MOTION PICTURE RESTORATION

by

Anil Christopher Kokaram

Churchill College

A dissertation submitted to the
University of Cambridge for the degree
of Doctor of Philosophy

Department of Engineering

May 1993

A. C. KOKARAM

SUMMARY

This dissertation presents algorithms for restoring some of the major corruptions observed in archived film or video material. The two principal problems of impulsive distortion (Dirt and Sparkle or Blotches) and noise degradation are considered. There is also an algorithm for suppressing the inter–line jitter common in images digitized from noisy video signals.

In the case of noise reduction and Blotch removal the thesis considers the image sequences to be a three dimensional signal involving evolution of features in time and space. This is necessary if any process presented is to show an improvement over standard two–dimensional techniques. As such some attention is given to motion estimation and the examination of the Three Dimensional Autoregressive model as a useful image sequence model.

Impulsive noise removal in image processing has been traditionally achieved by the use of median filter structures. A new three dimensional multilevel median structure is presented in this work with the additional use of a detector which limits the distortion caused by the filters. This technique is found to be extremely effective. A model based technique using the 3D AR model is also developed for detecting and removing Blotches. The technique achieves better fidelity than the median filter at the expense of heavier computational load.

Motion compensated 3D IIR and FIR Wiener filters are investigated with respect to their ability to reject noise in an image sequence. They are compared to several algorithms previously presented which are purely temporal in nature. The 3D filtering process is superior to the purely temporal process as expected.

There are several problems which have not been addressed as far as motion picture restoration is concerned. Chief among these are Line scratches and picture shake or roll. This provides ample oppourtunity for further research. The video supplement to the dissertation gives examples of the restorations which have been achieved for actual degraded sequences digitized from television. It shows that the algorithms can achieve worthwhile restorations of real degraded movies.

To Mom, Dad, Vashiest, Nalini and Kavishti

I was having a little trouble adding up.....
but its alright now.

Clint Eastwood,
For a Few Dollars More

Declaration

The research described in this dissertation was carried out by the author between October 1989 and April 1993. Except as indicated in the text, the contents are entirely original and are not the result of work done in collaboration. No part of this dissertation has been submitted to any other University. The dissertation contains not more than 55,000 words.

Anil Kokaram.

Keywords

The following keywords may be useful for indexing purposes:

Image sequence restoration; Motion estimation; Image restoration; Median filters; Three dimensional Autoregressive modelling; Wiener filters; Interpolation; White Noise suppression; Multilevel median filters; Discontinuity detection;

Video Supplement

The video supplement that accompanies this thesis can be obtained from

The Signal Processing Laboratory,
Cambridge University Engineering Department,
Trumpington Street,
Cambridge CB2 1PZ,
England.

ACKNOWLEDGEMENTS

Writing this section is not the simplest thing I have written in the thesis[1]. The tone is light, suiting the end of a long journey.

I would like to say thanks to all my colleagues in the Signal Processing Laboratory who aided and abetted my endeavours through these years. Thanks go to R. Young and S. Godsill for useful discussions about motion estimation and Autoregressive modelling. I owe a great deal to Dr. M. Lai who showed many of us that the PC is not quite a twilight zone; and who was always willing to help us mere mortals who frequently found the worlds of DOS and Intel too confusing to bear. I wish to thank P. Wilson, S. Karunasekera, C. Canagarajah for their immeasurable support during the writing of this thesis. Thanks must also be ladeled on P. Wilson, R. Auchterlounie and R. Morris for taking the time to upgrade the Novell network in the lab. Also to D. Bott (for hilarious e-mail and believing me when I said I'll write soon), Dr. P. Spencer, Dr. I. Morrison, R. Boyle, P. Jokhi, Dr. N. Kingsbury and everybody else; thanks.

L. Rubenstein must be thanked for providing me with the first material to start an ancient manuscript restoration project, and for her enthusiasm which led directly to the meeting with Prof. Hanley of the Classics department. This also gives me a reason to reference [48, 52].

I wish to thank my supervisor, Dr. Rayner for his support and encouragement over the years, and Dr. Fitzgerald for his enthusiasm. I am also grateful to the British Library and Cable and Wireless for funding my research; Dr. C. Roads, Dr. T. Cannon, Mr. B. Perry and those that serve on the research advisory comittee.

Of course I have to thank my family for always encouraging me in the pursuit of further education, and providing the impetus to complete this degree[2]. And I cannot fail to mention G. White who has brought some good signal processing aplications to my attention[3] [47]. I feel that I owe much of the success of this project to the support and kindness of one person whom I have known for almost 7 years. Sadly, she has forbidden me to mention her at all on pain of having the hardbound version of this thesis impacted soundly upon my head when I least expect, thereby causing internal bleeding, concussion and general unhappiness. Therefore I recognize that I shall incurr certain discomfort, probable disfigurement and/or disfunctionality when I say: _____ I thank you.

It is nice to be able to get 8 hours sleep once more.

[1] The Bibliography was pretty straightforward.

[2] Finish de damm ting and go and make some money boy! We want a Benz.

[3] Like frog sounds (no, they do not sound like Budgies!) and population dynamics of moths that live on citrus.

ABSTRACT

This dissertation presents algorithms for restoring some of the major corruptions observed in archived film or video material. The two principal problems of impulsive distortion (Dirt and Sparkle or Blotches) and noise degradation are considered. There is also an algorithm for suppressing the inter–line jitter common in images decoded from noisy video signals. In the case of noise reduction and Blotch removal the thesis considers image sequences to be three dimensional signals involving evolution of features in time and space. This is necessary if any process presented is to show an improvement over standard two–dimensional techniques.

It is important to recognize that consideration of image sequences must involve an appreciation of the problems incurred by the motion of objects in the scene. The most obvious implication is that due to motion, useful three dimensional processing does not necessarily proceed in a direction 'orthogonal' to the image frames. Therefore, attention is given to discussing motion estimation as it is used for image sequence processing. Some discussion is given to image sequence models and the 3D Autoregressive model is investigated. A multiresolution BM scheme is used for motion estimation throughout the major part of the thesis.

Impulsive noise removal in image processing has been traditionally achieved by the use of median filter structures. A new three dimensional multilevel median structure is presented in this work with the additional use of a detector which limits the distortion caused by the filters. This technique is found to be extremely effective in practice and is an alternative to the traditional global median operation. The new median filter is shown to be superior to those previously presented with respect to the ability to reject the kind of distortion found in practice. A model based technique using the 3D AR model is also developed for detecting and removing Blotches. This technique achieves better fidelity at the expense of heavier computational load.

Motion compensated 3D IIR and FIR Wiener filters are investigated with respect to their ability to reject noise in an image sequence. They are compared to several algorithms previously presented which are purely temporal in nature. The filters presented are found to be effective and compare favourably to the other algorithms. The 3D filtering process is superior to the purely temporal process as expected.

The algorithm that is presented for suppressing inter–line jitter uses a 2D AR model to estimate and correct the relative displacements between the lines. The output image is much more satisfactory to the observer although in a severe case some drift of image features is to be expected. A suggestion for removing this drift is presented in the conclusions.

There are several remaining problems in moving video. In particular, line scratches and picture shake/roll. Line scratches cannot be detected successfully by the detectors presented and so cannot be removed efficiently. Suppressing shake and roll involves compensating the entire frame for motion and there is a need to separate global from local motion. These difficulties provide ample opportunity for further research.

GLOSSARY

AR	**A**utoregressive
AWBME	**A**daptive **W**iener **B**ased **M**otion **E**stimator
3D AR	Three **D**imensional **A**utoregressive
BBM	**B**oyce **B**lock **M**atching (A Block Matching algorithm due to Boyce [13])
BM	**B**lock **M**atching
DFD	**D**isplaced **F**rame **D**ifference
DFDMSE	**D**isplaced **F**rame **D**ifference **M**ean **S**quared **E**rror (The mean squared DFD)
DFT	**D**iscrete **F**ourier **T**ransform
FDMSE	**F**rame **D**ifference **M**ean **S**quared **E**rror (The mean squared frame difference without motion compensation)
FFT	**F**ast **F**ourier **T**ransform
FIR	**F**inite **I**mpulse **R**esponse
IDFT	**I**nverse **D**iscrete **F**ourier **T**ransform
IIR	**I**nfinite **I**mpulse **R**esponse
ISR	**I**mage **S**equence **R**estoration
MAE	**M**ean **A**bsolute **E**rror
MMF	**M**ultilevel **M**edian **F**ilter
MSE	**M**ean **S**quared **E**rror
Pel	**Pixel**
PIMSE	**P**ercentage **I**mprovement in **MSE**
PMSE	**P**ercentage **MSE**
ROC	**R**eceiver **O**perating **C**haracteristic
RMS	**R**oot **M**ean **S**quared
SDI	**S**pike **D**etection **I**ndex
WBME	**W**iener **B**ased **M**otion **E**stimator

CONTENTS

INTRODUCTION

In recent years there has been a growing interest in the area of image sequence processing primarily because there is much more information in a sequence of images than in one still picture. The number of published works in this field has traditionally been small because of the lack of fast and cheap processing architectures suitable for the task. But with the ever increasing speed of microprocessors this is no longer a limitation. Research has been encouraged by the wide ranging applications that demand a knowledge of image sequence processing. Such areas as satellite imaging, Nuclear Magnetic Resonance imaging of living tissue, Electron Microscopy, traffic monitoring, have much to gain from a study of the moving image.

This thesis concerns itself with the problem of image sequence restoration. The application that motivates the work is the restoration of archived motion pictures. All the film shot in the silent era was made with Cellulose ~~Acetate~~ Nitrate which becomes unstable with time. Typical problems are 'blotches' or 'Dirt and Sparkle', due to missing information, and noise due both to the photographic process and the slowly decaying material itself. An interim solution to the problem would be to transfer the image onto video tape. However, the telecine process itself can introduce distortion. Scratches can be introduced due to dirt in the apparatus and noise can be introduced in the recording process.

Modern film is also subject to degradation. The same effects of Dirt and Sparkle as well as noise, are found. Repeated projection will also damage the film since it must be physically transported through the projection apparatus each time. Similarly, video tape is also subject to degradation. Repeated playback will damage the quality of the tape. A loss of synchronization information will occur at some point in time causing frames or lines to be displaced vertically or horizontally on the screen. The restoration of film and video is therefore seen to have useful applications in the entertainment market.

Of course the applications of the algorithms presented in this thesis are not limited to motion pictures. Any degraded image sequence presents itself for similar treatment. The noisy video sequences from electron microscopes are a good example as well as the images produced in remote sensing applications where the environment is quite harsh with respect to image quality.

Much of the work in image processing to date has concentrated on stills, with the notable exception of video compression research. Despite the fact that image sequence analysis for video compression is now an extremely well established pursuit, work in image sequence restoration is only just emerging. Previous to 1991 the references to work in this area have been limited

to [35, 17, 19, 62, 61] a few attempts at motion compensated temporal filtering. More recently, researchers such as [1, 10, 20] have shown that restoration processes in particular have a great deal to gain from a treatment of the image sequence as a spatio–temporal signal. In this mode there is the chance to overcome the high spatial nonstationarity of images by taking advantage of the high temporal correlation in the image sequence.

The thesis addresses the problems of Dirt and Sparkle detection and removal, and the problem of white noise suppression. The theme has been to design algorithms that are effective yet computationally light. In dealing with image sequences, attention must be drawn to the fact that regions in the image are moving. It is not correct to presume that the sequence restoration problem is a 3–D extension of the 1–D audio restoration techniques. Like areas must be treated together. Therefore, it becomes necessary to track the motion of regions in a sequence and consider the same region in separate frames, paying attention to its position in each frame.

So far, to achieve a restoration, the approach has been to find the motion of regions in each frame and then restore the information in a particular region by filtering along motion trajectories. Unfortunately, motion estimation is not a solved problem and is also the subject of much research. There are several approaches to motion estimation and these can be divided into three groups

- Region Matching
- Gradient Based Methods
- Transform Methods

Because of the lack of a tractable model of the image sequence no algorithm can claim perfect results. The thesis therefore spends some effort on assessing the advantages and disadvantages of a few of the more popular motion estimation techniques. However, it is not the role of this thesis to evaluate motion estimation techniques. These are of interest purely through their effect on the final restoration. The fact that one motion estimator may give more inaccurate motion vectors than another is secondary to the performance of the restoration system.

For actual restoration techniques this thesis draws from the work of Veldhuis [96], Vaseghi [94], Rayner and Godsill [31]. These researchers have all considered the problem of reconstruction of missing samples in a data stream. Their work forms the basis of one of the algorithms proposed for missing data interpolation in image sequences. The suppression of noise is treated through a comparison of the standard methods proposed so far. Two spatio–temporal Wiener filters for image sequences are introduced. The chapters which follow can be divided into two main parts; Dirt and Sparkle suppression and Noise suppression. The following is a brief outline of each chapter.

Chapter 2: Motion Estimation for Image Sequence Processing

This chapter reviews some of the approaches to motion estimation to date. The thesis begins with motion estimation since it is the ideal vehicle for investigating the nature of the image sequence. Consideration is given to image sequence models and it is pointed out that the lack of a suitable model continues to hamper the development of robust estimators. An estimator based on an autoregressive model of the image sequence is reviewed. The chapter sets the basis for the algorithms in the thesis since the performance of the restoration techniques depends on the behavior of the motion estimator.

Chapter 3: Three dimensional autoregressive modelling for image sequences

This work investigates and extends the technique developed in [20] to use the 3D AR model as a motion estimator. The technique is shown to give a better prediction error than other estimators, but cannot produce reliable vector estimates.

Chapter 4: Image Sequence Restoration: A Review

A review of the few reported techniques are presented. Median filtering has been put forward as an efficient technique for removing impulsive noise. This is an example of a global filtering operation applied to a problem which involves only a local distortion, yet not much work has been done in the detection of such distortion. The lack of spatio–temporal algorithms is also highlighted. Attention is also given to viewing the motion estimation algorithm in perspective with the rest of the system.

Chapter 5: Registering the lines from noisy digitized TV imagery

Before restoration can begin, the analogue sequence must be converted to a digital format. In order to capture a frame successfully, the digitizing equipment must recognize the synchronization information that is presented with the input video signal. Unfortunately, this information is sometimes lost due to noise or inefficient playback. The resulting inter-line jitter is disturbing to the viewer. A method using 2D AR modelling is presented for removing this distortion.

Chapter 6: Spatio–Temporal Median filtering for suppressing local distortion in image sequences

As stated previously, a common technique for removing impulsive distortion in images is the median operation. Arce [5, 7] has presented multistage median structures for the image sequence that are robust to motion and maintain image detail. This chapter extends that approach by employing motion estimation as well as incorporating more spatio–temporal information. A detector for impulsive noise is also presented which allows the median operation to be controlled. This has the twofold benefit of improving image quality and decreasing the computation necessary.

Chapter 7: Three Dimensional AR Modelling for suppressing local distortion in image sequences

This work reports on the use of a modelling technique that is able to both detect and remove impulsive noise in the same framework. The use of this technique was prompted by the success

of the equivalent 1–D process reported by Veldhuis, Vaseghi and Rayner. Veldhuis also showed good results for the removal of missing data in an image using a 2–D AR model. The method can give better results than the previous technique introduced although it involves more operations.

Chapter 8: Noise Suppression in Image Sequences

Some of the standard approaches to image sequence restoration are compared. The 3D IIR and FIR Wiener filter for image sequences are also presented. The chapter discusses the effects of the algorithms on real degraded motion pictures.

Chapter 9: Conclusions and Further Research

This chapter highlights the achievements of the thesis with respect to the solutions presented for problems in degraded film. It points out that although the 3D Autoregressive (3D AR) model gives a coherent structure for the removal of Dirt and Sparkle in particular, the simpler techniques that were presented achieved the same goal just as well without heavy computation. Several problems that have not been addressed are discussed as well as possible solutions that could guide future work.

MOTION ESTIMATION FOR IMAGE SEQUENCE PROCESSING: A REVIEW

An image sequence, such as a motion picture, consists of a set of images of a scene recorded at regular intervals in time. For the case of television in PAL format this interval is $\frac{1}{25}$th of a second.

