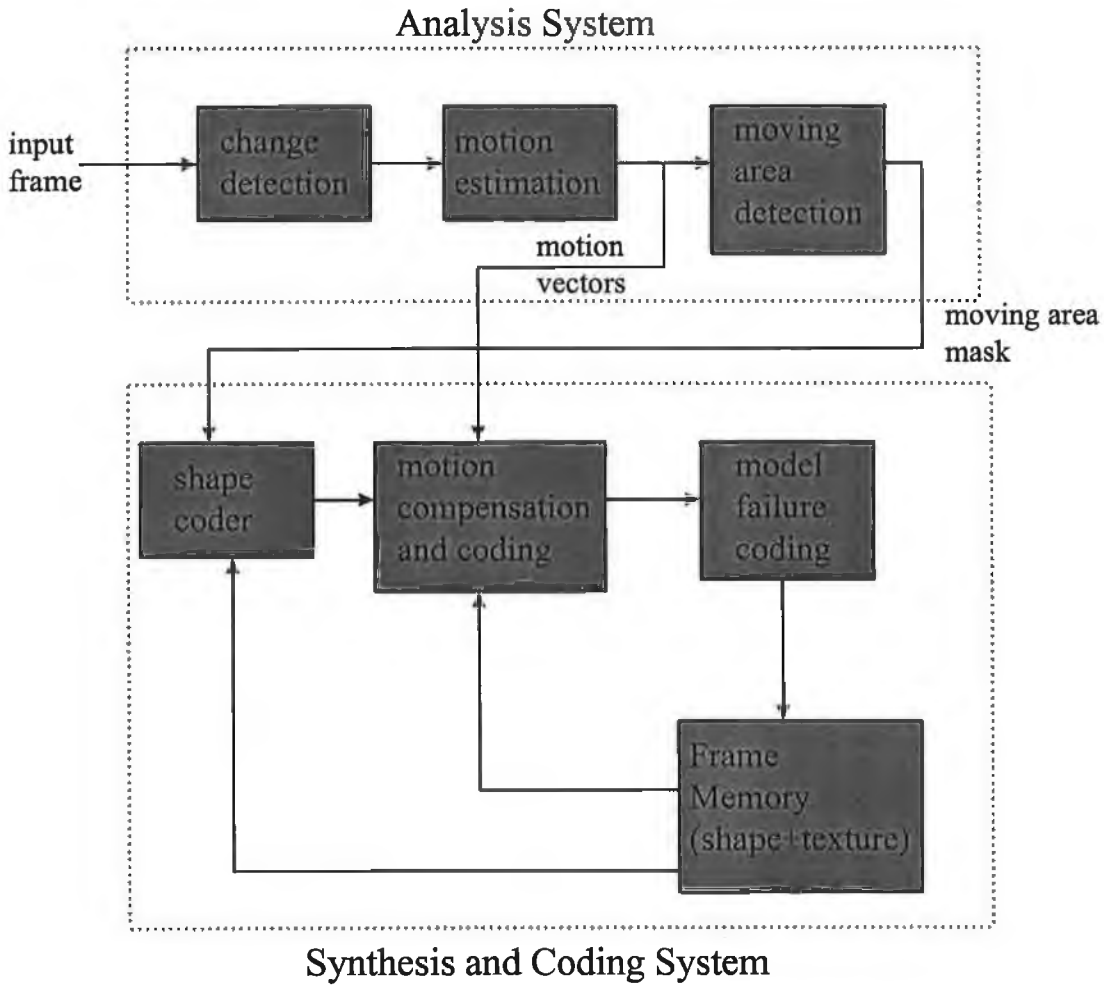


Figure 2-3: Simplified block diagram for SIMOC encoder.

2.3.1 Segmentation by Change Detection

Change detection represents one of the simplest methods for the segmentation of a moving object from a video sequence. When applied to segment an image at time *t*, the

algorithm has three inputs, i.e. the images at time t and t-1, plus the moving area mask at time t-1. Change detection is applied to the images at t and t-1 as follows.

1. Compute the absolute difference image.
2. Median filter the difference image.
3. Threshold the difference image with a fixed threshold to produce the binary change detection mask.
4. Clean the change detection mask using morphological filters and small region elimination.

The change detection is then modified by setting a pixel to CHANGED, if the corresponding pixel in the moving area mask at t-1 is classified as MOVING. This change detection mask is deemed to include uncovered background. The final step is to detect these uncovered areas, removing them from the change detection mask in order to generate the moving area mask for time t.

The median filtering, morphological filtering and small region elimination are intended to eliminate the effects of image noise. Interframe changes due to small noise-related variations in time, tend to be neglected. This is important because the production of a clean spatially coherent moving area mask means that the moving area mask (the shape of the moving object) can be highly compressed. The use of the previous moving area mask is to ensure some degree of temporal coherence in the resulting sequence of moving area masks. A temporally coherent shape sequence is generally defined to be one which possesses a high degree of temporal correlation or redundancy. Once again, this implies that a high compression ratio can be achieved through the use of interframe shape coding techniques. Moreover, actual moving objects in a given sequence generally exhibit this temporal characteristic and it is desirable that the moving area mask sequence should emulate it. That is, the moving area mask should accurately track the moving objects from frame to frame. As we shall see, most effective segmentation strategies encompass techniques which ensure both spatial and temporal coherence.

Several general criticisms are commonly levelled at this change detection approach **[96]**, **[12]**. The whole philosophy of SIMOC, which was built into the change detection method, was to isolate the moving area and concentrate the bit usage on that part of the picture. With respect to this philosophy, a sequence shot from a moving camera should theoretically result in all pixels being classified as part of the moving area and the object-based approach degenerates to a traditional frame-based approach. That is, SIMOC was an object-based coder only for scenes and sequences with static background. Many object-based purists viewed this as a great limitation. In fact, in a practical sense the situation was often somewhat worse. To explain, there is the inherent assumption in change detection that a pixel within a moving object will experience a change in its luminance value. This is not a very general assumption. Objects without significant spatial textural detail can move without changing all their pixel values. For this reason, with respect to the degenerate case given by the moving camera, the practical outcome of change detection was *not* that the whole frame was classified as moving. Very often, the moving area mask would span a significant (but not the whole) portion of the frame, yielding a highly complex shape to compress. This, of course, led to large bit usage for shape, when it would have been more sensible and efficient to signal in the bitstream that the whole frame was moving. For another example of useless bit usage for the same reason, consider the example of a white square moving on a darker background. As illustrated in Figure 2-4c, the moving area is deemed to be the changing area and as such, the coding system is requested to encode two rectangular shapes instead of just one. Another problem with change detection is that there is also the inherent assumption that all interframe luminance changes are due to an object motion. This assumption is seldom justified. For example, changes can be caused by sudden changes in the scene lighting conditions or by camera noise. The basic change detection approach is over-sensitive to these variations because it uses only frame difference information. This point is illustrated in Figure 2-5.

(a) (b) (c)

Figure 2-4: (a) Synthetic sequence, a section of Miss America moving at a constant horizontal velocity. (b) The change detection mask produced for the synthetic sequence in (a). (c) The change detection mask produced by a constant colour box moving horizontally on a black background.

From the general point of view, it could be argued that the change detection approach does not explicitly focus on coding oriented criteria. The task of segmentation in compression-oriented algorithms is to arrive at an optimal partition of the image given the models and tools employed by the coding system. The SIMOC segmentation algorithm is primitive in this sense. It only considers that changed pixels from one frame to the next need to be updated somehow. It does not consider how this updating will be performed, i.e. the motion compensation and the texture coding methods are not considered. In addition, no attempt is made to produce smooth shapes in the segmentation map. Thus, there is also no realisation that shape must be encoded. This can be noted from some of the change detection results provided in Figure 2-5. Contours are noticeably jagged and therefore require more coding effort. Furthermore, only a very weak control is exercised on how the moving area mask can change over time. Hence, even interframe shape coding strategies are not suitable. Because the change detection approach has no appreciation of the coding models, there is no guarantee that the resulting object-based representation of the coder will be efficient. In more recent times, the basic approach has been augmented by the use of noise modelling and relaxation labelling **[1]**. Sometimes, the moving area mask can be adapted so that its contours adhere to natural gradients in the original image. These additions tend to greatly improve the performance of change detection methods when applied to noisy sequences. The use of global motion compensation is being investigated to cope with the case when there is one foreground object being shot by a moving camera. Despite these undoubted

improvements, some additional use of coding oriented constraints as described in **[22]** are deemed necessary to improve compression efficiency.

Figure 2-5: (a) and (b) are consecutive frames of a 8.33Hz sequence exhibiting illuminatory/noise variations. (c) shows the result of the change detection process with a threshold of 2, while (d) is the change detection mask with a threshold of 3.

2.3.2 Vertex-based Contour Coding

Among the most common approaches to shape encoding is that based on polygon or spline models. These models have been used with relative success in SIMOC, where it is required to encode the shape of the foreground object moving on a static background. The intraframe approach is very simple. The contour of the object is described by a list of vertex locations. Each vertex is described by an (x,y) co-ordinate which resides on the contour. The remaining contour points are interpolated by means of a polygon or spline. Fixed parameter splines are used so that the spline parameters do not have to be transmitted. Only the vertex locations are encoded. This is done by a DPCM method. The contour vertices are scanned in a predefined order. The value of the first vertex is coded using a fixed length code. For subsequent vertices, the previous vertex is chosen as a predictor and only the prediction difference is encoded using an arithmetic encoding method.

The efficiency of such a scheme relies on a good method for choosing the vertices. In SIMOC, a top-down divide-and-conquer approach is employed. An initial small set of vertices are placed at selected contour points. While the contour is still badly approximated, further vertices are placed in between existing ones. The final

representation (within SIMOC) is a lossy one, with the approximated contours being at most two pixels distant from the original contours.

In SIMOC, an interframe coding method was developed based on this representation. Since each shape is represented at the decoder by a list of vertices, the vertices at frame t are used as the prediction for the vertices at frame $t+1$. In fact, the vertices at frame t are displaced according to their motion in order to provide an even better prediction of the shape at $t+1$. The prediction itself is formed by interpolating between the predicted vertices. Naturally, this prediction has certain associated errors. These errors are taken into account by the insertion and coding of new vertices, which are placed on contour points which are not well represented by the prediction. After the insertion of new vertices, there may be a certain amount of redundancy in the vertex list. That is, if three vertices roughly lie on the same polygon or spline curve, then the centre one is rejected from the vertex list. The result of this is that the encoder must transmit the list of new vertices, plus overhead information stating the locations of these new vertices and indeed, the rejected vertices.

This method is promising in that it effectively exploits the correlation in smooth object contours and it has been proven to be quite efficient for lossy coding. Furthermore, an interframe coding mode is possible, thus exploiting temporal correlation in the segmentations. Along with chain coding, this method is one of the most popular approaches to shape coding. There are still some criticisms of this method but these relate mainly to evaluation criteria other than coding efficiency, e.g. implementation complexity and end-to-end communications delay.

2.3.3 Grid Interpolated Motion Estimation and Compensation

It has been stated already that object-based coders rely very heavily on effective motion compensation. The block-based constant motion fields used within H.261, while requiring very little coding bandwidth, do not facilitate very good motion compensations. Block-based motion fields contain disturbing discontinuities at the block boundaries and these result in very nasty artefacts in the motion compensated

images. Furthermore, at low bit-rates, these artefacts also appear in the reconstructed images. The other problem is that only purely translational motion within the image plane can be well synthesised. The designers of SIMOC sought to avoid these blocky artefacts and settled on a motion representation that could deal with more general object motions. The approach is as follows.

A regular grid is defined on the current image. There is one grid point every 16 pixels on every 16th line. For each grid point, a robust hierarchical block-based search **[9]** is employed to compute a half pel motion vector in the range [-4.5,4.5]. This motion vector minimises the Mean Absolute Difference (MAD) in the local area of the grid point. Once all grid vectors are estimated, a bilinear interpolation of the motion field is carried out. This results in a half-pel motion vector for each pixel. Each pixel within the moving area is then motion compensated using its interpolated vector. Due to the smoother nature of the interpolated motion field, the motion compensated prediction is visually pleasing and is devoid of any blockiness. Moreover, the interpolated field turns out to be blockwise planar or affine. As we shall see in chapter 4, affine motion models are capable of approximating translations, rotations and zooms and hence the interpolated field is far more effective than the conventional blockwise constant fields. The described interpolated motion representation is therefore very promising.

While the grid interpolated approach is in itself a very sound technique, in the context of the object-based coding philosophy, it has a small disadvantage. As pointed out by Wuyts *et al* **[96]**, it is a globally flexible motion model in contrast to the more compact rigid object motion models that are recommended above. Thus, with a motion vector at each grid point, the representation uses as many bits as that of H.263 and in this sense, it does not help to compensate for the extra expenditure in shape.

2.3.4 Model Failure Detection and Coding

Model failure detection is a method intended to efficiently encode the prediction error within the object. The strategy involves detection of the subset of all the object pixels that are badly predicted by motion compensation. This subset, termed the model failure

(MF) region, is then encoded using a VQ technique. Of course, the shape of the MF regions also has be coded and transmitted. In the opinion of the author, this is one of the most naive aspects of SIMOC. It is a part of the encoder which is really not optimised on a representative set of sequences, but instead, it is designed to perform well only for very simple videophone scenes. The model failure regions were intended to detect and code unpredictable scene events such as eye openings etc. The assumption was that motion compensation would be able to accurately synthesise the remaining events. Of course, for sequences with noise, photometric variations or complex motion, motion compensation still results in large prediction errors and hence, in general, the MF regions detected by SIMOC are rather larger and more disjoint than first intended. When the assumptions underlying MF coding break down, the MF coding approach tends to be extremely inefficient.

2.3.5 Coding Results for SIMOC1

SIMOC1 is an open-loop coder. That is to say, it has no buffer regulation mechanism and is controlled only by a quality criterion. This criterion states that the PSNR of the luminance (Y) component of reconstructed pictures should be 34 dB. In fact, this is only enforced in model failure (MF) regions and uncovered background regions. The result is that bit usage varies wildly for different input sequences.

The average bit usages per second for each of the three test sequences used are shown in Table 2-1. This shows the breakdown of the total bit-rate into bits for shape, motion and colour. The bits for colour are entirely due to model failure regions being detected and coded. The bit-rate over time is also illustrated in the accompanying graphs of Figure 2-6, Figure 2-7 and Figure 2-8.

Table 2-1: The SIMOC bit usage on 3 common (QCIF) test sequences.

Sequence	Bits for shape (Kbit/s)	Bits for motion (Kbit/s)	Bits for colour (Kbit/s)
Miss America	7.4	2.3	12.2
Claire	5.8	4.6	20.5
Foreman	15.6	5.6	150.0

When the assumptions underlying the change detection, model failure detection and coding approach are correct, as in the cases of Miss America and Claire, reasonably efficient performance is possible. However, the sequence Foreman breaks many of the model assumptions, i.e. it is taken with a moving camera and the object is undergoing fast and complex motion. As a consequence of this, the bit-rate to encode model failure areas drastically increases. This illustrates the fragile nature of many object based algorithms.

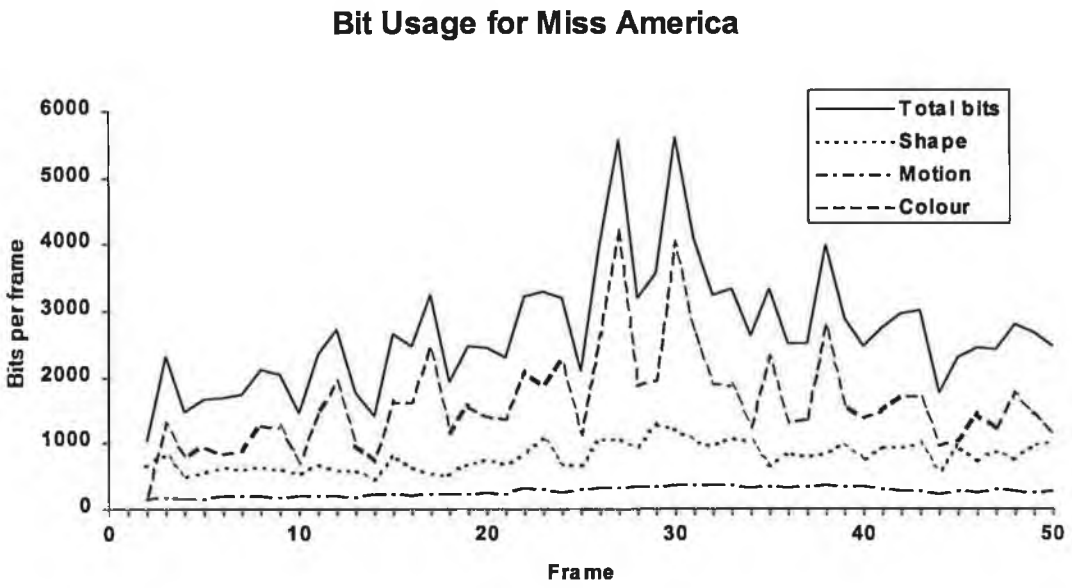


Figure 2-6: Graph of the bit-usage breakdown over 50 frames of Miss America.

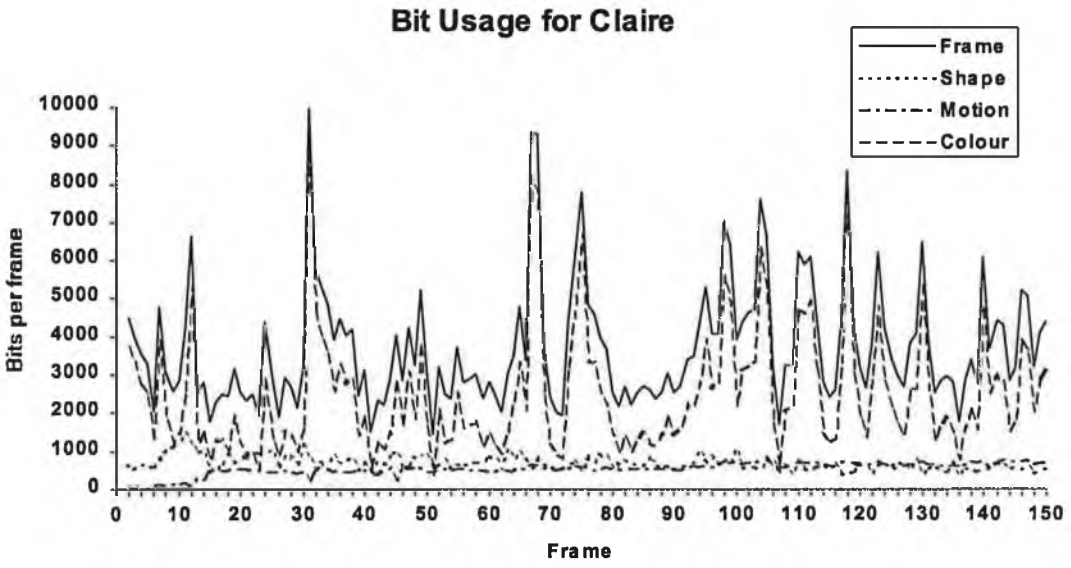


Figure 2-7: Graph of the bit-usage breakdown over 150 frames of Claire.

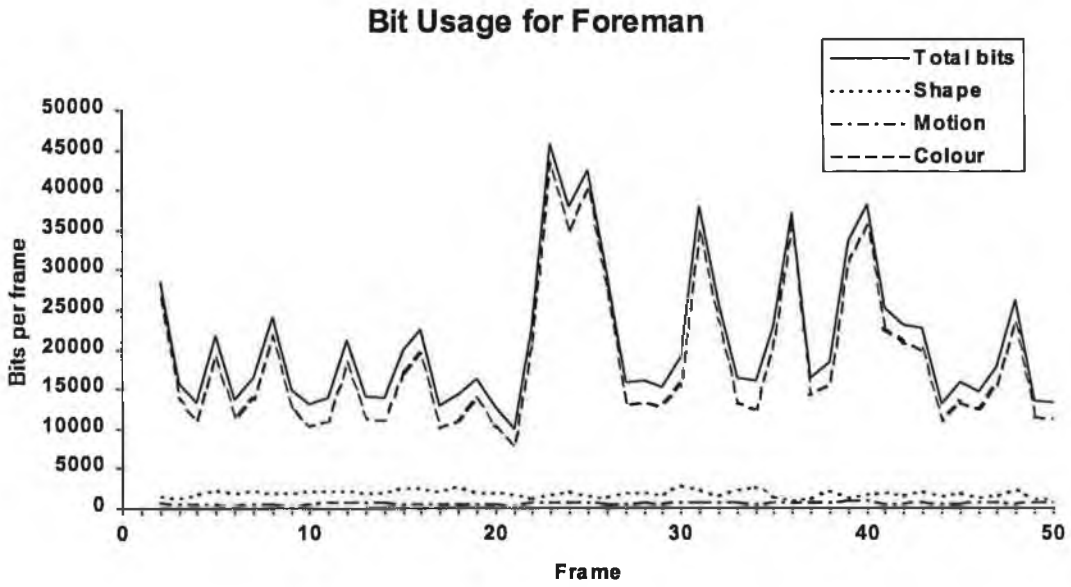


Figure 2-8: Graph of the bit-usage breakdown over 50 frames of Foreman.

SIMOC has been treated harshly in terms of its genericity, i.e. the segmentation approach works best on low-noise sequences with a static background and the model

failure coding approach works best when motion is very simple. However, genericity appears to be a problem in many object-based coding strategies. That is, most object-based coders are optimised for a certain set of conditions and these conditions are usually only fulfilled by a small set of sequences. Many people view this non-genericity as a fact of life: something which, though unsatisfactory, must be accepted. This has led to the development of switched mode coders which profit from being able to code in block-based mode or in object-based mode as appropriate **[20]**, **[21]**, **[64]**. These switched coders can choose on a frame by frame basis whether object-based coding is suitable, i.e. whether the scene conforms to a restrictive set of assumptions. If the assumptions are invalid, then the block-based coder is used as a fallback mode. This approach seems to produce good results, but it is rather unwieldy and inefficient from an implementation point of view.

2.4 MPEG-4 Video Work

Compression-oriented object-based coding methods have not lived up to the initial hopes and aspirations. More time may be required to allow the science to mature so that it can outperform the conventional block-based methods. Nevertheless, the recent MPEG-4 standardisation effort has placed considerably emphasis on object-based coding in order to provide for semantic content access.

MPEG-4 seems to have twin aims. The traditional requirement of coding efficiency is still very important, since MPEG-4 is committed to providing universal access to multimedia information via mobile networks and other low grade media. However, most of the emphasis appears to be on the requirement for content access **[56]**. The MPEG-4 video algorithm is of the content-oriented type, as defined in sub-section 2.1 above.

MPEG-4 began by specifying a flexible coding architecture, the MPEG-4 VM **[74]** based on VOPs or Video Object Planes, see Figure 2-9 and Figure 2-10. These VOPs are usually regarded as 2-D semantic objects, but on a more abstract level, they are merely arbitrarily shaped image regions. Each VOP is composed of YUV pixel data plus an alpha channel specifying the shape of the VOP. A video scene is viewed as being a composition of VOPs. The analysis/segmentation function (VOP definition) was

totally decoupled from the coding function. MPEG-4 only deals with the definition of the coding and decoding system. It is assumed, for the moment, that the alpha channel information is already available (via blue-screening) or that it will, in the future, be reliably provided by automatic or semi-automatic means. Note, that this approach is different than the SIMOC approach, whereby the analysis and coding were tightly coupled.

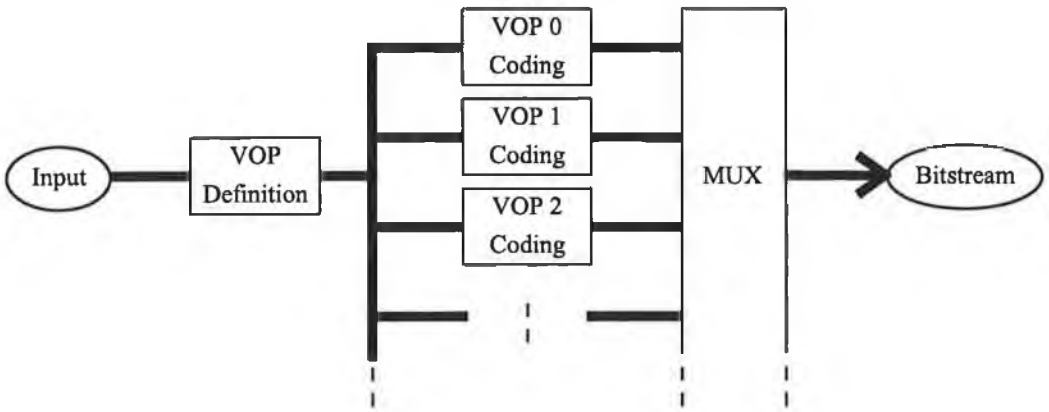


Figure 2-9: The MPEG-4 VOP-based Video Encoder. The VOP definition stage (the segmentation) is not subject to standardisation.

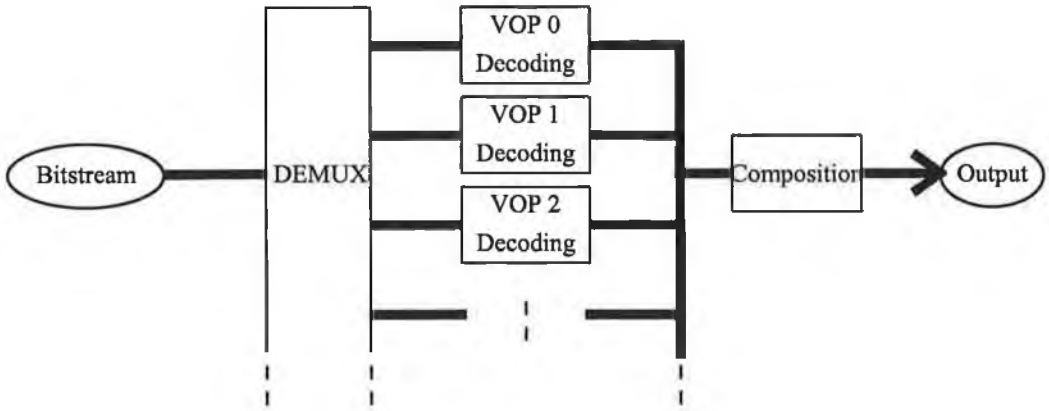


Figure 2-10: The MPEG-4 VOP-based Video Decoder

Through the use of VOPs, content-based access is satisfied. Moreover, by using just one VOP comprising all the pixels in the given frame, the coding scheme degenerates to the standard H.263-like specifications. In such degenerate cases, no shape information is transmitted and hence, a coding efficiency equivalent to or better than H.263 is

guaranteed. For the most part, very conventional block-based ideas have been tailored in order to code the VOP interior. Indeed, the VM is currently an amalgamation of H.263, MPEG-2 and the required shape coding algorithm. Therefore, MPEG-4 is wisely making use of existing robust technologies. This is in direct contrast to SIMOC, which suffered from the use of immature technologies such as model failure coding.

Apart from the required shape coding algorithm, the MPEG-4 video algorithm is not so much different from H.263, for example. The interior of the object shape is coded in terms of blocks. For interframe coding, each 16x16 block may be motion compensated using motion vectors and the residual is encoded by a DCT-based method. The major differences come when handling border blocks which are not fully inside the object-shape. In these cases, shape adaptive texture coding methods **[88]** can be used. Alternatively, methods of block padding are utilised. For instance, prior to DCT coding, an 8x8 pixel border block is padded by more or less extrapolating the pixel values inside the object to provide values for those block pixels which are outside. Then, the padded 8x8 block undergoes the standard DCT coding. For motion compensation, the process is similar. The motion vector points to blocks from the previous VOP and these block pixels are copied into the predicted VOP. However, since some of the copied block pixels may be outside the object shape in the previous frame, padding is used to evaluate such undefined pixels. A macroblock layer syntax is defined to represent the coded VOP, whereby each macroblock (16x16 pixels) may consist of shape information, motion vectors and DCT coefficients. If the shape information indicates that the macroblock is transparent, then no further information is coded for that macroblock. Similarly, measures are used to ensure no wasteful information is coded for transparent 8x8 blocks.

While both efficiency and content access requirements can be provided, a bitstream supporting content access is generally not as efficient as one which does not support it. Since coding efficiency is always sacrificed in order to provide content access (i.e. shape information must be coded), it is unlikely that MPEG-4 will succeed in providing content access **and** significantly higher quality video than is provided by H.263 in the same bitstream, for instance. On the other hand, for particular applications, e.g.

videophone, the content-based representations may present some advantages. These are related to content-based rate allocation. For example in videophone applications, it is the facial region which is important for effective communication. Let's suppose a videophone scene is decomposed into several VOPs, one of which corresponds to the face. For low capacity or noisy channels, it is possible to exploit the VOP-based representation to code the facial region with a higher quality. For instance, a higher proportion of coding and error protection bits might be allocated to the facial VOP, preserving its quality to the detriment of the other less important VOPs. Alternatively, the other VOPs might not be transmitted at all, facilitating a talking head videophone.

2.5 Summary

This chapter has reviewed the methods of and motivations for object-based video compression. Both compression-oriented and content-oriented approaches have been discussed. The failure of object-based approaches to significantly improve upon their block-based counterpart has been noted. This failure can, in part, be attributed to an immaturity in the segmentation methods and also in the shape compression. There is, therefore, a need for improved schemes for both compression-oriented segmentation and shape compression. While there are some doubts about the usefulness of compression-oriented object-based coders, there is little argument about the use of efficient object-based representations for facilitating content-based applications. In this regard, MPEG-4 will provide a solution that produces efficient video object representations by specifying methods for encoding the object shape and texture. However, in order to produce object-based video representations, there is a need for a convenient means for segmenting semantic video objects from their scenes of origin. These needs set the tone for the remainder of this thesis and new proposed solutions are described and tested.

The next chapter addresses the issue of shape compression, while chapters 4-6 confront the segmentation problem. For compression-oriented object-based coding, the need is for automatic segmentation methods capable of partitioning an image such that each partition can be compressed in a highly efficient way. For these applications, it has been established that coding-oriented criteria must be used by the segmentation approach. It is believed that many previous compression-oriented coders failed in this respect.

Chapter 5 develops such a segmentation approach based on motion analysis, the expectation-maximisation method and the minimum description length principle. For content-oriented applications, there is a clear need for segmentation methods which can capture the shape of semantic objects. It is doubtful that this can be achieved by fully automatic means and thus a supervised approach is envisaged. In this respect, tracking algorithms based on motion analysis are viewed as essential components. Chapter 6 builds on the motion analysis approach of chapter 5 in order to develop and test a new tracking algorithm.

3. SHAPE COMPRESSION

The outstanding difference between the conventional video standards on one hand and object-based video coding as exemplified by SIMOC and the MPEG-4 video Verification Model (VM) on the other, is that the compressed video bitstream contains shape information. The shape information describes the shape and location of each video object in the 2-D scene at every time instance in the sequence. In raw uncompressed form, for a particular object, this shape information is most conveniently represented by a 2-D binary image as illustrated by Figure 3-1. Pixels with the label WHITE are considered to be part of the given object and pixels with the label BLACK are considered not to be part of this object. In MPEG-4, this binary image is referred to as a binary alpha map. MPEG-4 also allows more general grey level alpha maps. In addition to specifying whether a pixel is part of the given object or not, they can specify levels of transparency for each pixel. All pixels labelled WHITE are totally opaque, all pixels labelled BLACK are fully transparent and those with any other grey level have some degree of transparency.

(a) (b)

Figure 3-1: MPEG4 test sequence KIDS. (a) The VOP (consisting of the two children) is composited onto a grey background, (b) The binary alpha map for the VOP.

This chapter focusses only on the encoding of binary alpha maps for digital video applications. A brief review of established and emerging techniques for shape coding is given in section 3.1. Section 3.2 presents a common set of requirements which should

be fulfilled by the shape encoding. The efforts within MPEG-4 have been responsible for the fast evolution of technology in this area and this evolution is described in section 3.3. Finally, some detail is provided on one of the main tools used in the MPEG-4 VM7 shape coder, i.e. context-based arithmetic encoding (CAE). This CAE method, which has been developed by the author, is very novel due to the fact that it is optimised for a block-based syntax and it is capable of exploiting temporal correlation.

3.1 Review of Shape Coding Techniques

Along the lines of a classification proposed by De Sequeira in **[87]**, the methods of coding shape and binary 2-D data and images can be classified into two broad types.

- Bitmap coding
- Contour coding

The bitmap coding class includes context-based arithmetic encoding (CAE) **[44]**, modified modified Read (MMR) encoding **[40]** and run length encoding (RLE) **[33]**. These methods directly encode the binary pixel values within the alpha map. The contour coding methods include chain coding and vertex-based coding. These perform an initial transformation which converts the binary alpha map into a contour image. Coding is then applied to the contour image. A contour image is again a binary image, but one where pixels with the value WHITE are those which reside on the edge of the object shape and those with value BLACK are either inside or outside the object shape.

3.1.1 Bitmap Encoding

Bitmap encoding strategies have certain advantages in terms of simplicity and flexibility due to the fact that they are applied directly to the binary image. They have been used for coding binary images in several FAX standards **[47]**. Two of the most common and well known binary label coding schemes are MMR, as used in FAX coding, and CAE, as used in JBIG **[41]**. In the more recent past, the key algorithms used within the FAX standards have come under study within MPEG-4. Initially, both MMR and CAE were

designed primarily for lossless coding of still images. However, in the context of MPEG-4, a means of applying these techniques to lossy coding is necessary.

G3 and G4 FAX Coding: MH, MR and MMR

In the G3 FAX standard **[39]**, several coding strategies are employed. The most basic is the Modified-Huffman (MH) method. This is a simple Huffman encoded RLE scheme which can be implemented without requiring large memory for the VLC tables. In this scheme, each raster scan line is subjected to MH coding and the line code is terminated with an end of line (EOL) code. The Modified-Read (MR) method introduced in the G4 standard **[40]** attempts to exploit correlations in the vertical direction as well as the horizontal direction. In adjacent lines, the locations of the black-white/white-black transitions are likely to be similar. The MR method requires that, while coding the current scan line, the previous line be retained in memory to be used as a reference or prediction for the current line. Through the use of special codes for representing common inter-line events, significant improvements in compression efficiency are achieved. The MH method is still used to encode line patterns which cannot be well predicted from the reference line. In fact, the MH method is used periodically every K lines. This provides error resilience by preventing bit errors in the compressed stream from propagating without limit through the reconstructed image. In terms of efficiency, the modified-modified-Read (MMR) method further improves upon the MR method. The gain in compression is achieved at the expense of error resilience. The EOL codes are removed and the periodic usage of MH is omitted. The MMR codes applied in this manner are only useful on error free channels or with the help of packet retransmission strategies such as automatic repeat request (ARQ). The G3 and G4 compression methods are successful due to their ability to provide significant compression while minimising computational complexity and memory requirements.

JBIG

JBIG (Joint Bi-Level Image Group) refers to a collaborative effort on the part of the ITU-T and ISO to define a progressive coding format for compressed binary images. The standard is also known as T.82 **[41]**. On scanned images of printed characters, the observed compression ratios for JBIG have been 1.1 to 1.5 times larger than those for

MMR. On computer-generated images of printed characters, JBIG is up to 5 times better.

One of the key algorithms used within JBIG is context-based arithmetic encoding **[44]**. The image is optionally decomposed into a number of resolution levels. Each resolution level is divided into rectangular bands called stripes. A stripe contains a contiguous group of lines. The pixels within each stripe are scanned in raster fashion. In JBIG, only a subset of the pixels are subjected to the CAE. For each of these pixels j, a so-called model template is defined. The template contains pixels previously coded, i.e. those in a causal neighbourhood of j and those from the already available lower resolution image. Based on the particular configuration within the template, a context number is produced for pixel j. This context number is used to access a probability table which provides the probability $P(0)$ that pixel j is BLACK and the probability $P(1)$ that pixel j is WHITE. An arithmetic encoder uses these probabilities to produce a highly compressed code for all the pixels in the stripe. In JBIG, the probability table is generated from the data being encoded, i.e. as each new sample is encoded, the probability model is updated accordingly. As such, the probability model can adapt to the statistical characteristics of each coded image. More details on the motivations for the CAE approach are given later in the chapter.

3.1.2 Contour Encoding

In contrast with bitmap methods, contour encoding techniques require that the initial binary image is transformed into a contour image and it is this contour image which is encoded. During the decoding process, the reconstructed contours must be filled in order to produce the reconstructed binary image. Here, two contour encoding methods are discussed, i.e. chain coding **[42]** and vertex-based methods **[23],[30],[36]**.

Chain Coding

A chain code begins with the (x,y) coordinate of the first contour pixel. This is usually transmitted without compression. Starting from the first contour point, the contour is traced pixel by pixel in a clockwise or anti-clockwise direction. Each contour pixel encountered is encoded by transmitting a value representing the direction passed

through in order to move from the previous pixel. An example of the possible directions (in 8-connectivity) is given in Figure 3-2. An uncompressed chain code in 8 connectivity is simply a list of 3 bit integers. The code is usually compressed by run length encoding (RLE).

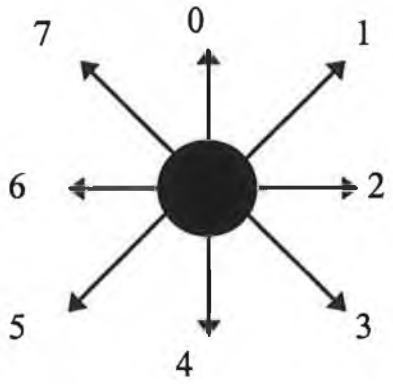


Figure 3-2: The basis of an 8-directional chain code.

Several improvements are possible on this basic scheme. For instance, the set of possible directions can be reduced by replacing diagonal directions by a composition of a vertical and horizontal direction. This is really a lossy chain code but generally the observable distortion is not disturbing. Another improvement to the basic scheme is the differential chain code. In this case, the uncompressed chain code is coded by predicting each direction from the previous one. This results in a list of prediction errors. The prediction errors are then entropy encoded using a VLC table or arithmetic encoding.

Vertex-based Methods

Vertex-based methods also belong to the class of contour encoders. As discussed in the previous chapter, the vertex-based representation consists of an ordered list of points in 2-D space. The reconstructed contour is produced by beginning at the top of the list and drawing lines or splines between each adjacent pair of vertices. Commonly, some kind of optimisation algorithm is required to choose an efficient set of vertices for the representation of a given shape. The vertex-based methods have been proved to be useful for lossy encoding of smooth contours. For such contours, a small number of suitably chosen vertices can be used and yet the reconstruction errors remain small.

3.1.3 Comparison of Bitmap vs Contour Encoding

In general terms, there is no reason to favour one of these classes of coding techniques over the other. Practically, however, the bitmap-based methods have some advantages.