The restoration of degraded image sequences can be performed with the repeated execution of the same two dimensional process on the separate images. Viewed in this light, image sequence restoration can be achieved by drawing from the many solutions already proposed for image restoration [36, 54, 89, 27, 78, 42, 83]. However, the recording of each image in a motion picture, for instance, generally occurs more rapidly than the change of information in the scene[1]. Therefore consecutive images in the sequence contain similar information. Hence, an algorithm that can take advantage of the high temporal correlations that exist between frames, has the potential to reject more distortion than a 2D operation. It is the development of such 3D (space/time) algorithms which is of interest in this thesis.

Unfortunately the design of a spatio–temporal restoration algorithm is made difficult by the motion of objects in the recorded scene. This motion implies that there exists useful information only along the trajectory of motion of an object in the sequence of images. Therefore, a 3D algorithm needs to involve knowledge of motion in the image sequence.

An alternative view is to perform spatio–temporal processing only in those areas of the image that are not moving. This would remove the need for motion information. However, such algorithms are limited in performance especially with regard to scratch removal and particularly in scenes with many moving objects.

The subject of Image Sequence Processing, in this case Sequence Restoration, is therefore inextricably linked with Motion Estimation. The restoration algorithms are more effective than 2D processes only if the motion estimator performs satisfactorily. Alternatively, some robustness to erroneous motion estimation must be built in to the restoration algorithm since the behaviour of the motion estimator cannot be guaranteed. There is also the consideration that the sequences being dealt with are degraded and so the motion estimator must be robust to noise.

The previous statements have implied that the motion estimator is a separate issue from the sequence restoration algorithm. Indeed, this has been the usual mode of image sequence

[1] With the exception of scene cuts

processing. The motion in the sequence is first estimated by some algorithm and then processing is directed along the calculated motion trajectories. In many respects, the work presented in this thesis does not deviate from that approach. This mode of thinking has been forced by the lack of image sequence models. If it were possible to develop a model for the image sequence that can predict the observed grey level intensities correctly, then it may be feasible to combine both restoration and motion estimation implicitly in the same algorithm. Unfortunately the development of an image sequence model which begins at the source of the information, i.e. moving objects in a 3D world, is extremely complicated [71] and at the moment impractical. It remains to be seen whether or not a combined model based restoration approach has any advantage over the current mode of thinking.

In some sense the current thinking with respect to image sequences revolves around a translational model. The model used for image sequence processing is therefore

$$I_n(\mathbf{x}) = I_{n-1}(\mathbf{x} + \mathbf{d}_{n,n-1}) \tag{2.1}$$

where $I_n(\mathbf{x})$ is the grey level of the pixel at the location given by position vector $\mathbf{x}$ in the frame n, and $\mathbf{d}_{n,n-1}$ is a displacement mapping the region in the current frame n into the previous frame $n-1$. Under this model, the grey level at each pixel in the current frame can be predicted from a shifted version of levels at pixels in the previous frame. The only parameter that is needed for this model is the motion vector $\mathbf{d}$. Hence the problem of image sequence modelling, using this model, reduces to one of motion estimation.

This chapter attempts to bridge the gap between the various ideas of image sequence modelling, motion estimation and image sequence processing. It points out that although no source model of the sequence has been employed to date, the current techniques can be related through the translational image model. The following sections outline some of the basic characteristics of the image sequence and also consider model based approaches that do not involve a translational approach. It is important to realize that motion estimation is a research topic in its own right. The various approaches to date are reviewed, the idea being to justify the approach taken to motion estimation in this thesis. An exhaustive review is not attempted since there have been many such articles published [81, 67, 70, 68, 63, 80]. The chapter lays the foundation for the ideas developed in the rest of the thesis.

2.1 Image Sequence Modelling: The Image Centric View

There are two approaches to image sequence models. The source approach would begin with consideration of lighting on a 3D world and the subsequent projection of that scene onto a 2D plane. The image centric view disregards the source of the information and tries to place constraints on the behaviour of pixel intensities by observing the sequence itself. This latter approach has been the more tractable one and almost all the motion estimation techniques to

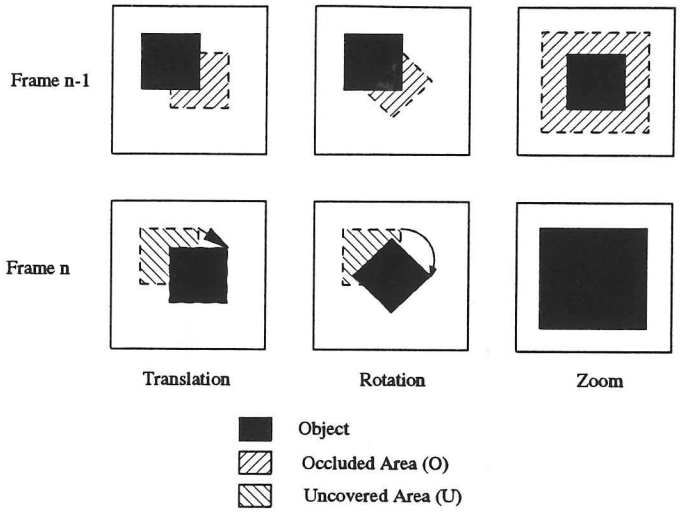

Figure 2.1: Examples of simple motion across two frames.

date can be related via a model as below.

$$I_n(\mathbf{x}) = I_{n-1}(\mathbf{F}(\mathbf{x})) \tag{2.2}$$

The vector function $\mathbf{F}(\mathbf{x})$ represents a linear transformation of image coordinates to represent motion such as zooming, rotation and translation. One restriction of the model is that the intensity changes in the sequence are due only to motion and not to lighting effects such as shadow. The model is intuitively correct since it implies that one image can be predicted from the previous one by rearranging the positions of objects in the scene.

Figure 2.1 shows three different forms of motion typically encountered in an image sequence. When the motion in the 3D world represents movement in a direction perpendicular to the image plane the vector transformation $\mathbf{F}(\mathbf{x})$ can become non–linear. This is the case with zooming, see figure 2.1. However, if the motion is small enough between frames, zooming (and rotation) of small portions of an object can be approximated by translation. In this case the model is as stated in equation 2.1. The vector displacement, $\mathbf{d}_{n,n-1}$, is called the motion vector representing motion between frames n and $n-1$. The modelling problem is reduced to finding the motion vector $\mathbf{d}$ for all the pixels in the image.

Although the model represented by equation 2.1 is at the root of most of the current techniques for image sequence processing, it ignores an important fact. This is well illustrated in figure 2.1. Motion vectors are only defined for areas which can be found in both frames n and $n-1$. Whenever an object moves, some area must be covered and some other area must be uncovered. For these areas there can be no motion vector since these areas represent some newly formed region. The area, labelled 'U' in figure 2.1, has been uncovered in frame n, and the area labelled 'O' has been occluded in frame n. This emphasizes the discontinuous nature of the motion field of a scene. Not only can there be many different objects showing different motion, but unless the entire image is occupied by a moving object, there will at least be uncovered and

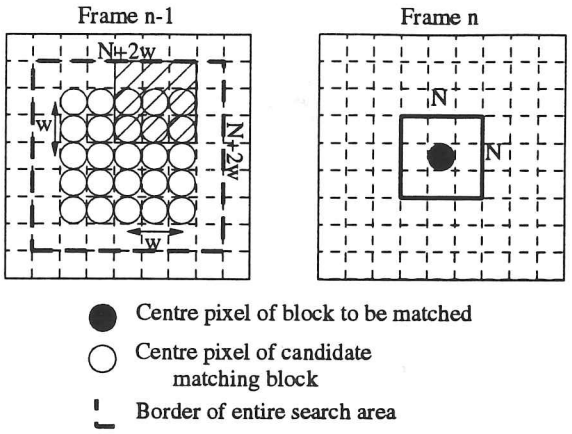

Figure 2.2: Motion estimation via Block Matching. The positions indicated by a ○ in frame $n-1$ are searched for a match with the $N \times N$ block in frame n. One block to be examined is located at displacement $[1\ -2]$, and is shaded.

occluded regions. In estimating motion fields, this problem has been ignored until recently in works such as [37, 97, 90].

The different approaches to motion estimation are identified by the way in which they solve equation 2.1 for **d**. Direct search techniques, called Block Matching have been used. There are also transform approaches which use some variations on the shift theorem [102]. There is a huge body of work using a linearisation of equation 2.1 to solve for the motion vector directly [81]. Finally there has recently arisen work [3, 90] using Gibbs distributions and a Bayesian methodology as introduced by Geman and Geman [27]. The rest of this section reviews some of these approaches.

2.1.1 Correspondence Matching

The most popular and to some extent the most robust technique to date for motion estimation is Block Matching (BM) [29, 103, 13]. The basic assumption in the technique is that pixels in some small region undergo the same translational motion. The image in frame n, is divided into blocks usually of the same size , $N \times N$. Each block is considered in turn and a motion vector is assigned to each. The motion vector is chosen by matching the block in frame n with a set of blocks of the same size at locations defined by some search pattern in the previous frame.

The separation of the candidate blocks in the search space determines the smallest vector that can be estimated. For integer accurate motion estimation the position of each block coincides with the image grid. For fractional accuracy [30], blocks need to be extracted between locations on the image grid. This requires some interpolation. In most cases bilinear interpolation is sufficient.

Figure 2.2 shows the search space used in a full motion search technique. The current block

is compared to every block of the same size in an area of size $(2w + N) \times (2w + N)$. The search[2] space is chosen by deciding on the maximum displacement allowed: in figure 2.2 the maximum displacement estimated is $\pm w$ for both horizontal and vertical components. The method used for determining the best matching block varies, but usually either the Mean Squared Error (MSE) or Mean Absolute Error (MAE) of the pixel intensities is used. The best match is the one that minimizes the error measure used.

The technique arises from a direct solution of equation 2.1. Defining the Displaced Frame Difference as below,

$$\mathrm{DFD}(\mathbf{x}, \mathbf{v}) = I_n(\mathbf{x}) - I_{n-1}(\mathbf{x} + \mathbf{v}) \tag{2.3}$$

where $\mathbf{v}$ is some displacement vector, the BM solution can be seen to minimize the Mean Absolute DFD (or Mean Square DFD) with respect to $\mathbf{v}$, over the $N \times N$ block. The chosen displacement, $\mathbf{d}$ satisfies the model equation 2.1 in some 'average' sense.

The Full Motion Search is computationally demanding, requiring about $N^2(2w + 1)^2$ operations per block for an integer accurate motion estimate. Several reduced search techniques have been introduced which lessen this burden. They attempt to reduce the operations required either by reducing the locations searched [29] or by reducing the number of pixels sampled in each block [103]. However, reduced searches may find local minima in the DFD function and yield spurious matches.

An important aspect of any motion estimator is the implementation of a motion detector. The motion detector attempts to limit searches to the regions where motion is suspected. If a search is undertaken at all locations spurious matches are likely. The motion detector employed usually takes the form of some threshold on the error measure recorded between two blocks in frames n and $n - 1$ if no motion is assumed. The usual mode of operation is to record the error measure at no displacement, and if this is greater than some set motion threshold, motion is flagged and a search is initiated.

The BM algorithm is noted for being a robust estimator of motion since noise effects tend to be averaged out over the block operations. In [13], Boyce introduces a technique to make the BM algorithm more robust to noise. The main error introduced by noise is that of spurious matches since the motion detector is likely to flag motion due to the presence of noise. To isolate this problem, Boyce compares the best match found (E_m), to the original 'no motion' match (E_0). If these matches are sufficiently different then the motion estimate is accepted otherwise no motion is assumed. The threshold acts on the ratio $r_b = \frac{E_0}{E_m}$. The error measure used is the MAE. If $r_b < t$, where t is some threshold chosen according to the noise level suspected, then no motion is assumed. This algorithm will be referred to as the Boyce Block Matching algorithm (BBM) in future references, and the threshold ratio called the *Boyce ratio.* This algorithm therefore verifies the validity of the motion estimate once motion is detected.

[2]There are $(2w + 1)^2$ searched locations

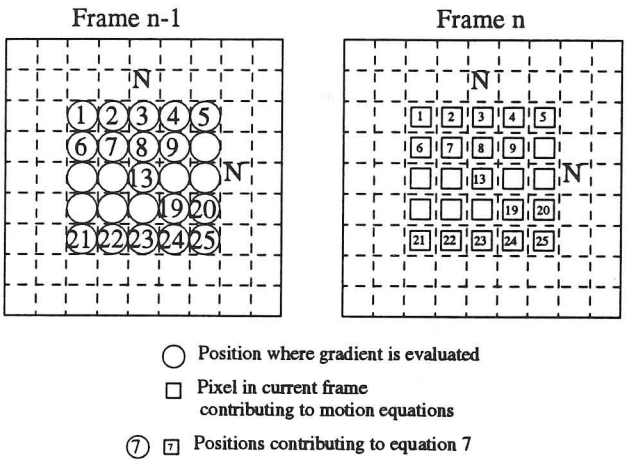

Figure 2.3: Motion estimation via spatial gradients. A block of pixels of size $N \times N$ in frame n is chosen to set up N^2 motion equations; $N = 5$. Each equation requires the evaluation of a gradient in the previous frame. The initial motion estimate is [0 0].

The main disadvantages of Block Matching are the heavy computation involved (although these are byte wise manipulations) and the motion averaging effect of the blocks. If the blocks chosen are too large then many differently moving objects may be enclosed by one block and the chosen motion vector would be unlikely to match the motion of any of the objects. The advantages are that it is very simple to implement[3] and it is robust to noise due to the averaging over the blocks.

2.1.2 Gradient Based Approaches

The second school of thought regarding motion estimation evolved out of a closer look at the image model in equation 2.1. The techniques were initially developed independently by workers in Video Coding and those in Computer Vision. In 1976 Cafforio and Rocca [15], introduced a direct solution for motion using a least squares approach. They were motivated by the need to find a motion estimator that required less computation than BM and also to get around the problem of the motion averaging of the blocks.

In 1978 Netravali and Robbins [73, 74, 81], improved upon that work, introducing the iterative, pel–recursive estimation scheme. Higgins et al. [33, 32], in 1980, were motivated by the study of the human perception of motion. Later on, workers such as Walker and Rao [98], and Biemond et al. [10] (1987), improved upon the technique introduced by Netravali et al. Recently, Efstratiadis and Katsagellos [20] have introduced a framework from which all the previous techniques can be derived.

The basic connection between all these techniques is that they rearrange the model equation 2.1 to have direct access to the motion parameter **d**. The equation 2.1 may be linearised

[3]It has been implemented on Silicon for video coding applications.

about $\mathbf{d}$ using a Taylor series expansion as follows.

$$I_n(\mathbf{x}) = I_{n-1}(\mathbf{x}) + \mathbf{d}^T \nabla I_{n-1}(\mathbf{x}) + e_{n-1}(\mathbf{x}) \tag{2.4}$$

Here $e(\mathbf{x})$ represents the higher order terms of the expansion, and the ∇ operator is the usual multidimensional gradient operator. The equation can be rearranged to yield an expression involving the Displaced Frame Difference with zero displacement.

$$\begin{aligned} \mathrm{DFD}(\mathbf{x}, 0) &= I_n(\mathbf{x}) - I_{n-1}(\mathbf{x}) \\ &= \mathbf{d}^T \nabla I_{n-1}(\mathbf{x}) + e_{n-1}(\mathbf{x}) \end{aligned} \tag{2.5}$$

This equation represents the relation between a number of observables and $\mathbf{d}$. However, there is one equation and 2 variables. To solve the equation therefore, at least one more equation is necessary. If the assumption is made that within a small region all the pixels behave the same with respect to motion, then several more equations can be set up. This assumption will be referred to as the Smooth Local Flow assumption. Therefore, neglecting the higher order terms allows an equation to be set up at each pixel in some defined region to yield a vector equation as below. The situation is illustrated in figure 2.3.

$$\mathbf{z}_0 = \mathbf{G}\mathbf{d} \tag{2.6}$$

$$\text{where } \mathbf{z}_0 = \begin{bmatrix} DFD(\mathbf{x}_1, 0) \\ DFD(\mathbf{x}_2, 0) \\ \vdots \\ DFD(\mathbf{x}_{N^2}, 0) \end{bmatrix}$$

$$\text{and } \mathbf{G} = \begin{bmatrix} \frac{\partial}{\partial x} I_{n-1}(\mathbf{x}_1) & \frac{\partial}{\partial y} I_{n-1}(\mathbf{x}_1) \\ \frac{\partial}{\partial x} I_{n-1}(\mathbf{x}_2) & \frac{\partial}{\partial y} I_{n-1}(\mathbf{x}_2) \\ \vdots & \vdots \\ \frac{\partial}{\partial x} I_{n-1}(\mathbf{x}_{N^2}) & \frac{\partial}{\partial y} I_{n-1}(\mathbf{x}_{N^2}) \end{bmatrix}$$

In figure 2.3, the set of N^2 equations is set up using the block indicated. The vectors are therefore set up by assembling the row–wise observations into columns. The gradient measurements are made using a simple difference technique, for example the horizontal gradient at pixel 2 in figure 2.3 is $\frac{I(3)-I(1)}{2}$. This is admittedly a noisy process and a more robust class of gradient estimation techniques, using curve fitting, was presented in [63].

A solution for $\mathbf{d}$ can then be generated using a pseudoinverse approach.

$$\mathbf{d} = [\mathbf{G}^T\mathbf{G}]^{-1}\mathbf{G}^T\mathbf{z}_0 \tag{2.7}$$

This solution yields $\mathbf{d}$ in one step by using gradient measurements in some region as in figure 2.3. It is one of the early solutions proposed by Cafforio et al, Netravali et al, Martinez [63] and others. The important drawback with this approach, recognized in [73], is that the

Taylor series expansion is only valid over very small distances. To overcome this problem, a recursive solution was introduced called the pel–recursive approach. This new solution refines an initial estimate for $\mathbf{d}$ until no further reduction in DFD is observed.

Rather than expand the model equation about $\mathbf{x}$, the image function is linearised about a current guess for $\mathbf{d}$, say $\mathbf{d}_i$. The idea is to generate an update, $\mathbf{u}$ for the current estimate that will eventually force the estimation process to converge on the correct displacement. The update is meant to be a small one to maintain the validity of the Taylor series expansion. Therefore, equation 2.4 becomes

$$\begin{aligned} \mathbf{u}_i &= \mathbf{d} - \mathbf{d}_i \\ I_n(\mathbf{x}) &= I_{n-1}(\mathbf{x} + \mathbf{d}_i) + \mathbf{u}_i^T \nabla I_{n-1}(\mathbf{x} + \mathbf{d}_i) + e_{n-1}(\mathbf{x} + \mathbf{d}_i) \end{aligned} \tag{2.8}$$

The solution may then continue as before. However, Biemond [10] presented a Wiener solution for the displacement which is more robust to noise. It handles the higher order terms more effectively by considering their effect on the solution to be the same as Gaussian white noise. The new vector equation to be solved becomes

$$\mathbf{z}_i = \mathbf{G}\mathbf{u}_i + \mathbf{e} \tag{2.9}$$

$e(\mathbf{x} + \mathbf{d}_i)$ represents the higher order terms which are considered to be Gaussian white noise of variance σ_{ee}^2. Note that the matrices $\mathbf{G}$, $\mathbf{z}_i$, now consist of terms similar to those defined earlier in equation 2.6, but the arguments are offset by the current estimated displacement, $\mathbf{d}_i$. The Wiener solution chooses the update $\hat{\mathbf{u}}_i = \mathbf{L}\mathbf{z}_i$ which minimizes the expected value of the squared error between the true update $\mathbf{u}_i$ and the estimate $\hat{\mathbf{u}}_i$. The objective function is therefore,

$$E[(\mathbf{u}_i - \hat{\mathbf{u}}_i)^T(\mathbf{u}_i - \hat{\mathbf{u}}_i)] \tag{2.10}$$

Minimizing this function with respect to $\mathbf{L}$ yields the following solution for $\hat{\mathbf{u}}_i$. The full derivation may be found in [10].

$$\begin{aligned} \hat{\mathbf{u}}_i &= [\mathbf{G}^T\mathbf{G} + \mu\mathbf{I}]^{-1}\mathbf{G}^T\mathbf{z} \\ \mu &= \frac{\sigma_{ee}^2}{\sigma_{uu}^2} \end{aligned} \tag{2.11}$$

Here, σ_{uu}^2 is the variance of the estimate for $\hat{\mathbf{u}}_i$, provided that both components of the motion have the same variance. Also, note the use of μ as a regularizing term in the required matrix inverse. The success of the method lies in treating the observables $\mathbf{z}, \mathbf{G}, \mathbf{e}$ as random variables. The estimator will be referred to as the Wiener based motion estimator (WBME) in future references.

After the update is estimated, $\mathbf{d}_i$ is refined to yield the displacement that is used in the next iteration, $\mathbf{d}_{i+1} = \mathbf{d}_i + \mathbf{u}_i$, and the process continues again. During the iterative process, the

observables may have to be evaluated at fractional displacements. Usually bilinear interpolation is employed to keep computation at a minimum. The iteration is terminated when the algorithm converges. This is usually detected by thresholding the size of the update vector and the size of the current Mean Square Error. When either of these two quantities are below their respective thresholds, convergence is assumed.

The main advantage of the Wiener based pel–recursive estimator is that it requires much less computation than the Block Matching algorithm[4]. It is also capable of resolving both fractional and integer displacements without changing the number of operations involved. Because the computational complexity is so low, many authors in the Video Coding field have used the technique to estimate a motion vector at each pixel. This is done so that there is less chance of the 'motion averaging' effect of blockwise manipulations. The approach is of limited use because some finite support region must always be employed to set up the necessary equations. Depending on the size of that region, errors are likely. The most important disadvantage of the estimator is that even as an iterative process, it cannot estimate large displacements because of the limited effectiveness of the first order Taylor series expansion. This is in contrast to BM which is only limited by the defined search space.

2.1.3 Ambiguity in Motion Estimation: Adaptive solutions and additional constraints

A fundamental fact about the translational model of motion is that it cannot be solved using one observation at one pixel alone. To overcome this difficulty the Smooth Local Flow assumption must be made to allow observations at several pixels to be used. In order to improve motion estimation, additional constraints must be found. A closer look at the gradient based approach to motion estimation led several authors to introduce further improvements to the basic estimator. Biemond et al. [10, 12, 53, 38, 37, 39] altered the WBME by introducing adaptive schemes. A separate approach was employed by Nagel [72, 69, 65, 67, 66, 71] and others [24, 34, 33, 32, 85, 86] who further refined the motion estimation solution by employing constraints on the motion itself. The ambiguity involved in motion estimation is illustrated in figure 2.4. The rectangular object of uniform intensity is moving diagonally and so has two non zero components of motion. If the block used for estimating motion in either a BM or WBME algorithm is located at an edge as shown in the figure then several motion estimates can be found which will cause a zero DFD. Only one of these is the true motion vector. The situation illustrates two points,

- The motion estimate is most accurate in the direction of maximum image gradient (or contrast). At edges this direction is perpendicular to the edge, at corners there are two directions along which motion estimation is accurate. Hence, for corners both components of motion (horizontal and vertical), can be estimated accurately.

[4] When convergence is achieved it is typically within 10 iterations

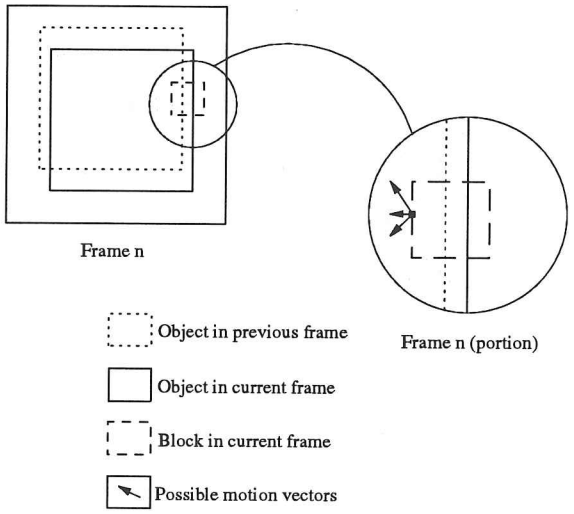

Figure 2.4: Motion ambiguity at an edge. *The motion estimate is most accurate along the direction of maximum image gradient.*

- The size of the block used for estimating motion affects the locality of the motion estimate. Because the block in figure 2.4 is small compared to the size of the object, a local motion estimate results which may not reflect the true motion of the object. It is this effect which causes the motion estimated in that block to be different from the global motion of the object.

If the block used for motion estimation is located in the centre of the moving object in figure 2.4, then it is likely that neither estimator will estimate any motion at all. There will be zero error for zero displacement and so the motion detector will not flag motion. Even if motion estimation were to be engaged then the zero or near zero DFD coupled with the zero gradient in the previous frame will bias the result toward a zero motion vector. Martinez [63] and Kearney at al. [46] quantitatively investigated this problem with respect to the gradient based estimation process. Those findings are discussed next.

Improved pel–recursion

Errors in the WBME can be linked directly to the extent of ill–conditioning in the matrix $\mathbf{M}_g = [\mathbf{G}^T\mathbf{G}]$. In this respect the WBME can be viewed as a regularized least squares solution for motion. The μ acts as a damper on the system, preventing the inverse from being unstable. Large μ encourages small updates and small μ allows large updates. The WBME is a regularized version of solution 2.7.

It is clear that μ should respond to ill–conditioning in $\mathbf{M}_g$. This ill–conditioning can be measured by the ratio of the eigenvalues of the matrix. Martinez was able to prove that this ratio was governed by the relative magnitudes of the two components of average spatial contrast (or gradient). His analysis is repeated in Appendix D. He showed that the solution for the

displacement was ill–conditioned when the size of one component of spatial gradient was much larger than the other; i.e. at an edge. The corollary to this result is that the motion estimate is most accurate in the direction of maximum image gradient. This is self evident from the figure 2.4, but the investigation was then able to employ the SVD decomposition of $\mathbf{M}_g$ intelligently.

Martinez presented in [63] a motion estimator, which was not pel–recursive, that was sensitive to this problem. It is stated below.

$$\begin{aligned} \mathbf{d} &= \begin{cases} \alpha_{\max}\mathbf{e}_{\max} & \text{if } \lambda_{\max} \gg \lambda_{\min} \\ \left[\mathbf{G}^T\mathbf{G}\right]^{-1}\mathbf{G}^T\mathbf{z} & \textit{otherwise} \end{cases} \\ \alpha_{\max} &= \frac{\mathbf{e}_{\max}^T}{\lambda_{\max}}\mathbf{G}^T\mathbf{z} \end{aligned} \tag{2.12}$$

Here, λ, $\mathbf{e}$ refer to the eigen values and eigen vectors of $\mathbf{G}^T\mathbf{G}$, and $\alpha_{\max}$ is a scalar variable used to simplify the final expression. The solution presents a specific response when the inverse operation is ill–conditioned. This response involves aligning the motion estimate along the direction of maximum contrast in order to make the most of an uncertain situation.

Workers such as Efstratiadis et al. [20], Driessen et al. [38, 37] and Böröczky [12, 53] have concentrated on adapting μ in the WBME to the ill–conditioning of $\mathbf{M}_g$ during the pel–recursive process. The estimator proposed by Driessen is shown here.

$$\begin{aligned} \mathbf{d} &= \left[\mathbf{G}^T\mathbf{G} + \mu\mathbf{I}\right]^{-1}\mathbf{G}^T\mathbf{z} \\ \textit{where } \mu &= |\mathbf{z}|\frac{\lambda_{max}}{\lambda_{min}} \end{aligned} \tag{2.13}$$

Therefore, as the required inverse solution becomes ill–conditioned (measured as a ratio of eigenvalues), μ increases, thus increasing the damping in the system. The additional multiplying factor of $|\mathbf{z}|$ helps to further stabilize the situation by relaxing the damping when the error is small and the motion estimate is close to convergence. There is obviously some scope here for a joint Driessen/Martinez algorithm and this is investigated in the later chapters of this thesis.

Additional Constraints

Rather than salvage the motion estimate when the solution is ill–conditioned, Nagel [72, 69, 65, 67, 66, 71] and others [24, 34, 33, 32, 85, 86, 22] used the Local Smooth Flow assumption itself as a constraint in the solution. These studies attempted to identify the true motion in a scene. They addressed the remaining problem facing standard gradient based motion estimators. When the textural information in a region is low, e.g. within the rectangular shape in figure 2.4, the gradient information is insufficient to provide any motion estimate. The resulting vector field is then non–zero only across the edges of the object, where there is some textural information.

To get around this problem, the motion estimate was constrained to be some value which did not violate the smoothness of the motion field. Therefore after several iterations, motion

vectors could be propagated from the boundaries of moving objects into the interior where there was less gradient information. The standard approach in these multiple constraint techniques is to consider a rearrangement of the expression 2.8 to yield an objective function $F(\mathbf{x})$ defined as below.

$$F(\mathbf{x}) = DFD(\mathbf{x}, \mathbf{d}_i) - u_i(x)\frac{\partial I(\mathbf{x}+\mathbf{d}_i)}{\partial x} - u_i(y)\frac{\partial I(\mathbf{x}+\mathbf{d}_i)}{\partial y} \tag{2.14}$$

The truncation error term is ignored. $F(\mathbf{x})$ is zero if the correct displacement is found and so to find this value the function $F(\mathbf{x})^2$ is minimized with respect to the update vector $\mathbf{u}_i = [u_i(x)\ u_i(y)]$ and a constraint as below, following Horn et al. [34].

$$(\frac{\partial u_i(x)}{\partial x})^2 + (\frac{\partial u_i(x)}{\partial y})^2 + (\frac{\partial u_i(y)}{\partial x})^2 + (\frac{\partial u_i(y)}{\partial y})^2 = K \tag{2.15}$$

Using Lagrange multipliers, the solution was then able to produce a smooth vector field since the constraint restricted the rate of change of the motion estimate with distance.

Nagel [72, 69, 65] considered the phenomenon of directional accuracy in motion estimation, as discussed earlier. He argued that

1. Motion estimation is most accurate in the direction of maximum image gradient, i.e. when aligned with the principal direction of image gradient given by the eigenvectors of the gradient matrix considered previously.

2. Motion field smoothness is violated at motion boundaries[5]. These motion boundaries would in general correspond to boundaries of objects in the image and so motion flow smoothness over such boundaries should be discouraged.

His ideas were implemented as a directed smoothness constraint as opposed to the homogeneous smoothness constraint of Horn. The constraints introduced were sensitive to the direction of image gradient and to edges in images.

These gradient based estimators improve upon the basic idea of the previous section, but also introduce another level of complexity into the motion estimation process. Although these estimators have better potential for yielding a true motion field it is questionable to what extent this is necessary for the application in this thesis. The fact that BM or the WBME cannot give true motion estimates in some image regions, is perhaps secondary to their role in providing a low prediction error. When the choice of several vectors gives the same low prediction error, does it matter that several of these vectors do not reflect the true motion? It is for this reason, together with the overriding necessity to keep computation at a minimum, that the multiple constraint techniques for motion estimation are not pursued in this work.

[5] Local Smoothness versus Global Discontinuity.

BLOCK MATCHING	GRADIENT BASED
Heavy Computation	Low Computation
Large motion OK	Small motion only
Fixed resolution motion[6]	Variable resolution
	Ill–conditioned when no texture
Only the actual data is used	Gradient measurements are noisy

Table 2.1: Comparing Block Matching and Gradient Based motion estimation.

2.2 Block Matching VS. WBME

The two more popular techniques to date are Block Matching and Gradient based motion estimation as presented by Biemond [10]. Block Matching is still the more popular for Video Coding techniques despite its greater computation requirement, for two reasons.

1. It is simple to implement and requires only integer operations.
2. It can handle any displacement size depending on the search space used.