- There is no need to extract the contour map at the encoder.
- There is no need for a filling algorithm at the decoder. Indeed, the design of a robust filling algorithm capable of dealing with generic shapes is non-trivial and many proposed solutions contain flaws, such as those described in [71].
- The memory access of the bitmap-based algorithms is regular and predictable, whereas the memory access of the contour methods is dependent on the nature of the data to be encoded. This means that on-chip memory caches cannot be easily used for contour based methods.
- Bitmap-based methods like MMR and JBIG are established standardised methods whose implementation is known to be feasible.

This very brief review has highlighted that some mature technologies for binary image/shape coding have existed for many years. For the extension of these techniques to video coding applications, attention needs to be devoted to the new nature of the content to be addressed in video applications, i.e. to the fact that high spatial *and* temporal correlations are common, to the need for lossy compression and to the need for error resilience. The next section explains what is expected of a shape coding algorithm for use in MPEG-4.

3.2 Shape Coding Requirements

While binary image coding has been a well researched topic, the use of binary shape coding in digital video systems presents somewhat different challenges. The challenges are the same as those which the standards developers of MPEG-1, MPEG-2, H.261 and H.263 were presented with, when they began work. These conventional challenges and requirements are discussed first and then, those new requirements specifically pertaining to the need for content access are outlined.

3.2.1 General Requirements for Shape Coding

The design of a video compression system begins with identifying the promising applications and then deriving a list of essential requirements which need to be fulfilled to best enable some or all of the applications. Table 3-1 gives a list of the most common requirements for digital video systems, i.e. encoders and decoders. All video compression standards up to and including MPEG-4 were developed on the basis of such a list of requirements. The difference with MPEG-4 is that an extra compression sub-system is required for encoding object shape information. This sub-system must be designed such that all the above requirements can still be met. That is, the shape encoding/decoding solution must not significantly hinder the fulfilment of the requirements.

While much work has been done previously on the representation of shape, not all offerings are designed to meet the stringent requirements of video communications. All the reviewed techniques of section 3.1, with the exception of the vertex-based methods, are only designed to deal with lossless coding of shape, whereas many video applications would benefit from a lossy mode allowing adaptive rate regulation and so on. Also, when dealing with video, it is known that extra coding efficiency can be gained by inter-frame prediction. This should also be true of shape coding. Therefore, the provision of an inter-frame coding mode is of prime importance. When dealing with video, the error resilience requirement is very important. In most applications, there is little possibility of using ARQ due to the delay and decoder buffering that it implies. Error resilience is a requirement with increased emphasis when it comes to video coding. Therefore, it is imperative that the coded representation for shape in a video context is able to provide, or fit into, some error resilient coding framework. Apart from these conventional requirements, the new need for content access places more limitations on the methods which can be applied, as discussed in the next section.

Table 3-1: Requirements placed on shape encoder and decoder systems.

Requirement	Definition	Comments
Coding Efficiency	The ability to achieve compression with the minimum loss in quality. Both lossless and lossy coding modes are required.	All applications require this.
Low Complexity	The encoding and decoding processes should be realisable in a practical manner and with reasonable cost.	For client terminals attached to video databases, it is particularly desirable if the decoding is process is simple and not so important if the encoding process is complex.
Low Delay	The delay between the start of encoding at the transmitter and the end of decoding at the decoder should be tolerable.	This is particularly important in real-time duplex communications, e.g. videophone, where delay hinders effective two-way communication.
Error Resilience	The representation should not be overly sensitive to bit-errors or cell loss.	Particularly tight constraints are imposed in real-time duplex communications where retransmission of packets is not permitted.
Rate Regulation	It should be possible to control the size of the representation by moving along the rate/distortion curve.	This is useful in most transmission environments but particularly when fixed-rate channels are being used.
Scalability	A receiver should easily be enabled to choose a subset of the bitstream and using the subset reconstruct some rendition of the fully coded entity.	This is useful in broadcast or database applications, where networks and receivers with varying capacities and capabilities exist.

3.2.2 Content Access Requirements

It is not discussed in the introduction of this chapter, but it is possible to represent several object (binary) shapes within one 2-D labelled image. Such a shape representation is called a segmentation map. Each object in a given scene has an ID number or label. The pixels within the segmentation map each have a label. A pixel with label i is associated with object i. To encode a segmentation map, there are two possibilities. The first is to encode the segmentation map directly, i.e. treat it as an N-bit image (where N bits are used to represent each label) and employ the appropriate techniques directly to this image. The second approach is to first decompose the segmentation map into binary alpha maps and apply binary shape coding techniques to each of these while associating the resulting independent codes with an object of given ID. The first approach is believed to be more efficient, as explained below. However, it does not lend itself well to the provision of content access. To access a particular object's data, the compressed segmentation map must be fully decoded. If the object is to be placed into a new scene (Content-based Video Editing), then the destination

scene's bitstream must be decoded and re-encoded with the new object included. The second approach avoids these problems, but it effectively means that there is a certain duplication of information in the bitstream. That is, the contours of object X may also form part of the contours of object Y and hence, by independently encoding the binary masks for object X and object Y, the shared contours are being encoded twice. This inefficiency is a necessary evil in order that flexible content access is provided for.

Although in this chapter only binary shape coding supporting content access is discussed, there are however, some applications for which limited content access is sufficient and for which direct segmentation map encoding may be more suitable. In a real-time encoding scenario, it is currently considered important that a decoder user be allowed to interact with the content of the video being viewed. One use of this interaction is that the user could assign priorities to one object or another. This priority could be transmitted back to the encoder and could be interpreted there as an instruction to allocate more bits to the chosen object. The result would be that the selected object is enhanced in quality. This scenario is useful in surveillance applications, where video is being transmitted over a low grade network, and it does not require full content access. In such an application, it would be best to use direct encoding of the segmentation map in order to achieve the utmost coding efficiency.

3.3 Evolution of MPEG-4 Binary Shape Coding

The most common techniques for binary shape coding have been reviewed and the requirements for MPEG-4 shape coding have been outlined. This section contains a review of how the MPEG-4 binary shape coding solution evolved.

MPEG-4 required an efficient shape representation which availed of interframe correlation and which could be lossless or lossy. Additionally, complexity, error resilience and delay were also important considerations. Prior to MPEG-4, these requirements were fulfilled by the development of block-based video encoding methods. That is, all previous video standards utilised a block-based algorithm. The reasons for this are best explained in relation to the requirements of Table 3-1.

Figure 3-3 shows how a QCIF video frame is partitioned into macroblocks, where a macroblock corresponds to a 16x16 block of image pixels. The macroblocks are then coded in some predefined order. For example, the MBs can be coded in raster scan order as indicated in Figure 3-3. Each macroblock is then represented in terms of its motion and YUV data in compressed form. This block-based video representation has been employed widely and for good reasons, as explained now.

Coding Efficiency

When encoding a video frame, it becomes apparent that there is no one coding approach which deals optimally with all areas of the frame. Some picture areas will require INTRA coding. Some picture areas will require INTER coding. Some picture areas will not require any update at all, but will be accurately predicted from previous frames. The use of a block-based representation for the video frame means that the coding approach can be adapted on a block basis such that every block is coded the best way possible. Blocks residing in still parts of the picture will not be updated. Blocks residing in parts of the picture exhibiting simple motions will be INTER coded. Blocks residing in parts of the picture where no useful prediction is possible will be INTRA coded. It is this local adaptation of the coding mode which gives the block-based approach the possibility of achieving very high compression ratios for video. Furthermore, a video scene typically contains one or more moving objects. Each object or each part of an object can be undergoing a different motion. For efficient interframe coding, some basis for local motion analysis and representation must be provided. A block-based framework provides a convenient platform for local motion representation.

1	2	3	4	5	6	7	8	9	10	11
12	13	14	15	16	17	18	19			
								97	98	99

Figure 3-3: A QCIF (176x144 pixels) video frame partitioned into 11x9 macroblocks, numbered 1 to 99.

Codec Complexity

Block-based algorithms have been successfully implemented on application specific integrated circuits and general purpose processors. This has been mainly due to the fact that a block-based approach reduces the hardware requirements in terms of memory bandwidth. The macroblock, which is the main coding data structure, contains six blocks each of size 8x8 pixels. This means that a macroblock contains only 384 bytes of data. The small size of the macroblock means that any processor can store several of them on-chip, avoiding the delays incurred for memory accesses across a system bus to the main memory. Additionally, block-based approaches have also enabled the use of parallel and pipelined hardware architectures.

Low Delay

The block-based approach has clear advantages in terms of delay. At the encoder, an input buffer receives the current frame from the capture device. The frame is received line by line. Once, the sixteenth line is received it is possible to begin encoding/transmitting the first line of macroblocks (macroblocks 1 to 11, in Figure 3-3). At the receiving end, decoding can begin immediately on receipt of the first compressed macroblock. In summary, a block-based representation, in principle, allows encoding

and decoding delays much less than one video frame in duration, thereby reducing the problems of delay and minimising encoder and decoder buffering.

Error Resilience

The general approach to providing error resilient video representations is to insert resynchronisation markers (RM) into the bitstream at regular intervals. If a bit error occurs, synchronisation is lost only until the next re-synchronisation marker. Effectively, a packetized structure is used as depicted below.

RM1	Compressed Data 1 (CD1)	RM2	Compressed Data 2 (CD2)	RM3	.....	.....

If an error is encountered in packet 1 then CD1 is discarded. RM2 is found and decoding continues with CD2. This implies that CD2 must have no dependence on the pixels represented by CD1. Hence, each packet, consisting of an RM and a CD field, contains coded data which has been produced independently of other packets. The block-based approach easily fits into this packetised framework. For example and with reference to Figure 3-3, CD1 might represent macroblocks 1 to 11, CD2 might represent macroblocks 12 to 22 and so on. Only one simple rule must be enforced, i.e. when encoding a macroblock within a given packet, no data from outside that packet may be used. This ensures that error propagation is limited to within the image area represented by the affected packet.

In terms of the requirements covered, the block-based approach is shown to be flexible in meeting the multiple demands placed on video systems. Despite this, MPEG-4 evaluated several non block-based shape coding algorithms based on contour coding techniques **[62]**. Ultimately, however, a block-based solution was favoured. A factor in this choice was the fact that the coding of the YUV texture data is performed on a macroblock basis. In order to facilitate an elegant syntax, it appeared to be sensible to perform the shape coding on a similar basis. The next sections plot the progress in shape coding within MPEG by describing chronologically the solutions of the various versions of the video Verification Model.

3.3.2 The VM3 and VM4 Shape Encoder

In July 1996, MPEG-4 published version 3 of the video Verification Model (VM) **[57]**. This contained a block-based shape coder capable of lossy coding and capable of inter-frame coding. The shape coder had been designed by Toshiba and Matsushita of Japan and was based on a motion compensated DPCM loop as depicted in Figure 3-4. The input to the loop is a binary alpha block (BAB) which is 16x16 pixels. When the video object is INTER coded, motion compensation (MC) can be carried out to give a predicted BAB. In the case of VM3 and VM4, this motion compensated BAB is generated using the same motion vectors which are used for motion compensating the corresponding YUV macroblock. This is input to a mode decision procedure. This procedure has the function of deciding how the current block is to be coded. The BAB may be represented in one of four ways as summarised in Table 3-2.

Table 3-2: Description of BAB coding modes in VM3/VM4

BAB Coding Mode	*Semantics of Decoding Process*
All Zero	The reconstructed BAB contains only BLACK/TRANSPARENT pixels.
All One	The reconstructed BAB contains only WHITE/OPAQUE pixels.
Not Coded	The reconstructed BAB is obtained by motion compensation using the motion vectors of the YUV macroblock. No replenishment of the BAB is performed.
Coded	The reconstructed BAB is obtained by decoding the INTRA MMR codewords.

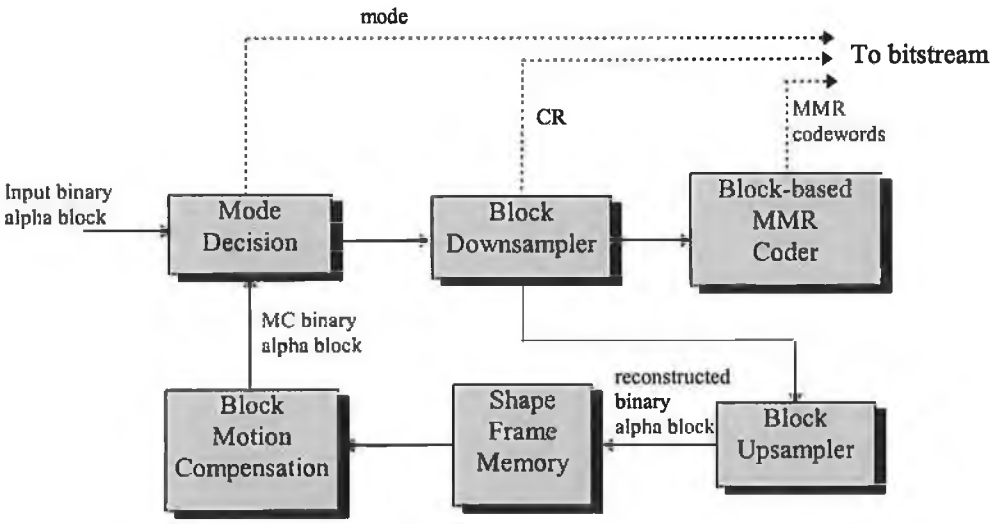


Figure 3-4: VM3/VM4 Binary Shape Coder. The method uses motion compensation to exploit temporal redunancy and a block-based MMR method to exploit spatial redundancy.

For coded BABs, three steps are required for their encoding:

1. Downsampling: This allows lossy coding of shape. A BAB may be downsampled from 16x16 to 8x8 or 4x4 and then the downsampled BAB is encoded. Rate-distortion trade-off points are chosen by setting the downsampling ratio on a per block basis to be 1, 2 or 4.
2. Modified-Modified-Read (MMR): This step applies MMR coding similar to G4 fax coding within the downsampled BAB. Compression ratios of 4:1 or greater are possible for the vast majority of BABs. MMR is inherently a lossless coding technique and the variety employed in VM3 exploits only spatial correlation.
3. Upsampling: Due to the temporal prediction employed in VM3, it is necessary to store the reconstructed alpha map at the encoder. Hence, any downsampled BABs must be upsampled to the nominal size of 16x16 prior to being stored in the reconstructed frame memory.

Using the described coding algorithm, the representation for each BAB contains three types of information (all elements of the representation are compressed using Huffman variable length codes):

- The mode decision to tell the decoder which decoding mode to adopt for the BAB.

- The conversion (downsampling) ratio (CR) to tell the decoder the size of the downsampled BAB and the upsampling ratio required for reconstruction of the BAB.
- The MMR stream consisting of a series of run-length codes and mode codes telling the decoder how to reconstruct a coded BAB.

The VM3 shape representation set the common basis for all further VM solutions. This common basis was defined by the fact that the representation was block-based, that it supported temporal prediction and that lossy coding was provided mainly (although not exclusively[3]) by means of prior downsampling.

3.3.3 The VM5 and VM6 Encoder

In November 1996, VM5 was published **[59]**. A competitive round of experiments resulted in a number of improvements being made to the shape coding algorithm. This improved algorithm outperformed many contour-based algorithms and also one (non-block) technique based on CAE. The major improvements were the following:

- The motion compensation of the BAB used a specially estimated motion vector, which was distinct from the YUV motion vector. The extra motion vector was communicated to the decoder and the coding used a spatial predictive method, whereby only the prediction error of the motion vector, i.e. the motion vector difference, was required to be coded. This change was included because, in general the motion vectors estimated on the basis of YUV data are not optimal for the motion compensation of shape data.
- The basic MMR method was extended to exploit the predicted BAB, i.e. an INTER MMR method was included. This resulted in improved performance towards the lossless end of the distortion range. When lossless or near lossless coding is required, there are many more coded blocks. The inclusion of INTER MMR meant that many of these coded blocks could be compressed more efficiently.

[3] A BAB with almost all white pixels could be encoded by the use of the All One mode. A BAB with almost all black pixels could be encoded by the use of the All Zero mode. A BAB whose motion compensated prediction was less than perfect could be coded using the Not Coded mode. Either of these three measures results in reducing the quality of the representation, while saving bit-rate.

- The MMR method which previously was applied using a horizontal raster scan could now also be applied using a vertical scan. For each BAB, the scan type which produced the smallest code was used. The chosen scan type was signalled in the bitstream.
- Several intricate modifications were made within the MMR method itself to improve the compression efficiency.
- For INTER coded video objects, the coding mode of each BAB was encoded utilising inter-frame correlation. Since VM5 involved the introduction of new coding modes, straightforward VLC coding of the mode information resulted in a considerable generation of bits. The rate of the mode information could be reduced by exploiting the fact that frequently the coding mode of a given BAB did not change much from frame to frame.

The VM5 syntax was more complex than that of VM3. The mode, downsampling and MMR information remained, but there were extra coding modes and some extra information fields. The coding modes are listed in Table 3-3. The increase in the syntax complexity was mainly due to the improved temporal shape prediction using the dedicated shape motion vector and the associated INTER MMR method. The shape motion vector was represented by a prediction process and the encoding of the prediction difference (MVDs – the motion vector difference of shape) in the bitstream. When, as occurs frequently, the prediction difference is zero, this is signalled by a special coding mode as indicated in Table 3-3.

Table 3-3: Description of BAB coding modes in VM5/VM6

BAB Coding Mode	*Semantics of Decoding Process*
All Zero	The reconstructed BAB contains only BLACK/TRANSPARENT pixels.
All One	The reconstructed BAB contains only WHITE/OPAQUE pixels.
INTRA MMR	The reconstructed BAB is obtained by decoding the INTRA MMR codewords.
Not Coded & MVDs ==0	The reconstructed BAB is obtained by motion compensation. No replenishment of the BAB is performed. The motion vector used for motion compensation is obtained by local spatial prediction.
Not Coded & MVDs!=0	The reconstructed BAB is obtained by motion compensation. No replenishment of the BAB is performed. The motion vector used for motion compensation is obtained by local spatial prediction and the addition of a motion vector prediction error (MVDs).
INTER MMR & MVDs==0	The reconstructed BAB is obtained decoding INTER MMR codewords. In order to decode these, a motion compensated BAB is required. The motion vector is obtained by local spatial prediction.
INTER MMR & MVDs!=0	The reconstructed BAB is obtained decoding INTER MMR codewords. In order to decode these, a motion compensated BAB is required. The motion vector is obtained by local spatial prediction and the addition of MVDs.

3.3.4 The VM7 Encoder

The last major step (in April 1997) in the MPEG standardisation process signalled the end of MMR-based coding in MPEG-4 **[61]**. Many researchers considered the method to be very crude and involved, even to the extent of not being understandable or implementable without great difficulty. A block-based context-based arithmetic encoding (CAE) method was proposed by the author. Results showed this CAE method to be superior to MMR for the BAB coding. Despite stiff competition from a very efficient vertex-based algorithm, it was decided that MPEG-4 should stay with a block-based bitmap algorithm. The VM7 shape coding algorithm (built on top of the VM5 method) is summarised in Figure 3-5. In comparison with VM3 of Figure 3-4, it can be seen that the VM7 encoder comprises its own shape motion estimator and an extra decision block for deciding whether the block will be coded by INTRA or INTER modes and for deciding upon the scan type. All these additions were inherited from VM5. The major step of VM7 was the replacement of INTRA MMR and INTER MMR with INTRA CAE and INTER CAE respectively. In the bitstream, the VLC codes of the MMR method were replaced by a single binary arithmetic code (BAC). Another small change was that the mode information for INTRA coded VOPs now utilised a method

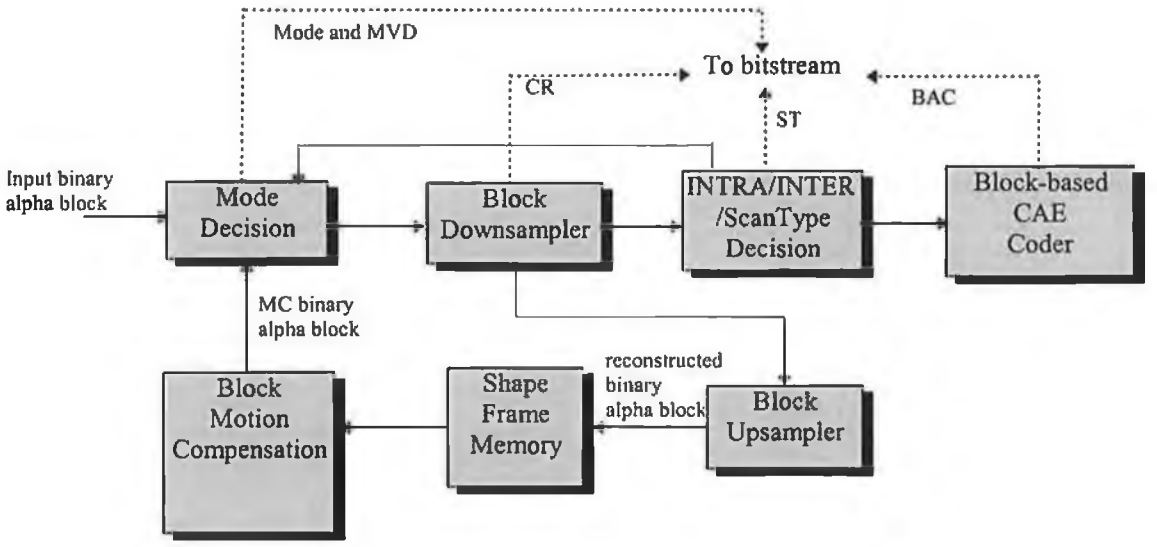


Figure 3-5: VM7 Binary Shape Coder. The encoder is effectively the same as that of VM5 except that the block-based CAE method is included in preference to the previous MMR method.

using spatial prediction, improving the efficiency of the INTRA shape coding algorithm by 4-6%. The remainder of this chapter is devoted to the details of CAE.

3.4 Context-based Arithmetic Encoding in MPEG-4

Context-based arithmetic encoding has been used in JBIG and has now been adopted for MPEG-4. This section discusses the principles and design choices related to CAE coding. The precise details of block-based CAE **[15]**, as employed in the MPEG-4 VM7, are also covered.

3.4.1 Context-based Arithmetic Encoding

Let's assume that there is a random information source which generates a sequence of samples. Each sample X may take on any integer value i between 0 and N-1. Successive samples are independent of each other and are distributed according to a common probability density function represented by $P(X=i)$, for $i=0,...,N$-1. For this source, there exists a lower bound on the number of bits/sample which can be used on average to code a sequence of samples generated by the information source. This lower bound is given by the first order entropy $H(1)$, where:

$$H(1) = -\sum_{i=0}^{N-1} P(X=i)\log_2 P(X=i)$$

Consider now a non-random source where successive samples are in some way dependent. In this case, a joint probability density function must be known. For instance, assuming that samples are to be coded in blocks of two, then the coding bound is given by the second order joint entropy, i.e.

$$H(2) = -\frac{1}{2}\sum_{j=0}^{N-1}\sum_{i=0}^{N-1} P(X=i,X=j)\log_2 P(X=i,X=j)$$

Once again, this represents the lower bound on the number of bits/sample assuming the sequence is coded in blocks of two samples. A lower bound on the bits per sample is obtained by dividing by two. In general, it may be proven that $H(M) \le H(M-1)$. In other words, the larger the block of samples coded, the smaller the coding rate. However, coding approaches based on very high order joint PDFs become very unwieldy. Lossless vector quantisation (VQ) may be considered as one example of a technique utilising joint PDFs. In VQ, coding large blocks of data results in large demands for code-book storage. Generally, VQ methods choose the largest data block size possible while staying within the complexity limits of the given application.

There is an alternative means of using high order statistics in coding. Consider the n-th order joint PDF $P(X(n),X(n\text{-}1),..,X(1))$. Rewrite this PDF as $P(X(n),C)$, where C represents the set of variables $\{X(n\text{-}1),\ldots,X(1)\}$. The n-th order conditional entropy may now be constructed according to:

$$H(n) = -\sum_{\forall j}\sum_{i=0}^{N-1} P(X=i,C=j)\log_2 P(X=i|C=j)$$

As may be noted, the conditional entropy depends on a conditional probability P(X|C) of order n-1. This representation of entropy suggests a context-based coding approach. In contrast to the joint coding approach, samples are coded one at a time and each variable X is coded using a conditional PDF which depends on a context C as defined

above. Context-based arithmetic encoding (CAE) is a coding system based on exploiting conditional probabilities in this manner. As with n-th order VQ, the lower bound on the CAE approach is still given by the n-th order entropy. In addition, this approach has similar difficulties with respect to the utilisation of high order statistics. As will be explained, as the order increases, the context-based approach the storage demands also increase.

The analysis to this point suggests that lossless VQ and CAE should yield similar coding performance. A simple example reveals that the conditional approach may be more efficient in some situations.

Assume a binary source with $P(0)=0.5$ and $P(1)=0.5$. Consider a coding approach exploiting the second order joint PDFs of this source, where $P(0,0)=0.375$, $P(0,1)=0.125$, $P(1,0)=0.125$ and $P(1,1)=0.375$. The second order joint entropy yields a figure of 0.91 bits/sample. Noting the relationship $P(A|B) = P(AB)/P(B)$, the first order conditional PDFs are computed as $P(0|0)=0.75$, $P(0|1)=0.25$, $P(1|0)=0.25$ and $P(1|1)=0.75$. The resulting conditional entropy evaluates to 0.82 bits/sample.

The hypothesis that conditional coding approaches are superior to joint coding approaches is not explored in any more general way. However, sub-section 3.4.7 includes some interesting comparisons between CAE and VQ for block-based binary shape encoding. The following sub-section explains how context-based arithmetic encoding may be used to encode binary images.

3.4.2 Binary Image Coding Using CAE

In most binary images, a high degree of local correlation exists. For a given pixel j, if all its neighbours are WHITE, then it is highly likely that the value of pixel j is also WHITE. Conversely, if all its neighbours are BLACK, then it is highly likely that the value of pixel j is also BLACK. In general, the values of the pixels in the neighbourhood of j will dictate the PDF of the pixel value at j.

When encoding/decoding a binary image using CAE, pixels are usually encoded/decoded in raster order. When decoding the value at pixel *j*, the decoder knows the values of all the pixels above and to the left of *j*. However, the decoder does not know what are the values of pixels below and to the right of *j*. This places a simple causal constraint on the neighbourhood pixels which can be used to infer the PDF. A very common neighbourhood which is used in JBIG is depicted in Figure 3-6. In the video coding community, the more usual term for these local neighbourhoods is a *template.*

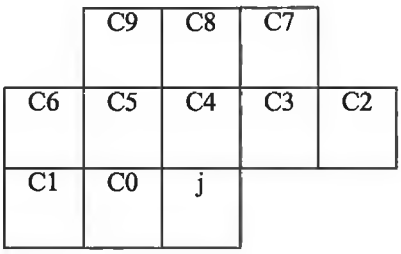


Figure 3-6: An example of a neighbourhood template. The pixel to be coded is indicated with a 'j' and all pixels which are part of the template are numbered C0 through C9.

The formation of the pixels in a given template can be represented by a ten bit number called the context number. The PDF of the pixel *j* is conditioned upon this context number. Effectively, the pixel *j* has 2^{10} possible PDFs. These together form what is termed the probability table. Hence, the encoding of the pixel *j* involves 3 simple steps:

1. Compute the context number.
2. Use the context number to access the correct PDF from the probability table.
3. Use the PDF and the actual value of the pixel *j* to drive an arithmetic encoder **[95]** which stores bits in the output buffer.

After applying this procedure to every pixel in the binary image, the whole image is represented by a single (atomic) arithmetic code stored in the output buffer. The decoding procedure at each pixel is equally straightforward:

1. Compute the context number.

2. Use the context number to access the correct PDF from the probability table.
3. Use the PDF and the bits from the arithmetic code to decode the pixel value.

The process is the same at every pixel. The efficiency of the resultant arithmetic code depends on how close the PDFs are to the real underlying PDFs of the binary image. There are two approaches to defining the PDFs to be used for encoding: adaptive models and fixed models.

3.4.3 Fixed vs Adaptive PDF Models

In a fixed model arithmetic encoder, the probability tables are fixed prior to encoding and they are never allowed to change during encoding. These fixed probability tables are derived by using a training procedure involving a large data set of representative images. The probability tables will, therefore, be optimal in the average sense across this training set. If an image whose statistical characteristics are not captured in this training set is to be coded, then coding inefficiency is to be expected.

The adaptive model overcomes this dependance on a training set by adapting the probabilities as each new pixel is encoded. After a considerable number of pixels have been encoded, it would be expected that the probability tables would have adapted to the specific statistics of the image being coded. If given a sufficiently large set of samples to allow effective adaptation, adaptive models can outperform fixed models. For example, adaptive schemes are used in JBIG because the image resolutions are typically very high, thereby containing sufficient data to allow for the adaptation to take place. On lower resolution images, a fixed table would be the best choice because there are not enough samples to cause proper adaptation.

Adaptive models have two disadvantages worthy of mention. The first relates to complexity. In order to maintain an adaptive table, extra processing is required at each pixel and many update strategies are quite involved computationally. The second disadvantage relates to error resilience. If the arithmetic code is subjected to bit errors then pixels will be decoded in error. Consequently, the adaptive probability tables at the

encoder and decoder will diverge and no resynchronisation will be possible without re-initialising the adaptive tables. Therefore, it is preferable to use fixed tables in situations where error-free transmission is unattainable and where sufficiently long periods of sustained error-free transmission are not guaranteed. Because video transmission systems do not, in general, guarantee sustained error-free transfers, the CAE method of VM7 uses only a fixed probability table. The probability tables were generated from a training set. For each context number, the table stores the probability that a pixel has a zero value. The probabilities are stored in 16 bit precision.

3.4.4 Templates and Contexts

In the MPEG-4 VM7, the JBIG template was the obvious choice for INTRA coded BABs. For INTER coded BABs, it was required to exploit the motion compensated prediction for the BAB. The most natural way to do this was to construct a template that included pixels from the current BAB *and* the predicted BAB. While doing this, it was decided to impose that the probability tables have no more than 1024 entries, i.e. no more than ten bits in the context. There were three reasons for this:

1. The probability tables should be small enough to be stored in on-chip processor cache memory.
2. The training of probability tables becomes very difficult when there are a large number of contexts.
3. The increase in efficiency by using larger contexts was found to be marginal, i.e. there is saturation of the performance as one increases the size of a template beyond a certain point. This point has been illustrated by the results presented by Moffat in **[52]**.

The following choices were made based on experimental findings: (1) For INTRA coded BABs, a 10 bit context $C = \sum_k c_k \cdot 2^k$ is built for each pixel as illustrated in Figure 3-7a. (2) For INTER coded BABs, a 9 bit context $C = \sum_k c_k \cdot 2^k$ is built for each pixel as illustrated in Figure 3-7b.

3.4.5 Block-based CAE

CAE had previously been applied to binary image coding within JBIG. It has also been specifically adapted for object shape coding **[10]**. However, in JBIG, CAE was used to compress image data blocks containing, typically, tens of thousands of pixels. In MPEG-4, CAE is applied to blocks, i.e. BABs containing at most 256 pixels. The process of coding each BAB is as follows. The arithmetic encoder is initialised. Pixels are scanned and coded in raster order. When the last pixel has been encoded, the arithmetic encoder is flushed and the arithmetic code is terminated.

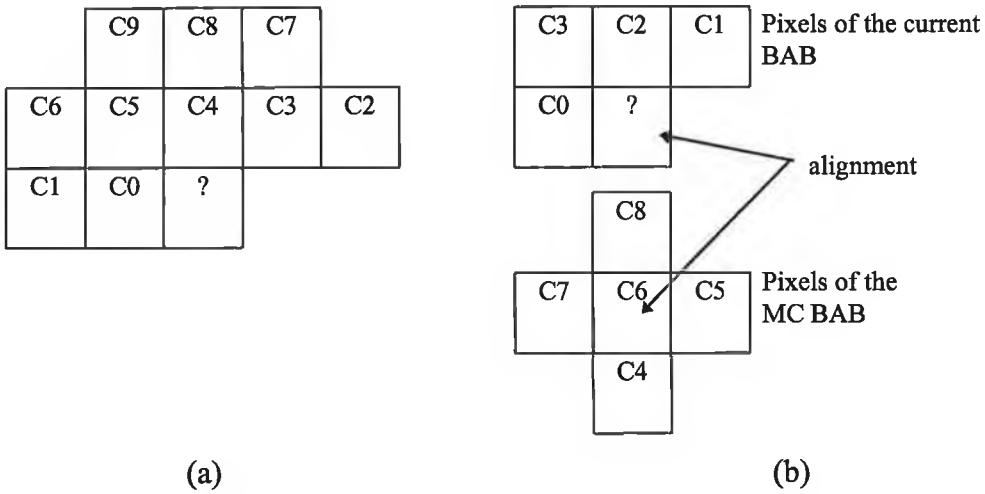


Figure 3-7: (a) The INTRA template and context construction. (b) The INTER template and context construction. The pixel to be coded is marked with '?'.

One disadvantage with this scheme is that the continual initialisation and flushing of the arithmetic encoder introduces inefficiency. It is important that these initialisation and flushing mechanisms are as efficient as possible. However, even with the most efficient of implementations (as in VM7), it is known through experiment that approximately 1 bit per BAB is being used for the repeated initialisations and flushings. In order to benefit from the aforementioned advantages of block-based coding syntaxes, this inefficiency must be tolerated. In any case, it is more than compensated for by the ability of the coding mode to be adapted on a block-by-block basis.

Another potential problem is that the efficiency of the final compressed BAB code is highly dependent on the manner in which pixels outside the BAB are treated. The

problem is caused by the fact that the coding of many pixels near the edge of the BAB relies upon a template and a context construction that includes pixels outside the BAB. The same problem is faced when operating in the non-block based mode, whereby the template for pixels close to the image borders may contain pixels that are outside the image border. The simple approach, in this case, is to set all pixels outside the image to BLACK. Applying an analagous strategy when coding a given BAB, all pixels outside that BAB are set to zero during context constructions. This is a very inefficient solution since it introduces artificial edges between BABs and these result in an increased coding bit-rate. A more efficient solution is to use, where possible, the actual pixel values, even if the pixels lie outside the current BAB. The approach taken is as follows.

Figure 3-8 shows a BAB surrounded by a 2 pixel wide border. Assume that this BAB is about to be encoded, and that already, the BABs above and to the left of this BAB have been encoded. This is the normal situation when the BABs of an image are processed in raster order. Of the border pixels, those in the region ABCD are contained within BABs which have already been coded. These pixels can be accessed to build the required contexts in the current BAB. On the other hand, those border pixels in the region marked U are contained within BABs which have not been encoded yet. These pixel values cannot be used in the context constructions for the current BAB. This is due to the fact that these pixels will not be known at the decoder, when it is about to decode the given BAB. A simple modification in the context construction is used to "estimate" the values of these "unknown" pixels. When constructing the INTRA context of Figure 3-7a, the following steps are taken in sequence.

1. If (*C7* is unknown), *C7=C8*,
2. If (*C3* is unknown), *C3=C4*,
3. If (*C2* is unknown), *C2=C3*.

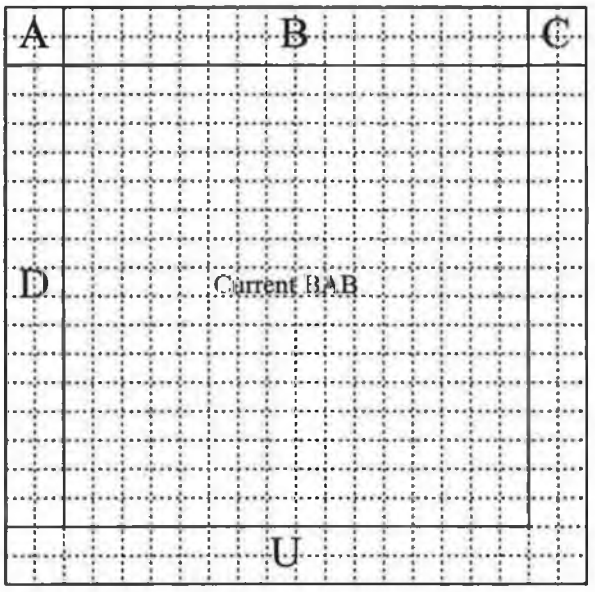


Figure 3-8: Bordered BAB. A:TOP_LEFT_BORDER. B:TOP_BORDER. C:TOP_RIGHT_BORDER. D: LEFT_BORDER. U: pixels which are unknown when decoding the current BAB.

When constructing the INTER context of Figure 3-7b, the following conditional assignment is performed. If (*C1* is unknown), *C1=C2*. This simple template padding approach provides a good way to avoid the effect of articifical discontinuities introduced by regular assumptions on the values of these "unknown" pixels.

Another important detail is in the construction of the motion compensated BAB. Although, a BAB is 16x16 pixels, the nature of the chosen INTER template requires access to motion compensated pixels within a one pixel border around the usual 16x16 motion compensated block as shown in Figure 3-9. That is, it is typical to copy an 18x18 motion compensated block from the previous alpha map, when constructing INTER contexts.

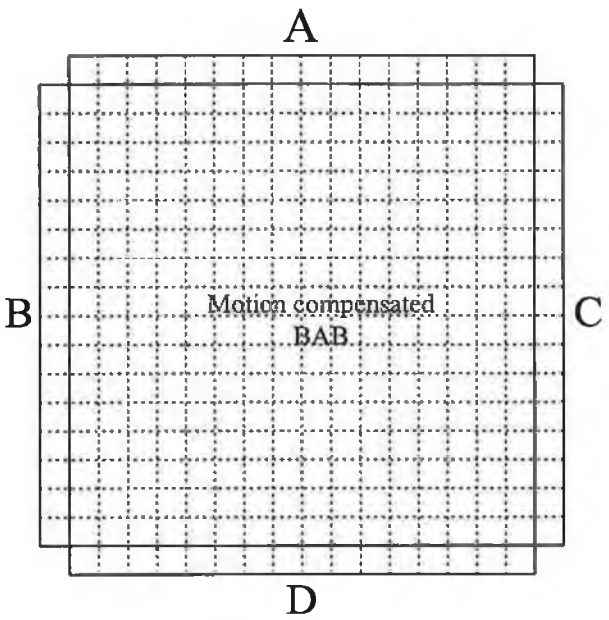


Figure 3-9: Bordered motion compensated BAB. A: TOP_BORDER. B: LEFT_BORDER. C: RIGHT_BORDER. D: BOTTOM_BORDER.