The WBME algorithm is computationally lighter, but it cannot handle displacements greater than 5 pixels in practice. This is because, for large displacements, the two regions being compared in the two frames may have no relation at all. Therefore the Taylor series expansion of the region in the previous image will have no bearing on the current region. This accounts for the lower popularity of the WBME. The basic tradeoffs between the two techniques are outlined in the table 2.1.

To increase the importance of the first order term in the Taylor series expansion, Kearney et al. [46] have proposed blurring the image using some low–pass filter prior to estimation. The method has merit and tends to stabilize the algorithm. There is some loss in precision but this is not catastrophic provided the size of the blur is controlled. The technique will be referred to again in later discussions.

2.3 Image Sequence Modelling: Alternatives

The previous sections have discussed the solutions proposed for the purely translational model for image sequences. The model represents an image centric view in that the constraints employed have not been derived from a consideration of the image formation process. This section attempts to briefly outline some of the work done in this area. It also reviews an Image Centric Model which breaks from the tradition of the translational model.

[6]The error field from the BM algorithm can be interpolated to yield a floating resolution vector field, but this is not usually done.

Schunck in [85, 86] shows that the image flow equation 2.14 is valid also across image discontinuities i.e. edges. In [85] he points out that it is a simple relation that does not consider lighting effects such as shadow, and depth motion effects such as zoom. In an attempt to incorporate such observations he defined a measure of 'feature' density in preference to grey level. He suggested this density to describe the occurrence of important structures e.g edges, corners etc that were of use in motion estimation. His arguments were speculative, but gave rise to a modified flow equation. The equation expresses the conservation of feature density in a unit element of image. The equation statement is as follows

- *The flux of feature fragments through unit image element must be balanced by the accumulation of such fragments within the element.*

The modified flow equation is as follows,

$$N\nabla.\mathbf{v} + \mathbf{v}^T\nabla N + \frac{\partial N}{\partial t} = 0 \tag{2.16}$$

Here N is the feature density, ∇N is the spatial gradient $\nabla N = [\frac{\partial N}{\partial x}\ \frac{\partial N}{\partial y}]$, and $\nabla.\mathbf{v} = [\frac{\partial v_x}{\partial x} + \frac{\partial v_y}{\partial y}]$ is the term representing the divergence of the flow field. Schunck recognized that only the consideration of perspective projection of a unit area in a real scene with certain irradiance and reflectance properties, would yield a true constraint.

Nagel [71] considered just that problem. He derived a displacement rate constraint equation by considering the projection of a planar surface element having some reflectance properties, onto an image element in the image plane. The surface element was presumed to rotate and translate in the 3D scene and the element was considered as a Lambertian source of reflected light. The constraint equation was derived on the basis of equating the instantaneous temporal change of intensities at a point in the 3D scene with respect to the spatial change of intensities at the image element. Nagel's equation was as follows.

$$\mathbf{v}^T\nabla I(\mathbf{x},t) + \mathrm{DFD}(\mathbf{x},t) = I'(\mathbf{x},t)f(\mathbf{x},t) \tag{2.17}$$

Here $I'(\mathbf{x},t)$ is not the grey level at $\mathbf{x}$ in the image plane, but a ratio between the power received by an image element divided by the size of the element. $f(\mathbf{x},t)$ refers to a combination of effects dealing with irradiance factors and the velocity of the object element in scene space. He found that although a divergence term similar to the one suggested by Schunck did arise, it was cancelled by the effects due to changes in size of the element in projection onto the image plane. The essential statement, however, is that the conservation of image grey level described by the simple flow equation is not satisfied. The right hand side of 2.16 does not vanish. Nagel concluded his findings by saying that his derivation was based on simplistic assumptions regarding the nature of image formation. Therefore more effort was necessary to derive a more general constraint.

These two contributions by Schunck and Nagel serve to highlight how difficult it is to derive image sequence constraints based on scene modelling. Particularly troublesome is the depth

ambiguity problem. Any motion in a direction perpendicular to the image plane causes a change in size of the object. This implies that constraint equations such as these invariably incorporate depth motion as an independent variable. The ambiguity arises in attributing image displacement to motion or a zooming effect. Both of these could yield similar image transformations over small regions.

The application of this thesis, however, is sequence restoration. Although it is probably better in the long term to have an estimate of the true motion in the image, it is not of primary concern. If over small displacements in scene space, depth motion is indistinguishable from motion parallel to the image plane, then an image centric motion model will suffice. Of particular importance to a model used for sequence restoration is the ability of that model to predict a frame. The translational model is sufficiently accurate in this respect for its use in Video Coding[7], and the next section reviews an Image Centric model which improves upon this idea.

2.3.1 An Image Centric Sequence Model which is not Purely Translational

One of the shortcomings of the translational model is that it is unable to explain the variation in image intensity from frame to frame in the same object. In 1990, Efstratiadis et al. [20] introduced the use of the Three Dimensional Autoregressive (3D AR) Model for image sequence processing. This model can deal with intensity changes and can be stated in words as follows

- *A prediction of the grey level at a pixel in frame n can be given by a linear weighted combination of pixels in some local region surrounding the pixel to be predicted.*

The model was actually presented earlier by Strobach [91], but that implementation used the model primarily for image segmentation rather than motion estimation.

The model tries to make the best prediction of a pel in the current frame based on a combination of intensities at pels in a predefined support region. This support region may occupy pels in the current frame as well as ~~previous~~ future and past frames.

Because of motion between frames in a sequence, the support region in each frame must be offset from the predicted pel by a distance related to that motion. Therefore, as opposed to the two dimensional AR model [42, 82, 41, 40, 99, 100, 92], a third parameter, considered to be the relative motion between frames, must be introduced.

In order to best explain the form of the final equations, the first consideration is given to describing a 3D model which does not involve motion. Following the notation of Kashyap [82], the location of each pixel in the support region is defined by an offset vector $\hat{\mathbf{q}}_k$. This vector consists of three components, the horizontal offset and the vertical offset within a frame and

[7] Of course in Video Coding applications, the error signal is transmitted to further improve the prediction, but the application of the model for that purpose still indicates some usefulness, especially with respect to robust performance, which cannot be ignored.

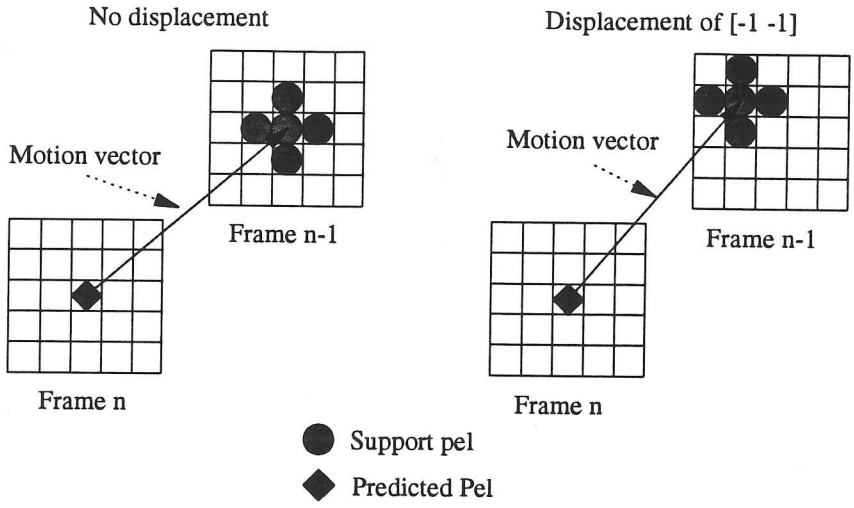

Figure 2.5: Handling motion with the 3D AR model.

then the temporal offset between frames. There are N vectors in the support which may define a completely general region in a (space/time) volume around the predicted position. If the grey level at a particular location $\mathbf{x}$ in frame n is defined as $I(\mathbf{x})$, then the model is

$$I(\mathbf{x}) = \sum_{k=1}^{N} a(\mathbf{q_k})\mathbf{I}(\mathbf{x} + \mathbf{q_k}) + \epsilon(\mathbf{x}) \tag{2.18}$$

$\epsilon(x)$ is a Gaussian random variable whose correlation properties depend on the structure of the support for the AR model (See Appendix B). The $a(\mathbf{q_k})$ are the weights corresponding to each particular offset vector. The figure 2.5 shows a typical configuration where the grey level at $\mathbf{x}$ is a linear combination of those levels in a + shape in the preceding frame.

In the case of motion however, the supports in various frames are necessarily displaced relative to each other. In order to describe the altered model, a 'motion' parameter, $\mathbf{d}_{k,l} = [sx_{k,l}, sy_{k,l}]$ is introduced[8].The 2-D vector $\mathbf{d}$ is the relative spatial displacement between the same region in frames k and l. Because $\mathbf{d}_{k,l}$ is a function of time (the k,l), it becomes necessary to separate the various components of the arguments in equation 2.18. Therefore a position vector $\mathbf{x} = [i, j, n]$ is defined, where i,j,n are the horizontal,vertical and temporal components respectively. Similarly, the offset vector $\mathbf{q}_k$ consists of the three components, $[q_k(x), q_k(y), q_k(n)]$. The three dimensional autoregressive model incorporating motion now becomes

$$I(i,j,n) = \sum_{k=1}^{N} a_k I(i + q_k(x) + sx_{n,n+q_k(n)}, j + q_k(y) + sy_{n,n+q_k(n)}, n + q_k(n)) + \epsilon(i,j,n) \tag{2.19}$$

Examination of the above equation will show that the correct relation between points in the support of the AR model has been achieved. That is, the support pel in a particular frame p say, must be offset by the relative displacement between the current frame n and p i.e. by $\mathbf{d}_{n,p}$.

[8]It will become clear why the word motion is placed in quotes after a few results from experiments have been presented in Chapter 3.

Figure 2.5 illustrates this point. Note that in the equation above, a_k has been used as shorthand for $a(\mathbf{q_k})$.

The model is plainly a superset of the translational model 2.1. If $a_1 = -1.0$, and there was only one tap at $\mathbf{q}_1 = [0\ 0\ -1]$, then model 2.1 results. $\epsilon(i,j,n)$ would then be some measure of the deviation of the real world behavior from the translational ideal. For the case of the general formulation, instead of modelling image behavior as a pure translation of the same grey level value, it is modelled as a translation and a modification by some gain factor. This modification attempts to explain effects which cannot be handled by intensity constant translation. To a certain extent, the model is able to handle effects such as occlusion by the ability to alter the model coefficients to make the best prediction. This cannot be always effective however and generally it will fail when this phenomenon occurs.

Parameter Estimation

The parameters which are required are the weights, a_k and the displacement $\mathbf{d}_{k,l}$. Unfortunately, the $\mathbf{d}_{k,l}$ is an argument in the intensity function, I(). If it were possible to write I() explicitly as a function of $\mathbf{d}$, then one would have ready access to this parameter. However, I() is only known by observation of its behaviour and in any typical informative scene, cannot be described easily by an analytic function. Unless some effective approximation to I() can be found which is expressed explicitly in terms of $\mathbf{d}_{k,l}$, this parameter can only be evaluated on the basis of a search which minimizes some cost function.

The coefficients $\mathbf{a}$[9] themselves can always be estimated from equation 2.19 on either a Maximum Likelihood basis or a least squared error basis if the displacement is known. The fact that both a and $\mathbf{d}$ are unknown then, necessitates some approximation that will make the solution tractable. The alternative is to minimize some cost function over the extended space $[a\ \mathbf{d}]$, a task which involves quite high dimensionality. In [20], the correlation structure of the image was assumed to be some known function. Given this function, the Normal equations could then be solved to yield some fixed set of model coefficients, $\mathbf{a}$. Knowing the coefficients, therefore, the task of modelling reduces to estimation of the displacement parameter.

In the derivation of a closed form solution for $\mathbf{d}$, the modelling equation is considered in its prediction mode. The 'noise' $\epsilon(i,j,n)$ is considered to be the error between the predicted grey level intensity, $\hat{I}$ and the actual intensity $I(i,j,n)$, where

$$\hat{I}(i,j,n) = \sum_{k=1}^{N} a_k I(i + q_k(x) + sx_{n,n+q_k(n)}, j + q_k(y) + sy_{n,n+q_k(n)}, n + q_k(n)) \tag{2.20}$$

The task then becomes that of choosing the parameters to minimize some function of the error, or residual,

$$\epsilon(i,j,n) = I(i,j,n) - \hat{I}(i,j,n) \tag{2.21}$$

[9] Vector notation used here to represent the vector of model coefficients.

The estimation [20] of the displacement parameter proceeded in a manner analogous to the approach by Biemond et al. discussed previously. The derivation is outlined in Appendix C. A Wiener based estimate for an update vector was found, and the final solution is stated here.

$$\hat{\mathbf{u}}_w = [\mathbf{G}_w^T \mathbf{R}_{vv}^{-1} \mathbf{G}_w + R_{uu}^{-1}]^{-1} \mathbf{G}_w^T \mathbf{R}_{vv}^{-1} \mathbf{z}_w \tag{2.22}$$

The form of the final solution is similar to the Biemond solution with two important differences. The gradient matrix, $\mathbf{G}_w$ consists of terms that result from a weighted[10] combination of gradients defined by the support region of the model. The error terms involved are a summation of both the truncation error of the Taylor expansion and the actual prediction error $\epsilon(i,j,n)$. In fact, the prediction error $\epsilon(i,j,n)$ is not white for non–causal model support (see appendix B). However, as long as the model support does not incorporate non–causal taps in the current frame[11] , the correlation structure of the error within the current frame (i.e. for the purposes of the solution 2.22), can be assumed to consist of one non–zero term at the zero lag position. In such a situation the update equation becomes,

$$\hat{\mathbf{u}}_w = [\mathbf{G}_w^T \mathbf{G}_w + \mu \mathbf{I}]^{-1} \mathbf{G}_w^T \mathbf{z}_w \tag{2.23}$$

where $\mu = (\sigma_{\epsilon\epsilon}^2 + \sigma_{ee}^2 \sum_{k=1}^{N} a_k^2)/\sigma_{uu}^2$. ($\sigma_{ee}^2$ = truncation error variance.) For a more complete treatment, see Appendix C.

The fact that the matrices used for this update have terms which are a weighted combination of observed values, may imply that the solution is more robust to noise. Efstratiadis et al. have indeed reported an increased robustness with this algorithm. The algorithm is more stable than the standard WBME and a smoother motion field was reported. However, an analysis similar to that carried out by Martinez [63], in Appendix D, shows that the conditions for ill–conditioning are still the same. If the points used to set up equation 2.23 lie along an edge, then the solution is ill–conditioned.

The work presented in [20] is extended in Chapter 3. In that chapter, consideration is given to adapting the coefficients to the image data by observing the image correlation structure at each iteration. One remaining point about the model is that the displacement parameter may not represent the true displacement of the scene. This is investigated further in Chapter 3 and the resulting observations have implications for the use of the model in a multiresolution scheme. This model is used later in Chapter 7 for scratch detection and removal as an alternative to the heuristic solution of Chapter 6.

2.4 Estimating Large Displacements

Throughout the discussion so far it has been noted that gradient based techniques can only effectively estimate small displacements. A correspondence technique, such as BM, is only limited

[10] Hence $\mathbf{G}_w$ for weighted gradient.

[11] That is, taps to the right or below the predicted location.

in this respect by the extent of the search space used. However, increasing the search space to deal with a large displacement, increases the computational requirement. Considering that motion pictures can involve displacements in excess of 10 pixels per frame, increasing the search space of BM to deal with this motion results in a huge increase in computation.

It has been widely accepted that the multiresolution schemes provide the most practical way of dealing with this problem effectively. The idea of a multiresolution representation was presented early on for use in Image Coding by Burt and Adelson [14]. The technique can reduce the computational requirement to estimate a particular displacement in the case of BM [11]. In the case of gradient based techniques, the scheme can make the problem better conditioned and increase convergence rates [22, 63].