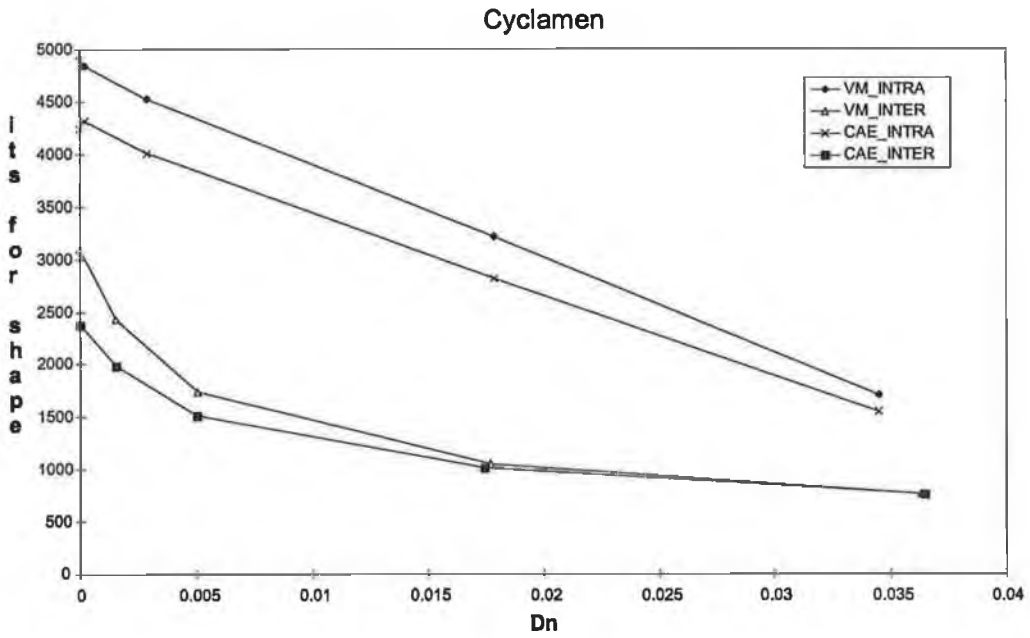
3.4.6 Simulation Results

In order to show the efficiency of the block-based CAE approach, it has been directly compared with a block-based MMR approach. The MMR algorithm used is that which was used in VM5 of the MPEG-4 development **[60]**. In the graphs provided in Figure 3-10, the effect of replacing the block-based MMR algorithm of VM5 with the block-based CAE algorithm is illustrated. Each graph plots the number of bits for shape per VOP against a shape distortion measure *Dn*. *Dn* is defined as the number of incorrectly coded pixels divided by the number of WHITE pixels in the original shape. The coding algorithms were run using 5 different distortion levels in order to produce the graphs. A given distortion level is achieved by setting a distortion parameter within the algorithms. This distortion parameter defines the maximum allowable distortion within each macroblock. For instance within each macroblock, the down-sampling factor, described in sub-section 3.3.2, is chosen such that this distortion threshold is not exceeded. In addition, a macroblock may not be compressed with the *not-coded* mode, if this also means exceeding the distortion threshold.

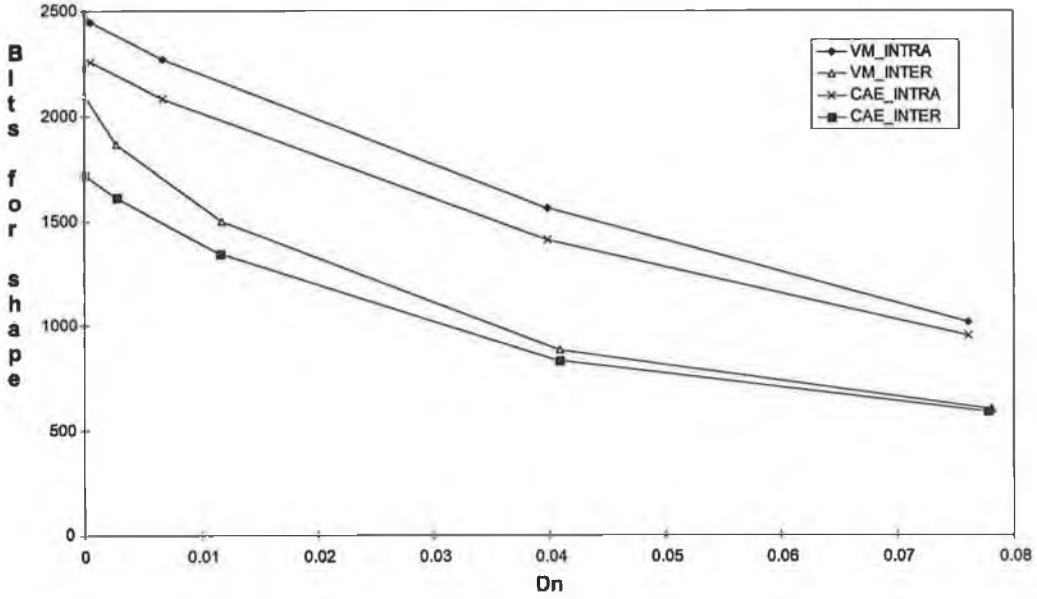
Four shape coding algorithms are compared:

- VM5 INTRA (including MMR coding)
- VM5 INTER (including MMR coding)
- VM5+CAE INTRA
- VM5+CAE INTER

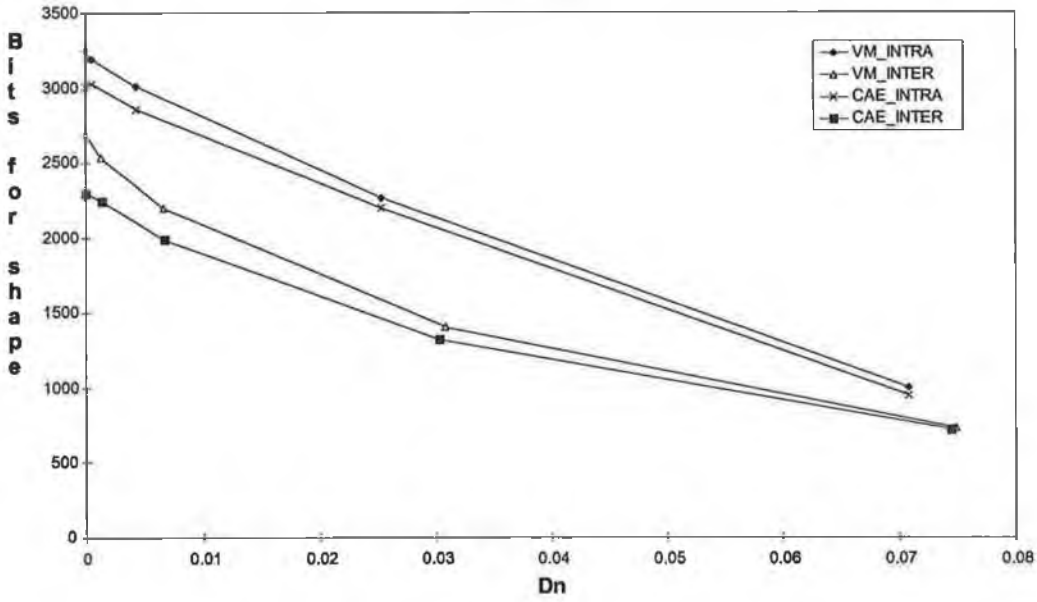
The sequences used were part of the MPEG-4 test set, both QCIF and SIF. The test set contained a mixture of synthetic and natural shapes, with varying degrees of motion complexity. All sequences were encoded at 10Hz.



Kids


Bits for shape
2500
2000
1500
1000
500
0
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
Dn
VM_INTRA
VM_INTER
CAE_INTRA
CAE_INTER

Robot


Bits for shape
3500
3000
2500
2000
1500
1000
500
0
0
0.01
0.02
0.03
0.04
0.05
0.06
0.07
0.08
Dn
VM_INTRA
VM_INTER
CAE_INTRA
CAE_INTER

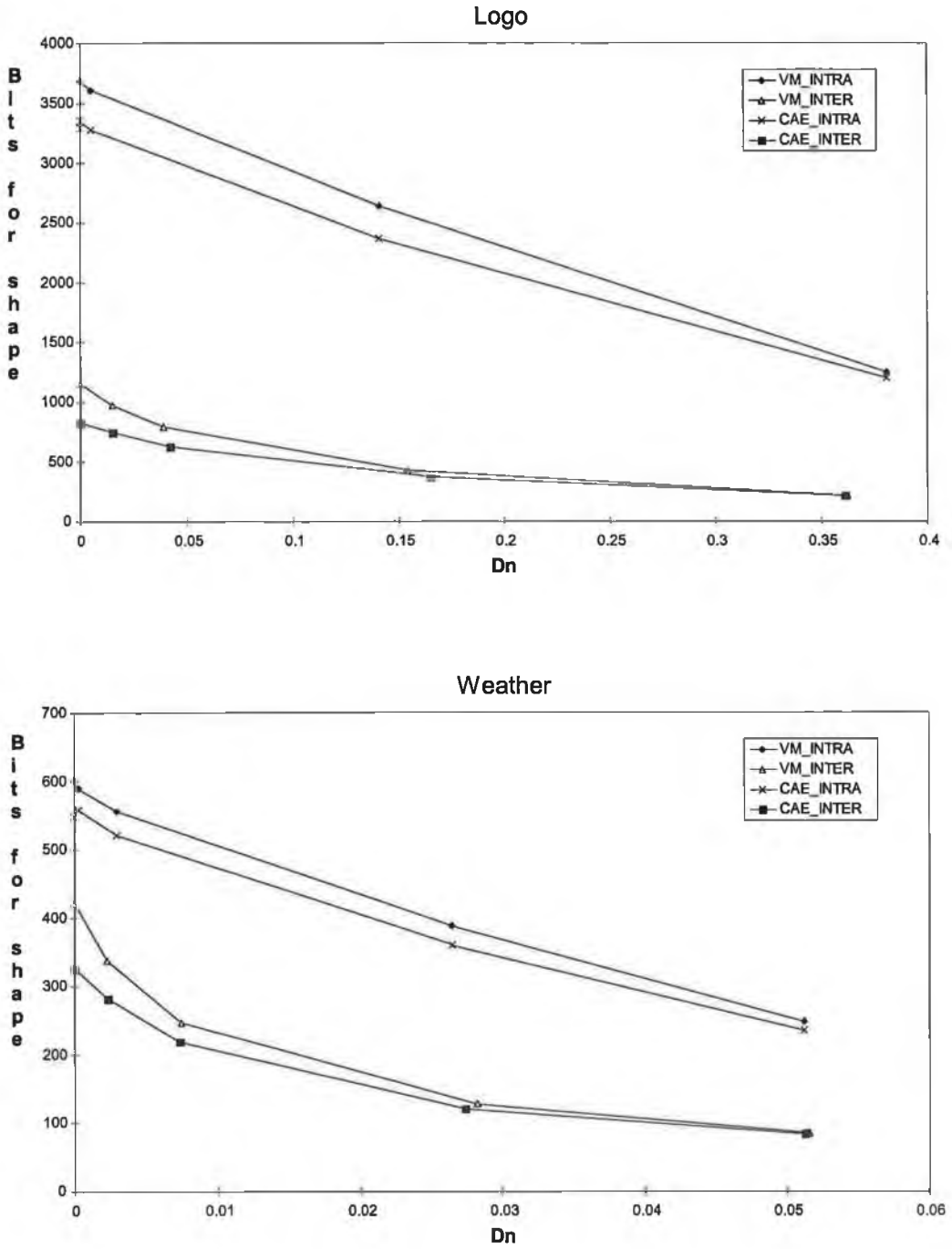


Figure 3-10: Simulation results illustrating the efficiency of CAE by comparison with the MMR solution of VM5. All sequences were encoded at 10 Hz frame rates. The sequences Cyclamen, Kids, Robot, and Logo are SIF (352x240 pixels) and the Weather sequence is QCIF (176x144 pixels).

The general trend shown in these graphs is that CAE_INTER > VM_INTER > CAE_INTRA > VM_INTRA where ">" may be understood as 'is more efficient than'. For INTRA coded shape, CAE results in improvements of between 3% and 11%. For INTER coded shape, the gains are in the range 10-28% towards the lossless end of the

distortion range. For very lossy (i.e. high *Dn*) INTER coded shape, the gains due to the use of CAE are very small. This is simply explained by the fact that the majority of shape blocks are coded by motion compensation and without subsequent MMR/CAE coding. That is, the *'not coded'* modes are used frequently and it is natural that the MMR and CAE-based algorithms give similar performance. It is believed, however, that binary shape coding algorithms will primarily operate in a lossless or near lossless mode. This is due to the fact that very lossy shape can lead to disturbing artefacts in the reconstructed sequence, as illustrated in Figure 3-11. These artefacts are more noticeable when viewing the sequence in real time and the resultant video is unlikely to be of sufficient quality for high-end applications, e.g. post-production for TV and film.

(a) (b)

Figure 3-11: Illustration of shape distortions. (a) coded VOP with lossless shape, (b) coded VOP with lossy shape.

It may be concluded from these results that CAE is a superior compression technique for binary shape coding. It is also interesting to note that the use of INTER coding brings appreciable gains over INTRA shape coding. On the 10Hz sequences used in the simulations, it is observed that up to 2000 bits/VOP may be saved by using INTER shape coding. This corresponds to a bit-rate saving of 20Kbits/s. Coding the sequences at 30Hz would naturally lead to increased savings by INTER coding.

3.4.7 On the Consideration of CAE vs VQ

Having presented the block-based CAE approach, the focus now returns to the suggestion made in sub-section 3.4.1, i.e. that VQ and CAE may be equally effective for

lossless compression. Here, some comparisons are made which suggest that CAE may be the more effective of the two techniques for the application of block-based binary shape coding.

The INTRA CAE approach proposed above for block-based shape coding uses a 10-bit context and requires 1024 (16-bit) words to store the probability table. According to the definitions of sub-section 3.4.1, this approach exploits 11-th order statistics. In order to construct a lossless VQ approach exploiting 11-th order statistics, a VQ block size of 11 pixels is required. With respect to the CAE approach, the VQ approach has two drawbacks. Firstly, the 11-th order VQ approach requires 2048 words to store the probability tables (assuming an arithmetic coding approach). Secondly, it is clearly not possible to completely tile a 16x16 shape block with a "tile" of size 11 without covering some pixels twice. This is inefficient in the coding sense, since it implies that the 11-th order VQ approach must code some pixels twice.

The choice of 11-th order statistics in the above comparison may be seen as slightly unfair. It is true that the same conclusion could not be made if 4-th order, 8-th order or 16-th order statistics had been chosen. After all, the tiling problem disappears if the VQ block size is chosen to be 4, 8 or 16. Nevertheless, the comparison does illustrate that the contextual coding approach is perhaps more flexible than the joint coding approach. In the block-based application, it appears that CAE has the ability to maximise coding efficiency irrespective of the statistics order. As illustrated in the tiling analogy, VQ does not have this property. This conclusion may be further expanded as follows. The flexibility inherent in CAE also allows total freedom in the choice of the template size *and* shape. Given a set of sources, each individual source may call for a different template size and shape. CAE can meet this requirement without being hindered by the 'tiling' constraint. In adapting VQ to the same sources, the 'tiling' constraint is always an obstacle to choosing the optimal block shape. As an example, consider an image exhibiting 8-th order joint statistical dependence. A VQ approach, wishing to avoid coding any given pixel twice, is constrained to employing a 2x4 or a 4x2 pixel block. No such constraints exist in the case of choosing the CAE template. It is ventured that this flexibility gives CAE a significant advantage over VQ.

Another interesting comparison can be drawn based on lossy compression. By its very name and by most of it applications, loss is an inherent feature of vector quantisation. In lossy VQ applications, uncommon block configurations are mapped to the closest code-book entry. It should be noted that lossy VQ increases requirements in terms of storage and necessitates the use of a search mechanism for finding the closest code-book entry for a given block configuration. Nevertheless, this lossy coding ability gives VQ an advantage over CAE. To the best knowledge of the author, CAE has never been applied for lossy coding purposes. For CAE, loss is usually introduced by suitable pre-processing, as in the down-sampling used in MPEG-4. This does not mean that no means exist to make CAE inherently lossy, it only suggests that a means remains to be found. Neither does it suggest that this makes the CAE approach any less efficient that VQ for lossy compression. Experimental evidence may be required to draw more decisive conclusions on this issue.

3.5 Summary and Future Work

Several classes of shape coders have been reviewed. The general advantages of bitmap-based techniques over contour-based methods are outlined. CAE, a bitmap-based method, has been proposed as a very efficient and flexible coding tool. Its adaptation to allow operation within a block-based framework has been described in detail. In particular, it is shown how CAE can be extended to exploit temporal correlation. Results have shown that CAE used in a motion compensated block-based coding algorithm is highly efficient, outperforming an advanced MMR algorithm. CAE forms the core of the MPEG-4 shape coding solution for this reason, thus enabling many new object-based functionalities in a most efficient way. The impetus of MPEG-4 has given rise to phenomenal technological advances in a short period of time. This has resulted in a very efficient and flexible solution for shape coding. The rate for the video object's shape information very much depends on the complexity and movement of the shape and the desired quality. It can be seen from the results presented above that bit-rates vary from 1 Kbits/s for small simple shapes at low quality, to 30 Kbits/s for larger more complex object shapes at high quality. Several directions of research are being followed to add the necessary remaining functionality. Efforts are being made to develop an

enhancement layer shape coder to enable a spatially scalable representation of shape. Other efforts are being made to provide interlaced shape coding tools based on CAE. Finally and most importantly perhaps, a large effort is being made to modify the basic representation so that it is more resilient to bitstream errors. Some preliminary ideas on these new aspects are given in **[16]**.

4. POLYNOMIAL MOTION MODELLING

The estimation of motion is extremely important for video compression applications. Most of today's video representations derive a substantial degree of their efficiency from exploiting interframe redundancy by way of motion compensation. For video coding purposes, simple translational motion models have been used most commonly. These have the advantage of being easy to compute and efficient to encode. Furthermore, motion is estimated for every 16x16 pixel block and more often than not, translational models suffice in synthesising the motion within these small image regions. However, there are other applications where the motion representation must be capable of representing general 3D motions. Some early 3D motion models and estimation methods were presented by Netravali and Salz in **[66]**. The more recent emphasis on object-based video representation has provoked interest in motion segmentation as a means of automatically recovering object shape. Segmentation approaches based on motion tend to utilize more complex motion models capable of representing more general types of motion, i.e. rotations and zooms. The use of these more complex models in segmentation systems typically results in simpler, more intuitive, segmentation results. Most of the successful applications of these more general models have been in segmentation, but they have also been applied for coding oriented tasks. For example, they have been experimented with in place of the translational models in block-based coders **[65]**, for temporal interpolation of video sequences **[7]**, and for region-oriented (non-block-based) motion compensation **[43]**. In addition, the use of motion in object tracking systems is very important and as such, accurate motion models are called for. For example, see the tracking system described in chapter 6.

The representation of general 3-D motion can be achieved very well by using simple polynomial functions. This chapter is devoted to the study of these polynomial motion models. The theoretical justification for their use is discussed and the estimation methods are covered in detail. The main new contributions by the author comprise comparative studies of several estimation methods and the development of fast estimation algorithms. Due to their relevance to the segmentation algorithms presented

in forthcoming chapters, the chapter concludes with a discussion of robust methods of estimation.

4.1 Motion Models and Optical Flow

In most image processing applications, motion is treated as a transformation which describes an inter-frame mapping of pixels. That is, the pixel (x,y) at time $t+1$ corresponds to a pixel at (x',y') at time t, which has effectively moved between the two spatial positions. This is also the essence of what is termed optical flow **[2],[5]**. The motion at (x,y) is described by a displacement vector $(dx,dy) = (x-x',y-y')$. While this adequately describes local motion, it is typically more useful to use a parametric form to describe all the local motions within a given region. In the current video compression standards, this parametric model corresponds to $(dx,dy) = (a_0,b_0)$, where a_0 and b_0 are the motion parameters to be estimated. This can be termed the constant model and is capable only of representing horizontal and vertical translational movements in planes parallel to the image plane. This model is inherently limited, but has proved useful for block motion compensation where the blocks are quite small, i.e. 16x16 pels. Recently, with the increased interest in object-based representations, more complex motion models are being investigated. For example, the affine model $(dx,dy) = (a_0+a_1x+a_2y, b_0+b_1x+b_2y)$ has six parameters and is capable of representing translations, rotations within the plane parallel to the image plane as well as other geometric transformations, e.g. zoom and shear. It is desirable, however, to be able to mathematically represent any arbitrary 3-D motion and in particular, to be able to represent the 2-D perspective projection of this motion onto the image plane. Fortunately, it is possible to derive the form of this representation **[34]**. We now follow the derivation of the equations describing the 2-D optical flow field due to an arbitrary 3-D motion.

Let us assume we have a moving camera and a fixed 3-D scene. A co-ordinate system is fixed with respect to the camera and the z-axis is parallel to the optical axis. Take a point P in the 3-D scene with co-ordinates (X,Y,Z). This point can be undergoing a translation and/or a rotation and the velocity is given by $\mathbf{V} = -\mathbf{t} - \mathbf{w} \times \mathbf{r}$ where

$\mathbf{r} = [X \quad Y \quad Z]^T$

$\mathbf{t} = (U \quad V \quad W)^T$, translations in X, Y and Z directions.

$\mathbf{w} = (A \quad B \quad C)^T$, rotations about X, Y and Z axes.

The velocity equation can be expanded out in component form.

$$\dot{X} = -U - BZ + CY$$
$$\dot{Y} = -V - CX + AZ$$
$$\dot{Z} = -W - AY + BX$$

Equation 4-1

Assuming perspective projection and a focal length of 1 in the camera, the corresponding image point $p = (x,y)$ is related to P as follows

$$x = \frac{X}{Z} \text{ and } y = \frac{Y}{Z}.$$

Equation 4-2

The optical flow at a point (x,y) in the image is represented by (u,v) where,

$$u = \dot{x} \text{ and } v = \dot{y}.$$

Equation 4-3

By differentiating the expressions for x and y and substituting the derivatives of X, Y and Z, the velocities within the image plane (the optical flow) are given by:

$$\dot{x} = \frac{-U + xW}{Z} + Axy - B(x^2 - 1) + Cy$$
$$\dot{y} = \frac{-V + yW}{Z} + A(y^2 + 1) - Bxy - Cx$$

Equation 4-4

Notice that this is, in fact, a quadratic function of the image co-ordinate (x,y) and has 6 parameters, i.e. (U,V,W,A,B,C) plus a surface or depth parameter Z. In general, the static 3-D scene structure may be represented by a function $Z(X,Y)$. Therefore, the optical flow field is due to some 3-D motion and some surface structure. When estimating motion according to the above models, there exists the problem of estimating the surface function $Z(X,Y)$. Surface structure can be recovered by stereo image processing **[70]** or in the absence of multiple cameras and under the assumption of rigid object motion, so-called structure from motion approaches can be used. In our application, the surface structure is assumed to have a predefined structure, i.e. planar or parabolic surface models are assumed. Under this assumption, the surface model parameters become implicit within a polynomial motion model form as explained in the following text.

As a common example, letting the surface be a planar surface, it can be shown that:

$$\frac{1}{Z(X,Y)} = K + Lx + My$$

Substituting this into the Equation 4-4 yields the 8 parameter motion model (labelled B in Table 4-1). Table 4-1 summarises several common motion models and highlights the important assumptions which have been made in their derivation. Both Dugelay and Sanson in **[27]** and Diehl in **[28]** discuss the usefulness of these model forms.

Table 4-1: Common motion models and underlying assumptions

	Model Form	Assumptions
A	$x' = (1+a_1)x + a_2 y + a_3$ $y' = a_4 x + (1+a_5)y + a_6$	• planar surface, translational motion or rotational motion about an axis perpendicular to the image plane
B	$x' = a_1 + (1+a_2)x + a_3 y + a_7 x^2 + a_8 xy$ $y' = a_4 + a_5 x + (1+a_6)y + a_7 xy + a_8 y^2$	• planar surface, general 3-D motion
C	$x' = a_1 x^2 + a_2 y^2 + a_3 xy + (1+a_4)x + a_5 y + a_6$ $y' = b_1 x^2 + b_2 y^2 + b_3 xy + b_4 x + (1+b_5)y + b_6$	• parabolic surface, general 3-D motion

As can be seen from Table 4-1, with strict planar or parabolic assumptions on the nature of the surface structure, the motion model form is required to be a quadratic function of the (x,y) image co-ordinates. The commonly used affine models thus constitute an approximation to the general motion forms. This approximation is quite accurate when the moving objects are distant from the camera or when the rotational motion components are within a plane parallel with the image plane.

4.2 Problem Formulation for Motion Estimation

Given images I_{t+1} and I_t, i.e. consecutive images from a video sequence, the task is to compute a parametric description of the inter-frame motion. Since the sequence is likely to contain several independently moving areas, the usual approach is to partition the image into non-overlapping regions or, in the simplest approach, blocks. As such, the final motion description is a combination of the partition method and the motion parameters defined on each partition segment. Equivalently, it can be said that the inter-frame motion field is a piece-wise continuous function. Let us therefore assume that some partition of the image space is available, Z_{t+1}, which describes various non-overlapping regions within the image. For each region R in Z_{t+1}, a motion model form must first be chosen. For the reasons given previously, the general polynomial function is a good choice. This function is described by:

$$\mathbf{x}' = \mathbf{x} + \mathbf{B}^T(\mathbf{x})\theta$$

Equation 4-5

where $\mathbf{x} = (x \quad y)^T$ is the pixel location within the image I_{t+1} and $\mathbf{x}' = (x' \quad y')^T$ is the corresponding displaced location within the image I_t. The coefficients of the polynomial, i.e. the motion parameters in this case, make up the vector θ. The *xy* power terms are in the matrix **B**. A simple example illustrates the mathematical structure of the motion model. For a first order model (affine), we have

$$\mathbf{B}^T = \begin{pmatrix} 1 & x & y & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & x & y \end{pmatrix}$$
$$\theta^T = (\theta_0 \quad \theta_1 \quad \theta_2 \quad \theta_3 \quad \theta_4 \quad \theta_5)$$

Note that, for an n^{th} order polynomial motion model, θ is an m-dimensional vector, where $m = (n+1)(n+2)$.

Equation 4-5 describes a mapping between two co-ordinate systems. In terms of motion or optic flow, it implies that the pixel at *(x′,y′,t)* has moved to *(x,y,t+1)*. Thus, the motion estimation problem can be formulated by:

$$I_{t+1}(\mathbf{x}) \approx I_t(\mathbf{x}')$$

or, alternatively,

$$I_{t+1}(\mathbf{x}) = I_t(\mathbf{x}') + \Delta e(\mathbf{x}).$$

Equation 4-6

The construction of an objective function based on the brightness constraint leads to a least squares optimisation approach. For a given region R, the objective function is

$$e(\theta) = \sum_{\forall \mathbf{x} \in \mathbf{R}} \left(\Delta e(\mathbf{x})\right)^2 = \sum_{\forall \mathbf{x} \in \mathbf{R}} \left(I_{t+1}(\mathbf{x}) - I_t\left(\mathbf{x}'(\theta)\right)\right)^2$$

Equation 4-7

i.e. a sum-squared error function. Equivalently, the objective function may be the mean of the squared errors, i.e.

$$e(\theta) = E\left(\left(I_{t+1}(\mathbf{x}) - I_t\left(\mathbf{x}'(\theta)\right)\right)^2\right)$$

Equation 4-8

where *E()* is the expectation or mean over the region R. The task of motion estimation is then to find the motion parameter vector θ^* which minimises Equation 4-8. Note that, this is a non-linear least squares problem because the function $I_t\left(\mathbf{x}'(\theta)\right)$ is not linear in the unknown motion model parameters.

4.3 Estimation Methods

The optimisation of non-linear least squares objective functions is performed using variants of the Newton method. An interesting comparison of methods was carried out by Dugelay and Sanson **[28]**, involving the Newton method, the Gauss-Newton method and the adaptive gain gradient method. The conclusion was that the Newton methods were comparable, while being more effective that the gradient method. Here, the Gauss-Newton (GN) method of **[81]**, **[82]** and **[83]** and a Quasi-Newton (QN) method similar to that in **[27]** are compared. As will be seen, these optimisation algorithms are iterative and must be initialised with an appropriate guess. For the particular problem of motion estimation, these techniques are usually embedded in a multiresolution image framework **[6]** since this:

- aids in the accurate modelling of large motions,
- adds robustness by reducing the effect of noise ,
- reduces implementation complexity, and
- reduces the chances of the estimator being trapped in local minima.

4.3.1 Newton's Method

The Newton method is an iterative method used to find function minima. Let $f(\mathbf{x})$ denote an arbitrary multivariate function to be minimised. An initial estimate $\mathbf{x}_0$ of the function's minimum must be provided. Starting from this point, the Newton method computes a search direction $\mathbf{p}_0$, which is said to be a descent direction. That is, by moving from the starting point along this direction, a reduction in the evaluation of the function can be attained. Given this descent direction, a new search point $\mathbf{x}_1$ lying some specified distance in the descent direction is chosen. The minimisation proceeds by stepping along the descent directions through the parameter space until a minimum is encountered. The minimum is assumed to have been encountered when a specified termination criterion has been satisfied. Most termination criteria rely on the fact that the function gradient (i.e. its first derivative) in the neighbourhood of the minimum is close to zero. A minimum $\mathbf{x}^*$ might be chosen which satisfies $\left|f'(\mathbf{x}^*)\right| \leq T$ where T is some predefined threshold with a value close to zero.

The Newton method, while not being the only technique used in minimisation, is widely acknowledged as the most reliable and most efficient. Nevertheless, it has several problems. Firstly, it is only capable of finding the local minimum in the catchment area of the initial estimate $\mathbf{x}_0$. For complex functions with many spurious minima, a straightforward application of the Newton method is unlikely to give satisfactory results, since the search is very likely to terminate at a local minimum. Secondly, the specification of foolproof termination criteria is difficult. Very often, Newton searches will terminate on function saddle-points rather than minima, since saddle-points also have zero gradients. For each particular problem, the Newton method must be carefully adapted to avail of prior knowledge of the function characteristics or be augmented with procedures for function simplification and search initialisation.

The basic Newton iteration is obtained by locally approximating the error function using a second order Taylor series, i.e.

$$e(\mathbf{x}+\mathbf{p}) = e(\mathbf{x}) + \mathbf{g}(\mathbf{x})^{\mathrm{T}}.\mathbf{p} + \mathbf{p}^{\mathrm{T}}\mathbf{H}(\mathbf{x})\mathbf{p}$$

Equation 4-9

where

$$\mathbf{g}(\mathbf{x}) = \nabla \mathbf{e}(\mathbf{x}) = \left\{ \frac{\partial e(\mathbf{x})}{\partial x_0}, \ldots\ldots, \frac{\partial e(\mathbf{x})}{\partial x_n} \right\}$$

is the gradient vector of the error function containing the first order derivatives of the error function and where $\mathbf{H}(\mathbf{x}) = \nabla^2 \mathbf{e}(\mathbf{x})$ is the Hessian matrix of the error function containing the second order partial derivatives $\frac{\partial e(\mathbf{x})}{\partial x_i \partial x_j}$. For error functions which can be locally approximated by the Taylor series of Equation 4-9, we should expect robust optimisation results and fast convergence properties. Non-optimal results can be expected when functions do not adhere to this assumption. The Newton iteration finds the step $\mathbf{p}$ which minimises this locally approximated quadratic function. This is done by taking the derivative and setting the result equal to zero. The result at the iteration k is as follows:

$$\mathbf{p}_k = -\mathbf{H}(\mathbf{x}_k)^{-1}.\mathbf{g}(\mathbf{x}_k)$$

Equation 4-10

where $\mathbf{x}_k$ is the current search point and $\mathbf{p}_k$ is a vector denoting the next search direction. If, the Hessian matrix is positive definite, then it is assured that the vector $\mathbf{p}_k$ is a descent direction. The resultant step in the space of $\mathbf{x}$ is given by:

$$\mathbf{x}_{k+1} - \mathbf{x}_k = \alpha \mathbf{p}_k$$

Equation 4-11

It should be noted that while $\mathbf{p}_k$ may be a descent direction, moving *any* arbitrary distance along this direction is *no* guarantee of obtaining a reduction in the error function. In particular, when the current estimate is far for the minimum, it is advisable to incorporate a secondary search technique (a line search along the descent direction) to find the step size α. It is safe to neglect line searches only in cases when the current

estimate is close to the minimum, and consequently the magnitude $\mathbf{p}_k$ is small. Unfortunately, line searches add significantly to the computational complexity of the overall estimation process. Results dealing with the use of line searches in the motion estimation problem are presented in sub-section 4.3.6

The overall iterative structure of the Newton method is depicted in Figure 4-1. There are several classes of Newton method. The pure Newton method is used when the Hessian matrix (i.e. the second derivatives of the error function) can be computed directly based on the error function. However, there are functions for which the Hessian is not available and/or would result in very complex calculations. In such cases, the Hessian is iteratively constructed using previous values of the gradient vector. These methods are termed Quasi-Newton (QN) methods.

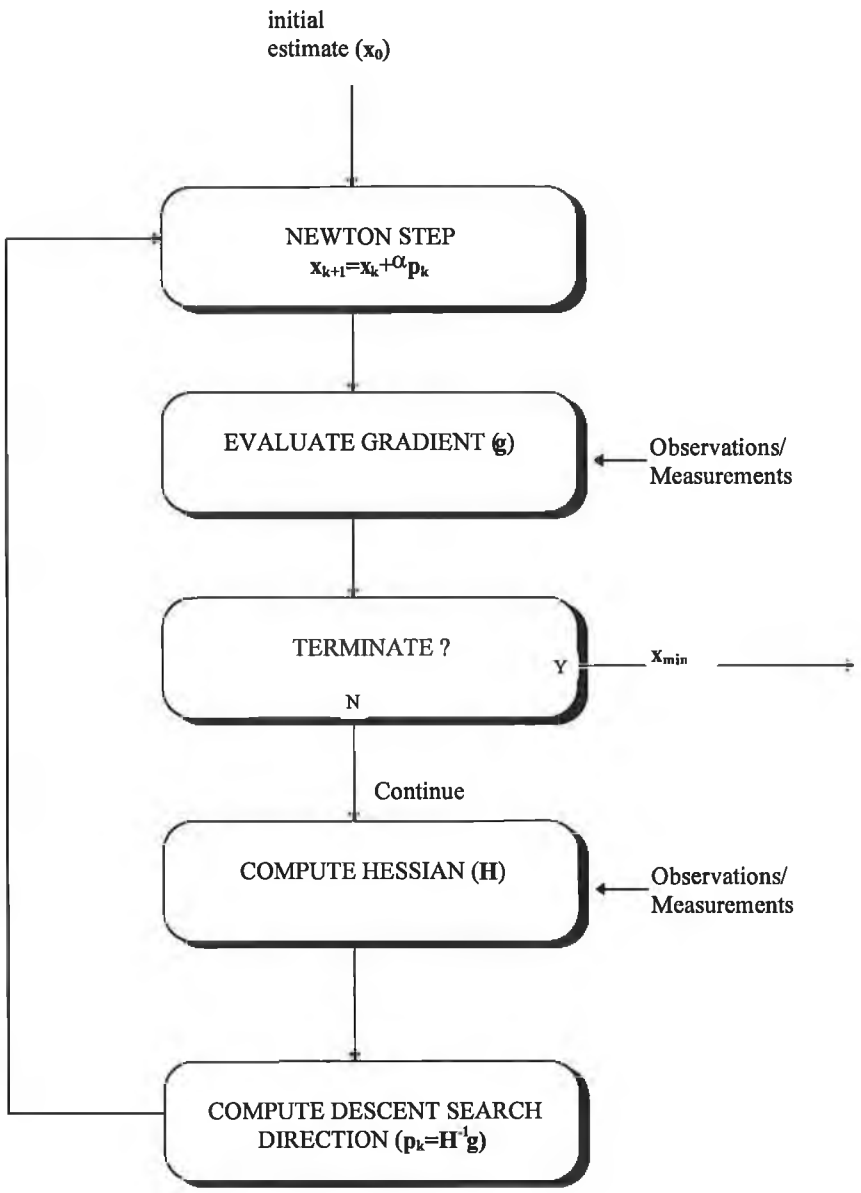


Figure 4-1: Illustration of a Newton method for the optimisation of an arbitrary error function in the variables x. For a motion modelling optimisation, x denotes the motion variables and the observations or measurements are the current and previous images.

Background material on the theory and practice of using Newton optimization can be found in an article by Brodlie **[17]**, and in books by Dennis and Schnabel **[26]** and Fletcher **[29]**. The next sections go into more detail on the problem of motion modelling and the proposed solutions, i.e. the GN and QN methods. In the following, the main emphasis is on the implementation-related aspects of GN and QN. Hence, mathematical equations relating to their implementation are presented. To keep the use of complex mathematics to a minimum, explanations for the equations are presented in a non-rigorous fashion. The referenced publications contain more rigorous justifications and derivations, although several derivations are included in Appendix A. In addition, Appendix A contains simplified examples on how the various formulae are utilised.

4.3.2 Quasi-Newton(QN)

Diehl **[27]** presents a QN system for the solution of the motion modelling problem. The QN method used here is based to a certain extent on this previous work. Quasi-Newton methods are used when the second derivatives of the error function are difficult to evaluate directly. In QN methods, the second derivatives (i.e. the Hessian matrix) are approximated from a knowledge of the first derivatives (i.e. the error gradients) at several points on the error function. To begin with, some initial estimate $\mathbf{C}_0$ of the Hessian matrix $\mathbf{H}$ is made. Based on the mean least squares form of the error function (i.e. Equation 4-8), Diehl chose to evaluate the initial Hessian approximate as in Equation 4-12[4].

$$\mathbf{C}_0 = \mathrm{E}\left\{\left(\frac{\partial \mathbf{x}'}{\partial \theta}\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}\right)\left(\frac{\partial \mathbf{x}'}{\partial \theta}\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}\right)^T\right\}\Bigg|_{\theta=\theta_0}$$

Equation 4-12

[4] A brief justification of this choice is presented in Diehl's paper **[27]**.

As derived in Appendix A, the gradient of the error function in Equation 4-8 is computed by:

$$\mathbf{g}(\theta_k) = \mathrm{E}\left\{ \left(I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')\right) \frac{\partial \mathbf{x}'}{\partial \theta} \left[\frac{\partial \mathbf{x}'}{\partial \mathbf{x}}\right]^{-1} \frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}} \right\} \Bigg|_{\theta=\theta_k}$$

Equation 4-13

A first iteration can now be carried out as follows:

$$\theta_1 = \theta_0 - \mathbf{C}_0^{-1} \cdot \mathbf{g}(\theta_0)$$

Subsequent iterations are given by:

$$\theta_{k+1} = \theta_k - \mathbf{C}_k^{-1} \cdot \mathbf{g}(\theta_k)$$

However, the matrix $\mathbf{C}_k$ at each iteration is given by the Broyden-Fletcher-Goldfarb-Shanno (BFGS) update formula **[17]** i.e.

$$\mathbf{C}_{k+1} = \mathbf{C}_k + \frac{1}{\mathbf{g}(\theta_k)^{\mathrm{T}} \mathbf{s}_k} \mathbf{g}(\theta_k)\mathbf{g}(\theta_k)^{\mathrm{T}} + \frac{1}{\mathbf{y}_k^{\mathrm{T}} \mathbf{s}_k} \mathbf{y}_k \mathbf{y}_k^{\mathrm{T}}$$

with

$$\mathbf{y}_k = \mathbf{g}(\theta_{k+1}) - \mathbf{g}(\theta_k) \text{ and } \mathbf{s}_k = \theta_{k+1} - \theta_k$$

Equation 4-14

The BFGS update formula returns an estimate of the Hessian matrix at the current search point. This estimate updates the Hessian (estimated at the previous search point) based on the gradient vectors obtained at the current and previous search point. Intuitively, it may be helpful to think of the method as extrapolating second order derivatives based on a knowledge of the first order derivatives at two distinct search points, i.e. θ_k and θ_{k+1}. Given a first estimate of the Hessian at the initial search point, the intention of the formula is to iteratively build an accurate estimate of the Hessian through accumulating knowledge of the first derivatives. Thus, on each iteration k, it is

only necessary to compute the gradient vector $\mathbf{g}(\theta_k)$ of the error function and then to employ the BFGS formula to find the next approximation of the Hessian matrix. The evaluation of the BFGS update formula is relatively simple. The main computational burden is the calculation of the gradient vector. For a given image region, the following procedure is employed to compute the gradient vector on each iteration. (The reader is required to consult the examples given in Appendix A in order to fully understand the procedure).

For all pixels $\mathbf{x} = (x \quad y)^T$ in the image region, the following are required:

1. Compute $\mathbf{B} = \dfrac{\partial \mathbf{x}'}{\partial \theta}$, an mx2 matrix.
2. Evaluate the model and the displaced co-ordinate $\mathbf{x}' = \mathbf{x} + \mathbf{B}\theta$.
3. Compute the error $\varepsilon = I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')$, a scalar.
4. Compute the image gradient $\nabla \mathbf{I} = \dfrac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}$, a 2x1 vector.
5. Compute the Jacobian inverse $\mathbf{J}^{-1} = \left[\dfrac{\partial \mathbf{x}'}{\partial \mathbf{x}}\right]^{-1}$, a 2x2 matrix.
6. Compute $\mathbf{v} = \varepsilon \mathbf{B}\mathbf{J}^{-1}\nabla \mathbf{I}$, an mx1 vector.

The $\mathbf{v}$ vectors are summed over the whole region and the mean is computed to give us the error gradient vector.

The sparsity of the matrix **B** can be exploited in any matrix products with which it is involved. The total number of operations required for the computation of the error function gradient is directly proportional to the number of pixels in the image region. It is also dependant on the model order n since higher model orders mean there are more elements in the associated vectors and matrices, i.e. m is large if n is large.

Diehl **[27]** suggested that the evaluations of image gradients (step 4) could be avoided at each iteration by a simple substitution in Equation 4-13. Although, no mathematical justification was provided in the paper, a fast QN (FQN) algorithm is yielded by

replacing the image gradient $\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}$ with $\frac{\partial I_{t+1}(\mathbf{x})}{\partial \mathbf{x}}$. The FQN computes the error gradient using

$$\mathbf{g}(\theta_k) = \mathrm{E}\left\{\left(I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')\right)\frac{\partial \mathbf{x}'}{\partial \theta}\left[\frac{\partial \mathbf{x}'}{\partial \mathbf{x}}\right]^{-1}\frac{\partial I_{t+1}(\mathbf{x})}{\partial \mathbf{x}}\right\}\Bigg|_{\theta=\theta_k}$$

Equation 4-15

The new image gradient is of course independent of the current value of the motion model and can be evaluated, once and once only, prior to the start of the first iteration. This in itself saves computation time, but there is an additional benefit to this method. Because the image gradient is known in advance, it is possible to sum the **v** vectors only over those pixels where the image gradient has a substantial value. This can result in large speed gains for most images and since pixels with small local gradients do not contribute much to the overall error gradient, a relatively accurate approximation (of the error gradient) can still be attained. This idea of pixel selection will be exploited in later sections of this chapter.

4.3.3 Gauss-Newton(GN)

The GN method can be viewed as a simplification of the pure Newton method, which is most suitable when the problem can be expressed as a least-squares regression, whereby the task is to fit a mathematical model to the observed data. It does not use an iterative update formula for the Hessian (as is done in the QN method), but it does approximate the local characteristics of the model function. These approximations result in making the direct computation of the second derivatives more feasible. A general treatment of Gauss-Newton theory is available in Fletcher's book **[29]**. The GN scheme was presented in relation to polynomial motion modelling by Sanson **[83]**. The following text presents this GN method.

Consider an iterative approach to the minimisation of the least-squares error function in Equation 4-8. At iteration k+1, the current model estimate is denoted by $\theta_{k+1} = \theta_k + \Delta\theta_k$

and the displaced position in the previous image is given as usual by $\mathbf{x}'_{k+1} = \mathbf{x} + \mathbf{B}\theta_{k+1}$. Thus, the current displaced position may be written as

$$\mathbf{x}'_{k+1} = \mathbf{x} + \mathbf{B}\left(\theta_k + \Delta\theta_k\right) = \mathbf{x}'_k + \mathbf{B}\Delta\theta_k$$

At each pixel, the error at this iteration is expressed by the difference between the observed data and the model function, i.e. $I_{t+1}(\mathbf{x}) - I_t\left(\mathbf{x}'_{k+1}\right)$. Assuming that $\Delta\theta_k$ is small, the model function can be approximated as a first order Taylor series as follows:

$$I_t\left(\mathbf{x}'_{k+1}\right) = I_t\left(\mathbf{x}'_k + \mathbf{B}\Delta\theta_k\right) \approx I_t\left(\mathbf{x}'_k\right) + \left(\mathbf{B}\Delta\theta_k\right)\frac{\partial I_t\left(\mathbf{x}'_k\right)}{\partial \mathbf{x}}$$

The derivation given in Appendix A5 shows how this local image approximation leads to the Gauss-Newton step $\theta_{k+1} = \theta_k + \mathbf{H}\left(\theta_k\right)^{-1}.\mathbf{g}\left(\theta_k\right)$, where the following expressions are used to compute the gradient vector $\mathbf{g}$ and the Hessian matrix $\mathbf{H}$:

$$\mathbf{g}\left(\theta_k\right) = \mathrm{E}\left\{\left(I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')\right)\frac{\partial \mathbf{x}'}{\partial \theta}\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}\right\}\Bigg|_{\theta=\theta_k}$$

Equation 4-16

$$\mathbf{H}\left(\theta_k\right) = \mathrm{E}\left\{\left(\frac{\partial I_t(\mathbf{x}')}{\partial \theta}\right)\left(\frac{\partial I_t(\mathbf{x}')}{\partial \theta}\right)^T\right\}\Bigg|_{\theta=\theta_k} = \mathrm{E}\left\{\left(\frac{\partial \mathbf{x}'}{\partial \theta}\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}\right)\left(\frac{\partial \mathbf{x}'}{\partial \theta}\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}\right)^T\right\}\Bigg|_{\theta=\theta_k}$$

Equation 4-17

The Gauss-Newton method is the same as the pure Newton method in terms of overall approach and structure. The only difference lies in the underlying Taylor series approximation. For images which are locally smooth, then the Taylor series approximation should be reasonably accurate and thus the GN should be robust and exhibit speedy convergence, as would be expected from the pure Newton method.

In comparison with the QN iteration, the GN iteration is just slightly more computationally complex. This is due to the fact that both the Hessian and gradient vector must be directly computed. For a given image region, the following procedure is adopted on each iteration.

For all pixels $\mathbf{x} = (x \quad y)^T$ in the image region, the following are required:

1. Compute $\mathbf{B} = \frac{\partial \mathbf{x}'}{\partial \theta}$, an mx2 matrix.
2. Evaluate the model and the displaced co-ordinate $\mathbf{x}' = \mathbf{x} + \mathbf{B}\theta$.
3. Compute the error $\varepsilon = I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')$, a scalar.
4. Compute the image gradient $\nabla \mathbf{I} = \frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}$, a 2x1 vector.
5. Compute $\mathbf{v} = \mathbf{B}\nabla \mathbf{I}$, an mx1 vector.
6. Compute $\mathbf{g} = \varepsilon \mathbf{v}$, an mx1 vector.
7. Compute $\mathbf{H} = \mathbf{v}\mathbf{v}^{\mathrm{T}}$, an mxm matrix.

The $\mathbf{g}$ vectors are summed over the whole region and the mean is computed to give the error gradient vector. The $\mathbf{H}$ matrices are summed over the whole region and the mean is computed to give the Hessian matrix. One should notice that the GN procedure is entirely similar to the QN algorithm up to, but not including, step 5. It is the multiplications involved with step 7 which mainly account for the GN method's slightly higher complexity. Fortunately, the degree of symmetry existing in step 7 can be exploited to reduce the number of computations, see Equation 4-21. As with the QN algorithm, the computational complexity of this algorithm is related to the size of the image region and to the model order. A more detailed analysis of the computational complexity of GN is described in sub-section 4.4.1.

4.3.4 From Discrete to Continuous Image Representations

The Newton method of optimisation is based on continuous functions. However, when applying this technique to image processing, there is a general difficulty due to the fact

that a digital image is a discrete function. To overcome this difficulty, it is necessary to convert the digital image into a continuous function in 2-D. In addition, the implementation of either QN or GN also requires the evaluation of image derivatives (gradient) in continuous space. To minimise any problems due to errors in computing these derivatives, a continuous-space representation of the image was constructed. For the sake of clarity, the methodology is first illustrated in the case of a 1-D signal and the extension to the 2-D case is then outlined.

Given some interpolation function $f(x)$ and a discrete signal $y(k)$ with sampling interval Δx, a signal value can be evaluated at any point in continuous space according to:

$$y(x_c) = \sum_{k=n}^{n+3} y(k) f(x_c - k\Delta x)$$

where $(n+1)\Delta x$ corresponds to the discrete sample directly before the continuous position x_C. This equation describes a 4 tap filter. The taps of the filter are obtained by evaluating the interpolating polynomial at the specified points. The chosen interpolation function is a piecewise bi-cubic polynomial based on the work of Mitchell and Netravali **[51]**. It has the following form.

$$f(x) = \begin{cases} |x|^3 - 2x^2 + 1, & \text{if } |x| < 1 \\ -|x|^3 + 5x^2 - 8x + 4, & \text{if } |x| < 1 \end{cases}$$

This polynomial was developed specifically for the purpose of reducing aliasing in converting from discrete to continuous image representations. It has already been found be useful by Sanson in his work on motion estimation. For 2-D image interpolation, a second filter is constructed (based on the same polynomial) and the two filters are applied in separable fashion according to the following:

$$I(x_c, y_c) = \sum_{l=m}^{m+3} f(y_c - l\Delta y) \sum_{k=n}^{n+3} I(k,l) f(x_c - k\Delta x)$$

Equation 4-18

where $(m+1)\Delta y$ corresponds to the discrete sample directly above the continuous position y_C. This separable filter allows the evaluation of the image intensity at any point in continuous space. In addition, it may be seen that the continuous image gradient may also be obtained by differentiating Equation 4-18. This yields:

$$\frac{\delta I(x_c, y_c)}{\delta x} = \sum_{l=m}^{m+3} f(y_c - l\Delta y) \sum_{k=n}^{n+3} I(k,l) f'(x_c - k\Delta x)$$

and

$$\frac{\delta I(x_c, y_c)}{\delta y} = \sum_{l=m}^{m+3} f'(y_c - l\Delta y) \sum_{k=n}^{n+3} I(k,l) f(x_c - k\Delta x)$$

Equation 4-19

Therefore, in computing the gradient at continuous positions, it is only necessary to derive two new 4 tap filters based on the derivative of the interpolating bi-cubic polynomial. This derivative polynomial is bi-quadratic in form.

In computational terms, the task of computing the image intensity and gradient at a given point in continuous space is highly complex. A total of four filters must be constructed. There are two filters used to evaluate the continuous intensity and two additional filters to evaluate the continuous gradient. Each filter tap requires the evaluation of either the bi-cubic polynomial or the bi-quadratic polynomial. It should be noted that constructing the filters in this way requires a total of 60 multiplications. This is in addition to the 20 multiplications involved in the convolution of Equation 4-18 and the 40 multiplications involved in the convolutions of Equation 4-19.

4.3.5 Details of Implementation

The two optimisation methods described above, i.e. QN and GN, have been implemented by the author. An essential difference between the chosen implementation and the standard implementations is that a co-ordinate system normalisation is performed. This can be simply described by the mapping,

$$\begin{pmatrix} x \\ y \end{pmatrix} \rightarrow \begin{pmatrix} \dfrac{(x - x_0)}{x_{\max}} \\ \dfrac{(y - y_0)}{y_{\max}} \end{pmatrix}$$

where the image region undergoing the motion estimation is bounded by a rectangle whose top-left co-ordinate is (x_0,y_0) and whose dimensions are $x_{\max}$ and $y_{\max}$, see Figure 4-2. This was required because it was noted that the Hessian matrices are frequently ill-conditioned[5]. The extent of the ill-conditioning depends upon the model order being used (higher order models yielding more ill-conditioned matrices) and on the size of the region within the image (larger regions producing more ill-conditioned matrices). The normalisation had the effect of producing better conditioned systems, less susceptible to noise in the data and to the effects of finite precision arithmetic. In this way, considerable improvements in performance were possible, especially for quadratic models.

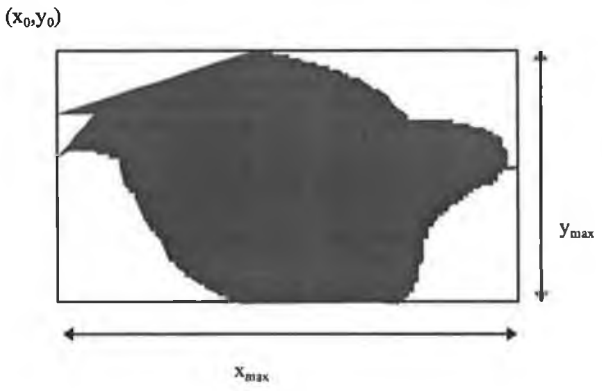


Figure 4-2: An arbitrary region bounded by a rectangle.

In the author's implementation, the optimisation system was also embedded in a multi-resolution image pyramid. Both current and previous images were applied to a three level pyramidal decomposition. It can be shown experimentally that this is of benefit in terms of robustness and computational complexity, when scene motion is large, or more

5 A matrix is said to be ill-conditioned if its rows or columns are nearly linearly dependent.

generally, when the minimum is substantially different from the initial estimate. Also, for the GN method, suitably chosen filters can lead to images at low resolution which locally conform to the aforementioned Taylor series approximation. A low-pass Gaussian filter was used in the pyramid construction. The use of a multiresolution pyramid necessitates the transfer of the polynomial motion models from low resolution images to high resolution images. This transfer is achieved as follows.

The general polynomial can be represented by:

$$d(x,y)=\sum_{i=0}^{n}\sum_{j=0}^{i}a_{ij}x^{i-j}y^{j}$$

Equation 4-20

The task is to transfer a polynomial motion model $\mathbf{d}'$ related to the image at resolution R' to a resolution twice this resolution, i.e $R=2R'$, such that $\mathbf{d}(x,y)=2\mathbf{d}'(x',y')$. (Here, $x=2x'$ and $y=2y'$). Therefore, the following equality must be satisfied.

$$\begin{aligned}\sum_{i=0}^{n}\sum_{j=0}^{i}a_{ij}x^{i-j}y^{j} &= 2\sum_{i=0}^{n}\sum_{j=0}^{i}a'_{ij}(x')^{i-j}(y')^{j}\\ &= 2\sum_{i=0}^{n}\sum_{j=0}^{i}a'_{ij}\left(\frac{x}{2}\right)^{i-j}\left(\frac{y}{2}\right)^{j}\end{aligned}$$

This is satisfied if $a_{ij}=\frac{a'_{ij}}{2^{i-1}}$.

The situation is different if a co-ordinate system normalisation is applied as defined by:

$$\mathrm{x_n}=\frac{x-x_0}{x_{\max}}\ ,\quad \mathrm{y_n}=\frac{y-y_0}{y_{\max}}\quad\text{and}\quad \mathrm{x'_n}=\frac{x'-x'_0}{x'_{\max}}\ ,\quad \mathrm{y'_n}=\frac{y'-y'_0}{y'_{\max}}\ .$$

Then a different equality is satisfied, i.e.

$$\sum_{i=0}^{n}\sum_{j=0}^{i} a_{ij} x_n^{\,i-j} y_n^{\,j} = 2\sum_{i=0}^{n}\sum_{j=0}^{i} a'_{ij} (x')_n^{\,i-j} (y')_n^{\,j}$$

Now since,

$$x = \frac{x'}{2}, x_0 = \frac{x'_0}{2}, x_{max} = \frac{x'_{max}}{2}, y = \frac{y'}{2}, y_0 = \frac{y'_0}{2} \text{ and } y_{max} = \frac{y'_{max}}{2},$$

then $x_n = x'_n$ and $y_n = y'_n$ and it follows that $a_{ij} = 2a'_{ij}$ satisfies the equality.

A final point on the GN method is also worthy of mention. Regarding the Hessian matrix **H** of Equation 4-17, it can be shown that ,

$$\mathbf{H} = \begin{pmatrix} \mathbf{H}_{xx} & \mathbf{H}_{xy} \\ \mathbf{H}_{xy} & \mathbf{H}_{yy} \end{pmatrix} \text{ where } \mathbf{H}_{xx}, \mathbf{H}_{yy} \text{ and } \mathbf{H}_{xy} \text{ are symmetric matrices.}$$

Equation 4-21

In the GN implementation, the sub-matrices off the main diagonal have been set to zero. This approach was taken by Sanson **[83]** and has two effects. Firstly, it simplifies the computation of **H** and secondly, it leads to better conditioned systems and generally better overall performance.

4.3.6 Performance Comparison of Gauss-Newton and Quasi-Newton

Tests were conducted to compare the performances of GN, QN and FQN. The test and nature of results are described and a discussion of the findings follows. Each estimation method was applied to some video-phone sequences. Both manual and automatic segmentations were used to define the various independent regions to which the motion estimation was applied. Table 4-2 summarises the test material together with the motion models used in each case.

Table 4-2: Summary of test material for comparative tests.

Case	*Sequence*	*Frames used*	*Frame Rate (Hz)*	*Segmentation/ #regions*	*Motion Model*
1	Foreman	200-340	25	Manual/5	quadratic
2	Foreman	200-340	8.33	Manual/5	quadratic
3	Foreman	0-54	8.33	Automatic/70	affine
4	Claire	0-140	5	Automatic/12	affine

The test material is diverse in the sense of the segmentations used. Test cases 1 and 2 use segmentations containing only a small number of regions, each assumed to be moving independently. The other test cases use segmentations containing larger numbers of regions[6]. The nature and magnitude of the motion within the scenes themselves is also diverse, ranging from fast/3-D motions in test cases 1 and 2 to simpler motions in test case 4.

At each frame of each sequence, the motion of each of the regions was estimated. The motion estimates were used to motion compensate the previous original image. Motion compensation errors were not propagated in time. Results are presented in terms of the motion compensation error magnitude, quantified by the PSNR and the numbers of iterations required for convergence. Table 4-3 presents a brief overview of the experimental findings via the mean PSNR and iteration figures. The graphs of Figure 4-3 illustrate the PSNR result over the length of the sequences.

6 These segmentations were performed using the morphological watershed, see chapter 5.

Table 4-3: Summary of results of comparative tests. Mean PSNR figures taken over each sequence are given as well as the number of iterations until convergence. Here, the format is as follows: #iterations in low resolution pyramid layer/ medium resolution pyramid layer/ high resolution pyramid layer.

	GN		*QN*		*FQN*	
Case	PSNR	#iterations	PSNR	#iterations	PSNR	#iterations
1	32.89	5/4/3	32.10	4/3/3	31.69	4/2/2
2	26.53	6/3/3	25.69	4/3/2	24.93	3/2/2
3	32.87	5/5/5	32.23	5/4/4	31.57	5/3/3
4	37.75	6/4/3	37.57	5/3/2	37.32	4/2/2

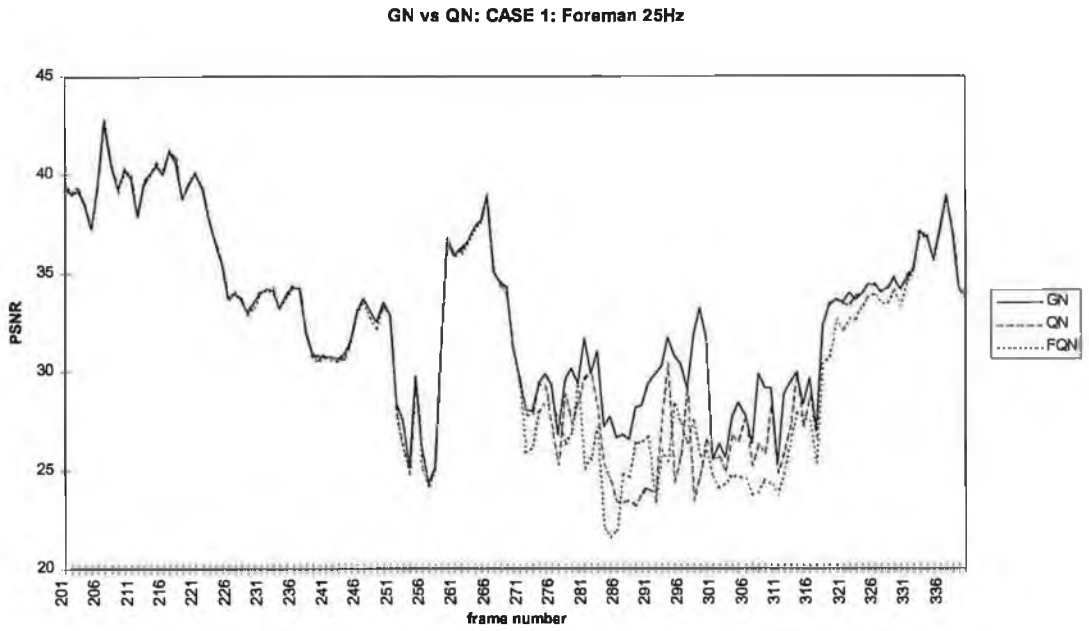
From Table 4-3 and Figure 4-3, it is clear that the QN method is, in general, inferior to the GN method. This is particularly evident if scene motion is large, i.e. the fast camera pan in Foreman, see graphs 1 and 2 (frames 280-310). In such cases, the GN can often produce motion compensated images with PSNRs 5dB greater than that of the QN method. It would appear that the QN method is not very reliable when provided with initial estimates far from the solution. In terms of convergence speed, the QN uses slightly fewer iterations. But since the QN generally does not converge to a good optimum, this advantage is a little dubious.

The fast QN method was also investigated, but proved ineffective relative to the GN method. Again, performance suffered mainly during instances of large motion. Motion modelling based on FQN, while having the potential to be fast, is considered to lack reliability and robustness and may only be suitable when motion magnitudes are known to be very small

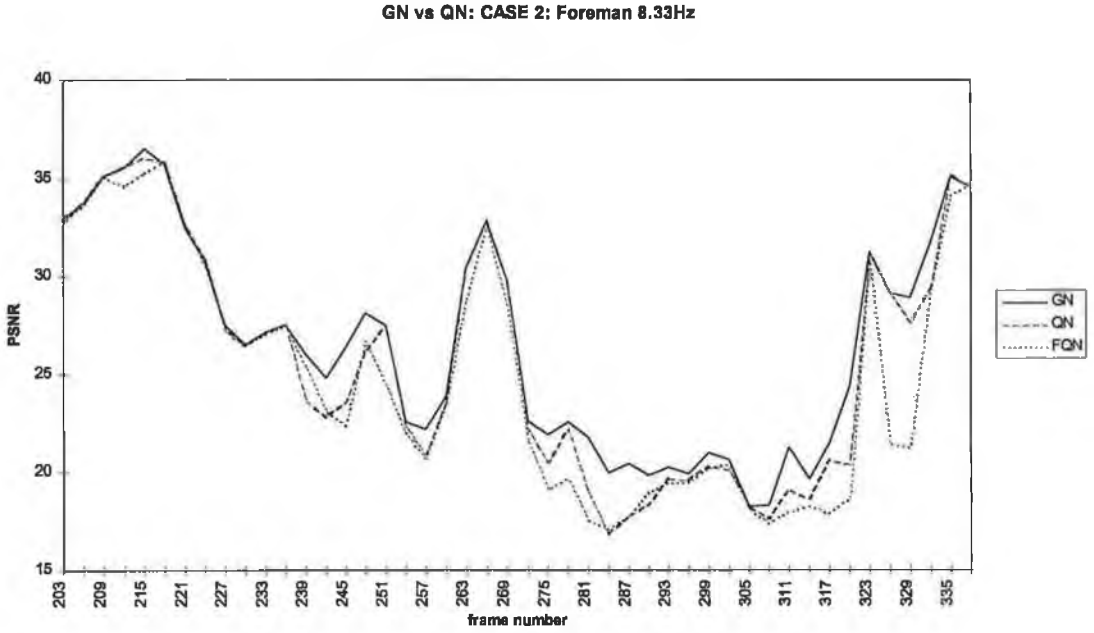
In summary, the GN method demonstrates good relative robustness even when given bad initial estimates of the minimum. Despite this, we can not be sure that the GN is always converging to the global minimum of the error function. Indeed, with the addition of a line search **[75]**[7], it is observed that, at times, up to a 2dB improvement can be attained in prediction quality. This is demonstrated by Figure 4-3e. Although the use

[7] An exact line search algorithm based on minimum bracketing and a 1-d minimisation algorithm, devised by Brent, was used.

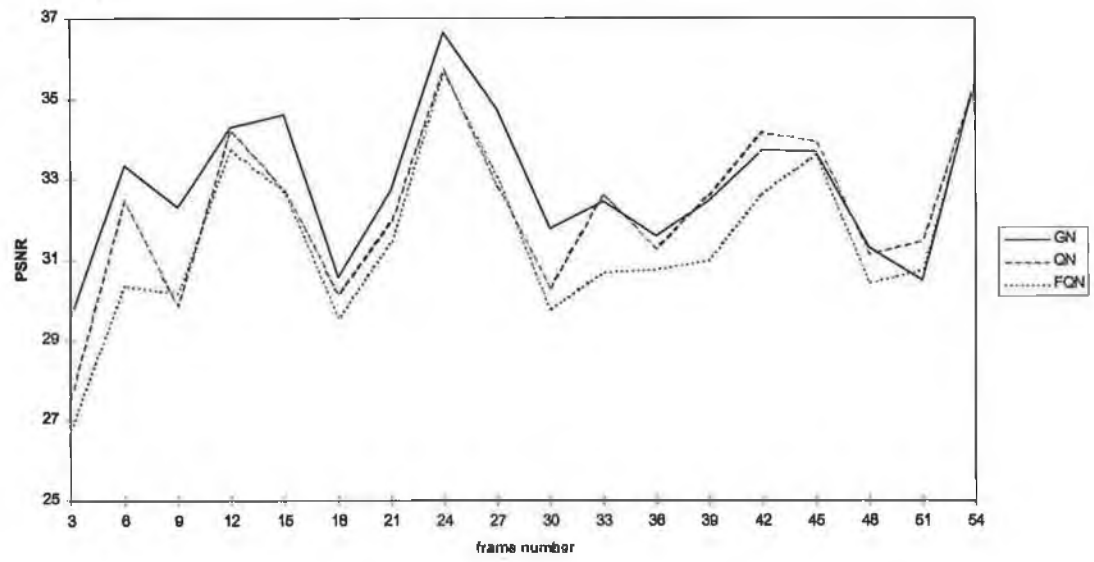
of line searches is inappropriate due to the substantial additional complexity, i.e. 10-20 extra function evaluations per iteration, this result does demonstrate that there is some scope for improving the GN motion modelling algorithm via the application of alternative low complexity methods. Alternatively, the use of line searches can be neglected if an attempt is made to find a good initial estimate of the minimum. In the application of motion estimation to tracking (as described in Chapter 6), the motion estimates for a given video frame may be initialised using the motion estimates of the previous frame.



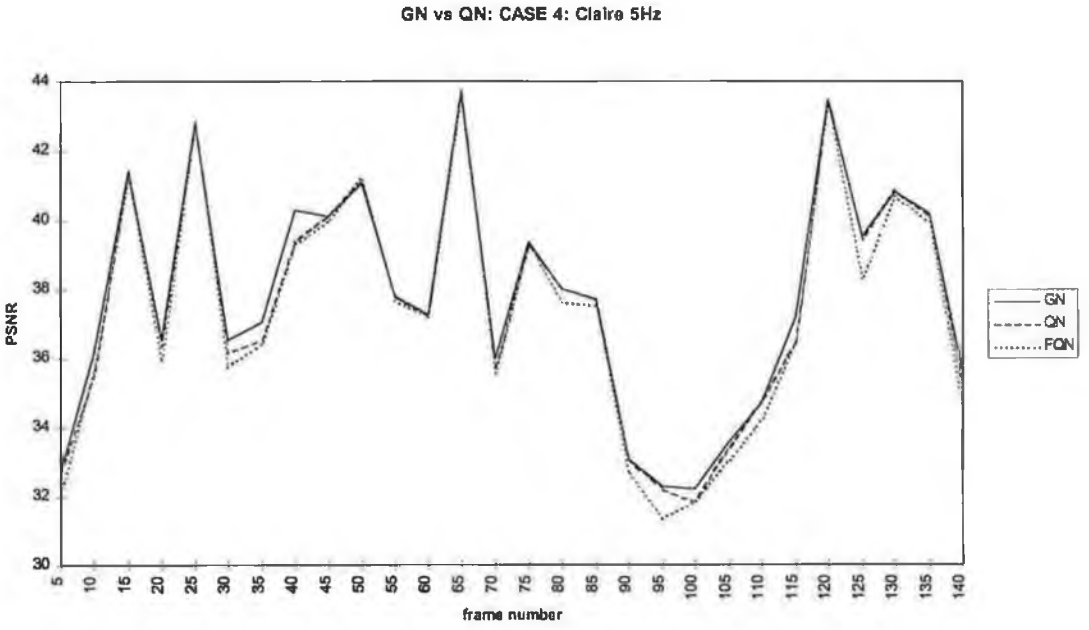
(a)


GN vs QN: CASE 2: Foreman 8.33Hz
PSNR
frame number
GN
QN
FQN

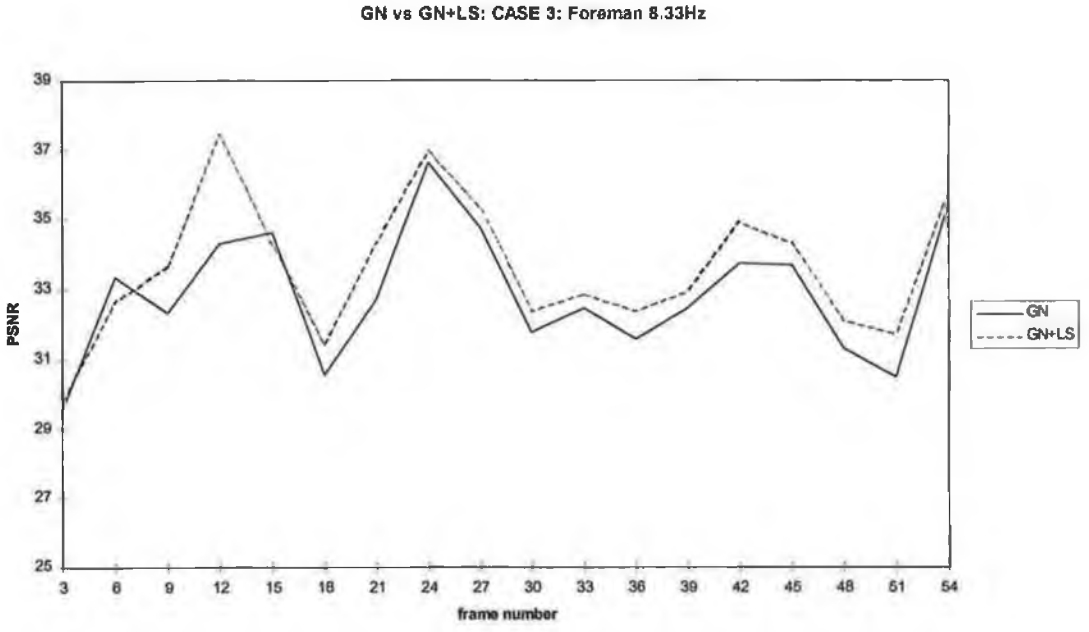
(b)


GN vs QN: CASE 3: Foreman 8.33Hz
PSNR
frame number
GN
QN
FQN

(c)



(d)



(e)

Figure 4-3: The five graphs plot PSNR versus time. This illustrates the relative effectiveness of each motion estimation technique.

4.4 Fast Gauss-Newton Estimation

The previous sub-sections introduced the GN and QN estimation methods and a comparision showed that the GN method is superior in terms of finding the optimum. A major problem with all Newton-like methods for motion estimation is their computational complexity. This sub-section summarises work by the author **[13]** which examines the complexity of GN and suggests methods of reducing it.

4.4.1 Computational Analysis of GN

For the estimation of a general n^{th} order motion model over a region R, the operations listed in Table 4-4 must be carried out at each pixel within each iteration. Note, for an n^{th} order polynomial motion model, θ is an m-dimensional vector, where $m = (n+1)(n+2)$. Note also, that at step 7 in Table 4-4, a fast vector product algorithm is already used in our implementation. This avails of all the symmetry present and neglects the sub-matrices off the main diagonal, setting the elements to zero, as discussed.

Table 4-4: Multiplications per pixel per iteration (*MPI*) for GN-based motion estimator (*m* refers to the dimension of the model vector).

Step #	*Operation*	*# Multiplications*
1	$\mathbf{B}$	$m/2$ [8]
2	$\mathbf{x}' = \mathbf{x} + \mathbf{B}\theta$	m
3	$\nabla\mathbf{I} = \frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}$	G
4	$\varepsilon = I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')$	I
5	$\mathbf{v} = \mathbf{B}\nabla\mathbf{I}$	m
6	$\mathbf{g} = \mathbf{g} + \varepsilon\mathbf{v}$	m
7	$\mathbf{H} = \mathbf{H} + \mathbf{v}\mathbf{v}^{\mathbf{T}}$	$m/2+m^2/4$

Summing the rightmost column of Table 4-4, we get what is termed the *MPI* (multiplications/pixel/iteration) figure for GN:

[8] This is only a rough approximation but does not significantly effect results or conclusions.

$$MPI = I + G + 4m + {}^{m^2}/_{4}$$

For the gradient and intensity interpolations, 4-tap filters, based on **[51]**, are applied in a separable fashion as discussed in sub-section 4.3.4. For this choice, $G=100$ and $I=20$. It is emphasised that G includes the construction of all the interpolation filter kernels, i.e. even those used for the intensity interpolation. Based on all this information, it can be computed that a quadratic modelling task (m=12) has an *MPI* figure of 204. For a given image resolution, it is possible to derive the overall requirement (M) in multiplications per second by using

$$M = MPI.N.K.F\ ,$$

where N is the number of pixels in the image, K is the average number of iterations until convergence and F is the frame rate of the video sequence. By way of example, we can compute the computational requirement for quadratic motion estimation based on the QCIF format at 25Hz to be 388 million multiplications per second. This is based on the conservative assumption that only 3 GN iterations are required (i.e. $K=3$).

4.4.2 Towards a Fast Implementation

The processing power discussed above is beyond the capabilities of any of today's general purpose processors and therein lies the motivation to look for means of simplifying the estimation procedure. Two methods are now suggested through which faster algorithms can be achieved. These two methods have been used by the author to implement a fast GN algorithm.

LUT-based Interpolation. Much effort is expended in computing the image intensity and gradient at the displaced position. A significant part of this expense is due to the construction of the interpolation filter kernels themselves, while the remainder is in performing the actual convolutions. The construction of the filter kernels is achieved by the evaluation of bi-cubic (for intensity) and biquadratic (for gradient) polynomials. This can be avoided by using look-up tables (LUTs) to approximate these polynomial

functions. Effectively, this means the algorithm stores the filter coefficients instead for computing them each time. In this way, a new reduced figure for G can be achieved, i.e. G=40. This figure is accounted for solely by the convolution of the image pixels with the filter taps. Note that this modification was suggested previously in **[7]**.

Gradient-based Pixel Elimination. The computation of the Hessian and gradient involves performing the 7 operations of Table 4-4 at each pixel in the given region and summing over all these pixels. One could instead choose to use only every second pixel on every second line. This choice is rather arbitrary and has been found to compromise the estimation procedure. Instead, a pixel elimination procedure is used based on thresholding the image gradient norm. This approach of gradient thresholding is motivated by the fact that the equations for the Hessian and gradient are largely dependent on the image gradient and that pixels with little gradient information will not contribute greatly to the total sum. Diehl **[27]** adopted a similar strategy within a QN estimation algorithm. Due to the nature of the QN algorithm of **[27]**, the gradient information was available prior to start of the first iteration. This made it possible to apply the pixel elimination strategy from the start. For GN, this is not possible because the image gradient utilised depends on the current model estimate. The FGN algorithm using this pixel elimination procedure is now summarised.

4.4.3 The Fast GN Algorithm (FGN)

The fast algorithm for the computation of the Hessian and gradient at iteration k is summarised by the following:

For all pixels $\mathbf{x} = (x \quad y)^T$ in the image region, it is required to:

1. Compute the matrix $\mathbf{B}$.
2. Compute the displaced co-ordinate $\mathbf{x}' = \mathbf{x} + \mathbf{B}\theta_k$.
3. Compute the image gradient $\nabla \mathbf{I} = \frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}$.
4. If $\|\nabla \mathbf{I}\| \geq T$, continue with step 5. Otherwise, the procedure moves back to step 1 and processing begins with the next pixel in the region.

5. Compute the error $\varepsilon = I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')$.
6. Compute the vector $\mathbf{v} = \mathbf{B}\nabla\mathbf{I}$.
7. Compute the gradient update $\mathbf{g} = \mathbf{g} + \varepsilon\mathbf{v}$.
8. Compute the hessian update $\mathbf{H} = \mathbf{H} + \mathbf{v}\mathbf{v}^T$.

It can be noted that at each pixel, steps 1-3 must be performed before the threshold-based decision is carried out. If desired, the interpolative filters can be derived from LUTs as described in sub-section 4.4.2. In the chosen implementation, the threshold T is adapted on a pixel-by-pixel basis in order to achieve a given factor of reduction in the amount of pixels considered.

4.4.4 Computational Analysis of FGN

It can be shown that the *MPI* figure for the FGN algorithm is computed thus:

$$MPI = \frac{3m}{2} + G + r\left(I + \frac{5m}{2} + \frac{m^2}{4}\right) \quad \text{with} \quad 0 < r \leq 1$$

where r is the fraction of pixels not rejected by the thresholding procedure. To reiterate, if LUT-based interpolation is used *G=40*, otherwise *G=100*. Table 4-5 and Table 4-6 provide examples of what can be achieved with the two proposed improvements. It can be seen that under the joint application of the LUT-based interpolation and the pixel elimination strategy, complexity is reduced almost by a factor of 3.

Table 4-5: mpi figures for affine motion estimation where r = 0.2.

	G=100	*G=40*
GN	153	93
FGN	117.8	57.8

Table 4-6: mpi figures for quadratic motion estimation where r = 0.2.