The multiresolution scheme begins by building successively smaller versions of the images in terms of spatial extent, while still maintaining useful image information in each version. The original images are low pass filtered and subsampled until the displacements measured in the subsampled frames are small enough to allow reasonably accurate pel–recursive, or BM motion estimation with a small search space. The motion vectors found at the lower resolutions are then projected onto the higher resolutions where the motion estimation process is allowed to continue. Successive projections of the estimated motion field eventually leads to an initial motion field at the highest resolution and then it is presumed that only a few iterations will be needed for a pel–recursive algorithm to converge. In the case of BM the search space at each level can be drastically reduced, because a small displacement at the higher levels represents a large displacement at the original level. It is typical to subsample by a factor of 2 so that if the original level is of resolution $H \times V$ then level l of N levels has a size $\frac{H}{2^l} \times \frac{V}{2^l}$. Motion of magnitude k pixels at level 0 (the original resolution level), is then reduced to $k \times 2^{-(N-1)}$ at level $N-1$. Figure 2.6 illustrates the pyramidal representation.

Many schemes have been proposed that generate multiresolution pyramids, they all employ different low pass filters. The Wavelet transform [60] in particular has been demonstrated to provide an analytic basis for multiresolution analysis and fast algorithms have been developed for its implementation. It is not the purpose here to launch into a discussion of the merits of the various multiresolution schemes. It must be noted that the underlying motivation for a multiresolution approach here is to create a representation of the original image in which the motion is small enough to allow a motion estimation algorithm to terminate successfully. To this end the basic character of the image at the varying scales must be maintained. This is necessary to ensure that the motion fields estimated at the lower resolutions bear some resemblance to the actual motion field at the original resolution.

The multiresolution algorithm of Enkelmann and Nagel [22] is adopted for use in this thesis. It uses a Gaussian shaped low pass filter which is able to provide a basis for the pyramid and also slightly blur the image at each scale. This latter point is useful in that it artificially increases the proportion of the first order term in the Taylor series expansion of the image function, thus

stabilizing the iterative process, (see Kearney et al [46]). Further, the filter is non directional and so does not favour motion estimation in any direction. The implementation used here is the standard refinement strategy common to multiresolution motion estimators. The low pass filter is defined as

$$\begin{aligned} f(x,y) &= w(x,y)\frac{1}{A}\exp^{-\left(\frac{r^2}{2\sigma^2}\right)} \\ r &= \sqrt{x^2+y^2} \\ w(x,y) &= \left\{ \begin{array}{ll} 1 & \text{for } x^2+y^2 \leq R^2 \\ 0 & \text{otherwise} \end{array} \right\} \\ A &= \sum_{|x|\leq R}\sum_{|y|\leq R} w(x,y)\exp^{-\frac{r^2}{2\sigma^2}} \end{aligned} \tag{2.24}$$

The multigrid algorithm is enumerated below

1. Generate L levels of the multiresolution pyramid, such that level $l = 0$ is the original resolution image, and level $l = L-1$ is the image with the smallest resolution.
2. Set the initial level to be the level with the most coarse resolution, level $l = L-1$. Set the initial field at this level to be a set of zero vectors.
3. Generate an estimate of the motion field at the resolution level l using the current starting field.
4. if $l = 0$ then goto 8.
5. Propagate the motion field from the coarse resolution level l down to the next higher resolution level $l-1$.
6. Set the current resolution level to be $l = l-1$.
7. Goto 3.
8. Stop.

Bilinear interpolation is used to propagate the motion field one level down, i.e. onto the higher resolution level. This will assign all pixel sites at the new resolution with a start vector. It is arguable what interpolating function would perform best, but it is found that the Bilinear operation gives satisfactory results.

There is a drawback with the multiresolution motion estimation scheme. It brings similar regions, not related by motion, closer together. A typical effect is the assignment of displacement vectors to stationary regions just outside moving regions. For gradient based algorithms this could cause a lack of convergence at the lower levels. The effect is called the vector halo effect. Bierling [11] solves the problem with respect to BM by using a motion detector at the lowest

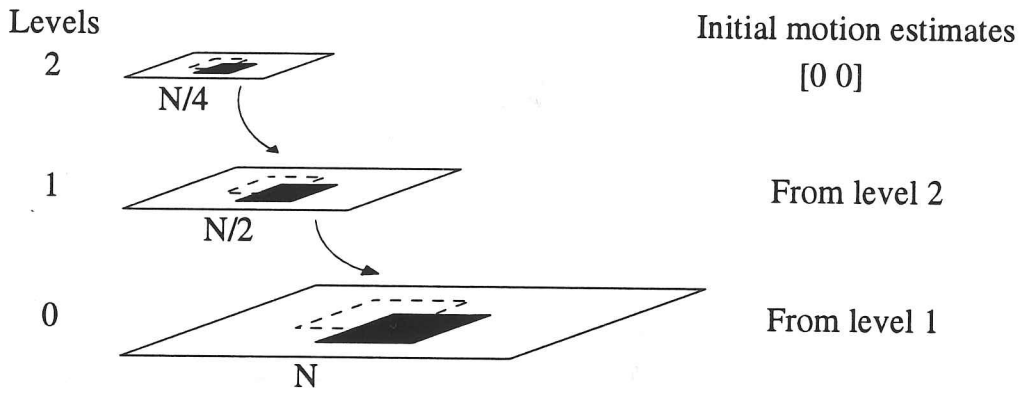

Figure 2.6: Multiresolution pyramid showing 3 level decomposition of an $N \times N$ frame. The dotted lines indicate the position of the object in the previous frame. Note this displacement is reduced at the same rate as the level dimensions.

level, regardless of the final vector estimate. Therefore, before assigning a motion vector to a region, Bierling verifies that it is in fact moving.

An alternative approach was used by Enkelmann[22] to improve the convergence of his pel–recursive algorithm. When it is detected that at a particular level, level l say, the algorithm does not converge quickly enough, the vectors are propagated up to the coarser level $l+1$. Estimation then proceeds at that level. He found that it was better to propagate just the update vectors between levels rather than the entire motion displacement. This bi–directional control strategy, as opposed to top down only, was found to give good results for real scenes.

In the interests of computation, this thesis employs only the top down approach with motion detection as implemented by Bierling. This is illustrated in figure 2.6.

2.5 Motion Prediction

A few authors, dealing with the gradient based motion estimators, have used the continuity of motion flow to generate an initial estimate for the iterative schemes. The phenomenon is easily understood by considering a region, at position $\mathbf{x}$ in image n say, moving with a constant velocity $\mathbf{v}$ pixels/frame. Suppose that using frames n and $n-1$, the velocity of this region was correctly estimated in frame n. Given the constant velocity assumption, the region at position $\mathbf{x}+\mathbf{v}$ in frame $n+1$ should also be moving at $\mathbf{v}$ pixels/frame. Hence, an initial guess for the velocity of that location in frame $n+1$ would be $\mathbf{v}$.

The initial estimate can be predicted both by temporal means, as just outlined, and also through some spatial support [38, 20]. This phenomenon arises directly from the Local Smooth Flow assumption outlined earlier. In practice, of course, objects will have some acceleration and so the temporal prediction of motion may not be accurate. One of the advantages of incorporating this scheme is that it enables the motion estimator to converge quickly, and can reduce the search

space required in BM for instance.

However, occlusion phenomena can adversely affect the motion prediction. Depending on the magnitude of the motion involved, this could mean abandoning the prediction over large areas of the image. Note also that at every scene change, there will be no available motion estimates and the algorithm will necessarily increase in computation at that point. This has interesting implications for real time applications. In such applications the motion estimator needs to generate a motion field for each frame in one frame interval. Therefore, even at a scene change, the algorithm would need to operate in one frame interval[12] starting with no previous estimate. If this is the case, then one could use the zero estimate at all frames. The advantage then, of using the motion prediction idea, would be one of more accurate motion estimation rather than reduction in computation required for an estimate.

Consider the multiresolution scheme discussed previously. Given that displacements are large, the multiresolution approach is appropriate. To use temporal motion prediction, initial estimates for the vectors at the smallest resolution image would need to be generated by propagating initial vectors up from level 0. However, at that level, all motion is quite small. Given that errors will arise in projecting vectors up the pyramid, and that the constant velocity assumption will be violated, it may well be that an initial motion estimate of $\mathbf{v} = 0$ is as useful as one projected up from level 0. Of course the alternative approach would be to use the multiresolution scheme at scene changes to reduce computation and help estimate large displacements, then depend upon temporal motion prediction to give useful estimates of motion thereafter, without using the multiresolution approach.

Some experimental work is necessary to verify how much is gained by using some motion prediction scheme, but this is outside the scope of this thesis. Instead, the motion estimators in this work use multiresolution schemes that have no 'motion memory' and use zero initial motion estimates at the top level of the pyramid. This keeps the motion estimation aspect of the work as simple as possible.

2.6 Summary

It has been stated that image sequence restoration is invariably related to motion estimation. Several motion estimation approaches have been reviewed. It has been pointed out that BM, although robust, is computationally expensive and gradient based approaches provide an attractive alternative. Several gradient based approaches have been outlined and it was noted that there is scope for an adaptive gradient based scheme that incorporates the work of Biemond [10], Martinez [63], Driessen [38] and Böröczky [12, 53]. However the simpler gradient based approaches, such as the WBME, are still not as robust as BM, which is also the easier algorithm

[12] Unless, of course enough frame buffer memory was available to hold the incoming frames. This can involve an impractical amount of memory considering a data rate of $768 \times 576 \times 25$ bytes/sec for PAL television.

to implement. For this reason, the work in this thesis primarily uses BM schemes.

Image sequence processing in general is hampered by the lack of a general model. It is possible to treat the simple flow equation 2.1 as a sequence model, but the effect of irradiance and 3D motion in general are then ignored. It has been stated that the development of 3D/perspective projection based models is fundamentally difficult. Instead it has been proposed to use the 3D AR model, previously presented in [91, 20], as a first approximation to a model of the image sequence. Further experimental work on the model is presented in Chapter 3 where a multiresolution adaptive 3D AR scheme is developed. Several interesting points arise about the nature of the displacement estimated from the model.

Finally, difficulties with large displacements have been highlighted. The approach by Bierling [11] appears to be the simpler and more computationally effective. For work on real television sequences, multiresolution schemes are essential.

Additional work on motion estimation is given in [106 - 109].

THREE DIMENSIONAL AUTOREGRESSIVE MODELLING FOR IMAGE SEQUENCES

In Chapter 2 various systems for motion estimation were reviewed. In particular, a model of the image sequence based on the 3D AR model was considered [20]. In that discussion the model coefficients were fixed for the entire sequence. Further, the implementation presented in [20] did not involve a multiresolution scheme, and as such it is difficult to use that scheme for motion estimation in a fast moving sequence.

Vaseghi and Rayner [95] have already introduced the use of the AR model for detection of impulsive noise in audio signals. They and Veldhuis [96], also show that the model can then be used for interpolation of the missing data. Veldhuis goes on to show that the 2D AR model is also effective in interpolating missing data in images.

In the light of these investigations, it is worthwhile to consider the usefulness of the 3D AR model as an image sequence model. In this chapter the model is compared with the WBME and integer accurate BM. Several improvements to the proposed scheme in [20] are made. The AR coefficients are made to adapt to the varying image information, and an adaptive scheme for estimating the motion parameter is presented which combines the efforts of Martinez [63] and Driessen et al. [38, 37, 39]. Some important points are highlighted with respect to the accuracy of motion estimation using the adaptive scheme and this eventually leads to a new adaptive multiresolution 3D AR algorithm for image sequence modelling.

Chapter 7 uses the model framework to detect and remove scratches in motion pictures.

3.1 The Model

It is important to acknowledge from the onset that the AR model is not necessarily *the* model of the image sequence. In fact, the choice of this structure is motivated more by computational reasons and its spatio–temporal nature than by any physical basis.

Simply put, the model tries to make the best prediction of a pel in the current frame based on a weighted linear combination of intensities at pels in a predefined support region. This support region may occupy pels in the current frame as well as ~~previous~~ future and past frames. The model was

already described in Chapter 2, and the equation is repeated below.

$$I(i,j,n) = \sum_{k=1}^{N} a_k I(i + q_k(x) + sx_{n,n+q_k(n)}, j + q_k(y) + sy_{n,n+q_k(n)}, n + q_k(n)) + \epsilon(i,j,n) \quad (3.1)$$

For the purposes of parameter estimation, the model is considered in its prediction mode. In this mode the model is reconfigured as a predictor of images in the sequence. The predicted intensity is as follows.

$$\hat{I}(i,j,n) = \sum_{k=1}^{N} a_k I(i + q_k(x) + sx_{n,n+q_k(n)}, j + q_k(y) + sy_{n,n+q_k(n)}, n + q_k(n)) \quad (3.2)$$

The task then becomes to choose the parameters in order to minimize some function of the prediction error, or residual,

$$\epsilon(i,j,n) = I(i,j,n) - \hat{I}(i,j,n) \quad (3.3)$$

The equation 3.3 is just a rearrangement of the model equation 3.1 with the emphasis placed on the prediction error, $\epsilon(i,j,n)$.

3.2 Parameter Estimation

As stated previously, the parameters which are required are the weights, a_k and the displacement $\mathbf{d}_{k,l}$. Because it is not possible to find a closed form solution for both the coefficients and the displacement at the same time, an iterative solution is proposed. The coefficients are estimated based on an initial guess for the displacement, this set of coefficients is used to generate an update to the initial guess for the displacement. The new displacement is then used to estimate a new set of coefficients and so on until the error is as low as required. To this end therefore, the following sections treat the coefficient estimation and displacement estimation separately. The model is then investigated with respect to artificial sequences and finally a few different algorithms are compared.

3.2.1 Estimation of the AR coefficients

It was decided, in the interest of computational load, to use a least squared estimate for the model coefficients in order to adapt the coefficients to the image function prior to motion estimation. The coefficients are chosen therefore, to minimize the square of the error $\epsilon()$, above. This leads to the Normal equations. The derivation is the same as the one dimensional case and the solution can be arrived at by invoking the principle of orthogonality. Solving the normal equations yields the model coefficients, a_k. Appendix A gives a more complete treatment, but the final set of equations to be solved is stated below.

$$\mathbf{Ca} = -\mathbf{k} \quad (3.4)$$

Here, $\mathbf{C}$ and $\mathbf{k}$ represent terms from the correlation function of the image sequence. $\mathbf{a}$ is the vector of model coefficients.

3.2.2 Estimating the Displacement

In order to gain an explicit relation for $\epsilon()$, in terms of $\mathbf{d}$, the approach used by Biemond [10] and Efstratiadis [20], was to expand the image function, $I()$, in the previous frames, as a Taylor series about the current displacement guess. This effectively linearizes the equation for $\epsilon()$ and allows a closed form estimate for $\mathbf{d}$ through a recursive update process. It is this iterative solution that is used for estimating $\mathbf{d}$ in this work. The reader is referred to Appendix C for a derivation and definition of the terms.

Recall that the displacement for the next iteration, $\mathbf{d}_{i+1}$ is found by updating the current displacement, $\mathbf{d}_i$, using $\mathbf{d}_{i+1} = \mathbf{d}_i + \hat{\mathbf{u}}_w$. The final solution for the update is

$$\hat{\mathbf{u}}_w = [\mathbf{G}_w^T \mathbf{G}_w + \mu \mathbf{I}]^{-1} \mathbf{G}_w^T \mathbf{z}_w \tag{3.5}$$

The case treated in this work is a causal model, so that the support region for the model does not have any taps in the next frame. Therefore, only one displacement parameter mapping the current frame into the previous one is considered. The terms in the solution equation are defined as below[1]. The equations constituting the matrix framework are assembled from observations of pixels in a block of size $N_1 \times N_2$ in the current frame. (See figure 2.3).

- $\mathbf{z}$, $(N_1 \times N_2 \times 1)$ is a column vector of prediction errors given the current displacement, as defined in 3.3, at all the points in the region used for estimation. Note that $\epsilon_o(\mathbf{x}_k, n)$ denotes the kth current observed prediction error at position $(\mathbf{x}_k, n)$ and not the actual model excitation $\epsilon(\mathbf{x}_k, n)$.

$$\mathbf{z}_w = \begin{bmatrix} \epsilon_0(\mathbf{x}_1, n) \\ \epsilon_0(\mathbf{x}_2, n) \\ \vdots \\ \epsilon_0(\mathbf{x}_{(N_1 N_2)}, n) \end{bmatrix} \tag{3.6}$$

- $\mathbf{G}_w$, $(N_1 \times N_2 \times 2)$ is a matrix of gradients at the support positions in the previous frame. In the following expression, $I_n()$ refers to the nth pixel in the block in the current frame used to set up the motion equations and $D()$ is a function representing the necessary offset terms corresponding to support vector $\mathbf{q}_k$. The model is assumed to be purely temporally causal and has no taps in the next frame.

$$\mathbf{G}_w = \begin{bmatrix} \sum\sum a_k \frac{\partial I_1(\mathbf{D}(f,k,n))}{\partial x}) & \sum\sum a_k \frac{\partial I_1(\mathbf{D}(f,k,n))}{\partial y} \\ \sum\sum a_k \frac{\partial I_2(\mathbf{D}(f,k,n))}{\partial x}) & \sum\sum a_k \frac{\partial I_2(\mathbf{D}(f,k,n))}{\partial y} \\ \vdots & \vdots \\ \sum\sum a_k \frac{\partial I_{(N_1 N_2)}(\mathbf{D}(f,k,n))}{\partial x}) & \sum\sum a_k \frac{\partial I_{(N_1 N_2)}(\mathbf{D}(f,k,n))}{\partial y} \end{bmatrix} \tag{3.7}$$

[1] See Appendix C

- $\mathbf{u}$ is the (2×1) vector of updates defined as

$$\mathbf{u} = \begin{bmatrix} u_{n,n-1}(x) \\ u_{n,n-1}(y) \end{bmatrix} \tag{3.8}$$

- μ is the regularizing parameter.

$$\mu = \frac{\sigma_{\epsilon\epsilon}^2 + \sigma_{\nu\nu}^2 \sum_{k=1}^{N} a_k^2}{\sigma_{uu}^2} \tag{3.9}$$

Where $\sigma_{\epsilon\epsilon}^2$ is the variance of the actual model error, $\sigma_{\nu\nu}^2$ is the variance of the truncation noise, and σ_{uu}^2 is the variance of the estimated update components.

In practice, neither $\sigma_{\epsilon\epsilon}^2$ nor $\sigma_{\nu\nu}^2$ are available. The standard approach is to assume some constant value for μ or to use some adaptive scheme. Such an adaptive scheme is presented later in this chapter.

The algorithm for finding the displacement is recursive and some criterion must be used for halting the process when convergence is sufficient. Several criteria are normally used in conjunction and they are as follows.

- Threshold the estimated update magnitude. *if* $|\mathbf{u}| < t_u$ *then halt.*
- Threshold the MSE. *if* $|\mathbf{z}| < t_z$ *then halt.*
- Limit the number of iterations to ensure the process does terminate even if convergence satisfying the above two criteria are not satisfied.

3.3 Experiments on Artificial Autoregressive sequences

The previous arguments have introduced a number of assumptions. These are

- The image sequence obeys an autoregressive model.
- The model can be linearized effectively using the Taylor expansion.
- The higher order terms in the expansion can be considered to have the same effect as adding white noise to the linearized equations.
- That the components of the vector estimate are uncorrelated.

It is educational to consider the effects of these assumptions in a controlled environment where the model is known. To this end it is first necessary to generate a sequence which obeys a three dimensional autoregressive model.

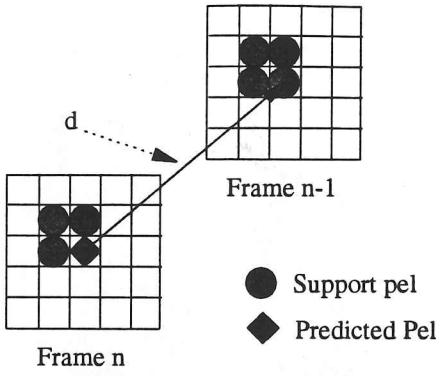

Figure 3.1: The structure of the AR system used for the synthetic sequences.

3.3.1 Generating Three Dimensional Autoregressive Sequences

To create these sequences, the prediction equations must be viewed from a different perspective. The observed intensity can be considered to be the result of an IIR system driven by an 'innovations' sequence, $\epsilon(\mathbf{x}, n)$. The z-transform of the denominator of the IIR system transfer function is the z-transform of the prediction equation 3.3. Therefore the standard method of creating autoregressive signals is to drive some IIR filter with a Gaussian noise sequence.

In the multidimensional case it is more natural to consider non-causal models than causal ones. In this case of image sequence modelling, there does not exist any notion of causality in the spatial domain other than the direction of processing and the z–transform of the system. However, the generation of non–causal 3D AR sequences is complicated by the fact that the driving function required is not white noise as is the case with causal systems[2] (See Appendix B).

There is the further complication that stability criteria for multidimensional IIR filters are extremely convoluted and difficult to use for general filter design [55]. However, if the IIR filter is separable then the stability criteria involve only consideration of the n one dimensional filters involved. To simplify matters, a specific case is considered and this is used in the ensuing experiments. The model used is a purely causal one defined by 3.10 and illustrated in fig 3.1.

$$\begin{aligned} I(i,j,n) &= a_1 I(i-1,j,n) + a_2 I(i-1,j-1,n) + a_3 I(i,j-1,n) + a_4 I(i+\mathrm{sx}, j+\mathrm{sy}, n-1) \\ &+ a_5 I(i-1+\mathrm{sx}, j+\mathrm{sy}, n-1) + a_6 I(i-1+\mathrm{sx}, j-1+\mathrm{sy}, n-1) \\ &+ a_7 I(i+\mathrm{sx}, j-1+\mathrm{sy}, n-1) + \epsilon(i,j,n) \end{aligned} \tag{3.10}$$

Taking the z–transform, the model equation becomes

$$\begin{aligned} I(z_1,z_2,z_3) &= I(z_1,z_2,z_3)[a_1 z_1^{-1} + a_2 z_1^{-1} z_2^{-1} + a_3 z_2^{-1} + a_4 z_1^{sx} z_2^{sy} z_3^{-1} \\ &+ a_5 z_1^{sx-1} z_2^{sy} z_3^{-1} + a_6 z_1^{sx-1} z_2^{sy-1} z_3^{-1} + a_7 z_1^{sx} z_2^{sy-1} z_3^{-1}] \\ &+ E(z_1,z_2,z_3) \end{aligned} \tag{3.11}$$

[2]This is not unique for multidimensional systems, it is the same for the one dimensional case.

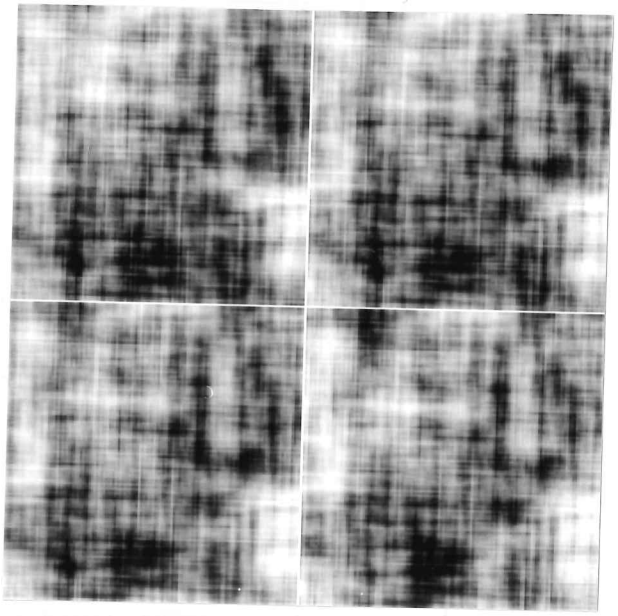

Figure 3.2: Top right to bottom left: A sequence of frames similar to AR95 but no motion. The intensities have been scaled to 8 bits.

Defining

$$H(z_1, z_2, z_3) = \frac{I(z_1, z_2, z_3)}{E(z_1, z_2, z_3)} \tag{3.12}$$

makes it easier to realize that the original predictor equation, 3.2 , can be rearranged so that the image sequence $I(i, j, n)$ is the output of a system $H(z_1, z_2, z_3)$ driven by the function $E(z_1, z_2, z_3)$. The model transfer function is seen to be an all pole model. For this causal case the output is a Markov autoregressive sequence if the driving function is Gaussian white noise.

In order to generate an autoregressive sequence one must choose the coefficients **a** so that $H(z_1, z_2, z_3)$ is stable. This is much simplified if $H()$ is made separable to enable the function to be implemented as the cascade of 3 one dimensional filters operating in orthogonal directions.

$$H(z_1, z_2, z_3) = \frac{1}{(1 - r_1 z_1^{-1})(1 - r_2 z_2^{-1})(1 - r_3 z_1^{sx} z_2^{sy} z_3^{-1})} \tag{3.13}$$

From the above equation therefore, $H()$ is stable provided that the three values, r_1, r_2, r_3, all have a magnitude less than 1. To simplify further references to these values they will be called poles, even though strictly speaking, the third system in cascade (involving z_3) is a multidimensional one and so has some sort of pole surface.

Two sequences were used for the experiments. They were generated by driving a 3D AR system with noise until 910 frames of 900×900 resolution[3] were produced. The first 900 were regarded as transient and the experiments used the bottom right hand corner (resolution 32×32) portion of the last 10 frames. The two sequences, AR95 and AR99, had poles at $r_1 = 0.95, r_2 = 0.95, r_3 = 0.95$ and $r_1 = 0.99, r_2 = 0.99, r_3 = 0.99$. The variance of the white noise used was 1.0 and in both cases the motion was $sx = -1.0$, $sy = -1.0$. This small motion was used in order to keep the spatial transients in the generation process to a minimum. The resolution

[3]This unusual resolution was forced due to memory constraints in the machine used.

Model	Coefficients						
	a_1	a_2	a_3	a_4	a_5	a_6	a_7
AR95	-0.95	0.9025	-0.95	-0.95	0.9025	-0.8574	0.9025
AR99	-0.99	0.9801	-0.99	-0.99	0.9801	-0.9703	0.9801

Table 3.1: The coefficients used for generating 3D AR signals.

of these frames is too small to include a useful print here, instead, fig 3.2 shows 4 frames (of resolution 128×128) from a sequence similar to AR95 but with no motion. It is hoped that these prints will give the reader a better appreciation for these synthetic sequences.

Once the poles of the AR system have been chosen and the AR sequence generated, the optimum prediction coefficients, **a** can be found by equating the denominator of the system function 3.13 to the expression for the system function in terms of the model coefficients, equation 3.12. The coefficients for the two processes examined are given in table 3.1.

3.3.2 The experiments

Having created two sequences for which the model is known, the task is now to examine the behavior of the displacement estimation algorithm. For each of the two sequences, displacements were estimated for 9 out of the 10 frames. To estimate the displacement at a pel, 49 equations were set up using a block of 7×7 pixels centred on the tested pel. The known coefficients for the model were used in setting up the equations. A separate displacement parameter was estimated for each pixel in each frame. To evaluate pixel intensities at fractional positions, bilinear interpolation was used. Two sets of experiments were carried out with μ fixed at 100.0 and 500.0. A border of 5 pixels around the frame was ignored in the results to reduce edge effects. No thresholds were used to test for convergence since the investigations themselves observed the convergence rate with increasing iterations.

3.3.3 Results and Observations

With respect to the prediction error, figure 3.3, shows the Percentage MSE over the 9 frames from both sequences. This measure is the percentage ratio between the Mean Squared prediction error across the entire frame and the Mean Squared intensity across the original frame. It is defined below.

$$\text{PMSE} = 100.0 \frac{\sum_{\mathbf{x}} \epsilon_o(\mathbf{x})^2}{I(\mathbf{x})^2} \tag{3.14}$$

The PMSE is about 10 times smaller for the AR99 sequence than for the AR95 sequence.

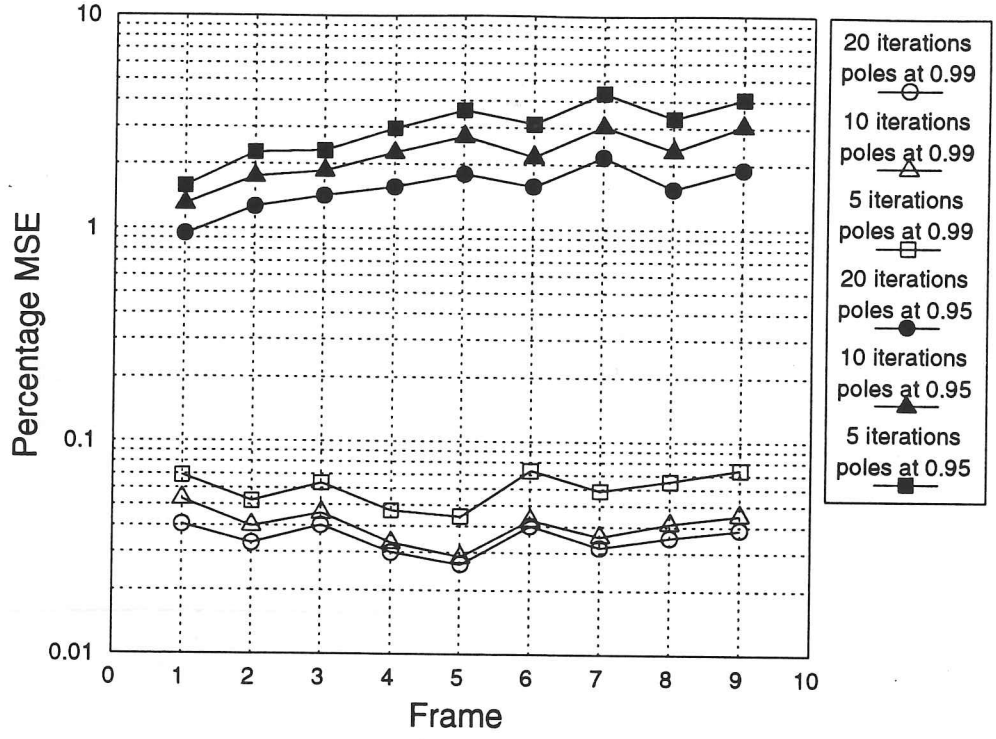

Figure 3.3: Comparison of percentage mean square prediction error.

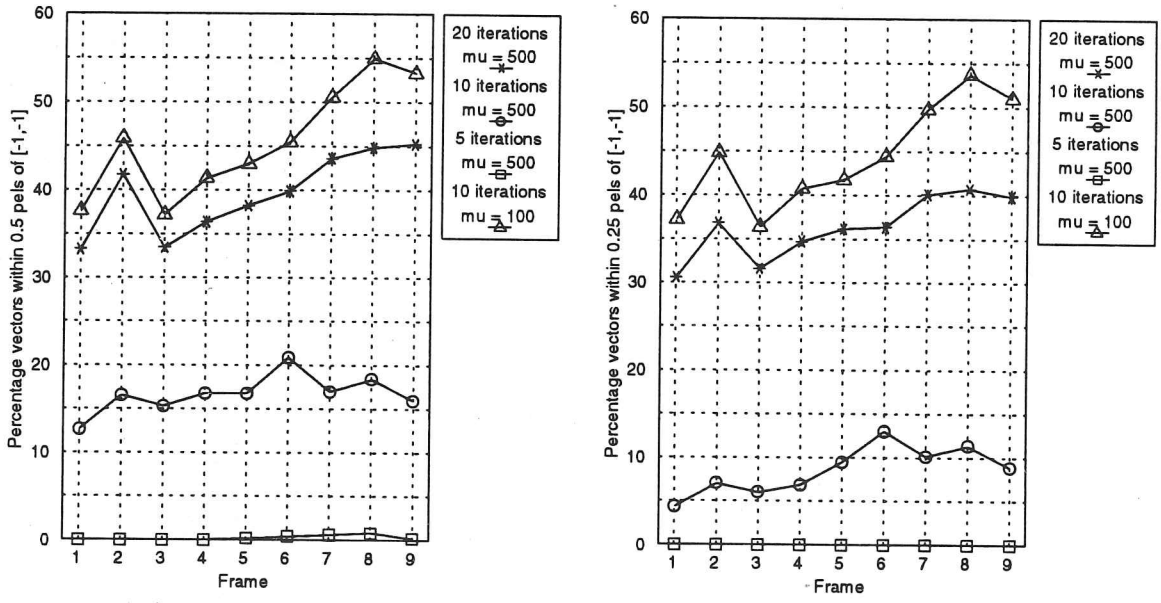

Figure 3.4: Accuracy of the algorithm with AR95.