	G=100	*G=40*
GN	204	144
FGN	135.2	75.2

It is clear that all the proposed modifications result is a more efficient implementation. The experimental findings reported in **[13]** show that the complexity may be reduced by a factor of 2-3 without significantly effecting estimation performance.

4.5 Robust Gauss-Newton Estimation

This final sub-section is dedicated to the estimation of the same polynomial motion models, but with a slight difference concerning the error function which is optimised. The straightforward sum-of-squares error function is overly sensitive to outliers. Outliers are defined to be errors within the error signal/image whose absolute magnitude is far beyond the average. In the estimation of visual motion, outliers can be due to image noise and the appearance or revealing of new objects, previously unencountered. It is often very desirable to ignore such phenomena in the computation of motion. For instance, often it is required to capture the real motion in the scene. This would, for instance, be important in a computer vision application where some action is triggered by a particular motion type. In the area of motion segmentation and object tracking, it is required to compute models which describe the pure object motion and which are not influenced or distorted by the fact that a new region has been uncovered or by the fact that there is a lot of image noise. For this reason, it is best to use what are termed robust estimators **[85]**.

Sawhney *et al* **[84]** performed the estimation of motion parameters θ in the region R by minimising the following generalised error function:

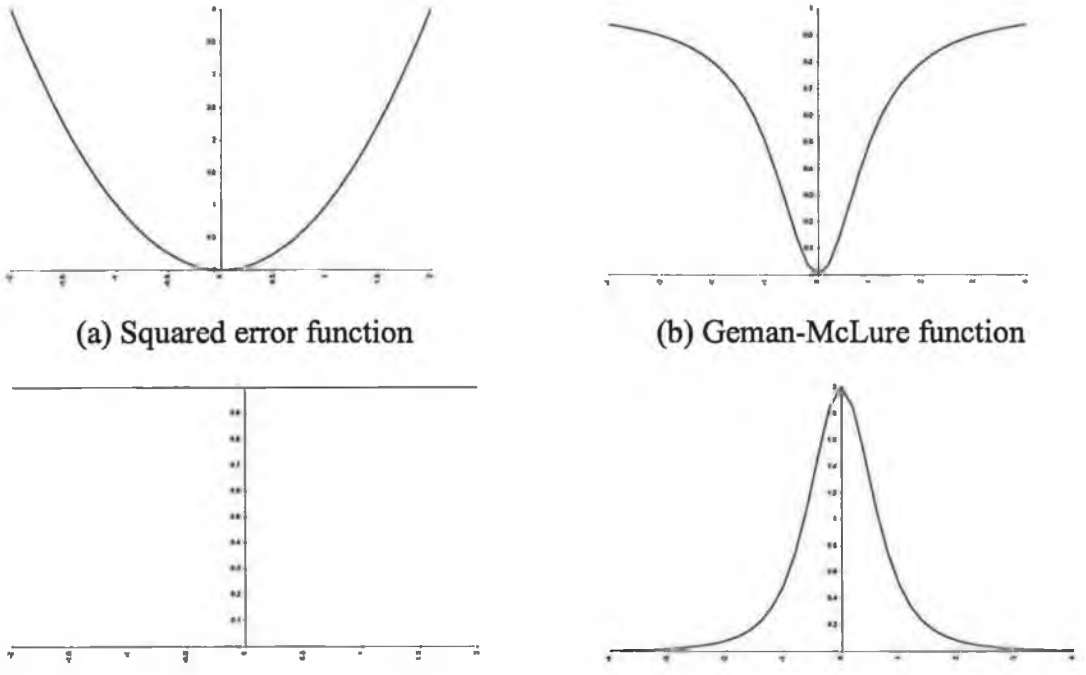
$$e(\theta) = \sum_{\forall \mathbf{x} \in \mathbf{R}} \rho(r(\mathbf{x}), \sigma)$$

Equation 4-22

where the residual error at each pixel $\mathbf{x}$ is $r(\mathbf{x}) = I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}'(\theta))$ and the parameter σ is an error scaling factor usually derived by computing the variance of the residual errors in the region R. For the previously introduced sum-of-squares error function of Equation 4-8, $\rho(r,\sigma) = \frac{r^2}{\sigma^2}$. For robust estimation, the Geman-McLure (GM) function was used instead, i.e.

$$\rho(r,\sigma) = \frac{\frac{r^2}{\sigma^2}}{1+\frac{r^2}{\sigma^2}}$$

A comparison of the squared error (SE) and GM functions are given in Figure 4-4. Note, when using the Geman-McLure function, the larger errors contribute less to the overall error function.



(a) Squared error function

(b) Geman-McLure function

(c) Influence function of squared error function

(d) Influence function of the Geman-McLure function

Figure 4-4: The top line shows the SE function and GM function. The bottom line shows the respective influence functions for each.

As demonstrated by Sawhney, the gradient and Hessian of the given robust error function can be approximated as:

$$\mathbf{g}(\theta_k) = \mathrm{E}\left\{ w(\mathbf{x})\left(I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')\right)\frac{\partial \mathbf{x}'}{\partial \theta}\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}} \right\}\Bigg|_{\theta=\theta_k}$$

Equation 4-23

$$\mathbf{H}(\theta_k) = \mathrm{E}\left\{ w(\mathbf{x})\left(\frac{\partial \mathbf{x}'}{\partial \theta}\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}\right)\left(\frac{\partial \mathbf{x}'}{\partial \theta}\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}\right)^T \right\}\Bigg|_{\theta=\theta_k}$$

Equation 4-24

where,

$$w(\mathbf{x}) = \frac{2\sigma^2}{\left(\sigma^2 + r(\mathbf{x})^2\right)^2}$$

is a weighting function relying on the residual error ($r(\mathbf{x}) = I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')$) at the given pixel **x**. Comparing the above Hessian and gradient to those of Equation 4-16 and Equation 4-17, it is clear that the presence of this error weighting function is the only difference between the robust estimator and the least squares estimator. This error weighting/influence function is depicted in Figure 4-4. It can be concluded that errors with large magnitude have very little effect on the estimation process.

By applying these influence functions in the estimation procedure, a technique known as weighted least squares (WLS) estimation is being used. To effectively use the WLS estimator, the variance of the error distribution σ must be known or ascertained. The computation of the error variance itself is highly influenced by outliers and hence Sawhney suggested that the median of the error distribution be used to arrive at a more representative variance. The following equation was used:

$$\sigma = 1.4826 \text{ median}|r(\mathbf{x})|$$

Justifications for the use of this equation are given in **[84]**. In the framework of the iterative GN estimator, this variance value should be re-computed before each iteration and hence this method is often referred to as iteratively re-weighted least squares (IRLS).

4.6 Summary

Generalised polynomial models are capable of representing the effects of rigid 3D motions. The estimation of the model parameters requires an iterative approach to optimisation. Several optimisation algorithms have been tested by the author and compared, here and in other literature, in terms of their ability to minimise the given error function. It would appear that the Gauss-Newton method is better than most. However, it is noted that the estimation of polynomial motion has some problems. For example, the global minimum of the error function is not always found. There is possibly some scope for improving results by some suitable initialisation of the search. Another problem with the estimation of the motion parameters is that a significant amount of computing power is required if real-time performance is to be attained. To address this problem, fast Gauss-Newton algorithms have been developed which reduce computation time by a factor of 2-3, while only marginally effecting the estimation performance. Finally, the basic least squares error function that is usually the basis of motion estimation is overly sensitive to outliers. It is shown how a minor modification to the general least squares minimisation results in an estimation algorithm that is more robust to the effects of noisy images and other factors resulting in statistical outliers.

While most of this chapter focusses on the estimation of motion within a defined region of an image, the motion within any given image can be conveniently represented by a finite set of polynomial motion models and a corresponding set of support maps defining the model which is supported at each pixel. For a given image, the estimation of its motion is equivalent to the joint estimation of the model parameters and model

supports. This is the problem of motion segmentation which is discussed in the next chapter. Motion segmentation is one important tool which may allow the efficient recovery of object shapes from video sequences and even more importantly, it may play a role in facilitating the tracking of objects from frame to frame, as discussed in chapter 6.

5. IMAGE SEGMENTATION

Image segmentation is a widely studied topic. With regard to the subject of object-based video representation, image segmentation is required in order to identify the position and shape of the various visual objects in an image or video. For a variety of reasons, image segmentation is a very difficult task. The kind of segmentations required for object-based content access and editing are very demanding. That is, it is usually required to identify semantic objects in a scene. With current image representations, general methods for the identification of semantic objects from images are not available. Semantic segmentation relies on the kind of intelligence which is currently present only within the human brain. Existing semantic segmentation systems involve much laborious work on the part of the human user, i.e. mouse pointing and clicking. Computers can only be trained to understand and process very primitive image attributes, i.e. texture and motion. Texture segmentation can be applied in order to separate the pixels of the image into regions of coherent texture **[19]** and similarly, motion segmentation is used to identify pixels which are moving with the same velocity **[11]**. While the results of computer generated texture and motion segmentations seldom capture the semantic content, computer-based algorithms can be used to ease the task of semantic object segmentation by performing these so-called "primitive" processes at high speed.

In this chapter, several prominent tools and methodologies for image segmentation based on both texture and motion are reviewed. The purpose of the review is:

- to illustrate the difficulties with existing automatic image segmentation and the fact that semantic interpretation by automatic means is currently infeasible,
- to introduce the basic techniques of statistical and morphological segmentation that are utilised in chapter 6 to develop a new object tracking algorithm, i.e. the segmentation of moving objects,
- to show the strength of a joint motion estimation/segmentation approach by Ayer and Sawhney **[4]** based on the Expectation-Maximisation algorithm and the Minimum Description Length principle (the EM-MDL approach).

In addition to the review elements of this chapter, some new work by the author is presented. The approach of Ayer and Sawhney emphasised the aspect of motion estimation and somewhat neglected the requirement for clean and consistent segmentations. The author has augmented their basic algorithm by encorporating contextual labelling constraints into the EM algorithm based on local Markov random field models. In addition, the computation of the description length used in the MDL is improved by considering local correlation in the segmentation labels.

The first sub-sections focus on a statistical approach to image segmentation. In sub-section 5.1, the segmentation problem is viewed as an optimisation problem of high dimensionality and a general iterative framework is presented as a possible solution. Several examples applying this framework are given. Sub-section 5.2 presents two estimation tools, i.e. the Expectation-Maximization method and the Minimum Description Length principle, which provide this iterative framework with some theoretical basis. The basic premise of statistical segmentation is the allocation of pixels to various classes based on the models underlying those classes. However, it is generally not useful to limit the pixel labelling task to the consideration of model suitability. It is more appropriate to further constrain the labelling according to some local spatial contexts. Sub-section 5.2.4 discusses contextual labelling algorithms that impose local spatial dependancies using Markov Random Field (MRF) models. Finally, departing significantly from the preceeding discussions, morphological image processing methods for image segmentation are presented due to the relative success which has been achieved by their use in texture segmentation.

5.1 Problem Formulation and Solution

Image segmentation involves a classification of image pixels where two or more classes of image pixels exist. Segmentation may be achieved by answering the following simple question: Which pixels belong to which classes? However, for most segmentation tasks, the following facts are not known, (*i*) how many classes exist in the image? and (*ii*) what are the models for each class in the image? This is why segmentation is often referred to as a "chicken and egg" problem. Without the segmentation, it is unknown

how many classes exist and it is not possible to directly compute the model parameters underlying each class. Without the knowledge of the classes and associated model parameters, it is not possible to generate the segmentation. This sub-section formulates the problem as an optimisation task and presents an iterative framework that can be used to unify many approaches to generalised segmentation.

5.1.1 Problem Formulation

There are basically two image characteristics that can currently be used for automatic segmentation purposes, i.e. texture and motion. The basic approach to both types of segmentation is the same and only differs in terms of the models used.

Texture segmentation involves the derivation of models based on the pixel colours, i.e. YUV data or RGB data for colour images, or simply Y data for monochrome images. Texture models are basically mathematical functions or processes that approximate the image colour over some region of support. Under these models, each pixel coordinate (x,y) is assigned a colour value based on the model function $g(x,y,\theta)$, where θ represents the model parameters. Any image can be approximated in terms of (*i*) one or more of these models and (*ii*) a segmentation map identifying the image regions where each model is supported. Let $I(x,y)$ be the pixel value at (x,y) and let $I^*(x,y)$ be the approximated pixel value based on the model. To represent the segmentation, let each pixel possess a label $\mathbf{z}(x,y)$, where $\mathbf{z}$ is a vector of binary values with elements z_i, such that

$$z_i(x,y) = \begin{cases} 1 & \text{if } I^*(x,y) = g(x,y,\theta_i) \\ 0 & \text{otherwise} \end{cases}$$

The image $I(x,y)$ can then be approximated by:

$$I^*(x,y) = \sum_i z_i(x,y) g(x,y,\theta_i)$$

Given an image $I(x,y)$, the task of texture segmentation is to identify both the segmentation labels $\mathbf{z}(x,y)$ and the model parameters θ_i, $i=0,1...,G$, where G is the number of models.

The task of motion segmentation may be summarised in a similar way, except that the models used are different. In the case of motion, the models are used to represent an image at time $t+1$ based on the image at time t. The image $I_{t+1}(x,y)$ is approximated in terms of the segmentation and motion models as follows:

$$I^*_{t+1}(x,y) = \sum_i z_i(x,y) g(x,y,I_t,\theta_i)$$

Note that this model function relies on the image at time t. The segmentation labels $\mathbf{z}(x,y)$ and motion parameters must be estimated based on the two images, i.e. I_{t+1} and I_t.

Based on the above model-based image approximations, segmentation can be treated as a minimisation problem. That is, the segmentation labels and the model parameters are chosen in order to minimise some chosen error function $e()$, taking account of the error between the actual image and the approximated image. This methodology is denoted as follows,

$$\hat{\mathbf{z}}, \hat{\mathbf{T}} = \arg\min_{\mathbf{z},\mathbf{T}} e(I, I^*)$$

Equation 5-1

where $\mathrm{T} = \{\theta_1, \theta_2, ... \theta_G\}$ represents the model parameters for each of the G classes. The above problem formulation for image segmentation is slightly restrictive, relying only on the pixel intensities as the observation data. More generally, image segmentation is carried out on the basis of observation vectors, sometimes called feature vectors. A special case of a feature vector is a scalar representing the pixel intensity, $I(x,y)$. In the general case, the observation is represented by:

$$\mathbf{X} = \left[\mathbf{x}_1 \quad \mathbf{x}_2 \quad .. \quad .. \quad \mathbf{x}_M\right]^T$$

where M is the number of image pixels and $\mathbf{x}_j$ represents the feature vector of the pixel j. The feature vector may be consist of any local measurements made at, or centred on the pixel in question, e.g. Y, U and V values, local motion parameters **[93]**, local texture moments and so on. The choice of features is critical to the performance of the segmentation system. Some bad choices of features will be highlighted in the examples given in section 5.1.3. It is assumed that the set of observations has a probability density function of some known form and that this function is dependent on a set of unknown parameters. The unknown parameters are denoted by the vector Ψ and the density function is written: $f(\mathbf{X};\Psi)$. For segmentation, the unknown parameter vector Ψ defines the parameter set $\{Z,T\}$, where $Z = \{\mathbf{z}_1, \mathbf{z}_2, \ldots\ldots \mathbf{z}_M\}$ represents the segmentation labels (one label for each pixel).

The unknown parameters are found using the principle of maximum likelihood (ML) estimation. The ML principle may be summarised as follows. Given some sample observation $\mathbf{X}=\boldsymbol{x}$, the density function becomes a function of the unknown parameters, sometimes called the likelihood function. The ML estimate is that parameter vector which makes the observations most likely, i.e. it is the value of Ψ which maximises the likelihood function $f(\boldsymbol{x};\Psi)$. Very often, it is the logarithm of this likelihood function (the log-likelihood function) that is maximised, i.e.

$$\hat{\Psi} = \arg\max_{\Psi} \log f(\boldsymbol{x};\Psi)$$

Equation 5-2

To illustrate this principle in a simple manner, consider a case where a random source generating independent events is being sampled. Assume that each observation x is independently distributed according to a normal density function with an unknown mean m and a standard deviation σ equal to one i.e.

$$f(x;m,\sigma)=\frac{1}{\sqrt{2\pi}}\exp\left(\frac{-(x-m)^2}{2}\right)$$

The only unknown parameter is the mean m. It is desired to estimate the mean of the source given 2 or more observations. This can be performed using the ML principle as follows.

Assume the source is sampled twice yielding observations $x1$=2 and $x2$=3. With these two observations, a joint density function is constructed noting that $x1$ and $x2$ are independent:

$$f(x1,x2;m)=f(x1:m)f(x2:m)=\frac{1}{2\pi}\exp\left(\frac{-(x1-m)^2-(x2-m)^2}{2}\right)$$

This density function may be further evaluated by substituting the sample values, yielding an expression in one unknown m:

$$\frac{1}{2\pi}\exp\left(\frac{-(2-m)^2-(3-m)^2}{2}\right)$$

This function is called the likelihood function and it is maximised to find the mean. This can be done most conveniently in this case by maximising the logarithm of the likelihood function. Maximisation is performed by taking the derivative of the log-likelihood function, setting it equal to zero and solving for m. This procedure yields the expected result: $m = (x1+x2)/2 = 2.5$.

This is a very simple example and the ML solution to the segmentation problem is more complicated requiring the use of the Expectation-Maximization algorithm as discussed in sub-section 5.2.

5.1.2 Iterative Solutions for Segmentation

The segmentation task may be thought of as a minimisation of error, as in the case of Equation 5-1, or a maximisation of probability, as in Equation 5-2. In either case, it is

clear that segmentation is an optimisation problem of very high dimensionality. There is one label to be estimated for very pixel and there are a number of model parameters. Assuming that there are M pixels in the image, G different classes and K model parameters per class, then there are $M+GK$ unknown parameters. In addition to the high dimensionality, there is the outstanding problem of choosing how many classes to apply in the segmentation. This number G shall be referred to as the model complexity. Naturally, a larger number of models will produce a lower approximation error or a higher value in the likelihood function. However, the purpose of segmentation is not to use a very large set of models and to partition the image into as many regions as there are pixels, for example. Instead, useful segmentations result by finding a good trade-off between the fidelity of the image representation and the complexity of the overall model.

Due to these difficulties, very few researchers have attempted to solve the segmentation problem by direct and simultaneous estimation of all the unknown parameters. Instead, the problem is broken down into a number of manageable steps arranged within an iterative framework as follows:

I. *Parameter Initialisation*: A finite number of classes are decided upon and the model parameters for each are initialised, i.e. $\mathrm{T}^{(0)}$. There are many initialisation strategies and some are described later in this chapter.

II. *Iterate until stability*

   A. *Pixel Labelling*: Each pixel is allocated to a particular class based on the model parameters $\mathrm{T}^{(k)}$. This yields a new segmentation $\mathrm{Z}^{(k)}$.

   B. *Model Parameter Estimation:* The model parameters are re-estimated based on $\mathrm{Z}^{(k)}$ to give $\mathrm{T}^{(k+1)}$

   C. *Model Complexity Adjustment*: It is ascertained whether the number of models/classes are consistent with the observation data. For instance, there may be redundancy, i.e. too many classes, in which case, some models may be eliminated. Alternatively, there may not be enough

models, in which case a new model is added. In either case, a change in the model complexity G necessitates some pixel reallocation.

In the above iterative approach, stability can be defined as the state where little or no pixels are being re-allocated at each iteration or where the values of the model parameters have stabilised. Several examples of this framework exist in the literature. Some of these are discussed in the next sub-section.

5.1.3 Segmentation Examples

Texture Segmentation and Clustering

Consider that it is desired to segment an image into regions of constant or approximately constant intensity. Assume that it is known that G classes exist. The model i has the form:

$$g(j,i) = m_i$$

where j represents a particular pixel. As previously, the image may be approximated as follows:

$$I_j^* = \sum_i z_{ij} m_i$$

and the task is to find the segmentation labels $\mathbf{Z}$ and the model parameters m_i for $i=1,2...,G$. The error at each pixel is specified as follows:

$$e(I_j, I_j^*) = \left| I_j - \sum_{i=1}^{G} z_{ij} m_i \right|$$

Equation 5-3

The image is initially partitioned into G non-overlapping blocks. For each of the blocks, the mean intensity value is computed and used to initialise the model parameter set.

Based on the initial models, each pixel j is allocated to the model which produces the smallest error as specified in Equation 5-3. Based on this segmentation, the mean parameters are re-estimated. This process of estimation and allocation continues until stability is reached.

The above illustration of texture segmentation was based only on a one-dimensional feature vector. Most images have a three-dimensional colour space, e.g. RGB or YUV. Each pixel has an associated vector representing the value of each colour component:

$$\mathbf{x}_j = \begin{pmatrix} y_j \\ u_j \\ v_j \end{pmatrix}$$

It is generally better to allow all three components to form the observation data than to restrict the observation data to just a single component. In this case, the model function is simply a vector-valued function of a vector valued parameter.

$$\mathbf{g}(i,j) = \begin{pmatrix} m_i^{(y)} \\ m_i^{(u)} \\ m_i^{(v)} \end{pmatrix}$$

Once again some error criterion is used. For instance,

$$\left\|\mathbf{x}_j - \mathbf{g}(\mathbf{i},\mathbf{j})\right\| = \left(y_j - m_i^{(y)}\right)^2 + \left(u_j - m_i^{(u)}\right)^2 + \left(v_j - m_i^{(v)}\right)^2$$

Equation 5-4

The process of segmenting the image on the basis of these 3-dimensional observation vectors is exactly the same as in the 1-dimensional case. The image space is sampled to obtain initial estimates for the G models. Based on these initial estimates, allocation is carried out by choosing at each pixel j the model which minimises the error criterion. This allocation procedure results in an initial estimate of the segmentation $\mathbf{Z}$. The model

parameters are re-estimated by taking the mean of the YUV data within each model's region of support. That is, for the Y component

$$m_i^{(y)} = \frac{1}{N_i} \sum_{j=1}^{M} z_{ij} y_j$$

where N_i is the number of pixels allocated to model i. The U and V means are computed in an identical fashion. The vectors storing the Y, U and V mean values are termed the cluster centres and the error function in Equation 5-4 denotes the distance between the feature vector at pixel j and the cluster centre i. Based on the new cluster centers, pixels are again re-allocated according to the distance metric and so on until stability is reached.

The approaches that have been outlined for texture segmentation are purposely simple. They illustrate the iterative nature of texture segmentation, but they ignore the model complexity issue and assume that pixels may be independently classified. Firstly, it is assumed that the model complexity G is known in advance and that the initial model parameters for each of the G models may be captured by a simple sampling of the image. This is rarely the case. The examples on motion segmentation in the next two sub-sections will illustrate how some determination of model complexity can be achieved. They are not necessarily the best ways, but they will give the reader some feeling for the concept of model complexity. Secondly, the pixel allocation procedures described so far, concentrate only on minimising the model approximation error at each pixel. As will be demonstrated in this chapter, this is insufficient for the production of coherent segmentations. The presence of image noise and the inevitable fact that the models are inexact will lead to very incoherent segmentations, where a given model is supported by many small spurious and disconnected groups of pixels. The examples on motion segmentation serve to present some mechanisms for imposing coherency on the segmentation. Additionally, the use of contextual pixel labelling is discussed in sub-section 5.2.4. For further and more detailed treatment of texture analysis and segmentation, see the work of Chellappa *et al* **[19]** and Tuceryan and Jain **[90]**.

Motion Segmentation Using Clustering

The clustering schemes illustrated in the previous sub-section can also be used to segment an image on the basis of motion. This is what is attempted in the work of Wang and Adelson **[93]**. This approach begins by estimating a motion vector for each pixel using a hierarchical optical flow method **[6]**. These motion vectors formed the observation data, i.e.

$$\mathbf{x}_j = \begin{pmatrix} dx_j \\ dy_j \end{pmatrix}$$

where *dx* denotes the horizontal displacement and *dy* denotes the vertical displacement. Roughly speaking, the optical flow field is estimated to minimise the brightness constancy constraint at each pixel co-ordinate $j=(x,y)$, i.e.

$$e(x,y) = I_{t+1}(x,y) - I_t(x+dx, y+dy)$$

The image is partitioned into G non-overlapping blocks and within each block an affine motion model is fitted to the optical flow field by a least squares approach. The affine model is of the form:

$$dx^*(x,y) = a_0 + a_1 x + a_2 y$$
$$dy^*(x,y) = b_0 + b_1 x + b_2 y$$

This initialisation step results in an initial set of motion models with parameters $\mathrm{T} = (\theta_1 \quad \theta_2 \quad \ldots \quad \theta_G)^T$, where each model is represented as:

$$\theta_i = \left(a_0^{(i)} \quad a_1^{(i)} \quad a_2^{(i)} \quad b_0^{(i)} \quad b_1^{(i)} \quad b_2^{(i)}\right)^T.$$

The allocation rule relies on the distance measure:

$$\left(dx_j - \left(dx^*\right)^{(i)}\right)^2 + \left(dy_j - \left(dy^*\right)^{(i)}\right)^2$$

Equation 5-5

The pixel j is deemed to be represented by the model i if model i produces the lowest value of the expression in Equation 5-5. The least squares parameter estimation and pixel allocation are alternated until stability.

In Wang and Adelson's work, the clustering approach was augmented by techniques addressing some of the general problems with segmentation, i.e.

- the assessment of model complexity,
- the attainment of spatial coherence in the segmentation, and
- *outlier* detection and processing.

The value of G, i.e. the number of models, is initially chosen to be very large and initial model parameters are obtained by dividing the image into small blocks and fitting an affine model to the optical flow field within each block. Some models are eliminated immediately due to the fact that the model fit within the particular block is not good enough. Subsequently, a model clustering process is used to reduce the model complexity. The initial step in this model clustering is to divide the parameter space of θ into a number of equal sized regions, each one with its own cluster centre. Each model of the initial set is then allocated to one of these cluster centres. Following this first allocation of models, each cluster centre is re-computed by simply taking the mean of all the models allocated to that cluster. Given the new cluster centres, the models are re-assigned and cluster centres are updated. If during this iteration, two cluster centres come within some pre-defined distance of each other, then the two clusters are merged into a single cluster. The iterations comprising reassignment, cluster centre updating and cluster merging continue until no more clusters are merged and no models are reassigned. The model complexity G is given by the number of clusters remaining and the initial model parameters are given by the cluster centres. Beginning with these G models, the steps of pixel allocation, parameter estimation and model clustering are iterated. Therefore, even in subsequent steps, models may be merged.

After each pixel allocation step, the resultant segmentation is post-processed to remove small regions, i.e. small connected groups of pixels supported by one or other of the models. This gives the segmentation a cleaner more coherent appearance and avoids the problems associated with the instability of model parameter estimation in small regions.

A final point to note is that not all pixels are allocated to a model. In order for a pixel to be allocated to a model, the model parameters not only have to minimise Equation 5-5, but also this minimum value must be lower than some pre-defined threshold. Pixels for which no model meets these criteria are left unassigned. These pixels are usually termed *outliers* since they are not well represented by any of the available models. The final step in the algorithm is to assign these outlier pixels to the model which minimises the intensity distortion between the current image intensity and the motion compensated previous image.

The work of Wang and Adelson is impressive and ambitious since they attempt to solve many of the general segmentation problems in their scheme. The scheme works very well on certain image sequences and less well on others. Two major problems with the scheme are apparent:

- The choice of local motion as the observation data was probably not good due to the fact that local motion is difficult to accurately estimate. Direct computation of optical flow **[2],[5]**, as was used, does not necessarily result in a true and meaningful motion representation of the scene. The first pitfall is that an estimation procedure based on local image characteristics is prone to the detrimental influences of image noise. To a certain extent, multiresolution approaches **[6]** have addressed this robustness issue. Secondly, for any kind of motion estimation, there is the requirement for a sufficient level of image detail or structure to drive the algorithm. Many pixels contain no local intensity variation and hence, local motion cannot be recovered unambiguously. Furthermore, for totally unambiguous recovery of motion, the local intensity structure should contain variations in both the x and y directions. When 2-D structure is lacking, the well-known "aperture problem" comes into effect. It has been

generally agreed that due to these difficulties, optic flow estimation is an underconstrained problem **[2]**. To further constrain the problem, smoothness constraints are used, but these are inconsistent with the natural motion discontinuities which occur at the boundaries of moving objects. The task then is to somehow avoid applying these smoothness constraints at occlusion boundaries. Thus, arises the "chicken and egg" problem of segmentation. No accurate motion can be measured without shape and no shape can be recovered without motion.

- The determination of model complexity relies upon the use of rather arbitrary thresholds. In the model clustering process, it is ordained that no two model clusters may be within some pre-defined distance of each other without being coalesced. Such thresholds can only be derived on the basis of experimental experience and typically, no single threshold suffices for every situation. When thresholds are used like this, it is important to have some criterion for adapting the threshold values to the context of each new situation. Notice also, that the system provides no possibility to recover from mistakes made in the initialisation stages. That is, it is possible that some valid motions are eliminated in the model clustering, but it is not possible to add new models. The determination of a suitable level of model complexity is a very difficult problem.

Quadtree-based Motion Segmentation

Sanson **[83]** developed a motion segmentation approach based on quadtree-partitioning of the image. At the first level, the image is partitioned into G non-overlapping blocks. On each of these blocks, an affine motion model is estimated directly from the image data using a Gauss-Newton method. At the second level, each block is split in quad-tree fashion, see the shaded block in Figure 5-1. It is tested whether it is appropriate to re-allocate any of the sub-blocks to any of the models existing in adjacent blocks. Four connectivity defines the adjacency relationship as shown in Figure 5-1.

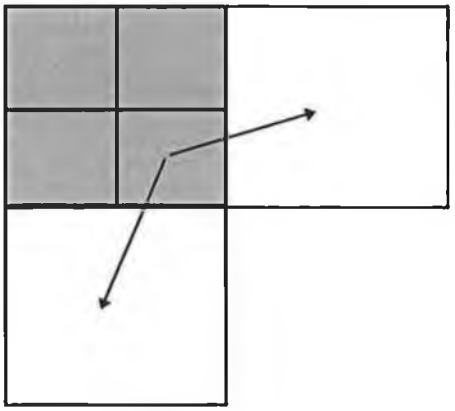


Figure 5-1: Quadtree decomposition and adjacency relationship

In fact, for each sub-block, there are 3 possibilities:

- The sub-block is not reallocated and remains with the parent model.
- The sub-block is allocated to a neighbouring model.
- The sub-block is attributed a newly estimated model different from either the parent model or the neighbouring model.

The sub-block allocation rule is based on the mean-squared error (MSE) computed over the sub-block pixels for each of the possible models. To decide between the first two cases, that path is taken which leads to the lowest MSE. To decide whether the sub-block might benefit from a new model, this new model is estimated and the corresponding MSE is measured. If this new model reduces the MSE by more than some specified percentage, the new model is adopted.

Following the re-allocation of the sub-blocks, the motion parameters within each connected region are updated. The third step in the iteration is to merge adjacent regions possessing a similar motion. This is done by constructing a weighted region adjacency graph, with each edge weight in the graph corresponding to the increase in the MSE which would result if the region's model was replaced by the model of its neighbour. A

minimal spanning tree approach is utilised to merge the nodes in the graph. Merging stops when the global error increases beyond some predefined threshold.

At each level/iteration, the same steps are repeated, i.e. sub-block allocation, parameter updating and region merging. It is unclear from the paper how successful this approach is. However, it is one of the first papers that viewed motion segmentation in an image coding context. The obvious goal of Sanson's system was to generate a segmentation with a minimum associated set of motion models, such that the some acceptable global motion compensated MSE is attained. It was acknowledged that the majority of bits in video coding are used to code the residual error, but it was also clear that an unlimited number of models/regions could not be supported while maintaining adequate coding efficiency. Sanson sought to find a good trade-off between the model complexity and the residual error. Once again, however, ad-hoc methods based on arbitrary thresholds were used to tackle this problem. While segmentation, by its nature, must use threshold-based decisions, it is best if these thresholds can be derived from, or traced back to, some high level concept or mathematical criterion. When the algorithm reacts badly, it is easier to re-analyse and modify a high level mathematical criterion than to search aimlessly for a threshold that works better. Just like Wang and Adelson, Sanson lacked such a formal criterion to help him solve the problem of model complexity. Furthermore, Sanson seemed to realise this fact, as he states: "segmentation can also be viewed as an optimisation issue. However, in that case, the criterion cannot be obviously formulated".

With regard to the problem of spatial coherency, the adjacency constraints imposed on the sub-block re-allocation and merging procedures are designed to produce large connected regions of support for each motion model.

5.2 Expectation-Maximisation, Minimum Description Length and Contextual Enhancements

It has been highlighted above that segmentation is usually achieved by adopting an iterative framework alternating between parameter estimation and pixel allocation. In

this sub-section, the segmentation problem is formulated as a maximum-likelihood problem and the EM algorithm is introduced as the solution. It is noted that the EM algorithm fits neatly into the general iterative framework. The difficulty of determining the required model complexity has also been highlighted. The minimum description length estimate is presented as a tool to properly address the model complexity estimation problem. The final aspect of segmentation that is highlighted in the previous sub-section is spatial coherency. It is shown how contextual pixel allocation rules can be formulated based on local Markov Random Field (MRF) models.

5.2.1 Maximum-Likelihood Estimation

Segmentation may be formulated as a maximum likelihood estimation problem as follows. Let the observation set be denoted by $\mathbf{X} = \left[\mathbf{x}_1 \quad \mathbf{x}_2 \quad .. \quad .. \quad \mathbf{x}_M\right]^T$, where $\mathbf{x}_j$ is the feature vector at the pixel *j*. The observed data is thought of as a realisation of an underlying piece-wise model. The model is described by the model parameters $\mathbf{T} = \{\theta_1, \theta_2, ... \theta_G\}$ and the support maps $\mathbf{Z}=\{\mathbf{z}_1,\mathbf{z}_2,...\mathbf{z}_G\}$. As before, the value of the support map $\mathbf{z}_i$ at the pixel *j* is denoted z_{ij}. It is equal to 1, if the model *i* is supported at the pixel *j* and equal to zero otherwise. Given **Z**, the observations **X** are assumed to have a conditional probability density function of some known form and dependent upon the unknown parameters describing the underlying model, i.e. $f_{X|Z}(\mathbf{X}|\mathbf{Z};\mathbf{T})$.

For a particular observation set **X**=*X*, this density function becomes a function of the unknown parameters and is called the likelihood function. The maximum likelihood estimate of the unknown parameters is given by maximising the likelihood function with respect to the unknown parameters. More often, the unknown parameters are found by maximising the log-likelihood function:

$$\hat{\mathbf{T}} = \arg\max_{\mathbf{T}} \log f_{X|Z}(X|\mathbf{Z};\mathbf{T})$$

To allow a more convenient representation of the density function and simplify the estimation resulting procedure, it is assumed that every observation is conditionally independent and distributed according to a common density function, thus:

$$f_{X|Z}(\mathbf{X}|\mathbf{Z};\mathbf{T}) = \prod_{j=1}^{M} f_j(\mathbf{x}_j|\mathbf{Z};\mathbf{T})$$

Additionally, since only one model is assumed to be supported at each pixel, the distribution of each observation $\mathbf{x}_j$ is dependent only on one model, i.e.

$$f_j(\mathbf{x}_j|\mathbf{Z};\mathbf{T}) = \prod_{i=1}^{G}\left(f_{ij}(\mathbf{x}_j;\theta_i)\right)^{z_{ij}}$$

Hence, the ML estimate of **T** corresponds to maximising the following log-likelihood function:

$$\log f_{X|Z}(\mathbf{X}|\mathbf{Z};\mathbf{T}) = \sum_{j=1}^{M}\sum_{i=1}^{G} z_{ij}\log\left(f_{ij}(\mathbf{x}_j;\theta_i)\right)$$

Equation 5-6

The problem with applying this strategy is that the support maps **Z** are not known and as such the estimation of the model parameters is not possible by direct MLE. The estimation of the model parameters **T** is therefore said to be a problem of incomplete data. Fortunately, if with a knowledge of **Z**, maximisation of Equation 5-6 is possible, then the joint estimation of **T** and **Z** can be given by the expectation-maximisation algorithm.

5.2.2 Expectation-Maximisation (EM)

The EM algorithm computes both **Z** and **T** by utilising an iterative 2-step approach, comprising the expectation step (E-step) and the maximisation step (M-step). This iterative structure is depicted in Figure 5-2. Applications of the EM algorithm require

that a finite initial set of models can be provided, i.e. $\mathbf{T}^{(0)} = \left\{\theta_1^{(0)}\theta_2^{(0)},....\theta_G^{(0)}\right\}$. On each EM iteration, the E-step computes the probability that each pixel belongs to each given model, i.e. the ownership probabilities. The ownership probabilities can be viewed as a soft segmentation. A hard segmentation may be derived by allocating each pixel to that model with the highest ownership probability. Given the new ownership probabilities, the M-step updates the model parameters. The iterations continue until the model parameters have stabilised, or alternatively, until the hard segmentation has stabilised. The description of each step is presented below. Note that, the mathematical derivation of the EM algorithm is omitted. However, it is emphasised that the presented formulae for the E-step assume that the segmentation labels are independently distributed. That is, in computing the ownership probability at a given pixel, no consideration is given to the neighbouring pixel classifications. For theoretical background on the EM algorithm, see **[25]**, **[48]**, **[76]** and **[89]**.

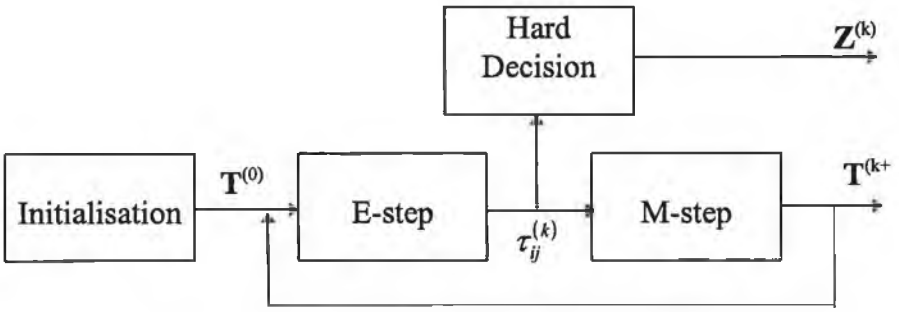


Figure 5-2: Basic iterative structure of the EM algorithm

To better illustrate the EM formulae, examples are given below. The examples are based on an observation set consisting of the intensity at each pixel, and each pixel's intensity value is distributed according to a Gaussian PDF, expressed as a function of the error with respect to a given model function $g(j,\theta_i)$, i.e.

$$f_{ij}\left(x_j;\theta_i\right)=\frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(\frac{-r_{ij}^2}{2\sigma_i^2}\right),$$

where, $r_{ij}=x_j-g\left(j,\theta_i\right)$

Equation 5-7

The Expectation Step

The E-step uses Equation 5-8 to compute the ownership probabilities for each pixel *j*, given the current set of model parameters. That is, τ_{ij} is the probability that pixel *j* belongs to model *i* given the current model estimates. The formula can use prior probabilities $\pi_{ij}=\Pr\left(z_{ij}=1\right)$. These prior probabilities may be chosen in a variety of different ways and are evaluated as part of the M-step:

$$\tau_{ij}\left(\mathbf{x}_j;\mathbf{T}\right)=\pi_{ij}f_{ij}\left(\mathbf{x}_j;\theta_i\right)\Big/\sum_{t=1}^{G}\pi_{ij}f_{ij}\left(\mathbf{x}_j;\theta_t\right)$$

Equation 5-8

Substituting the Gaussian PDF of Equation 5-7 into Equation 5-8, the E-step results in:

$$\tau_{ij}\left(\mathbf{x}_j;\mathbf{T}\right)=\frac{\pi_{ij}}{\sigma_i}\exp\left(\frac{-r_{ij}^2}{2\sigma_i^2}\right)\Big/\sum_{t=1}^{G}\frac{\pi_{ij}}{\sigma_t}\exp\left(\frac{-r_{ij}^2}{2\sigma_t^2}\right)$$

In basic terms, given unbiased prior probabilities, the ownership probability τ_{ij} is large if the residual error r_{ij} is small in relation to the variance parameter σ_i.

The Maximisation Step

The M-step uses the ownership probabilities to formulate a ML criterion for the estimation of the model parameters **T**, as given in Equation 5-9:

$$\mathbf{T}^{(k+1)} = \arg\max_{\mathbf{T}} \sum_{i=1}^{G} \sum_{j=1}^{M} \tau_{ij} \log f_{ij}\left(\mathbf{x}_j; \theta_i\right)$$

Equation 5-9

Substituting the Gaussian PDF and eliminating terms that are independent of the model parameters, the maximisation becomes a minimisation problem as expressed by:

$$\mathbf{T}^{(k+1)} = \arg\min_{\mathbf{T}} \sum_{i=1}^{G} \sum_{j=1}^{M} \tau_{ij} r_{ij}^2$$

Each individual model parameter is given by:

$$\theta_i^{(k+1)} = \arg\min_{\theta} \sum_{j=1}^{M} \tau_{ij} r_{ij}^2$$

This is a weighted least squares problem where the weights are defined by the ownership probabilities. The model parameters may therefore be obtained by a Gauss-Newton method as presented in chapter 4. The M step is also responsible for computing any other unknown parameters such as the variance parameter of the PDF, σ. In addition, the prior probabilities must be computed in anticipation of the next E-step. One possible approach is to set them equal to the current ownership probabilities, i.e. $\pi_{ij}^{(k+1)} = \tau_{ij}^{(k)}$.

The Hard Decision

The support maps **Z** may be obtained by setting z_{ij} to 1 if,

$$\tau_{ij} > \tau_{tj}, \forall t \neq i$$

Equation 5-10

and to zero, otherwise.

In many ways, the EM fits neatly into the iterative framework for segmentation which was presented in sub-section 5.1.2. It provides a formal well-understood method for solving the extremely difficult problem of joint estimation and segmentation. Nevertheless, the EM in no way answers the question of model complexity estimation.

The next sub-section presents the minimum description length concept. This provides a sound theoretical framework for segmentation problems that do not have prior information regarding the model complexity.

5.2.3 Minimum Description Length (MDL)

Segmentation, like many other signal processing tasks, may be thought of as fitting some model to a set of observed data so that the observed data is accurately represented. In particular, segmentation involves the fitting of a piece-wise model to the observed data, i.e. the pixels or feature vectors. Given a fixed number of initial models, the EM algorithm will fit these models to the observed data. Unfortunately, it is not known in advance how many local model pieces are sufficient to describe the observation data. If the task of segmentation is to obtain a perfect fit to the data, then it is conceivable to choose as many model pieces as required. However, a degenerate case of segmentation occurs when there are as many models as pixels. On the other hand, if few models are used, then the model representation will be inaccurate and the segmentation will not reflect the image content. Several approaches, e.g. those of Sanson and Wang, to obtaining a suitable level of model complexity have been reviewed in sub-section 5.1.3. These generally follow a top-down strategy, involving model and region splitting, or a bottom up strategy involving model merging. Threshold-based rules are used to make the split/merge decisions. Very often, however, no justifications exist for the choice of the threshold value. The threshold is usually obtained by trial and error and when a given image causes the algorithm to fail badly, one is reduced to adjusting the thresholds on a case by case basis. The use of arbitrary thresholds and parameters in decision criteria is futile. Critical decisions should be based on some sound criterion which fits in with the overall goal of the segmentation. One such decision-making criterion is described now.

Consider that the task is to compactly represent the image in terms of a finite set of models and a segmentation. The finite model set may be chosen from a infinitely large set of models. For simplicity, it is assumed that only models of a common form may be

used, e.g. the segmentation procedure may be limited to using only affine models. Given this, the representation length will be made up of 2 main parts:

1. Model Representation: The model parameters of each model chosen from the infinite set will have to be encoded. Additionally, the support of each chosen model will have to be encoded.
2. Model Error: The model fit error at each pixel will have to be encoded.

In this scenario, it may be seen that increasing the number of models will reduce the length of the model error part of the representation, but it also results an increase in the length of the model representation. Conversely, minimising the model representation length by limiting the number of models, only results in increasing the model error. Therefore, minimisation of the overall coding length is achieved through finding the optimal balance between the information length of the model representation and that of the model error. This idea is the essence of what is referred to as the Minimum Description Length principle **[77],[78]**. While the MDL principle is easily explained by consideration of image coding, more generally, it is apt in the information theoretic sense to believe that the best model is that which results in the most compact representation of the observed data. Using this concept, segmentation is formulated as an MDL estimation problem as follows:

$$\hat{\Psi} = \arg\min_{\Psi} H_{X(\Psi)}(\mathbf{X}(\Psi))$$

where $\Psi = (\mathbf{Z}, \mathbf{T})$ represents the model parameters and segmentation support maps, as before and $H_{X(\Psi)}(\ )$ is the length of the representation of the observed data in terms of the model parameters and segmentation. This length is now expanded into the constituent parts of model parameters, segmentation and model error/residual.

Let R denote the residual of the model fit as represented by:

$$\mathbf{R} = \mathbf{X} - M(\Psi)$$

Now, the MDL estimate of model parameters may be written as:

$$\hat{\Psi} = \arg\min_{\Psi}\left(H_{R(\Psi)}(\mathbf{R}(\Psi)) + H_{\Psi}(\Psi)\right)$$

The description length is further developed by specifying probability density functions for the quantities being encoded and where applicable, the accuracy with which each quantity is encoded. The relationship between the probability density and the encoding length is simply:

$$H_X(\mathbf{X}) = -\log_2 f_X(X)$$

For convenience, the natural logarithm is used from now on, bearing in mind that $\log_2 x = C \ln x$ where the constant $C = (\ln 2)^{-1}$.

To within this constant multiplier, the description length can be then written as:

$$H_{X|Z,T}(\mathbf{X}|\mathbf{Z},\mathbf{T}) = -\ln f_{R|Z,T}(\mathbf{R}|\mathbf{Z},\mathbf{T}) - \ln f_Z(\mathbf{Z}) + H_T(\mathbf{T})$$

Equation 5-11

Assuming that the residual data are independently distributed according to a normal density function and that these residuals are encoded to the nearest integer then (see **[3]** for details):

$$\ln f_{R|Z,T}(\mathbf{R}|\mathbf{Z},\mathbf{T}) = \sum_{j=1}^{M}\sum_{i=1}^{G} z_{ij}\left(-0.5\ln(2\pi) - \ln\sigma_i + \frac{r_{ij}^2}{2\sigma_i^2}\right)$$

Equation 5-12

Using the same independance assumption for **Z**,

$$\ln f_Z(\mathbf{Z}) = \sum_{j=1}^{M}\sum_{i=1}^{G} z_{ij} \ln \pi_{ij} ,$$

where π_{ij} = Pr(z_{ij}=1)

Equation 5-13

Finally, some means is required to represent the length of the model parameters **T**. Let each of the *G* models have *k* real-valued parameters. As proposed initially by Rissanen and commonly used:

$$H_T(\mathbf{T}) = \sum_{i=1}^{G} \frac{k}{2} \ln M$$

Equation 5-14

Combining these last three equations, we get the MDL estimate which has been used for the purposes of motion estimation and segmentation by Ayer and Sawhney **[3]**.

The main use of the MDL criterion is for hypothesis testing rather than the direct estimation of the model parameters and model complexity. In line with the iterative framework discussed in sub-section 5.1.2, the MDL concept is used to adaptively adjust the model complexity at each iteration. It may be applied in one of the following ways.

Model Elimination and Merging

In order to test if two models denoted A and B can be merged into one model, the description length is firstly computed based on the initial set of models and the associated supports. The pixels supporting model A are re-assigned to model B and the description length is re-computed. If the description length is reduced, then it is assumed that model A can be eliminated and its support is transferred to model B. An alternative hypothesis test would involve reassigning the support of model B to model A and re-computing the description length. A positive result would cause the elimination of model B. A third possibility exists whereby the supports of A and B are merged and a new model parameter vector is estimated for the merged support. This new model replaces A and B if it results in a reduction of the description length. This latter kind of

model merging has been used by Zheng *et al* in **[98]** to design a system very similar to that of Sanson. In contrast to the top-down approach of Sanson, Zheng *et al* designed a bottom up system relying totally on region merging. Similarly, however, model merging was carried out under the constraints of adjacency relationships by the construction of a weighted region adjacency graph. Each node in the graph represented a single model and it associated support. Each edge connecting each pair of nodes had a length equal to the reduction in the description length which would result from merging the two model supports. The minimisation of the description length was achieved by searching the graph for the longest edge and merging the two associated nodes. All edge lengths were then re-computed and a further search for the longest edge was carried out. This process continued until no further reduction could be achieved, i.e. the edge lengths were all negative.

In their work on motion segmentation, Ayer and Sawhney used a similar pruning strategy. At each iteration of the segmentation, the most redundant model, i.e. that model whose removal led to the largest decrease in the description length, was removed. In contrast to the previous examples, no adjacency constraints were imposed. In fact, in the general case, any pixel of the redundant model's support could be re-assigned to any other model of the set.

New Model Hypothesis Test

Examples of MDL hypothesis testing are not confined to the reduction of the model set but can also be used to test if the current model set is inadequate. For this, it is assumed that some new candidate model is estimated. The new model is allowed to compete for support with the existing models using some simple assignment rule or based on an E-step. Given the new support, the description length can be computed. The new model and its support are retained if the description length is decreased.

The disadvantage of using the MDL criterion in the above fashion is related to the complexity of performing so many hypothesis tests. When the set of hypotheses is very large, this brute-force approach quickly becomes infeasible. While region adjacency constraints can be used to reduce the number of hypothesis combinations, it would be

more convenient if some direct estimation approach could be designed to extract the minimal set of hypotheses from some initial set. This is what is attempted in **[24]** wherein a gradient descent method is applied to directly minimise the MDL. In simple situations, this is reported to work well. For more complex situations where there are a large number of hypotheses or where the supports are complex, the method is embedded within a continuation method in order to avoid local minima. Unfortunately, no comparative study has been performed as yet to evaluate the relative performance of this method against the brute force method.

5.2.4 Contextual Enhancements of EM and MDL

Having covered the fundamentals of the Expectation-Maximization method and the Minimum Description Length principle, this sub-section examines the role of statistical dependance assumptions in applying these tools. Consider that there are a number of image models with known parameters and it is desired to classify each pixel to one of the models. The non-contextual allocation rule is to allocate each pixel to the model which maximises the likelihood function of the observed data. However, non-contextual labelling rules are sensitive to model inconsistencies and image noise and result in complex incoherent segmentations. These noisy segmentations may be unusable for subsequent purposes. If the application is compression, then noisy segmentations are costly in terms of bit-rate. If MDL-based hypothesis testing is being used, then models with such noisy supports will almost certainly be eliminated. If the application is image interpretation, then such noisy segmentations can be unintelligible and over-complex. To get around this difficulty, conditional dependence in the segmentation labels is introduced implying a change in the underlying probability density function. The new density function results in a likelihood function which contains terms encouraging spatial coherence among the segmentation labels. Frequently, conditional dependence is expressed in the form of a local Markov random field (MRF). This idea was first used by Besag when he developed the Iterated Conditional Mode (ICM) algorithm **[8]**. Similar ideas are easily incorporated into the EM algorithm.

The use of contextual information is also very important when computing the description length in MDL-based hypothesis testing. The description length example given in the previous sub-section made some very loose assumptions on the probability distributions used to describe the residual data and the support map data. For effective MDL testing, it is important that the information content in each part of the representation is not over-estimated. Independence assumptions, while simplifying the derivation and implementation, often lead to over-estimation of the coding length. Some improvements in coding length estimates are discussed here also.

Contextual Expectation-Maximisation

Many examples of the Expectation-Maximisation (EM) method arise in the segmentation literature **[4]**, **[94]** and **[97]**. It has been found that the use of the basic EM algorithm, as outlined above, usually results in very noisy segmentations. This has been attributed to the nature of the assumptions on which the basic EM algorithm is based. The first assumption of the observations **X** being independently distributed given the segmentation **Z**, is retained in most applications of EM, and of other related techniques such as Iterated Conditional Modes (ICM). The second assumption that the labels within **Z** are independently distributed is much to blame for bad performance. Many of the attempts to avoid the implications of this second assumption are based on locally dependent MRF models[9].

Recently, Zhang *et al* **[97]** produced an EM algorithm for model-based texture segmentation that proposed a means of avoiding the detrimental effects of the independence assumptions. Their work resulted in an EM algorithm, whereby the E-step involved the consideration of contextual information in the form of an MRF model for the segmentation map. Effectively, the E-step becomes a *maximum-a-posteriori* (MAP) estimator of the soft posterior probabilities τ_{ij}. The MAP estimator is implemented using a simulated annealing approach or by an approximate recursive method similar to the ICM approach. By using the simulated annealing approach, both the independance assumptions with regard to the observed data **X** and those associated with the segmentation labels **Z** are overcome. With the ICM-like approach, only the

[9] For a brief summary of MRFs and their uses, see **[49]**, Chapter 13.

independance assumption of **Z** is tackled. The effect of the ICM-like approach is that the prior probabilities computed within the M-step and used in the E-step are given by:

$$\pi_{ij} = f\left(z_{ij} = 1 \middle| \mathbf{z}_{(j)}\right) \text{ or } \pi_{ij} = f\left(z_{ij} = 1 \middle| \tau_{(j)}\right)$$

depending on whether the hard or soft probabilities of the neighbourhood were used.

Any local MRF model is characterised by its neighbourhood and the conditional probability density function. One of the most common MRF models is the Ising model **[49]**, which is based on an eight neighbourhood scheme as depicted in Figure 5-3. Under this model, the M-step (of the EM algorithm) simply becomes

$$\pi_{ij} = \exp\left(\beta u_{ij}\right) \Big/ \sum_{t=1}^{G} \exp\left(\beta u_{ij}\right)$$

Equation 5-15

where u_{ij} is the number of pixels in the neighbourhood of j that belong to model i. The parameter β is a positive constant and G is the number of models. To utilise the soft probabilities generated by the E-step, it has been suggested to evaluate u_{ij} by $\sum_{v} \tau_{iv}$, where the summation is taken over the prescribed neighbourhood of j. Again, it is worthy of note that the same contextual constraints were applied in developing extensions to the ICM method, see **[49]** and **[69]**.

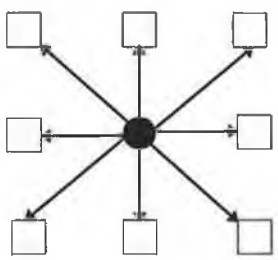
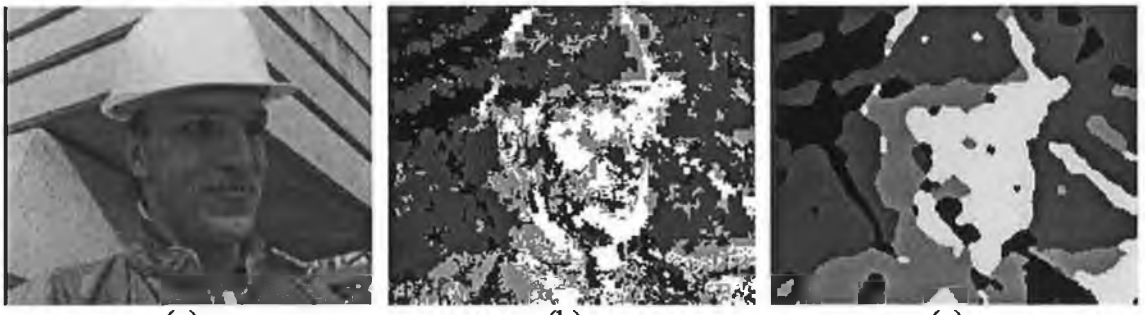


Figure 5-3: The square pixels constitute the 8-neighbourhood about the circular pixel.

For the purposes of illustration, an EM-based motion segmentation algorithm was applied to an image. Four initial motion models were obtained by partitioning the image into 4 blocks and using a robust motion estimator to compute the motion parameters. Based on the initial models, an EM algorithm was applied to segment the image. Both non-contextual EM and contextual EM approaches were used The contextual approach used Equation 5-15 with $u_{ij} = \sum_{v} \tau_{iv}$ defined on a local 8-neighbourhood. The results are shown in Figure 5-4. Neither result is very impressive due to the ad-hoc initialisation procedure and the lack of any mechanism to ascertain the model complexity. Nevertheless, the effect of the contextual constraint is clearly manifested in a cleaner looking segmentation.



(a) (b) (c)

Figure 5-4: Illustration of contextual EM using a local MRF model: (a) the image to be segmented, (b) motion segmentation using non-contextual EM, (c) motion segmentation using contextual EM.

In an application of EM to motion segmentation by Weiss and Adelson **[94]**, departing a little from the idea of locally dependent MRFs, it was proposed to compute u_{ij} using a weighted function of the soft probabilities within the local neighbourhood, i.e.

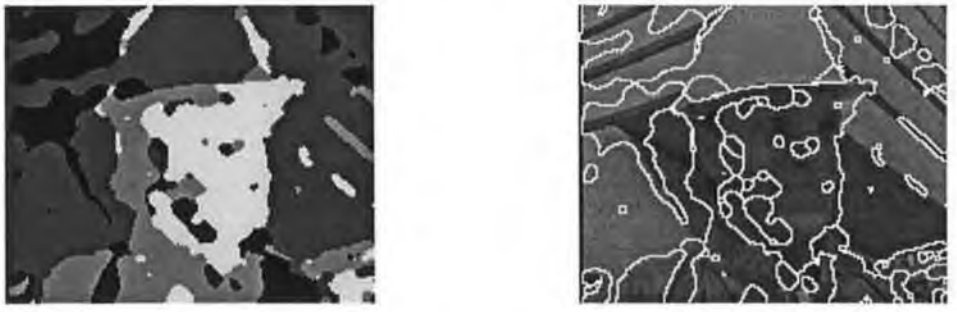
$$u_{ij} = \sum_{v} \tau_{iv} w(j, v)$$

Equation 5-16

The authors suggest the use of a weighting function $w(j,v)$ which favours those pixels which are closer to j in geometric and intensity terms, e.g.

$$w(j,v) = \exp\left(-\frac{\|j-v\|}{\sigma_1^2} - \frac{\|I(j)-I(v)\|}{\sigma_2^2} \right)$$

The overall effect of this idea is that local groups of pixels with similar intensity values tend to be attributed to the same motion model. This approach is motivated by the reasonable assumption that motion boundaries often coincide with transitions in the image intensity or colour. Hence, the resulting segmentations are intended to align the contours to texture edges, unlike those contours shown in Figure 5-4. This contextual constraint was implemented with an 8-neighbourhood and with $\sigma_1^2 = 200, \sigma_2^2 = 40$. These settings effectively nullify the effect of the pixel proximity, i.e. all pixels v in the 8 neighbourhood of j are assumed to be the same distance from j. The results in Figure 5-5 illustrate that the alignment of the contours with the real moving objects of the scene is not significantly better than in Figure 5-4c. Different results can be obtained by enlarging the neighbourhood and by varying the parameters of the contextual constraint. This is a criticism of the method, since it is not generally desireable to have many parameters within the algorithm without any means to estimate their value. Despite the difficulty with this approach, the general motivation for aligning motion and texture boundaries is worthy of note.



(a) (b)

Figure 5-5: Illustration of contextual EM using colour constraints. (a) output segmentation (b) segmentation contours overlaid on image.

To conclude the discussion on contextual EM, it is clear that simple contextual strategies vary only in the manner in which the prior probabilities are computed. It

should also be noted that cleaner segmentations are obtained if, instead of using Equation 5-10 for the hard decision, the support maps **Z** are obtained by setting z_{ij} to 1 if,

$$\pi_{ij} > \pi_{tj}, \forall t \neq i$$

Equation 5-17

Contextual MDL

In the formulation of the description length of sub-section 5.2.3, independance assumptions were made for the probability distributions of the residual data **R** and the segmentation labels **Z.** In tackling the independance assumptions within the description length, it is tempting to assume that some given coding scheme is in operation and utilise these coding tools directly. For instance, in Zheng and Blostein, a simple chain coding procedure was used to compute the length of the information associated with **Z.** In line with this thinking, one could apply DCT-based coding to the residual data. The complexity associated with these coding approaches may be prohibitive, however, and it may be more appealing merely to capture some of the coding tool's assumptions within a probabilistic model. For approximations to the coding length for the shape information, it is probably reasonable to exploit local correlation via the same MRF models used in the contextual EM algorithms. After all, shape compression methods using context-based arithmetic encoding (CAE) approaches **[44]**, **[15]** rely on the same kind of correlations. Assuming that the Ising model is used, then Equation 5-13 becomes

$$\ln f_Z(\mathbf{Z}) = \sum_{j=1}^{M}\sum_{i=1}^{G} z_{ij}\left(\beta u_{ij} - \ln \sum_{t=1}^{G} \exp\left(\beta u_{tj}\right)\right)$$

Equation 5-18

This equation is used to estimated the coding length for shape within the MDL hypothesis testing framework. However, a better approach would be to utilise a simple context-based probability table storing the probability of binary events given the context in a predefined neighbourhood. The probability table could be trained on a typical set of

good segmentations. This would more closely mimic the behaviour of CAE coding approaches and could not be considered overly complex.

With regard to the encoding of the residual data, it is well known that in motion-compensated video coding systems, the degree of local correlation in the prediction error image is very small and most compression is achieved simply by the fact that the errors are tightly distributed about zero. Hence, independance assumptions for the residual data in the case of motion segmentation are not so inaccurate. The situation may very well be different for the case of texture segmentation where it can be imagined that significantly more correlation will exist in the residual data. In cases where the residual data is still correlated, it is advisable to use some kind of simple local linear predictor and encode the prediction errors rather than the residual data themselves.

5.2.5 Motion Segmentation using an EM-MDL Framework

Ayer and Sawhney [4] developed a system primarily for the purpose of simultaneously detecting multiple motions in a scene. Their framework, depicted in Figure 5-6, utilises the EM algorithm for segmentation and parameter estimation and MDL for hypothesis testing. The EM-MDL steps are applied in a coarse to fine manner to the image data based on a multiresolution image pyramid.

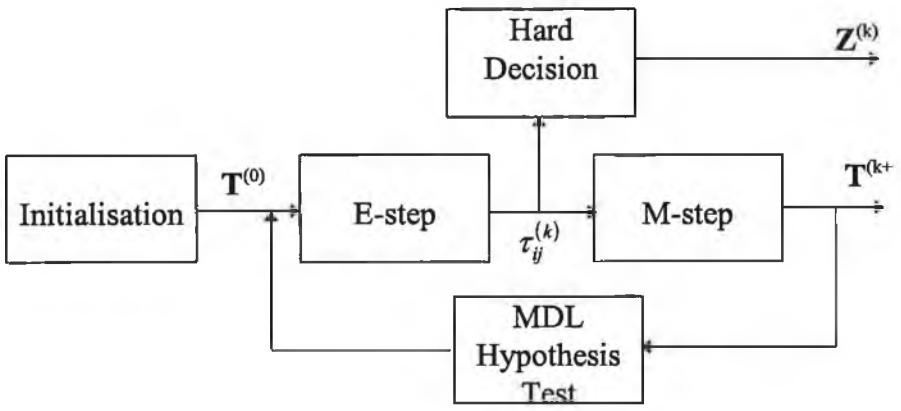


Figure 5-6: EM-MDL framework for computing multiple motions.

The initialisation step computes an initial set of polynomial motion models based on a block-based partition of the image. A non-contextual EM step is then carried out based on the assumption that the residual errors $r_{ij} = I_j(t+1) - I_j(t, \theta_i)$ are independently distributed with a zero mean Gaussian distribution.

The E-step is given by

$$\tau_{ij} = \frac{\pi_i}{\sigma_i} \exp\left(\frac{-r_{ij}^2}{2\sigma_i^2}\right) \Bigg/ \sum_{t=1}^{G} \frac{\pi_t}{\sigma_t} \exp\left(\frac{-r_{ij}^2}{2\sigma_t^2}\right)$$

where the prior probabilities obey: $\pi_{i1} = \pi_{i2} = \ldots\ldots = \pi_{iM} = \pi_i$

The M-step comprises the estimation of the motion parameters for each model and an update of the prior probabilities.

$$\pi_i = \frac{1}{m} \sum_{j=1}^{m} \tau_{ij}$$

$$\theta_i = \arg\min_{\theta} \sum_{j=1}^{m} \tau_{ij} r_{ij}^2$$

The motion parameters are robustly estimated by an iteratively re-weighted least squares approach. The M-step also computes the values of the variance parameters σ_i using a robust estimate. The means of providing these robust estimates has already been described in chapter 4.

Following the EM-step, the MDL-step attempts to eliminate the most redundant of the models and the iterations continue. The coding length is estimated in the way described in sub-section 5.2.3 except that in coding the support maps, linear predictive coding is used to incorporate some notion of statistical dependance. When the iterative steps have settled upon a minimum set of models, a hard segmentation is computed by means of an ICM-like algorithm resulting in a relatively clean final segmentation.

To illustrate the results of the EM-MDL approach, a variation of this system has been implemented by the author. In subtle contrast to the application of Ayer and Sawhney which was to recover motion estimates, the main goal of the author's implementation was to recover the coherent shape of the moving objects. For that reason, the author's implementation places more emphasis of contextual approaches, i.e. a contextual EM method is used and the MDL exploits local correlation in the segmentation labels. Some results are presented in Figure 5-8 and Figure 5-9. The details of implementation are as follows:

- A three layer pyramid is used.
- Translational, affine or quadratic motion models can be used.
- The initialisation step begins with the lowest resolution image of the pyramid by dividing it into 16 square blocks and estimating the initial motion models. A number of non-contextual EM steps are performed until stability. At this lowest resolution, the MDL steps are not carried out. Models are only eliminated if the EM iteration results in an unsupported model, i.e. if no pixels are assigned to a particular model, it is eliminated.
- For the remaining pyramid layers, contextual EM-MDL steps are used for each iteration. The β parameter of the contextual model used in the EM is chosen such that the strength of context is greater for the finest pyramid layer. This approach is chosen so that in the early stages of motion estimation, the context has only a small influence. Once the motion parameters are more stable then the strength of context can be increased in order to get a more coherent segmentation. The value of β in the finest pyramid level was chosen to be 1.5 based on recommendations given in **[49]**. Values for the coarser levels were chosen in arbitrary fashion in order to lessen the MRF field strength. It is conceivable that improvements may be obtained by the use of an estimator for β.
- The contextual EM used for the results of Figure 5-8, is characterised by the use of an 8-neighbourhood local MRF model, i.e. it is the same as that used for the results in Figure 5-4.

- The coding length used in the MDL step is the basic formulation of sub-section 5.2.3 but using statistical dependence when encoding the shape. The context is taken into account along the lines of Equation 5-18 but with the use of a causal local neighbourhood as shown in Figure 5-7. Additionally, with respect to encoding the residual, it is necessary to detect what pixels are outliers and what pixels are not. A uniform probability density function is applied in the coding of the outlier pixels.
- For each MDL step, each existing model is removed in turn. For each model removed, it is necessary to reallocate the pixels prior to computing the coding length. This reallocation is performed by first normalising the posterior probabilities, contextually adjusting these probabilities according to Equation 5-15 and applying the rule described by Equation 5-17. In this way, pixels that are associated with the missing model are re-allocated in a contextual manner.

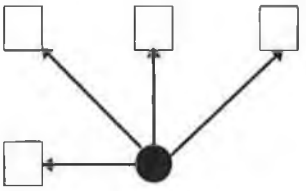


Figure 5-7: Causal 4-neighbourhood used in coding length estimate of shape in MDL-step

(a) (b)

Figure 5-8: Results of the EM-MDL framework for motion segmentation (Foreman). Foreman is viewed with a moving camera and the man is the only moving object. Quadratic motion models were used.

(a) (b) (c)

Figure 5-9: Results of the EM-MDL framework for motion segmentation (Calendar). Calendar is viewed with a zooming camera. The moving objects are the calendar, the ball and the toy train. Affine motion models were used.

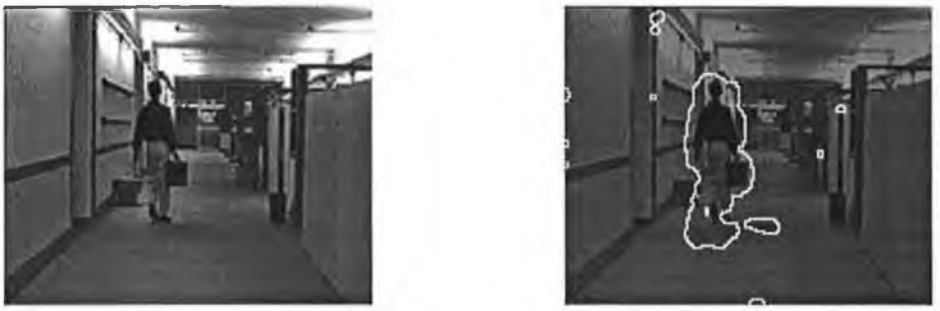
The results of this EM-MDL approach to motion segmentation are quite impressive, but clearly the results do not lend weight to the belief that semantic segmentations can be achieved by automatic means. Apart from this general comment, some specific observations can be made on the basis of the results.

The first observation relates to the benefits of MDL-based hypothesis testing. As can be noted by comparing Figure 5-4 and Figure 5-8, the use of more and better initial model hypotheses allied to the MDL-based model elimination mechanism results in significantly better segmentations. Starting from 16 initial motion hypotheses, all but 3 remain in the final segmentations. Given no initial knowledge about the locations and numbers of moving objects in each scene, the algorithm returns segmentations which distinguish the moving objects with a minimum of motion models. However, on the more negative side, it should not be overlooked that the brute force approach to MDL hypothesis testing is extremely intensive computationally.

The second observation relates to the fact that the regions of the segmentation clearly coincide with the real moving objects. This shows that the affine and quadratic motion models, as estimated by the EM algorithm, can be relied upon to accurately model real image motion. A high confidence can be placed on the fact that the estimated motion parameters constitute real and meaningful information about the motion in each scene. This is an extremely important fact which means that the motion parameters can be safely relied upon for object tracking purposes. Following on this observation, the EM-MDL algorithm is used to develop a new object tracking algorithm in chapter 6.

The third observation relates to the effects of using MRF contextual approaches for labelling the pixels within the EM. On one hand, it has to be acknowledged that the segmentations are extremely clean and coherent, as is intended. On the other hand, the MRF approaches have done nothing to ensure that the segmentation contours are accurate, i.e. in most cases contours do not become aligned with the real object's contours. Additional measures are required to improve upon this aspect.

In relation to the inaccurate placement of contours, an element of blame must be placed on the fact that the ownership probabilities (used to eventually give the final hard segmentation) are computed based only on the suitability of one motion model over the others for accurately representing the pixel intensity. It can happen in un-textured picture areas that motion is not a good discriminant. In the case of Foreman, it can be observed that the intensity edges defining the man's hat are classified as belonging to the man, whereas the un-textured interior is classified as part of the background. This can happen because both the man's motion and the background motion may be equally suited to representing the un-textured area. A similar problem is evident in the Calendar test case where all the textured lettered area of the calendar is classified as one moving object, whereas the untextured area of the calendar is classified as belonging to the background. The previously discussed approach of Weiss and Adelson suggests a likely way of improving this situation by utilising intensity and colour information in computing the prior probabilities. In chapter 6, a new contextual EM-based motion segmentation algorithm is developed along these lines. The new algorithm makes use of the morphological watershed method for texture segmentation with the intention of generating segmentation contours on common motion and texture edges. The morphological watershed is introduced now.



(a) (b)

Figure 5-10: Results of the EM-MDL framework for motion segmentation (Hall Monitor). The only moving object is the man walking down the corridor. Affine motion models were used.

5.3 Morphological Image Segmentation

This chapter has concentrated on statistical approaches to segmentation and most of the examples have emphasised motion segmentation. Motion analysis is a natural tool for segmenting video sequences since very often the objects of interest have a distinct motion within the scene. Sometimes, however, the object of interest is contained within a still image or it is a still object within a video sequence. In cases like this, texture analysis is more useful than motion analysis. Additionally, it has been noted that even in motion segmentation approaches, an appreciation of the intensity structure within the image is very important. That is, very often motion boundaries coincide with texture edges. This idea is being used to improve motion segmentation methods **[14]**, **[94]**.

In the domain of texture analysis and segmentation, morphological tools **[86]** are extremely simple and effective. The classical morphological segmentation approach relies on three different tools:

- filters for image simplification,
- the morphological gradient, and
- the watershed.

The process of graylevel image segmentation is described below by way of illustration. The discussion relies upon an understanding of the basic morphological concepts such

as the structuring element, the operations of erosion and dilation and so on. See **[79]** for basic tutorial style coverage of morphology and **[86]** for theoretical aspects.

5.3.1 Filters and Image Simplification

The first step of the process is image simplification. There is a particular class of morphological filter that significantly simplifies the image by removing unwanted intensity structures, while exactly preserving the shape of the remaining intensity structures. There are two basic classes of filter in morphology: the *open* filter and the *close* filter. The basic *open* filter is used to remove peaks in the intensity function. The basic *close* filter removes valleys in the intensity function. For the 1-D case, the effects of these filters may be seen in Figure 5-11. The severity (the extent/width of the peaks and valleys that are removed) of the filter is controlled by the size of the structuring element. The open and close filters may be applied in cascade (open-close) to eliminate both peaks and valleys of the intensity function.

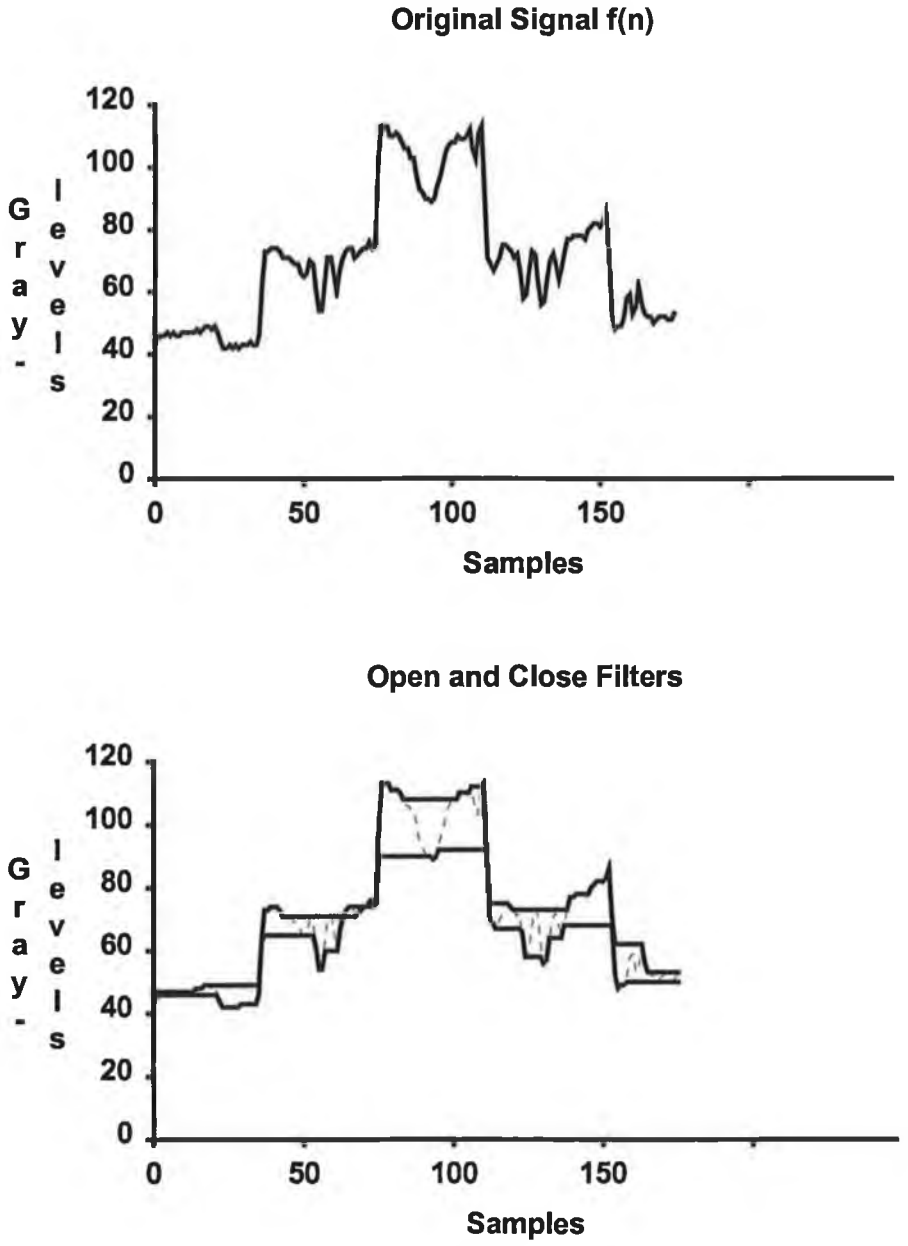


Figure 5-11: 1-D open/close filters. The open filter results in a signal which is always less than the original. The close filter results in a signal which is always greater than the original filter. The upper graph shows the original signal. The lower graph shows this same signal enveloped above and below by the close filtered signal and the open filtered signal respectively.

When applied in 2 dimensions as in image processing, these basic filters cause unwanted distortion. Figure 5-12b illustrates this point. For the basic open-close filter, almost every important feature of the original image is completely obliterated. A segmentation based on this 'simplified image' would not produce desireable results.

(a) (b) (c)

Figure 5-12: 2-D morphological filters. (a) the original image, (b) the result of an open-close, (c) the result of an open close by reconstruction. All filters used a 5x5 square structuring element.

By utilising what is termed morphological reconstruction **[91]**, it is possible to develop a more sensitive class of simplification filters. Figure 5-12c shows the result of an open-close by reconstruction. Notice how much of the superfluous detail, e.g. the text on the calendar, the spots on the ball, has been removed, while at the same time, the remaining contours are positioned in the same locations and maintain their shape almost precisely. Filtering by reconstruction is part of the class of connected operators **[80]**, so-called because they can discriminate between various connected components in the image. The small connected components corresponding to the spots and text are filtered out whereas the larger connected components, e.g. the calendar, are retained.