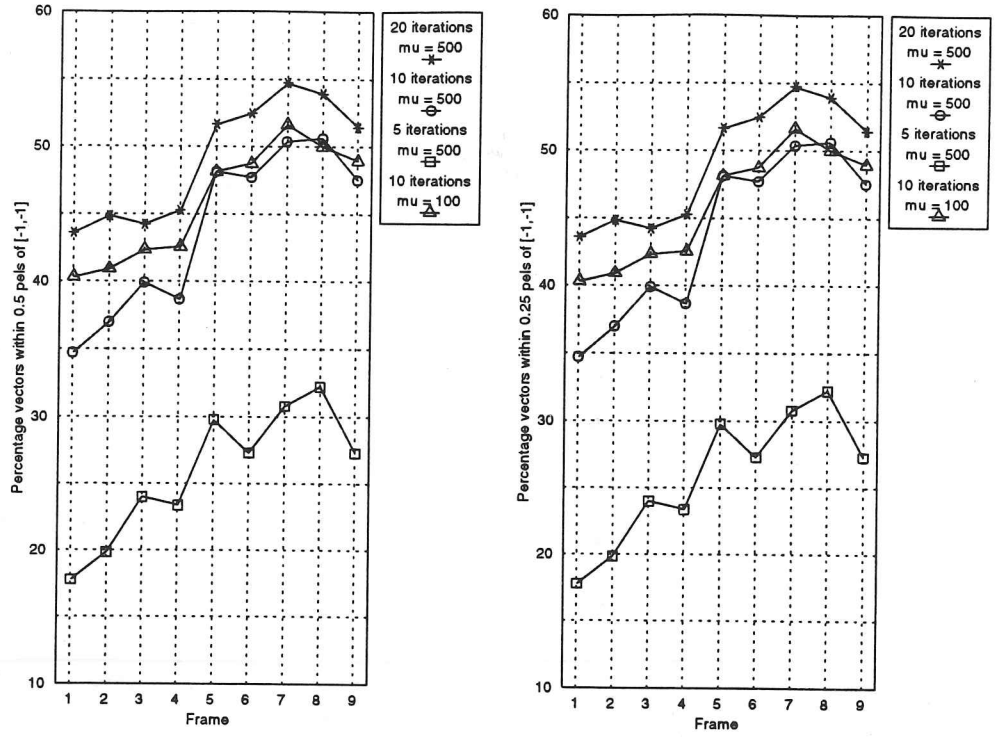

Figure 3.5: Accuracy of the algorithm with AR99.

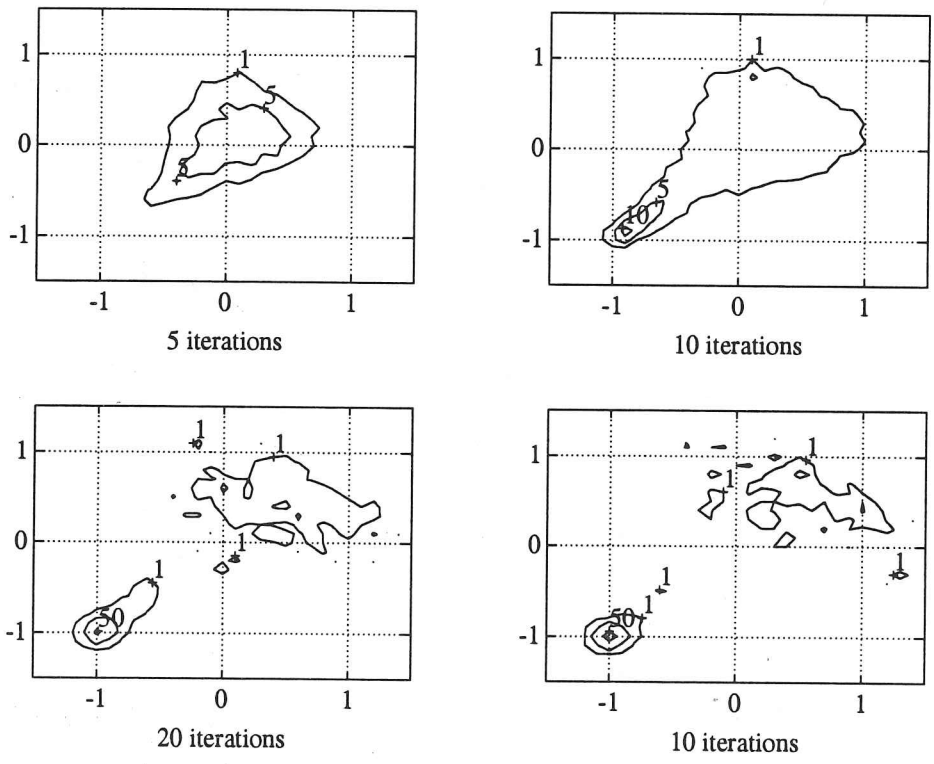

Figure 3.6: Distributions of the estimated vectors at 3 stages in the motion estimation process, $\mu = 500$. Poles at 0.95 (AR95). The bottom right plot shows the distribution for $\mu = 100$ at 10 iterations.

Figures 3.4, 3.5 show the percentage number of estimated displacements that were within radii of 0.25 and 0.5 pels of the correct displacement. These results show the different performance between the two sequences with respect to displacement estimation. Figure 3.6 is included to show the typical behavior of the algorithm as the iterations progress. The plots in the figure show averaged 2D histograms of the displacement estimates over the entire frame[4] at a particular iteration. Therefore, as the iterations progress the distribution 'flows' away from the initial cluster at [0.0, 0.0] to eventually converge in a peak at [−1.0, −1.0], the actual displacement. The plots give an overall view of the progression of the method. They also point out that although the number of vectors correctly estimated is just 50.0%, as shown by figure 3.4, the distribution does show a dominant peak at the correct displacement after 20 iterations. The remaining vector estimates do not converge and are distributed randomly near [0.0, 0.0].

The following observations can be made

- The motion estimation algorithm is approximately 50% accurate.
- Accuracy is increased with poles closer to the unit circles. The accuracy of the estimation algorithm with sequence AR95 is lower by about 5% at 20 iterations than the estimation with sequence AR99.
- With increasing iterations there is a corresponding increase in accuracy, but there is an upper bound.
- μ acts as a damper on the estimation system. Small μ yields higher accuracy at smaller numbers of iterations. Compare the right top and bottom plots in figure 3.11.

3.3.4 Discussion

The observations made above seem to suggest that even with a known system the algorithm is not satisfactory. However attention must be drawn to the fact that even with a given model, a Taylor series approximation is made to linearize the equations. The approximation requires essentially that the higher order terms in the expansion be negligible or can be modelled as a white noise error source. The prints of the sequences shown before indicate that rapid changes in the intensity of each frame are common. This implies that the higher order Taylor series terms can be appreciable. Therefore, contrary to the initial impression, it is understandable that the motion estimator is not quite as good as expected. Of course some of the discrepancy must be due to the random nature of the signal itself. It would be better perhaps to put a bound on the variance of the estimated vectors given that the driving function is random, and observe how close the estimator gets to that bound. Nevertheless, some perspective must be maintained, and it is clear from the prints that this particular driving function does not necessarily result in the type of image features encountered in a real case.

[4]The measured distribution over each frame is averaged over all 9 frames to give the distribution that is plotted.

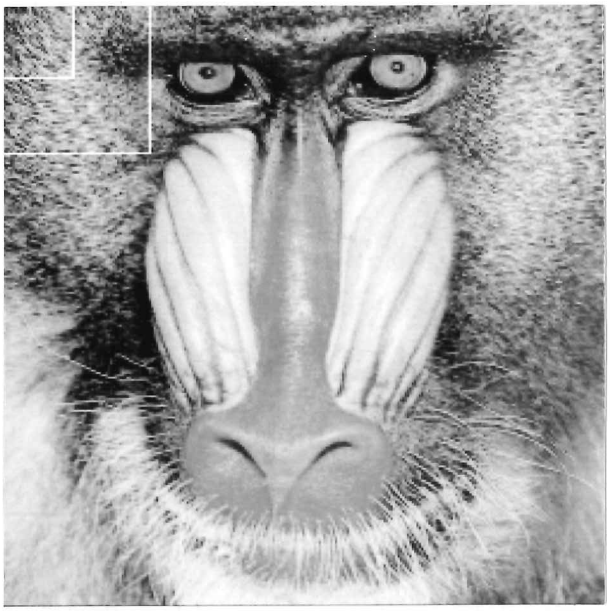

Figure 3.7: The Baboon image. The upper left borders delineate two quadrants used for experiments. The larger quadrant (64 × 64) is used for the multiresolution experiments.

The fact that the performance with poles at 0.99 is better can be attributed to the more 'resonant' nature of the generated signal frequency spectrum. This is a well known phenomenon with regard to generator/estimator experiments like these, and it is considered in [87]. The resulting sequence from the generator is better described by the generator coefficients if the poles are closer to the unit circle.

Finally, the behavior of the estimator with increasing iterations is expected. As the number of iterations is increased more of the estimated displacements can converge on a solution. The degree of this convergence is proportional to the damping factor μ. When μ is high the estimator is heavily damped and necessarily takes more iterations to converge.

3.4 Experiments with a more realistic sequence

To better assess the performance of the 3D AR model, sequences which obey both an AR model and a first order Taylor series expansion must be found. Unfortunately, these requirements conflict. A sequence obeying a first order Taylor series expansion implies a ~~smoothness~~ uniformity that can be obtained when the poles of the generating system are far from the unit circle. When the poles are far from the unit circle however, the coefficients of the generator system do not describe the generated image well. Given a certain number of trials of generator/estimator experiments, there is a large variance in the estimated coefficients with the same generating system. This does shed some doubt as to the usefulness of experiments such as these and instead this next set of experiments tries to represent a half way point between the real world case and the completely artificial case. Driessen et al. [38], in their paper on adaptation of the Biemond motion estimator, used the upper left hand corner of the Baboon image, (see Fig 3.7) to generate a sequence for investigation. Their motivation was that this image was a good test of the estimator since this

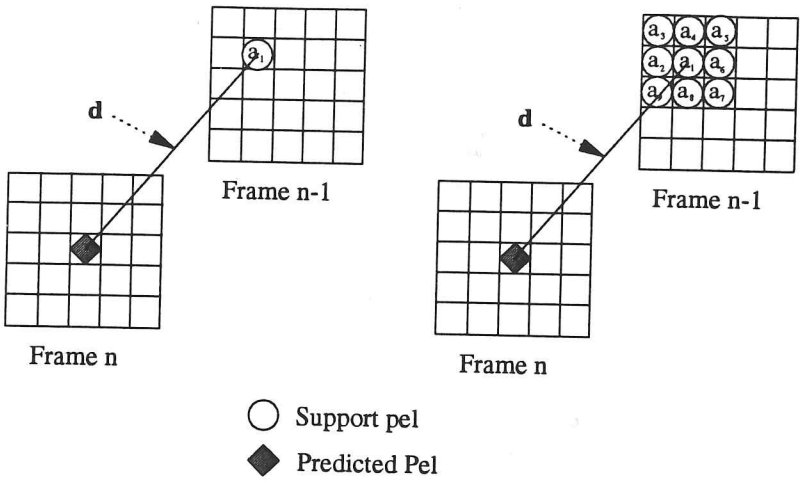

Figure 3.8: Two 3D AR models used in experiments. Left: AR1, Right: AR9.

highly textured image would result in a noisy **G** matrix. This same image was used to generate a test sequence for this algorithm. A window of 32 × 32 pixels was extracted from the upper left hand corner of the Baboon image to create one frame. This window was displaced by [-2,-2] pixels from frame to frame to create the desired sequence. The sequence was 10 frames long, the first frame of which is delineated by a white frame in fig 3.7.

In his work on 2D interpolation of images, Veldhuis [96] reported that a much better prediction was gained in real images by using a model as stated in equation 3.15 below.

$$\begin{aligned} I(i,j) - (p_1 + p_2 i + p_3 j) &= \sum_{k=1}^{N} a_k [I(i + q_k(x), j + q_k(y)) - (p_1 + p_2 i + p_3 j)] \\ &\quad + \epsilon(i,j) \end{aligned} \tag{3.15}$$

The coefficients, p_k, represent a 2D plane. This form dealt effectively with non–stationarity in the image blocks used. To linearize the parameter estimation problem, the plane coefficients were chosen on a least squared basis prior to the AR modelling. Therefore, the coefficients, p_k were chosen to minimize the following expression over the chosen image block.

$$[I(i,j) - (p_1 + p_2 i + p_3 j)]^2 \tag{3.16}$$

In this work, an extension of that idea is used for the sequences with real data, and a 3D trend replaces the 2D one of Veldhuis as shown in 3.17.

$$\begin{aligned} I(i,j,n) - (p_1 + p_2 i + p_3 j + p_4 k) &= \\ &\sum_{k=1}^{N} a_k [I(i + q_k(x) + sx_{n,n+q_k(n)}, j + q_k(y) + sy_{n,n+q_x(n)}, n + q_k(n)) \\ &-(p_1 + p_2 i + p_3 j + p_4 k)] \\ &+\epsilon(i,j,n) \end{aligned} \tag{3.17}$$

This form is equivalent to subtracting a 3D trend from the sequence data, $I(i,j,n)$ prior to the modelling step. The trend acts as a normalizing term and this subtraction is implemented prior

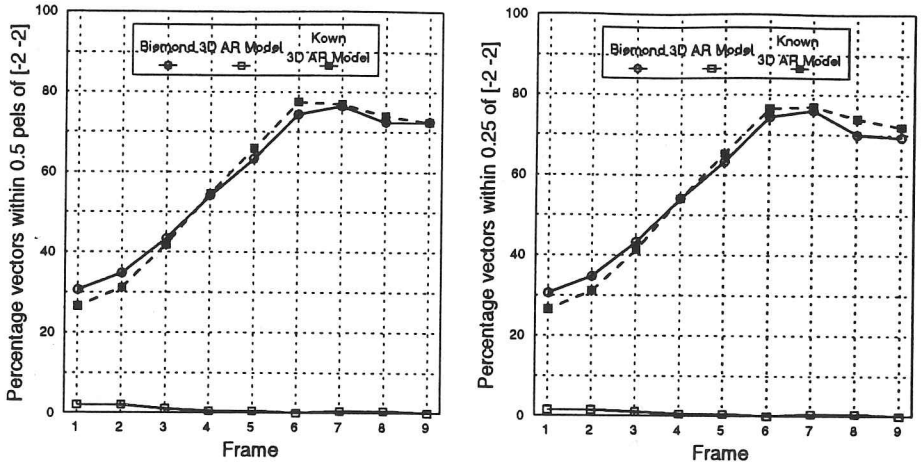

Figure 3.9: The accuracy of the motion estimation for various algorithms with respect to the Baboon sequence.

to modelling. The normalizing step does not affect the form of any of the equations and the trend is replaced after processing to yield a usable prediction.

3.4.1 Experiments.

In working with the Baboon test sequence it is possible to compare the performance of the AR algorithm with the algorithm developed by Biemond et al [10]. However, the AR model for the case of this translational sequence is not known. In order to overcome this problem, the coefficients were determined by solving the Normal equations at the correct, known displacement and then using those coefficients as required at each pixel. The following experiments were then performed,

1. The known coefficients were used in the estimation of the displacement $\mathbf{d}$, starting the iterations at $\mathbf{d} = [0.0,\ 0.0]$.