5.3.2 Morphological Gradient

The next step is to compute the morphological gradient. The morphological gradient is used to highlight intensity transitions within an image. The gradient image is constructed by eroding the input image, dilating the input image and then taking the difference between the erosion and the dilation. In the morphological segmentation approach, the watershed is usually applied to the gradient of the simplified image to yield the final segmentation. Figure 5-13a shows an example of the gradient image obtained from the simplified image.

(a) (b) (c)

Figure 5-13: The steps involved in morphological segmentation: (a) shows the morphological gradient of the simplified image for Calendar. (b) shows the marker image where each black connected component represents the interior of a region to segmented. (c) shows the final segmentation contours overlaid upon the original image.

5.3.3 Marker Extraction

The task of marker extraction is to mark the interiors of the regions that are to form the final segmentation. This step can be done manually. For example, a user may merely use a mouse to draw on the objects to be segmented. Alternatively, in a fully automatic system, some analysis may be done to perform the marking. As an example, the marker image shown in Figure 5-13b is derived by analysing the gradient image for flat regions and then filtering the resulting flat regions to remove very small ones. The black regions mark the interiors of the regions to be segmented, whereas the remaining (white) pixels are part of the *uncertainty region,* i.e. it is assumed that somewhere in this uncertainty region lie the contours of the segmented regions.

5.3.4 Watershed

The final step in the morphogical segmentation approach is to apply the watershed **[18]**, **[92]** to the gradient and the marker image. This step finds the segmentation contours within the uncertainty region. In fact, a connected operator (reconstruction by erosion) is first applied to remove small variations from the gradient image. The result of this is that the modified gradient only contains the dominant peaks corresponding the largest edges in the local sense, i.e. sub-peaks residing upon major peaks are removed or flattened. The watershed can be imagined as a flooding process in a mountainous terrain. The markers correspond to lakes in the valleys and the valleys are separated by

ridges as defined by the bright areas of the gradient image. The water level begins to rise. At a certain points, the water level will be such as to merge two adjoining lakes into one. The points where this merging takes place are marked as the watershed lines and these form the contours of the final segmentation. Figure 5-13c shows the watershed lines overlaid upon the original image.

It is noted that morphological segmentations have a cleaner appearance than those generated by statistical means. This is due to the fact that the idea of connectivity in inherent in most morphological tools, and hence connectivity is dealt with more elegantly than by means of adjacency graphs as sometimes used with statistical approaches. On the other hand, the exact segmentation obtained by the morphological approach is highly dependent on the nature of the simplification filter. Larger filters will eliminate more of the detail and hence the segmentation will have fewer regions. Automatic choice of the filter size for the task at hand is not easy. For instance, it can be seen that the calendar in the image Calendar is covered by approximately 12 regions. It would have been better if it was covered by only one. The problem is to choose a filter which eliminates all the other image detail and maintains the main structure of the calendar. This is far from trivial. Additionally, even on merging these 12 regions, the resulting contours are not in agreement with the human perception of the calendar shape. This is again due to the unavoidable inadequacies in the simplification process. In particular situations such as medical or biological imaging, morphology has proven effective. For general segmentation purposes, it is envisaged that this approach will be significantly improved by the use of user assistance. As a quick example, let's say the user is interested in segmenting the calendar from the scene of the example. The user draws a line over the image roughly indicating the location and shape of the calendar. This line is dilated by some amount such that the thin contour line becomes a thick one. This thick contour line is then treated by the watershed as the region of uncertainty. Given this and the modified gradient of the image, the contour is snapped by the watershed onto the intensity edges as defined in the gradient. There are other ways of achieving this snapping process, but it is believed that the use of the watershed is among the most intuitive and simple.

5.4 Summary

This chapter has reviewed some of the recent literature on image segmentation. Both texture and motion segmentations have been discussed. Both statistical and morphological means have been described.

For the statistical approach, it has been shown how segmentation can be construed quite conveniently as an optimisation problem. The high complexity of solving this problem is emphasised and it is illustrated how iterative algorithms can be used to find an elegant solution. In particular, the EM algorithm has been discussed at length. The difficulty of initialising these iterative algorithms with reasonable initial models is discussed and several approaches are outlined by way of example. The necessity for finding the simplest set of models for image segmentation is also illustrated and MDL-based approaches are described. Finally, the need for local contextual models to constrain the segmentation is demonstrated. A promising EM-MDL iterative framework is described and implemented. This provides a new improved MDL estimate by taking into account local correlation in segmentation labels and uses MRF contextual constraints in the computation of the EM. Based on this implementation, some results are given for the problem of motion segmentation. These results are among the best motion segmentation results presented to date. Importantly, it is emphasised that results for all sequences have been generated without adjusting the algorithm's parameters. This basic approach is believed to be among the most powerful segmentation frameworks available today. However, several problems have been noted from observations made on the basis of the results. While it is clear that the moving objects are captured in the segmentations, the contours are not sufficiently accurate for enabling high quality object-based editing, for example. In the next chapter, this approach will be further extended to deal with spatio-temporal segmentation based on motion, i.e. object tracking. Also, to tackle the hitherto observed problems, some contextual improvements are proposed and tested.

For the morphological approach, the watershed provides a powerful tool for texture analysis producing segmentations composed of connected components and with excellent contour localisation. This can be used to segment images automatically or in

some supervised manner. It can also be used to align rough segmentations with the images intensity edges, which is very useful indeed.

In conclusion, based on the results given in this chapter, the accurate segmentation of semantic objects cannot be attained by automatic means using today's technologies. Today's technologies do provide us with promising tools for tackling texture and motion segmentation. It is envisaged that a combination of motion and texture analysis augmented with some clever use of user input will yield a very efficient means of segmenting semantic objects from general image and video content. The next chapter takes this idea one stage further for video segmentation by combining texture and motion analysis within a system to be used for tracking identified objects through a video sequence.

6. TRACKING MOVING OBJECTS

With regard to requirement for high quality segmentation of semantic objects, it has been argued in the previous chapter that a user assisted approach is necessary. Given a video sequence, it is conceivable and even necessary that the user identifies the object of interest in the first frame of the sequence. Given this initial semantic segmentation, it is also conceivable that a computer-implemented algorithm could be employed to track the semantic object as it moves through the subsequent frames of the sequence. It is believed that effective tracking algorithms are among the most important means by which the computer can shorten the laborious task of the segmenting semantic video objects. This chapter describes a system developed by the author which allows the tracking of moving objects in a video sequence. The system is similar in structure to that discussed in **[14]**. However, the presented tracking system is novel in the following ways:

- It is implemented using statistical motion estimation and segmentation tools based on the same EM and MDL methods introduced in the previous chapter. This is first use of the EM-MDL tools in a tracking application.
- For tracking purposes, the EM algorithm has been implemented using a new contextual E-step encouraging both spatial and temporal coherence.
- The spatial aspect to this contextual step (described in sub-section 6.2) uses the morphological watershed to create segmentations with contours aligned on motion and texture edges. This idea is quite powerful and is not limited to the tracking application. It can be used to improve on the motion segmentation results presented in the previous chapter as demonstrated by the results in the following sub-sections.
- The temporal aspect takes the previous frame's segmentation into account when developing the probabilities used to classify each of the pixels in the current frame. This is described in sub-section 6.1.1.
- The coding length estimate used for the MDL hypothesis testing takes temporal correlation in the segmentation into account. Once again, this step is necessary to preserve temporal coherence in the resultant segmentations and it is described in sub-section 6.1.1.

- Finally, the system provides a framework by which objects may be tracked even when their motion complexity varies greatly over time. The tracking model complexity is adjusted by means of a simple hypothesis generation and MDL-based validation strategy. This strategy is discussed in sub-section 6.1.2.

6.1 Overview

In the system, each semantic object is regarded as a collection of moving regions. These moving regions are identified, tracked and finally summed to reform the semantic object again. This concept, which is very similar to that used in **[45]** and **[46]**, is illustrated in Figure 6-1 and provides the possibility of tracking objects that are not only moving with a rigid motion, but also those moving with flexible motion. This is very important, since generic objects exhibit generic (flexible) motion. In the following, the term "motion segmentation" is used to refer to the segmentation of an image or image region into moving regions as depicted in Figure 6-1a. When referring to a segmentation approximating the shape of a semantic object as in Figure 6-1b, the term "object segmentation" or "semantic segmentation" is used. The regions of an object segmentation are formed by the union of one or more regions of the motion segmentation.

(a) (b)

Figure 6-1: (a) Motion segmentation for the image. (b) Object segmentation by region merging, the semantic object shape is given by R1+R2+R8+R7+R6+R9.

The proposed tracking system relies on the estimation of accurate scene motion. The motion representation is a piece-wise polynomial model comprising a motion segmentation and a distinct polynomial model representing the motion within each motion region. At each frame, the shapes of the regions within the previous frame are

projected forward into the current frame by using the previous frame's motion representation. This gives an initial estimate of the new shapes. The EM is applied to refine the shapes and the motion estimates within each of the regions. The EM algorithm in this case uses a robust GN estimator in the M-step and an E-step availing of both spatial and temporal contextual information. The use of temporal information in the EM imposes the necessary consistency in the temporal dimension to facilitate tracking. The forward projection of the shape and the refinement of the projected shape comprise what is termed *the projection step.* Because the accuracy of the motion representation is of paramount importance, it is necessary to adapt the model complexity on a frame-by-frame basis. This is intended to be of benefit for tracking objects whose motion over time varies greatly from simple rigid motion to more flexible motion. To allow for this, a number of additional steps are introduced into the system. For a given frame, when the projection step has been completed, it is endeavoured to identify new motions within the scene. This is done by *the detection step*, i.e. producing new moving region hypotheses using an EM algorithm, and then by testing each of these hypotheses using the MDL approach, i.e. *the validation step*. An additional feature of the system is the use of texture analysis in the development of the contextual constraints placed on the EM algorithm. This results in the contours of the tracked objects being well aligned with the texture contours of the original images. The high level structure of the tracking system is shown in Figure 6-2 and is very similar to that presented in **[22]** for coding purposes.

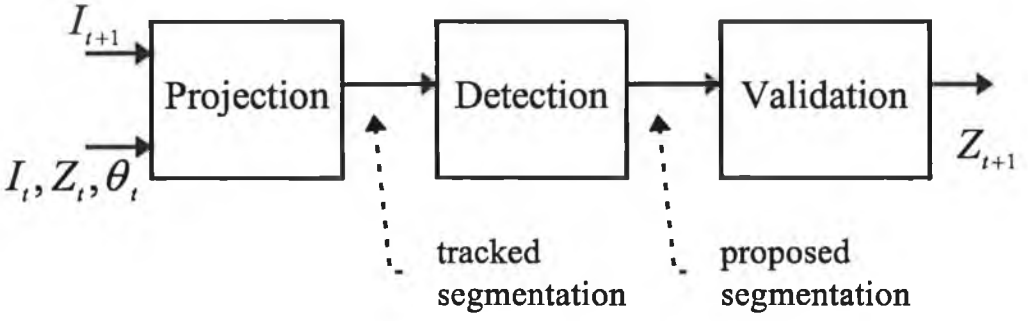


Figure 6-2: High level structure of the tracking algorithm as applied at frame *t+1*

6.1.1 The Projection Step

With reference to Figure 6-2, the inputs to the projection step are:

- the current image I_{t+1},
- the previous image I_t,
- the previous frame's motion models $\Sigma_t = \{\theta_i\}_t$, and
- the previous motion segmentation Z_t, denoting the supports for each of the motion models.

The first sub-step is to produce a projected segmentation for time t+1 and a set of prior probabilities for the segmentation at time t+1. This is accomplished by forward-projecting the support of each model. That is, a binary mask is created for each support i, whereby a pixel has a value of 1 if it belongs to support i and a value of 0 otherwise. The pixels of this binary mask are then displaced according to the inverted motion model parameters, i.e. θ_i. The displaced binary mask is then subjected to an NxN mean value filter[10]. This has the effect of blurring the mask around the edges, while leaving the interior values equal to 1 (see Figure 6-3). When this process has been applied for every support, there is one blurred displaced mask for each motion model present in the previous frame. From these masks, a prior probability on the tracked segmentation at time t+1 is computed as follows. Let b_{ij} be the value of the blurred displaced mask associated with the support of motion model i at the pixel j. It is assumed that the probability that pixel j supports model i at time t+1 is conditional upon the segmentation/motion at time t and that this probability is given by:

$$\delta_{ij} = \frac{b_{ij}}{\sum_{i=1}^{G} b_{ij}}$$

Equation 6-1

[10] N=7 was used for the results presented in the latter part of this chapter.

For pixels where the denominator of Equation 6-1 is 0, then the priors are set equal, i.e. $\delta_{1j} = \delta_{2j} = \ldots\ldots = \delta_{Gj}$. An initial motion segmentation of the image at time t+1 can be obtained by setting z_{ij} to 1 if,

$$\delta_{ij} > \delta_{ij}, \forall t \neq i$$

and to zero otherwise. In addition, the motion of each support within this initial segmentation is initialised with the previous frame's motion. This has the purpose of providing better initial estimates to the subsequent EM-MDL estimation step which is used to refine the initial shape and motion estimates.

(a) (b)

Figure 6-3: Illustration of how to develop the temporal priors. (a) a binary mask depicting a forward projection of the region shape into the current frame. (b) based on this mask, a low pass filter is applied to produce the blurred mask. The brightness of each pixel in this blurred mask is related to the probability that the pixel is part of the moving region.

The EM estimation steps avail of the prior probabilities as derived in Equation 6-1 in order to ensure that the final segmentation does not diverge far from the initial one. Additionally, some local spatial contextual information is integrated in order to allow a classification of the pixels which is spatially coherent and respecting the textural contours of the original image. The exact use of spatio-temporal context is described in sub-section 6.2.

In order to allow for moving regions leaving the field of view or being occluded, an MDL hypothesis test is carried out on certain motion models. This hypothesis test can result in the elimination of models which have very small supports. The MDL estimate

used is the same as that described in the previous chapter, with the exception that temporal correlation is taken into account in the shape coding part. That is, the shape code length is given by:

$$\ln f_Z(\mathbf{Z}) = \sum_{j=1}^{M}\sum_{i=1}^{G} z_{ij} \ln \pi_{ij} ,$$

where

$$\pi_{ij} = \frac{\delta_{ij} \exp(\beta u_{ij})}{\sum_{t=1}^{G} \delta_{tj} \exp(\beta u_{tj})}$$

Equation 6-2

and where u_{ij} denotes the number of neighbours of pixel j which support model i. A causal 4-neighbourhood scheme is recommended. The use of the temporal prior probabilities in this way averts any danger that a large tracked motion region can be eliminated.

The result of the projection step is two-fold:

- a tracked motion segmentation reflecting the new shapes and positions of the moving regions,
- a polynomial motion representation associated with each region.

At this stage, a semantic segmentation of the image can be attained by merging the regions of the motion segmentation. As shown later, the tracking system can be easily configured such that this region merging becomes trivial.

6.1.2 The Detection and Validation Steps

The detection and validation steps are present in the algorithm because it is clear that the motion complexity of an object will vary over time. At some times, the object will have a very simple motion capable of being represented by a single affine model, for example. At other times, the motion will be more complex and will consequently require many different supported motion models for accurate representation. The

tracking algorithm must be capable of staying in touch with the motion complexity, since if it does not, then the projection step already described will not be effective and the result will be a severe loss of tracking in situations of complex motion. This sub-problem of adapting the model complexity is tackled as usual by a system of hypothesis generation and testing. The detection step produces new moving region hypotheses and the validation step tests their efficacy.

As stated already, the projection step produces a number of distinct motion models and the tracked segmentation represents the support of each model. For each model/support, there exist what are termed *outlier* pixels. An outlier pixel is one whose intensity value is not well represented by the motion model that it supports. In the present system, a pixel j supporting a model i is an outlier if the residual error at the pixel has a sufficiently large magnitude. In particular, if $|r_{ij}| > 2.5\sigma_i$, then the pixel j is deemed to be an outlier of the model i, given that the residual errors of this model are normally distributed with a standard deviation of σ_i. Outliers are quite common and may be caused by image noise and other unpredictable events. However, outliers can also betray the presence of a motion which is presently not accounted for. Therefore, the detection step uses these outlier pixels to seed the estimation of new motion models. These new motion models are then allowed to compete for support with the existing motions within an EM algorithm.

For the purposes of reducing complexity in the system, the detection step has been kept quite simple. For each model present in the tracked motion segmentation, a single new model is set up to compete with each one. That is, if there are 2 models, i.e. A and B in the tracked segmentation, then 2 new models, i.e. C and D are set up, with C competing of support with A and D competing for support with B. Competition between A and C is restricted to the zones already supporting A and similarly for the competition between B and D. This effectively means that this detection step is a region splitting process. As mentioned, competition is enabled by the EM algorithm with fixed prior probabilities in order to delineate the desired regions of competition. Additionally, a new contextual constraint is utilised based on the morphological watershed. The exact details of the EM algorithm given here are presented in sub-section 6.2. At this point, it is enough to

describe the manner in which the initial estimates of the supports for the new motion models are arrived at.

The first step is the detection of the outlier pixels. This yields an outlier mask. The outlier pixels are first eroded using a 3x3 structuring element and then dilated using a 5x5 stucturing element. This has the effect of removing isolated outlier pixels and then enlarging any remaining outlier clusters. Each pixel of this modified outlier mask is then allocated to one of the new motion models. As an example, outlier pixels occupying the support of model A would be allocated to model C, whereas those occupying the support of model B would be allocated to model D. These allocations determine the initial supports for the new models and a first M-step is applied on that basis to arrive at a first estimate of the parameters for the new motion models. The reason for the morphological filtering described above, is so that a reasonable initial estimate of motion is possible, based on some consistent set of pixels.

Based on the initial motion estimates and supports for the new models, a number of EM iterations are performed to allow the new models to compete for support and to refine their parameter estimates. When the EM algorithm has stabilised, a segmentation is produced representing the supports of the existing motions and the new motions. This segmentation is termed the *proposed motion segmentation*. Hypothesis validation is carried out by firstly measuring the MDL based on the proposed segmentation and all its associated models. One by one, the new motions are removed. When a new motion is removed, the pixels of its support revert to being allocated to the original existing motion and the parameters of this existing motion are restored to the values they originally had in the tracked segmentation. The MDL is, once again, measured in the absense of the new motion. If the MDL is not reduced, then this new motion is accepted as a valid hypothesis. Otherwise, the new motion and its support are discarded. The final segmentation is the same as the tracked segmentation except when any new motion hypothesis is deemed valid. In that case, the final motion segmentation contains the support for the valid new motion(s). This segmentation and the associated motions are then used to perform the projection step for the next frame.

6.2 New Contextual Methods

In chapter 5, there has already been some discussion on local MRF fields used for the purposes of producing clean coherent segmentations. However, it is observed that results are seldom ideal. It can be noted that MRF fields will produce nice smooth contours in the segmentation, but these contours are often somewhat inaccurate with respect to the human perception of where the contours should be. When motion does not sufficiently constrain the segmentation field, then the MRF constraint will simply opt for the smoothest contour. Additionally and more to the point, MRFs can also be responsible for propagating more dominant labels into under-constrained images regions. To avoid these kind of problems, two new spatial constraints are introduced based on texture segmentations derived by the application of the morphological watershed. Both encourage the segmentation to align the motion contours to intensity edges as defined in the watershed.

The constraints presented here are motivated by the need for segmentations to be clean and coherent, while not suffering from the negative effects of local MRF constraints. Furthermore, they are motivated by the assumption that texture contours and motion contours should be aligned. That is, in general, it is assumed that each independently moving object in a scene has an occluding boundary which separates it from the other objects and that this occluding boundary is characterised by a transition in the motion field of the scene *and* the intensity map of the image. In chapter 5, the morphological approach to segmentation is highlighted due to the fact that very clean texture segmentations can be produced. Additionally, each class of pixels in the segmentation forms a connected component termed a watershed region. Finally, contours are generally very accurate due to the connected operators used in the simplification process. All these features make the morphological watershed very attractive in the domain of texture analysis and it is chosen here to form the basis of two related spatial constraints for application to the motion segmentation problem.

The first constraint may be summarised by contrasting it with the MRF-based constraints. The MRF constraint is characterised by a neighbourhood scheme where:

- the neighbourhood scheme is regular, i.e. the same neighbourhood scheme is used at each pixel and
- the neighbourhood scheme is overlapping, i.e. the neighbourhoods of adjacent pixels overlap.

Now, consider that each pixel *j* has a neighbourhood scheme which:

- may be unique to that pixel, i.e. the neighbourhood scheme is not regular,
- may not necessarily overlap with the neighbourhoods of neighbouring pixels, i.e. the neighbourhood scheme is not overlapping at all pixels, and
- is defined by those pixels which fall into the same watershed region as the pixel *j*.

Based on the usual mathematical form of the MRF constraint, the watershed-based constraint can be summarised as follows:

$$\pi_{ij} = \exp\left(\beta V_{ij}\right) \Big/ \sum_{t=1}^{g} \exp\left(\beta V_{tj}\right)$$

where

$$V_{ij} = \frac{1}{N_{Sj}} \sum_{v \in S_j} \tau_{iv} .$$

Equation 6-3

In Equation 6-3, S_j denotes the watershed region of the pixel *j*. It is a region/neighbourhood about pixel *j* defined by a spatial (watershed) segmentation of the current image. The parameter β is a positive number specifying the strength of the spatial constraint within each neighbourhood. For simulations, it has been set to 1.5 for every neighbourhood, but results do not exhibit a large dependance on its value.

The use of the watershed in this way was first presented in **[14]**. Results show that it can be extremely effective under certain conditions. Figure 6-4 demonstrates the strengths and weaknesses of the approach as applied within a motion-based EM-MDL segmentation framework. Note that these hard segmentations are attained by labelling

the pixels according to the maximum value of π_{ij}. It can be noted from the results that the use of this constraint and hard decision is equivalent to classifying watershed regions, instead of individual pixels, i.e. all pixels of a given watershed region are classified to the same model. This can be effective when, as happens in Foreman, the contours of the moving object are captured by the watershed segmentation. However, if the watershed does not accurately capture these contours, then the 'inaccurate' contours will also be reflected in the final motion segmentation. Some displeasing results of this nature can be observed in the Calendar results and to a lesser extent in the case of Hall Monitor. A worst case of this negative behaviour occurs when the moving object is small and the watershed has been produced using large filters. In such cases, the object is eliminated by the simplification filter used when deriving the watershed regions. Its contours are not, therefore, reflected in the watershed and hence the moving object can never be detected. An instance of this worse case effect is presented in **[14]**. In order to avoid this kind of effect, the results of Figure 6-4 are generated on the basis of a very fine watershed, i.e. a small simplification filter was used. While the worst case effect is not so much in evidence, it is clear that certain important contours are not captured, i.e. those of the spotted ball are not captured, while those of the calendar itself are quite inaccurate in certain areas. On the positive side, these segmentations are indeed very clean and coherently presented. More importantly and in contrast to the results given in the previous chapter for MRF constraints, the segmentation contours are extremely accurate in many cases.

The coherency exhibited by these segmentations is a feature which is used in the tracking algorithm by the detection/validation step. The detection step has the task of proposing new models and ascertaining the support of the new models. The validation step eliminates any model which is not useful in reducing the MDL. For a new model to pass the validation test, it is extremely important that the support is coherent, i.e. the shape part of the coding length is small. The constraint just described meets this need very well. On the other hand, it has been found that it is not suitable for use in the projection step due to the fact that incorrect contours ordained by the detection step are, more often than not, carried through into the subsequent frames. Additionally, it has

also been noted that the contours of a tracked moving region which at some point in time coincide with a watershed line, generally tend to cling to that watershed line. Hence, for the projection step, it is desirable to use the watershed in a weaker capacity. Additionally, for the projection step, it is required to integrate the spatial constraint with the temporal probabilities as derived by Equation 6-1.

A weaker version of the watershed-based constraint is obtained by modifying the basic MRF neighbourhood scheme at the pixel j such that the influence of the neighbourhood pixels not in the same watershed region as the pixel j is down-weighted. For example,

$$\pi_{ij} = \exp\left(\beta u_{ij}\right) \Big/ \sum_{t=1}^{G} \exp\left(\beta u_{ij}\right)$$

where

$$u_{ij} = \sum_{v \in N_j} w_v \tau_{iv} \text{ and } w_v = \begin{cases} 1 \text{ if } v \in S_j \\ 0.25 \text{ otherwise} \end{cases}$$

Equation 6-4

Note that N_j is the normal 8-neighbourhood and S_j is the watershed region of the pixel j.

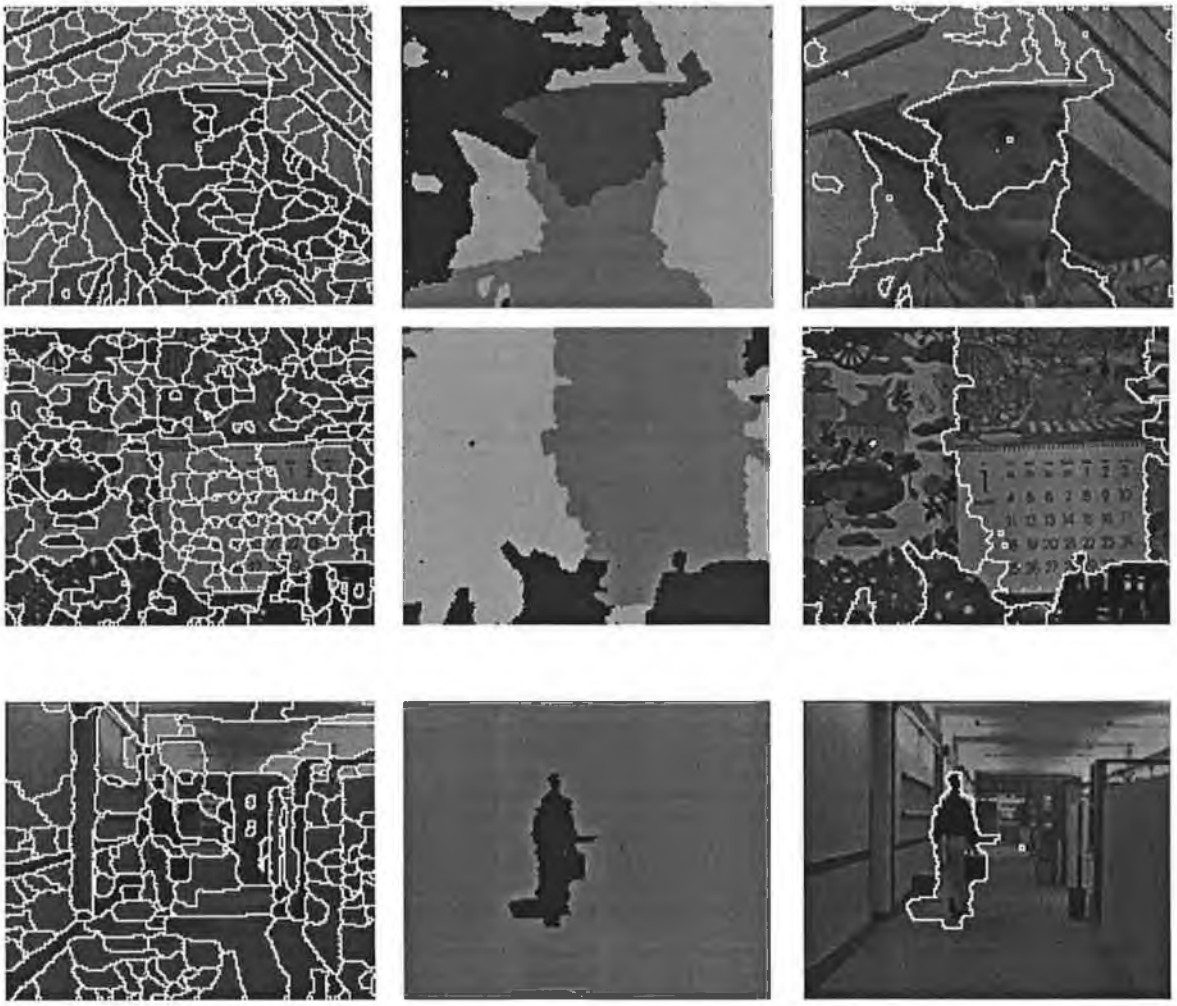


Figure 6-4: These are the results of EM-MDL motion segmentation using a watershed-based contextual constraint as implemented on Foreman (Frame 224, 25Hz), Mobile (Frame 25, 10Hz) and Hall Monitor(Frame 52, 10Hz). The left column shows the locations of the watershed lines. The middle column shows the motion segmentation map and the right column shows the motion contours overlaid upon the input images.

The constraint of Equation 6-4 is the same as the local MRF-based constraint, except that its ability to propagate dominant labels across the watershed lines is diminished. This constraint produces slightly noisier results than the full-blown watershed constraint. On the other hand, it avoids the negative behaviour of the straightforward MRF constraint, while having a tendancy to align motion contours to the watershed contours. The weakened watershed constraint is used effectively in the projection step where it is augmented by the temporal prior probabilities as follows:

$$\pi_{ij} = \delta_{ij} \exp\left(\beta u_{ij}\right) \Big/ \sum_{t=1}^{G} \delta_{ij} \exp\left(\beta u_{ij}\right)$$

Equation 6-5

This spatio-temporal constraint allows a classification of the pixels in the uncertainty regions surrounding the contours of the motion projected segmentation within the projection step. The classification is biased, encouraging results to be similar to the segmentation of motion-projected segmentation, while also encouraging smooth contours which are aligned to a watershed lines where appropriate.

To summarise on the use of contextual constraints, the detection step of the tracking algorithm requires spatial coherency and hence it uses an EM algorithm based on Equation 6-3. In the projection step, a spatio-temporal constraint with a weaker reliance on the watershed is used, based on Equation 6-5.

6.3 Simulation Results of Tracking Algorithm

The algorithm described above is designed for tracking moving regions within a video sequence. The algorithm relies on having a segmentation of the first frame. This can be provided by automatic means, for example, by a motion segmentation algorithm. Alternatively, a more accurate initial segmentation may be provided by a supervised approach. This sub-section presents results for both types of initialisation. It is demonstrated that in the case where the initial segmentation corresponds well with the semantic object, the tracking algorithm is capable of generating good semantic segmentations for subsequent frames under conditions of moderate motion.

6.3.1 Tracking with Automatic Initialisation

Figure 6-5, Figure 6-6 and Figure 6-7 show the results of the tracking system as initialised by an automatic motion segmentation. With this initialisation, the algorithm produces a temporally coherent motion segmentation sequence, but it is necessary to manually allocate each sub-region of the motion segmentation between the object of interest and the background. While this is not the ideal initialisation procedure, the

results still demonstrate the performance of the tracking capability. The top row of each figure shows the partition of the images into the moving regions and the bottom row shows examples of how these regions, when merged, form semantic objects. In the results shown, this merging of the regions is achieved manually. That is, for each frame shown, the user chooses the motion regions which best cover the semantic object. This is a non-ideal method for merging. As shall be seen with supervised initialisation, this manual work can be eliminated.

While tracking is good in general, some small problems are evident from the results.

- For small objects, e.g. the men in Hall Monitor, accurate contours are difficult to obtain.
- The initialisation procedure based on motion means that the first frame segmentation is non-ideal and often these problems are not eliminated by the tracking procedure. This can be seen when considering that the contours of the calendar in the Calendar sequence. The boundary of the calendar is not well captured in the first frame and although improvements are evident as tracking ensues, the exact boundary is never obtained.
- From the segmentations for Calendar, it would not be possible to extract only the moving ball from the scene without also taking the train. See the results for frame 150 in Figure 6-6. This is due to the fact that, at some point, the tracking algorithm no longer detects that the ball's motion is different from that of the train. This is mainly due to the fact that under an affine motion model, the rotating round surface of the ball cannot be well synthesised.
- Since the tracking and classification is highly reliant on motion criteria, it can happen that objects with a similar intensity to the background are not discriminated very well. The man's hat in Foreman is one example of this.
- Another problem evident in the Foreman sequence is that beginning from the imperfect initialisation, it takes some time for the whole moving semantic object to be identified.

It is expected that some of these problems can be eliminated by a better initialisation provided by a user controlled segmentation algorithm.

Figure 6-5: Hall Monitor tracking results: frames 52,130,220 (10Hz sequence was analysed). Frame 52 was the first frame analysed. Affine motion models were used. Top-row: the motion segmentations. Bottom row: the object segmentations.

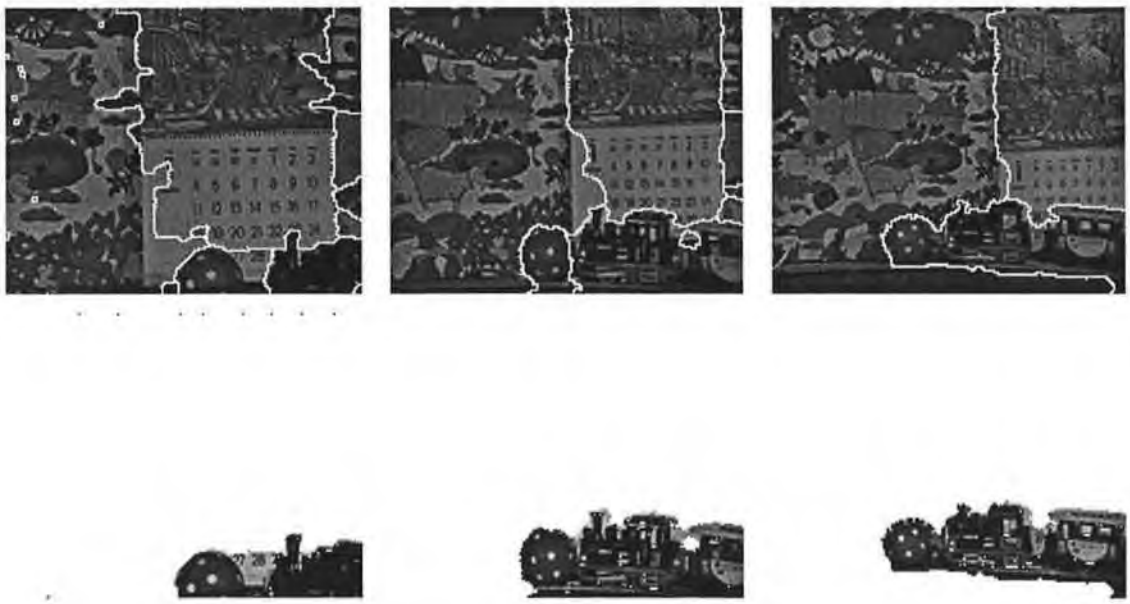


Figure 6-6: Calendar tracking results: frames 3, 90, 150 (10 Hz sequence was analysed) Frame 3 was the first frame analysed. Affine models were used. Top-row: the motion segmentations. Bottom row: the object segmentations.

Figure 6-7: Foreman result: frames 200, 225, 250. (25Hz sequence was analysed). Frame 200 was the first frame analysed. Quadratic models were used. Top-row: the motion segmentations. Bottom row: the object segmentations.

6.3.2 Tracking with Supervised Initialisation

For supervised initialisation, the initial segmentation of the first frame is provided by a supervised EM algorithm based on the work of O'Connor *et al* **[67]**. This is the only use of human intervention to aid the video segmentation algorithm. The segmentation of the first frame typically labels the object of interest with one value and the remainder of the image with another value. The remainder of the image is termed the background and the label value is termed the background label value. Provided with this segmentation, the overall function of the algorithm is to track the object of interest thoughout the subsequent frames of the sequence.

The first step in fulfilling this task is to estimate the motion within the object of interest and within the background region. There are a number of options in computing this motion. In the presented results, the initial motion over the whole object of interest is estimated as a single polynomial function, i.e. affine or quadratic. Alternatively, the

initial motion within the object of interest could be attained by generating a set of motion hypotheses and allowing them to compete. The former approach was chosen due to the fact that it is simpler and works well in cases where the initial motion of the object is rigid. The next step is to perform a modified detection step, whereby a new motion may be proposed only within the object of interest. That is, the detection step does not attempt to find new motions within the background area. The new motion is validated as usual by MDL-based criteria. At this point, the following information is available:

- the shape of the object of interest, i.e. the semantic segmentation for the first frame (as is provided by the methods presented in **[67]**),
- a motion segmentation of the object of interest (identifying at most two motion regions within the object), and
- the motion parameters associated with the motion segmentation.

These three pieces of information are used in the projection step to find the new shape of the object in the next frame. This is achieved by forward motion compensating the regions of the motion segmentation and running an EM algorithm to refine the so-given initial estimates of the projected segmentation and associated motion parameters. The union of all the projected sub-segmentation regions forms the new shape of the object as per Figure 6-1. Following this, the detection/validation step is again applied in order to detect and estimate any new motions. The implementation of the projection/detection/validation step is identical to that of the unsupervised algorithm and these steps are applied at every frame.

The only differences between the supervised and unsupervised algorithm are:

- The initial shape of the object of interest is given at the start of the algorithm.
- The detection step only sub-segments within the object of interest. In this way, the motion complexity of the object of interest is adapted, whereas the motion complexity of the background is not adapted. This detection approach is also such that pixels of background area are labelled with a single known value at the beginning and pixels labelled with any other value are deemed to be part of the object

of interest. Indeed, this allows a simple automatic process to be applied for making the union of regions and producing the final shape of the object of interest in each frame of the sequence.

Figure 6-8: Initial supervised segmentations for the two test sequences

Figure 6-9: Foreman tracking results: Frames shown are numbered 25, 50, 75, 100, 125, 150, 175, and 200.

Figure 6-10: Effect of occlusion on tracking : Frames shown are numbered 252, 254, 256 and 258.

The results of the tracking algorithm with the supervised initialisation are shown for the Foreman sequence and Mother and Daughter sequence. The initialised semantic objects are shown in Figure 6-8. A summary of the tracking results for Foreman over an 8 second duration is shown in Figure 6-9. The Foreman results are impressive when one considers that there are approximately five million pixels in this 8 second moving video sequence and that the tracking algorithm has rightfully classified the vast majority of them. However, the small percentage of mis-classified pixels are very noticeable in some cases. Some flaws are evident in the region of the man's hat. This is due to the fact that the pixel classification is based on luminance information only (i.e. the watershed is based on the luminance component; the suitability of a given motion model is quantified in terms of its ability to synthesise the luminance information). That is, no chominance information is used. Since the luminance content of the hat is very similar to the background, this makes discrimination very difficult. Figure 6-10 shows the case when the man's hand is swept across his face and away again. This kind of occlusion and unocclusion causes the tracking performance to rapidly deteriorate without the possibility of unassisted recovery.

Figure 6-11: Mother-Daughter (30 Hz) results: Frames shown are numbered 60, 120, 180, and 240.

The Mother-Daughter sequence possesses much simpler motion than the Foreman sequence and hence results would be expected to be very good. However, this expectation is not fulfilled. Firstly, when using the weakened contextual constraint of Equation 6-4, the segmentations become very noisy around the object edges and eventually, after several tracked frames the whole consistency of the object is eroded. The root of this problem is the almost total emphasis on motion criteria for allocating pixels. The motion of the semantic object is well estimated and it does the task of forward projection very well. However, when it comes to refining the shape of the projected object, it happens that the object motion can be equally applicable to the

background regions. That is, many background pixels get allocated to the foreground object due to the fact that the foreground motion is quite suitable for them also. This discrimination problem occurs because this sequence possesses a combination of untextured background and very small motion magnitudes in the foreground object. Hence, for many pixels, the motion-based discrimination criteria applied in the EM steps do not work well. This observation leads to the conclusion that the exact placement of the object contours at each tracking step should not be heavily reliant on motion. The projection step should rely on motion information to get a first estimate of the object shape, but in refining this first estimate texture-related criteria should be emphasised more.

Given that the weakened constraint did not operate well on this sequence, the full watershed constraint of Equation 6-3 is used for the presentation of results. The drawbacks with this constraint are evident from Figure 6-11. The problems are, as outlined previously, that the watershed is attracted to the dominant contours within the image. As the object moves, the tracked object contours may come close to a more dominant contour, i.e. a contour with a more contrasted intensity edge. At this point, the watershed has the effect of locking the tracked object contour onto the dominant contour and due to the temporal constraint inherent in the tracking algorithm, this dominant contour remains for the duration of the tracking process. This is the reason why the tracked object shape appears to grow in size over time.

6.4 Summary

A novel approach to semantic object tracking is presented in this chapter. This approach relies on piecewise motion models for an accurate estimate of the object's motion. The object's motion is used to forward project the object's shape and a new spatially and temporally constrained EM algorithm is formulated to refine the projected shape and the motion estimates. The reliance on accurate motion representation calls for the use of a hypothesis generation/validation process for adapting the motion model complexity to the varying degrees of motion being exhibited by the actual object.

The refinement of the object's shape is an aspect to which much attention has been paid. Various new constraints based on the watershed have been experimented with. The particular emphasis is on obtaining very accurate contour placement by aligning them to texture edges. The results of this approach show an improvement over the traditional MRF-based approaches for basic motion segmentation.

More generally, the results of the proposed tracking system are promising. It is believed that the overall approach is sound, but that much improvement is possible in the area of actually making the final decisions on where to place the contours of the tracked object. The contextual constraints presented here for this purpose are far from perfect and could undoubtedly be improved upon. Alternative approaches to contour localisation might also prove useful.

As is usual, complexity is always an important issue when processing video data. The complexity of the approach may be attributed to:

- the iterative nature of the EM algorithm,
- the motion estimation by the Gauss-Newton method (also an iterative estimator), and
- the brute-force MDL-based hypothesis validation procedures.

For a more practical deployment of the proposed algorithm, it would be advisable to look at methods which can maintain the necessary performance, but which might be less complex. The software implementing the algorithm is totally unoptimised but it should be said that it can require 10-20 hours run to process 10 seconds of video at full frame-rate on a Sparc-20 workstation.

In general terms, it can be concluded that the segmentation of semantic video objects might be efficiently performed by supervised approaches. One simple approach has been illustrated whereby a good initial supervised segmentation of the object is provided in the first frame and tracked throughout the remainder of the sequence. From results, it is clear that further supervision will need to be employed at certain locations in the sequence. This is due to inadequacies in the automatic tracking algorithm and due to

events which cannot be well coped with, e.g. occlusion/un-occlusion. When such flaws appear in the tracked segmentation, the user will be required to correct it. Supervised video segmentation is still very much a problem area and much work will be required to make it practical over a wide range of video source material. Future research should concentrate on the following aspects:

- supervised segmentation of semantic objects from single images as in **[67]**,
- improved tracking algorithms as presented here, with particular emphasis on suitable methods for maintaing accurate contour localisation throughout time, and
- methods and image representations to ease the correction of segmentations at any point in a sequence.

The overall goal of this research should be to minimise the intervention of the user so that accurate and useful semantic objects can be obtained with a minimum of effort.

7. CONCLUSIONS

Object-based video compression was proposed in the mid-eighties and aimed at providing higher quality compressed video. For a variety of reasons, this aim has not been achieved. Early efforts such as SIMOC suffered because of the immaturity of the topic. MPEG-4 has since adopted object-based methods and developed very efficient compression techniques. MPEG-4 is selling object-based technology to industry based on a belief that many new and useful functionalities can be provided. In particular, it eases the process of content provision and allows much needed interactivity in multimedia programmes. Several new technical challenges are presented by this focus on object-based technology. Two problems in particular are addressed this thesis, i.e. shape compression and video segmentation. In each area, this thesis has reviewed previous solutions and then presented new solutions which exhibit varying degrees of success.

The design of compression methods for object shape has been tackled very successfully. An algorithm utilising context-based arithmetic encoding has been developed by the author. The new aspect of this algorithm lies in the fact that a block-based paradigm is chosen and in the fact that context-based arithmetic encoding is used for inter-frame coding. The algorithm is capable of providing very efficient lossless and lossy codes for object shape, while also meeting ancillary requirements such as feasible implementation complexity. Indeed, the algorithm performed so well in competitive tests, that the MPEG-4 working group has adopted it as part of the new MPEG-4 international standard. In addition, the author has filed a patent application based on elements of this block-based CAE. Finally, it should be noted that some minor adaptations to the main algorithm are necessary to provide error resilient representations and compression of interlaced alpha channels. The author describes these adaptations in **[16]**.

The second problem addressed is video segmentation, i.e. the acquisition of the object shape (alpha maps) from the video sequence in which the object resides. The approach taken in this thesis is highly reliant on motion estimation and segmentation. In the area

of motion estimation, polynomial motion models have been used since these are effective in synthesising the effects of real 3-D object motion. Several well-known Newton-based methods, notably Quasi-Newton and Gauss-Newton methods, have been compared for estimating these models. This work has added to the volume of knowledge on the estimation of polynomial motion models and resulted in showing that, as in previous literature, the Gauss-Newton method is a superior estimator. The thesis has highlighted that the complexity of Newton-based estimators is sufficiently high that their use in real-time systems is difficult. To address this problem, a fast Gauss-Newton algorithm has been developed. This fast algorithm benefits by minimising the size of the observation set, i.e. the number of image pixels which are considered by the estimator, and by using look up tables for the implementation of the image interpolation filters. Computational analysis reveals that the fast algorithm can work 2-3 times faster while maintaining excellent estimation performance.

Chapter 5 discusses several automatic segmentation approaches. In particular, a motion segmentation approach was implemented using the Expectation-Maximisation (EM) algorithm and the Minimum Description Length (MDL) principle. The method treats segmentation as a complex model fitting optimisation problem. The optimisation criterion seeks to minimise the inter-frame coding cost of the video frame as represented by the segmentation and the motion parameters associated with each segment. To facilitate this goal, some effective coding model for the shape information is required in both the E-step (responsible for classifying pixels) and in the coding cost itself (used in the MDL-based hypothesis validation). The author has chosen to use local Markov random field (MRF) models for this purpose because these are quite similar in principle to the coding models used in the CAE shape compression method. Specifically, the description length (used for MDL hypothesis testing) uses an MRF model to estimate the coding length for representing the segmentation map and a similar MRF model has been incorporated into the E-step in order to produce spatially coherent segmentation maps. The results of this approach are judged (subjectively) to be at least as good as those presented by Ayer and Sawhney and better than those presented by Wang and Adelson. However, with respect to the goal of aiding in the task of high quality semantic segmentation, some deficiencies are very evident. While the general shape of the

moving objects can be recovered and the segmentations are very clean and coherent, the location of the contours is not sufficiently accurate to be used in real applications such as video editing and so on. The overall concept of minimising coding cost is intuitively attractive. However, the results suggest that a segmentation minimising inter-frame coding cost does not necessarily correspond with a segmentation which accurately captures the contours of the semantic objects. Based on this observation, the author has developed new methods for improving upon the placement of the segmentation contours, as summarised in the following paragraph.

As discussed in chapter 6, the observation that motion boundaries generally coincide with texture edges has led the author to develop a new and powerful contextual E-step based on the morphological watershed. When computing the ownership probabilities for a given pixel, the probabilities of other pixels in the same watershed region are considered. In effect, the watershed constraint results in an E-step which classifies watershed regions rather than pixels. When, as is very often the case, the watershed accurately captures the contours of the moving semantic object, then the resulting segmentations show significant improvement over the MRF-based E-step. However, when the watershed occasionally overlooks an important contour, then this contour will not appear in the final segmentation.

Since the overall aim of the segmentation work presented in this thesis relates to the segmentation of video sequences and not merely to the segmentation of single images, the development of tracking mechanisms is very apt. The EM-MDL scheme (augmented with the watershed-based E-step and improved description length) has been encorporated for the first time within a framework designed to track moving objects throughout a sequence The algorithm relies on the maintenance of a piecewise polynomial motion model for the moving object. This model is used to project the segmentation from frame to frame and this provides a good initial guess of the location and shape of the object in the future frame. Given the initial guess, contextual expectation-maximisation is used to refine the estimated shape. For this refinement step, the author has devised novel methods which allow the projected segmentation of the object to have influence within the segmentation process for the current frame. It

involves deriving conditional probabilities for the current segmentation based on the previous segmentation and then integrating these probabilities into the E-step which refines the initial guess given by the motion projected object shape. Thus, the E-step of the tracking algorithm combines these temporally-derived probabilities with the aforementioned spatial watershed-based constraint. The segmentations produced in this way are therefore clean and coherent in both space and time. Also significant is the fact the framework uses a piecewise motion model which is adapted at each frame to the complexity of the underlying object motion. This is achieved by mechanism for generating new hypotheses and an MDL-based hypothesis tester which can eliminate both old and new hypotheses. By this mechanism, the number of motion models and corresponding motion regions used to project the object shape varies in accordance with the varying nature of the object motion.

The performance of the proposed tracking algorithm is good in some cases and bad in others. The EM-MDL tools can produce motion representations which are consistent with the real object motion and these motion representations can be relied upon to accurately project the object shape from frame to frame. However, the criteria used in classifying pixels are not robust. These criteria suffer from the common difficulty in accurately placing a contour, especially when the associated intensity edge is afflicted by either of two extreme characteristics, i.e. when the edge has little or no intensity gradient or when the edge is located in a 'busy' part of scene amidst other possibly more dominant and sharper edges. In either case, the watershed can fail to help choose a suitable contour location. In addition to this common problem, the pixel classification criterion is founded on motion model suitability. That is, in the absence of any other constraint, a pixel is classified to the model which most accurately represents the pixel. In cases where the object motion is slight, this can cause some unexpected problems as has been demonstrated by applying the method to the seemingly simple Mother-Daughter sequence.

The results of the tracking algorithm have been obtained using a supervised segmentation of the first frame of the sequence and despite the aforementioned shortcomings, it must still be observed that, in each image, a significant proportion of

the object's contour is captured with an accuracy which could suffice in high quality applications. It is only is cases of highly complex motion and/or occlusion that the algorithm fails drastically. Hence, it is believed that some or all of the innovations presented in this thesis can prove useful in future supervised video segmentation systems. Due to the accurate tracking and the use of the texture oriented watershed constaint, it appears that the algorithm can relieve the user of a significant work-load.

An increasing number of researchers, including the author, firmly believe that the shape of semantic video objects in 2-D images can *only* be accurately recovered by a supervised approach. The supervised approach allows the human user to complete the tasks that a computer is incapable of. Future segmentation work should initially concentrate on building an integrated application around the presented image analysis engine or some similarly functioning engine. The major components of such a supervised segmentation tool can be envisaged. The tool should comprise an advanced graphical user interface for capturing the wishes and knowledge of the human user, and this information should be fed to a powerful automatic image analysis engine to perform the necessary remaining tasks of placing the exact object contours. The human interaction has the function of performing the semantic interpretation, i.e. selecting the object to be segmented, correcting the segmentation results: all tasks which the computer is incapable of. The graphical interface is the link between what the user intends, i.e. the segmentation of a particular semantic object, and the language that the image analysis engine understands, i.e. texture and motion primitives. It is believed that, currently, the graphical interface is a much under-estimated component of the system which will have a significant result on the eventual system performance and ease of use. This belief is motivated by the prediction (based on experience) that no matter how powerful the image analysis engine is, the time consumed in any given segmentation task will be dominated by the interaction of the user. The tools provided for this interaction should be thought over very carefully, as should their link with the underlying image primitives. The final part of the system, the image analysis engine, comprises all the automatic algorithmic elements for placing contours, classifying pixels and tracking moving objects.

Examples of similar commercial image editing systems, e.g. Corel, Adobe Photoshop and Automedia's Automasker, should be studied to glean valuable experience in the way graphical user interfaces can be designed effectively. When a first system prototype exists, it will provide a good platform for effective evaluation of new or improved versions of the image analysis engine. Ultimately, this approach, based on evaluation of the integrated system, may lead to high quality supervised segmentation systems and hopefully, this will arrive in time to enable the new multimedia applications which MPEG-4 promises.

8. REFERENCES

[1] T. Aach, A. Kaup and R. Mester, "Statistical Model-based Change Detection in Moving Video", *Signal Processing*, Vol. 31, No. 2, pp. 165-180, March 1993.

[2] J. K. Aggarwal and N. Nandhakumar, "On the Computation of Motion from Sequences of Images - A Review", *Proceedings of IEEE*, vol. 76, no. 8, August 1988.

[3] S. Ayer and H. Sawhney, "Layered Representation of Motion Video using Robust Maximum-likelihood Estimation of Mixture Models and MDL Encoding", *Proceeding of 5th International Computer Computer Vision*, Cambridge, MA. June 20-23, 1995.

[4] S. Ayer and H. Sawhney, "Compact Representations of Videos through Dominant and Multiple Motion Estimation", *IEEE Transactions on Pattern Analysis and Machine Intelligence, Special issue on Digital Libraries: Representation and Retrieval*, vol. 18, no. 8, pp. 814-830, August 1996.

[5] J. Barron, D. Fleet and S. Beauchemin, "Systems and Experiment: Performance of Optical Flow Techniques", *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43-77, 1994.

[6] J. Bergen et al., "Hierarchical Model-based Motion Estimation", in *Proceedings of 2nd European Conference on Computer Vision*, pp. 237-252, 1992.

[7] C. Bergeron and E. Dubois, "Gradient-based Algorithms for Block-Oriented MAP Estimation of Motion and Application to Motion-compensated Temporal Interpolation", *IEEE Trans. on Circuits and Systems for Video Technology*, vol 1., no 1., pp 72-85, March 1991.

[8] J. Besag, "On the Statistical Analysis of Dirty Pictures (with discussion)", *Journal of Royal Statistical Society*, B 48, pp 259-302, 1986.

[9] M. Bierling, "Displacement Estimation by Hierarchical Block Matching", *3rd SPIE Symposium on Visual Communications and Image Processing*, Cambridge, USA, pp. 942-951, November 1988.

[10] F. Bossen and T. Ebrahimi, "A Simple and Efficient Binary Shape Coding Technique based on Bitmap Representation", *Proceedings of IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, Munich, 1997.

[11] P. Bouthemy and E. Francois, "Motion Segmentation and Qualitative Dynamic Scene Analysis from an Image Sequences", *International Journal of Computer Vision*, vol. 10, pp. 157-182, April 1993.

[12] N. Brady, "Moving Area Segmentation within SIMOC", *COST 211ter Simulation Subgroup*, SIM 94(7), Paris, February 10-11, 1994

[13] N. Brady, N. Murphy and T. Curran, "Computationally Efficient Estimation of Polynomial Model-based Motion", *Proceedings of Picture Coding Symposium*, Melbourne, March 1996.

[14] N. Brady and N. O'Connor, "Object Detection and Tracking using an EM-based Motion Estimation and Segmentation Framework", *Proceedings of IEEE International Conference on Image Processing*, Lausanne, September 1996.

[15] N. Brady, F. Bossen and N. Murphy, "Context-Based Arithmetic Encoding of 2D Shape Sequences", ", *Proceedings of IEEE International Conference on Image Processing*, Santa Barbara, October 1997.

[16] N. Brady and F. Bossen, "Shape Compression of Moving Objects using Context-based Arithmetic Encoding", Submitted to *Signal Processing: Image Communication: Special Issue on Shape Coding*

[17] K. Brodlie, "Unconstrained Minimisation", Chapter III.1 in *"The State of the Art in Numerical Analysis" edited by D. Jacobs*, Academic Press, London, 1977.

[18] S. Buecher, "Segmentation Tools in Mathematical Morphology", *in HandBook of Pattern Recognition and Computer Vision, Ed. C. H. Chen, L. F. Pau and P. S. Wang*, World Scientific, pp 443-456, 1993.

[19] R. Chellappa,. L. Kashyap and B. Manjunath, "Model-based Texture Segmentation and Classification", in *Handbook of Pattern Recognition and Computer Vision, Eds. C. H. Chen, L. F. Pau and P. S. Wang*, pp. 277-310, World Scientific, 1993.

[20] M. Chowdury et al., "Architecture and Performance of a Switched Model/Block-based Image Coder", *Cost211ter European Workshop on New Techniques for Coding of Video Signals at Very Low Bitrates*, Hannover, December 1993.

[21] M. Chowdury et al., "A Switched Model-based Codec for Video Signals", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 3, pp. 216-217, June 1994.

[22] I. Corset et al, "Technical Description of SESAME", *ISO/IEC JTC1/SC29/WG11*, MPEG 95/408, November 1995.

[23] COST 211ter Simulation Subgroup, "SIMOC1: Simulation Model for Object-based Coding", SIM(95) 08, February, 1995.

[24] T. Darrell and A. Pentland, "Cooperative Robust Estimation using Layers of Support", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 17, no. 5, pp. 474-487, May 1995.

[25] A. Dempster, N. Laird and D. Rubin, "Maximum Likelihood from Incomplete Data via the EM Algorithm", *Journal of Royal Society of Statistics*, Series B, no. 1, pp 1-38, 1977.

[26] J. Dennis and R. Schnabel, *Numerical Methods for Unconstrained Optimisation and Non-linear Equations*, Prentice-Hall, New Jersey, 1983.

[27] N. Diehl, "Object-oriented motion estimation and segmentation in image sequences", *Signal Processing: Image Communication*, vol. 3, no. 1, pp 23-56, 1991.

[28] J. Dugelay and H. Sanson, "Differential Methods for the Identification of 2-D and 3-D Motion Models in Image Sequences", *Signal Processing: Image Communication*, vol. 7, vo. 1, pp. 105-127, March 1995.

[29] R. Fletcher, *Practical Methods in Optimisation*, 2nd Edition, John Wiley and Sons, 1987.

[30] P. Gerken, "Object-based Analysis-Synthesis Coding of Image sequences at Very Low Bit rates", *IEEE Circuits and Systems for Video Technology: Special Issue on Very Low Bit Rate Coding*, vol. 4, no. 3, pp. 228-235, June 1994.

[31] R. Gonzales and P. Wintz, *Digital Image Processing*, 2nd Edition, Reading, MA., Addison-Wesley, 1987.

[32] B. G. Haskell, A. Puri and A. N. Netravali, *Digital Video: An Introduction to MPEG-2*, ISBN:0-412-08411-2, Chapman & Hall, 1997.

[33] G. Held and T. Marshall, *Data Compression: Techniques and Applications, Hardware and Software Considerations*, 2nd edition, John Wiley & Sons, New York, NY, 1987

[34] B. Horn, *Robot Vision*, MIT Press, Cambridge, MA, 1986.

[35] M. Hötter and R. Thoma, "Image Segmentation Based on Object Oriented Mapping Parameter Estimation", *Signal Processing*, vol 15, pp. 315-334, 1988.

[36] M. Hötter, "Object-Oriented Analysis-Synthesis Coding Based on Moving 2-D Objects", *Signal Processing: Image Communication*, Vol. 2, No. 4, December

1990.

[37] ITU-T Recommendation H.261, "Video Codec for Audio Visual Service at Px64 Kbits/s"

[38] ITU-T Recommendation H.263, "Video Coding for Narrow Telecommunication Channels at < 64 Kbits/s"

[39] ITU-T Recommendation T.4, "Standardization of Group 3 Facsimile Apparatus for Document Transmission"

[40] ITU.T Recommendation T.6, "Facsimile Coding Schemes and Coding Control Functions for Group 4 Facsimile Apparatus"

[41] ITU-T Recommendation T.82, "Information Technology - Coded Representation of Picture and Audio Information - Progressive Bi-Level Image Compression"

[42] T. Kaneko and M. Okudaira, "Encoding of Arbitrary Curves based on the Chain Code Representation", *IEEE Transactions on Communications*, vol. 33, no. 7, July 1985.

[43] M. Karczewicz, J. Nieweglowski and P. Haavisto, "Video Coding using Motion Compensation with Polynomial Motion Vector Fields", *Signal Processing: Image Communication*, vol 10, Nos. 1-3, pp. 63-93, July 1997.

[44] G. Langdon and J. Rissanen, "Compression of Black-White Images with Arithmetic Coding", *IEEE Transactions in Communications*, COM-6, pp. 158-167, 1981.

[45] F. Marques, B. Marcotegui and F. Meyer, "Tracking Areas of Interest for Content-based Functionalities in Segmentation-based Video Coding", *International Conference on Acoustics, Speech and Signal Processing (ICASSP'96)*, vol. II, pp 1224-1227, Atlanta, USA, May 1996.

[46] F. Marques and C. Molina, "Object Tracking for Content-based

Functionalities", *Proc. SPIE Visual Communication and Signal Processing-94 Conference*, pp. 190-198, Feb. 1997.

[47] F. McConnell, D. Bodson and R. Schaphorst, *FAX: Digital Facsimile Technology and Applications, 2nd Edition*, Artech House, Boston, London 1992.

[48] G. McLachlan and K. Basford, *Mixture Models Inference and Applications to Clustering*, Marcel Dekker, New York and Basel, 1988.

[49] G. McLachlan, *Discriminant Analysis and Statistical Pattern Recognition*, John Wiley and Sons, New York, 1992.

[50] D. Le Gall., "The MPEG Video Compression Algorithm", *Signal Processing: Image Communication*, vol. 4, No. 2, pp. 129-140, 1992.

[51] D. Mitchell and A. Netravali, "Reconstruction Filters in Computer Graphics", *Computer Graphics*, vol. 22, no. 4, pp 221-228, August 1988.

[52] A. Moffat, "Two-level Context-based Compression of Binary Images", *Proc. IEEE Data Compression Conference*, pp. 382-391, 1991.

[53] A. Moffat, R. Neal, I. Witten, "Arithmetic Coding Revisited", *Proc. IEEE Data Compression Conference*, pp. 202-211, 1995.

[54] ISO/IEC 11172-2, *Information Technology - Coding of Moving Pictures and Associated Audio for Digital Storage Media at up to about 1,5 Mbit/s - Video*, Geneva, 1993

[55] ISO/IEC 13818-2, *Information Technology - Generic Coding of Moving Pictures and Associated Audio Information - Video*, March, 1995.

[56] MPEG-4 PPD

[57] MPEG-4 Video Group, Video Verification Model Version 3, MPEG meeting, Tampere, July 1996.

[58] MPEG-4 Video Group, Video Verification Model Version 4, MPEG meeting, Chicago, September 1996.

[59] MPEG-4 Video Group, Video Verification Model Version 5, MPEG meeting, Brazil, November 1996.

[60] MPEG-4 Video Group, Video Verification Model Version 6, MPEG meeting, Seville, February 1997.

[61] MPEG-4 Video Group, Video Verification Model Version 7, MPEG meeting, Bristol, April 1997.

[62] MPEG-4 Video Group, Core Experiment Descriptions Document, MPEG Meeting, Bristol, April 1997.

[63] H. Musmann, M. Hotter and J. Ostermann, "Object-oriented Analysis-synthesis of Moving Images", *Signal Processing: Image Communication*, vol. 1, no. 2, pp. 117-138, 1989.

[64] H. Musmann, "A layered coding system for very low bit-rate video coding", *Signal Processing: Image Communication*, vol. 7, no. 4-6, pp. 267-278, 1995.

[65] Y. Nakaya and H. Hasashima, "Motion Compensation based on Spatial Transformations", *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 4, no. 3, June 1994.

[66] A. Netravali and J. Salz, "Algorithms for Estimation of Three-Dimensional Motion", *AT&T Technical Journal*, vol. 64, no. 2, February 1985.

[67] N. O' Connor, N. Brady and S. Marlow, "Supervised Image Segmentation Using EM-Based Estimation of Mixture Density Parameters", *Proceedings of Cost 211ter Workshop on Image Analysis for Multimedia Interactive Services*, Louvain-la-Neuve, Belgium, pp. 27-32, June 24-25, 1997.

[68] M. Orchard, "Predictive Motion-Field Segmentation for Image Sequence Coding", *IEEE Transactions on Circuit and Sysyems for Video Technology*,

vol. 3, no. 1, February 1993.

[69] A. Owen, "Contribution to the Discussion of Paper by B. D. Ripley", *Canadian Journal of Statistics*, vol. 14, pp 106-110, 1986.

[70] G. Papadimitriou and T. Dennis, "3-D Parameter Estimation for Stereo Image Sequences for Model-based Image Coding", *Signal Processing: Image Communication*, vol. 7, vo. 4-6, pp. 471-488, November 1995.

[71] T. Pavlidis, *Algorithms for graphics and image processing*, Computer Science Press, 1982.

[72] D. Pearson, "Developments in Model-based Video Coding", Proceedings of IEEE, vol. 83, no. 6, pp. 892-906, 1995.

[73] D. Pearson, "Model-based Coding: Past and Future", *Proceedings of Picture Coding Symposium*, Melbourne, March 1996.

[74] F. Pereira, "MPEG-4: A new challenge for the representation of Audio-Visual Information", *Proceedings of Picture Coding Symposium* , Melbourne, March 1996.

[75] H. Press et. al, *Numerical Recipes in C: The Art of Scientific Computing*, Cambridge University Press, 1988.

[76] R. Redner and H. Walker, "Mixture Densities, Maximum Likelihood and the EM Algorithm", *SIAM Review*, 26, pp. 195-239, 1984

[77] J. Rissanen, "Modelling by Shortest Data Description", *Automatica*, vol. 14, pp. 465-471, 1978.

[78] J. Rissanen, "A Universal Prior for Integers and Estimation by Minimum Description Length", *The Annals of Statistics*, 11(2), 416-431, 1983.

[79] P. Salembier P and J. Serra, "Mathematical Morphology and Its Applications to Image Coding", *Special Course for the RACE-MAVT and COST 211ter*

partners, March 1994, Daimler Benz, Ulm, Germany.

[80] P. Salembier, P. Briggar, J. Casas and M. Pardas, "Morphological Operators for Image and Video Compression", *IEEE Transactions on Image Processing*, vol. 5, no. 6, pp. 881-897, June 1996.

[81] H. Sanson, "Motion Affine Models Identification and Application to Television Image Coding", *Proc. SPIE Visual Communication and Image Processing '91*, Boston, MA, vol. 1605, Part 2, pp. 570-581, 11-13 November 1991.

[82] H. Sanson, "Region-based Motion Estimation and Compensation for Digital TV Sequence Coding", *Picture Coding Symposium*, Lausanne, Switzerland, March 17-19 1993.

[83] H. Sanson, "Joint Estimation and Segmentation of Motion for Video Coding at Low Bit-rates", *COST 211ter Workshop*, Hannover, December 1993.

[84] H. Sawhney, S. Ayer and M. Gorkani, "Model-based 2D and 3D Dominant Motion Estimation for Mosaicing and Video Representation", *Proceedings of International Conference on Computer Vision '95*, Cambridge, MA, 1995.

[85] G. Seber and C. Wild, *Nonlinear Regression*, John Wiley and Sons, New York 1989.

[86] J. Serra, *Image Analysis and Mathematical Morphology, Vol.2: Theoretical Advances*, Academic Press 1988.

[87] M. Menezes de Sequeira and D. Cortez, "Partitions: A taxonomy of types and representations and an overview of coding techniques", *Signal Processing: Image Communication*, vol. 10, nos. 1-3, pp. 5-20, July 1997.

[88] T. Sikora, "Low Complexity Shape Adaptive DCT for Coding of Arbitrarily Shaped Image Segments", *Signal Processing: Image Communications*, vol. 7, no. 4-6, pp. 381-395, November 1995.

[89] C. Therrien , *Discrete Random Signals and Statistical Signal Processing*, Prentice Hall, New Jersey, 1992.

[90] M. Tuceryan and A. Jain, "Texture Analysis", in *Handbook of Pattern Recognition and Computer Vision, Eds. C. H. Chen, L. F. Pau and P. S. Wang*, pp. 235-276, World Scientific 1993.

[91] L. Vincent, "Morphological Grayscale Reconstruction in Image Analysis: Applications and Efficient algorithms", *IEEE Transactions on Image Processing*, vol. 2, no. 2, pp. 176-201, April 1993.

[92] L. Vincent. and P. Soille, "Watersheds in Digital Spaces: An Efficient Algorithm based on Immersion Simulations", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 6., June 1991.

[93] J. Wang and E. Adelson, "Layered Representations for Image Sequence Coding", *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, 1993.

[94] Y. Weiss and E. Adelson, "Perceptually Organised EM: A Framework for Motion Segmentation that Combines Information about Form and Motion", Technical Report No. 315, MIT Media Laboratory, 1995.

[95] I. Witten, R. Neal and J. Cleary, "Arithmetic Coding for Data Compression", *Communications of the ACM*, vol. 30, no. 6, pp. 520-540, June 1987.

[96] T. Wuyts et al, "Some Critical Remarks on 2-D Object-based Coding", *Cost211ter European Workshop on New Techniques for Coding of Video Signals at Very Low Bitrates*, Hannover, December 1993.

[97] J. Zhang, W. Modestino, and D. Langan. "Maximum-likelihood Parameter Estimation for Unsupervised Model-based Image Segmentation", *IEEE Transactions on Image Processing*, 3(4), 1994.

[98] H. Zheng and S. Blostein, "Motion-based Object Segmentation and Estimation using the MDL Principle", *IEEE Transactions on Image Processing*, vol. 4,

No. 9, pp. 1223-1235, September 1995.

Appendix A

A1. Quadratic Motion Model: An Example

The quadratic motion model is expanded below but first some generalised definitions.

$$\mathbf{x}' = \begin{pmatrix} x' & y' \end{pmatrix}^T ; \mathbf{x} = \begin{pmatrix} x & y \end{pmatrix}^T$$

For a model of order n,

$$\mathbf{x}' = \mathbf{x} + \begin{pmatrix} \sum_{i=0}^{n} \sum_{j=0}^{i} a_{ij} x^{i-j} y^j \\ \sum_{i=0}^{n} \sum_{j=0}^{i} b_{ij} x^{i-j} y^j \end{pmatrix} = \mathbf{x} + \mathbf{B}\theta,$$

$$\text{where } \mathbf{B} = \left(\frac{\partial \mathbf{x}'}{\partial \theta} \right)^T = \begin{pmatrix} \frac{\partial x'}{\partial a_{00}} & \frac{\partial y'}{\partial b_{00}} \\ \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \\ \frac{\partial x'}{\partial a_{nn}} & \frac{\partial y'}{\partial b_{nn}} \end{pmatrix}^T \text{ and}$$

$$\theta = \begin{pmatrix} \theta_0 & \theta_1 & .... & ... & \theta_{m-1} \end{pmatrix}^T = \begin{pmatrix} a_{00} & ... & a_{nn} & b_{00} & ... & b_{nn} \end{pmatrix};$$

and $m = (n+2)(n+1)$;

For a quadratic model (n=2) we get,

$$x' = x + \left(a_{00} + a_{10}x + a_{11}y + a_{20}x^2 + a_{21}xy + a_{22}y^2 \right)$$

$$y' = y + \left(b_{00} + b_{10}x + b_{11}y + b_{20}x^2 + b_{21}xy + b_{22}y^2 \right)$$

$$\theta = \begin{pmatrix} a_{00} & a_{10} & a_{11} & a_{20} & a_{21} & a_{22} & b_{00} & b_{10} & b_{11} & b_{20} & b_{21} & b_{22} \end{pmatrix}^T$$

$$\mathbf{B} = \begin{pmatrix} 1 & x & y & x^2 & xy & y^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & x & y & x^2 & xy & y^2 \end{pmatrix}$$

A2. A QN Estimation Example

For QN, it is necessary only to calculate the error gradient vector at each iteration according to:

$$\mathbf{g}(\theta_k) = \mathrm{E}\left\{\left(I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')\right)\frac{\partial \mathbf{x}'}{\partial \theta}\left[\frac{\partial \mathbf{x}'}{\partial \mathbf{x}}\right]^{-1}\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}\right\}\Bigg|_{\theta=\theta_k}$$

For a quadratic model,

$$\frac{\partial \mathbf{x}'}{\partial \theta} = \mathbf{B}^T = \begin{pmatrix} 1 & x & y & x^2 & xy & y^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & x & y & x^2 & xy & y^2 \end{pmatrix}^T$$

$$\frac{\partial \mathbf{x}'}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial x'}{\partial x} & \frac{\partial y'}{\partial x} \\ \frac{\partial x'}{\partial y} & \frac{\partial y'}{\partial y} \end{pmatrix} = \begin{pmatrix} 1 + a_{10} + 2a_{20}x + a_{21}y & b_{10} + 2b_{20}x + b_{21}y \\ a_{11} + a_{21}x + 2a_{22}y & 1 + b_{11} + b_{21}x + 2b_{22}y \end{pmatrix}$$

$$\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}} = \begin{pmatrix} \frac{\partial I_t(\mathbf{x})}{\partial x} \\ \frac{\partial I_t(\mathbf{x})}{\partial y} \end{pmatrix}\Bigg|_{\mathbf{x}=\mathbf{x}'}$$

A3. A GN Estimation Example

For GN, the Hessian and gradient are evaluated according to:

$$\mathbf{g}(\theta_k) = \mathrm{E}\left\{\left(I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')\right)\frac{\partial \mathbf{x}'}{\partial \mathbf{T}}\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}\right\}\Bigg|_{\theta=\theta_k}$$

and

$$\mathbf{H}(\theta_k) = \mathrm{E}\left\{\left(\frac{\partial \mathbf{x}'}{\partial \theta}\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}\right)\left(\frac{\partial \mathbf{x}'}{\partial \theta}\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}\right)^T\right\}\Bigg|_{\theta=\theta_k}$$

at each iteration.

For a quadratic model,

$$\frac{\partial \mathbf{x}'}{\partial \theta} = \mathbf{B}^T = \begin{pmatrix} 1 & x & y & x^2 & xy & y^2 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & x & y & x^2 & xy & y^2 \end{pmatrix}^T$$

$$\frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}} = \left(\frac{\frac{\partial I_t(\mathbf{x})}{\partial x}}{\frac{\partial I_t(\mathbf{x})}{\partial y}} \right)\Bigg|_{\mathbf{x}=\mathbf{x}'}$$

A4. Gradient derivation for QN

The error function is given by:

$$e(\theta) = \tfrac{1}{2} \mathrm{E}\left\{ (I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}'))^2 \right\}$$

The gradient of the error function with respect to θ is:

$$\frac{\partial e(\theta)}{\partial \theta} = \mathrm{E}\left\{ (I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')) \frac{\partial I_t(\mathbf{x}')}{\partial \theta} \right\}$$

Equation A-1

By the chain rule,

$$\frac{\partial I_t(\mathbf{x}')}{\partial \theta} = \frac{\partial \mathbf{x}'}{\partial \theta} \frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}'} = \frac{\partial \mathbf{x}'}{\partial \theta} \left[\frac{\partial \mathbf{x}'}{\partial \mathbf{x}} \right]^{-1} \frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}}$$

and substituting this into Equation A-1 yields

$$\frac{\partial e(\theta)}{\partial \theta} = \mathrm{E}\left\{ (I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}')) \frac{\partial \mathbf{x}'}{\partial \theta} \left[\frac{\partial \mathbf{x}'}{\partial \mathbf{x}} \right]^{-1} \frac{\partial I_t(\mathbf{x}')}{\partial \mathbf{x}} \right\}$$

A5. Gradient and Hessian derivation for GN

The error function is given by:

$$e(\theta) = \tfrac{1}{2} \mathrm{E}\left\{ (I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}'))^2 \right\}$$

Equation A-2

At iteration k+1, we can approximate $I_t(\mathbf{x}'_{k+1})$ with a first order Taylor series as discussed in Chapter 4.

$$\begin{aligned} I_t(\mathbf{x}'_{k+1}) &\approx I_t(\mathbf{x}'_k) + \left(\frac{\partial I_t(\mathbf{x}'_k)}{\partial \mathbf{x}} \right)^T (\Delta \mathbf{x}_k) \\ &= I_t(\mathbf{x}'_k) + \left(\frac{\partial I_t(\mathbf{x}'_k)}{\partial \mathbf{x}} \right)^T (\mathbf{B} \Delta \theta_k) \end{aligned}$$

and then substituting this into Equation A-2, we get an approximated error function which is quadratic in the the variable $\Delta\theta_k$.

$$e(\Delta\theta_k) = \tfrac{1}{2} \mathrm{E}\left\{ \left(I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}'_k) - \left[\frac{\partial I_t(\mathbf{x}'_k)}{\partial \mathbf{x}} \right]^T \mathbf{B}.\Delta\theta_k \right)^2 \right\}$$

For convenience, we will write $\nabla \mathbf{I} = \frac{\partial I_t(\mathbf{x}'_k)}{\partial \mathbf{x}}$ and $\varepsilon = I_{t+1}(\mathbf{x}) - I_t(\mathbf{x}'_k)$. The error function now is written as,

$$e(\Delta\theta_k) = \tfrac{1}{2}\mathrm{E}\left\{\left(\varepsilon - \nabla\mathbf{I}^T\mathbf{B}.\Delta\theta_k\right)^2\right\}$$

Equation A-3

To find the minimum, we differentiate Equation A-3 with respect to $\Delta\theta_k$ and set the result to zero.

$$\frac{\partial e(\Delta\theta_k)}{\partial\Delta\theta_k} = \mathrm{E}\left\{\mathbf{B}^T\nabla\mathbf{I}\left(\varepsilon - \nabla\mathbf{I}^T\mathbf{B}.\Delta\theta_k\right)\right\} = \mathrm{E}\left\{\mathbf{B}^T\nabla\mathbf{I}\varepsilon - \mathbf{B}^T\nabla\mathbf{I}\nabla\mathbf{I}^T\mathbf{B}.\Delta\theta_k\right\} = 0$$

Solving for $\Delta\theta_k$, we get

$$\Delta\theta_k = \mathrm{E}\left\{\left(\mathbf{B}^T\nabla\mathbf{I}\right)\left(\mathbf{B}^T\nabla\mathbf{I}\right)^T\right\}^{-1}.\mathrm{E}\left\{\mathbf{B}^T\nabla\mathbf{I}\varepsilon\right\} \quad \text{or} \quad \Delta\theta_k = \mathbf{H}(\theta_k)^{-1}.\mathbf{g}(\theta_k)$$