2. The iterations begin with $\mathbf{d} = [0.0,\ 0.0]$ and for each iteration the coefficients are estimated by solving the Normal Equations at the current displacement. Then that displacement is updated by solving for $\mathbf{d}$ using the previously estimated coefficients.

For all experiments, $\mu = 100.0$, blocksize $= 13 \times 13$. The algorithm was run for 20 iterations at all points considered. No threshold was set on the update magnitude nor the mean squared prediction error to halt the process before the maximum number of iterations were reached. The AR9 model, illustrated in figure 3.8, was used. The PIMSE (defined below) was calculated as well as the percentage of correct vectors estimated. These measures were used to compare the various approaches.

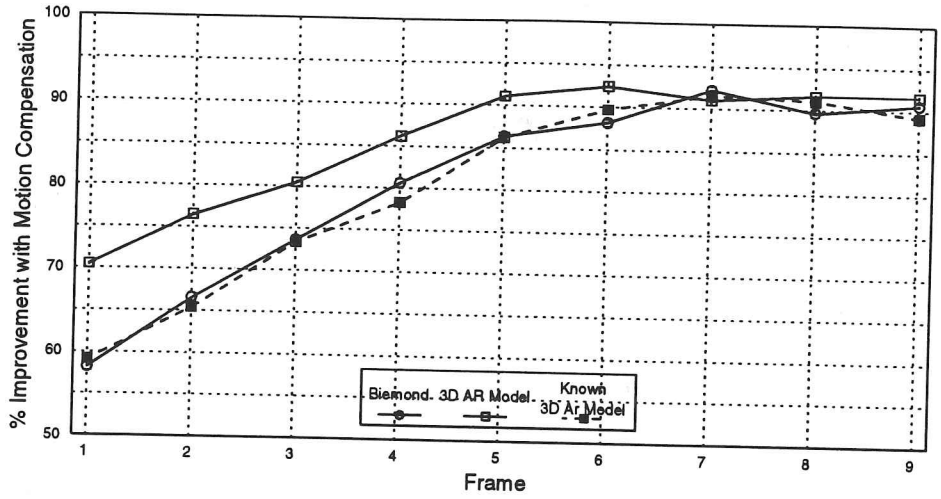

Figure 3.10: Comparing the percentage improvement in mean squared prediction error.

The Percentage Improvement in Mean Squared Error (PIMSE) is defined as follows.

$$\text{PIMSE} = \frac{\text{FDMSE} - \text{DFDMSE}}{\text{FDMSE}} \tag{3.18}$$

FDMSE is the mean squared frame difference without motion compensation and DFDMSE is the mean squared prediction error with motion compensation. For the Biemond algorithm, the DFDMSE is the mean squared displaced frame difference given the estimated vectors. Therefore the PIMSE is a measure of how much better the prediction error is with motion compensation when compared to the difference between frames. 100% PIMSE means that there is no error in predicting the next frame from the last. 0% PIMSE means that the prediction of the next frame using the considered compensation algorithm is just as good as using the previous frame itself as a prediction of the current frame, hence there is no advantage in using the algorithm at all.

3.4.2 Results and Discussion

It is clear that for this translational sequence, with integer pixel displacement, the optimal AR model has a one pixel support in the previous frame with associated coefficient value -1.0. This is so in practice. When the 'true' model coefficients are estimated using the Normal Equations at the correct displacement, both the models in fig 3.8 assign a value of -1.0 to a_1, and the rest are set at 0.0. This will always occur since this assignment of a_1 will always yield the minimum squared prediction error at the correct displacement. Therefore, the Biemond and AR algorithms must perform identically in the experiments given the 'true' AR coefficients. Figures 3.9, 3.10 illustrate this point in that the curves for both the AR algorithm with known coefficients and the Biemond algorithm are almost identical.

What is surprising is the discrepancy that arises for the case where the AR coefficients are not known and so are estimated at each iteration using the current value for $\mathbf{d}$. This is a more realistic assessment of performance. Figure 3.10, which shows the PIMSE using the AR9 model

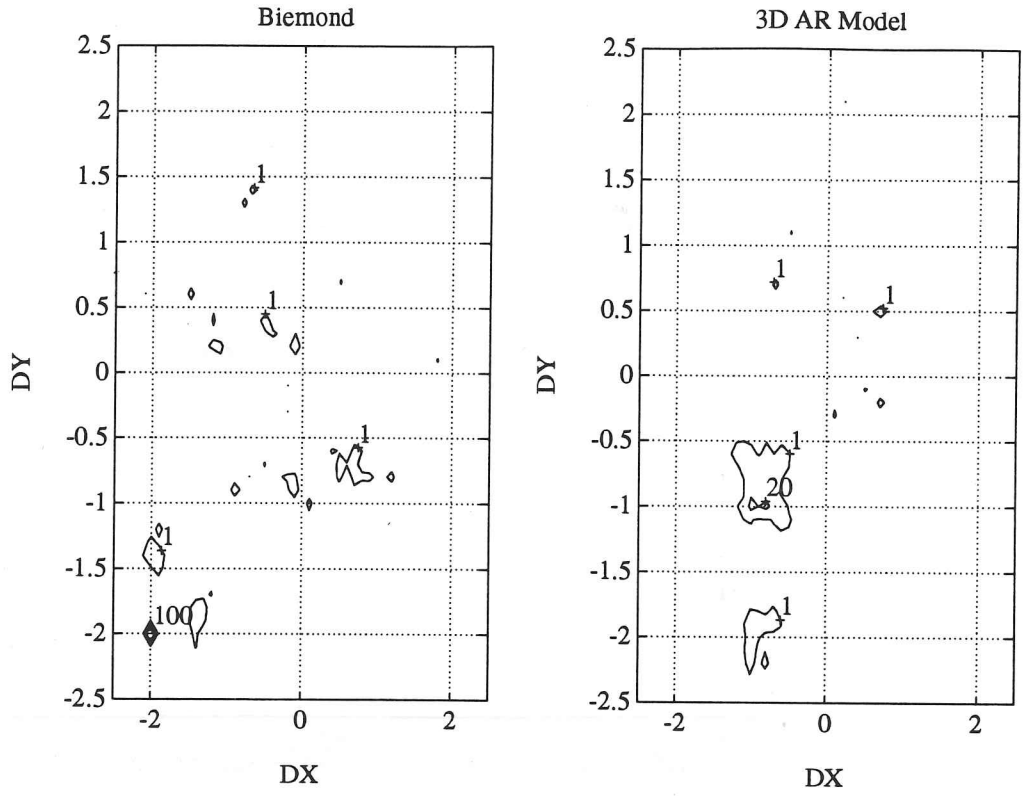

Figure 3.11: Comparing the distributions of the estimated vectors for Baboon using the AR9 model.

compared with the Biemond approach, indicates that the AR9 model gives a better prediction than the Biemond approach. However, as far as the accuracy of the motion estimate is concerned, figure 3.9 indicates the opposite result. The accuracy of the Biemond approach surpasses the AR model result. The experiment highlights the important point that the value for **d** given by the AR model does not necessarily reflect the true motion of the scene. This is not unreasonable. The major influence in this observation is the change in the coefficients as the iterative process proceeds.

When the coefficients are estimated using the current estimate for **d** the Normal Equations adjust the values to minimize the expected value of the squared prediction error. This implies that the PIMSE is maximized at each iteration. In contrast, the estimation of the motion **d**, tries to reduce the expected value of the error with respect to the vector **d**. This is done using the observed image gradients which do not necessarily indicate the correct direction in which to update the current vector. The coefficient adjustment therefore serves as a powerful method of compensating for errors in the vector estimation process. Furthermore, in this case (for AR9), once the iterations have brought **d** to within a 1 pixel radius of the correct displacement, the spatial support of the model overlaps the pixel at the actual displacement. The coefficient nearest to that pixel (in this case a_3 can coincide exactly with that location) can take on a value of -1.0 and the others forced toward 0.0 to allow a perfect prediction. Of course this is not exactly the case in practice and figure 3.11 shows a two dimensional histogram of the estimated vectors estimated using both algorithms. The histograms for each frame were averaged to give this result.

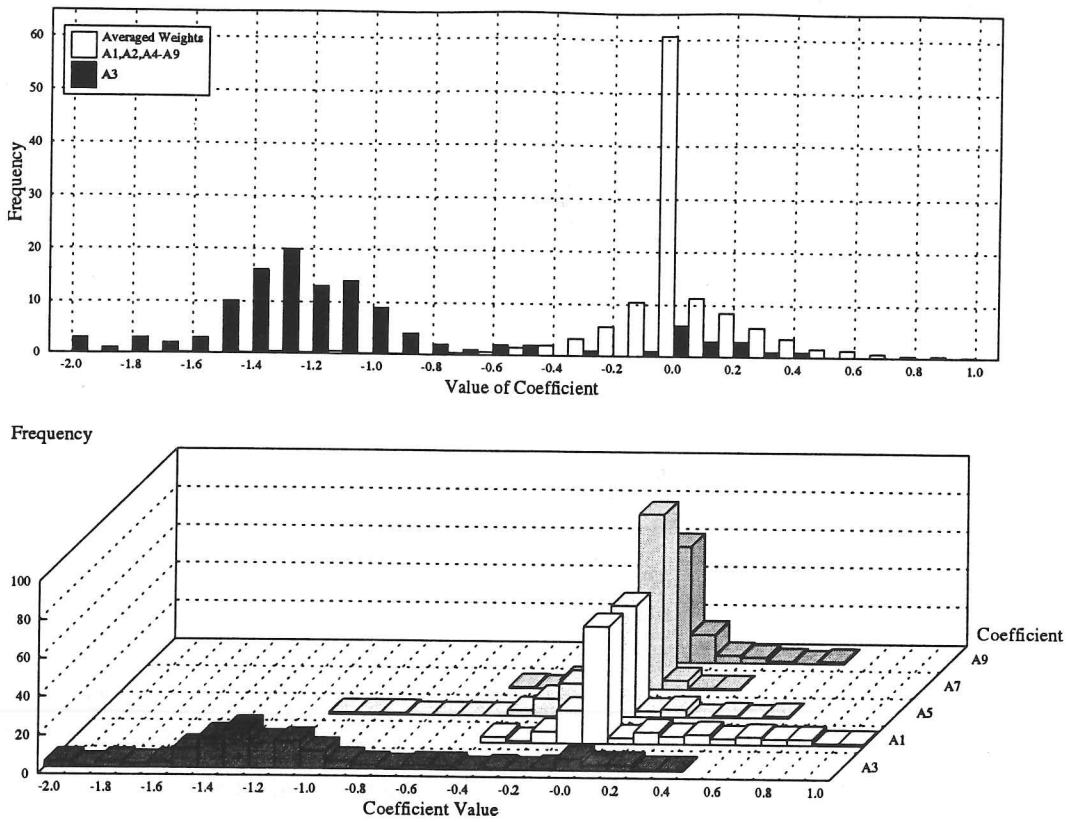

Figure 3.12: Distributions of model coefficients after termination of iterative AR process. Top : Comparing the averaged distributions of $a_1, a_2, a_4..a_9$ with a_3. Bottom : Distributions of selected coefficients.

The AR case shows a large cluster about $[-1, -1]$ whereas the case for the Biemond estimator shows a large cluster around the actual displacement $[-2, -2]$.

It is interesting to observe the values of the model coefficients after the final iteration. Figure 3.12 was produced by considering the 9 coefficients at each pixel in an 11×11 block at the centre of each 32×32 frame. The distribution of each coefficient was averaged over all the frames. The three dimensional bar chart in the bottom half of the figure shows the distributions of 5 of the 9 coefficients. Note that a_3 is different from any other coefficient and takes on values between -1.0 and -1.5. The plot at the top of the figure contrasts the distribution of a_3 with that of the averaged distribution of the other 8 coefficients. It is clear that all but a_3 are distributed about 0.0. This reinforces the statements made above in that when $\mathbf{d} = [-1.0 - 1.0]$, the support pel corresponding to a_3 overlaps the correctly displaced position and the corresponding correlation coefficient is high enough to cause $a_3 = 1.0$ and $a_1, a_2, a_4..a_9$ to be near 0.0. The resulting prediction error becomes small and the update vector produced is similarly diminished. The algorithm converges at that point.

It would appear that in order to get a good prediction, estimating the coefficients at each iteration is worthwhile. In order to get accurate motion estimation a model with a small spatial support must be used. To combine these two benefits it would seem reasonable to use an AR1

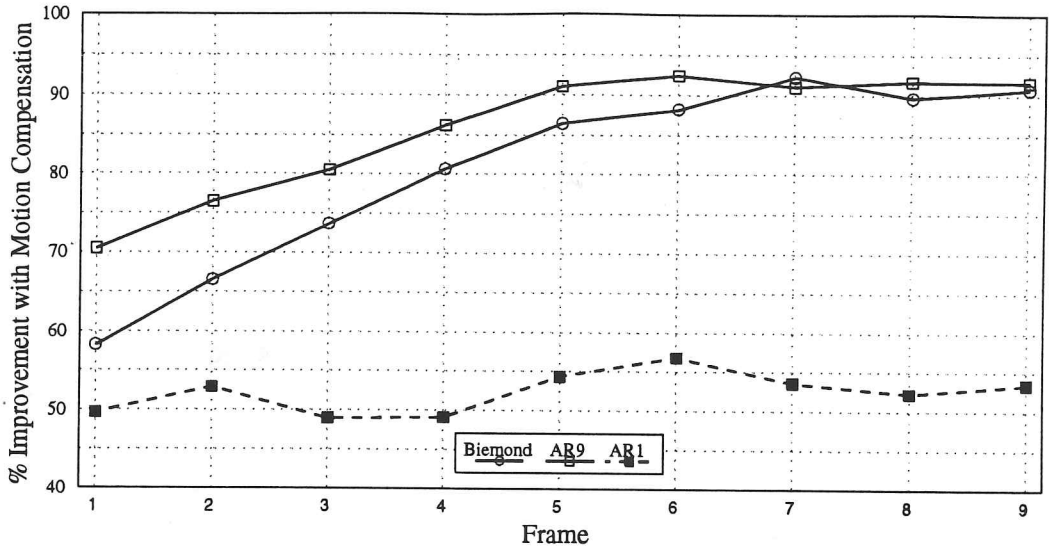

Figure 3.13: Comparing performance when using a model with a small spatial support.

model and update the coefficient at each iteration. This is not necessarily the case. Consider the small spatial support of model AR1 in figure 3.8. Depending on the image and the motion involved, there may not be a high correlation between the pixels at a location in the current frame, n, and the corresponding position in frame $n-1$. This is the situation if the iterations start with an initial zero displacement estimate. Of course, this can occur regardless of the support for the model but the effects are exaggerated if the spatial support is limited as in this case. When the Normal equations are solved with this initial value for $\mathbf{d}$ the low correlation will cause coefficient a_1 to be assigned a very low value. This will weight the elements of the gradient matrix toward zero causing the solution for the motion update to be ill conditioned. The algorithm will not converge on the correct displacement and a good prediction would not be possible. Alternatively, if the image information was such that the initial displacement guess allowed for some useful correlation between frames at the start of the iterations then the situation would be favourable.

Figure 3.13 illustrates this point with respect to the current experiments. It shows that the PIMSE with model AR1, with support as in figure 3.8, is much worse[5] than with the 9 point AR model. Understandably, the accuracy of the vectors is increased from almost 0% with the AR9 model to 10% with this model (this is not illustrated), but the improvement is insufficient to compensate for the loss of spatial support which in this case yields the low PIMSE. To counteract the ill conditioning caused by a low weighting of the gradient terms, μ can be increased. For this model, increasing μ to 500 results in behavior which is almost identical to the Biemond estimator, as expected.

The following points make clear the lessons of the preceding paragraphs.

[5]Recall that the Biemond estimator uses a model identical to AR1. The difference in performance is due purely to the process of estimating the coefficient at each iteration. In the Biemond technique, this coefficient is fixed at -1.0.

1. The general 3D AR model of the image sequence has a motion estimation accuracy equal to the extent of its spatial support in the previous frames. The model AR9, has a motion estimation error of ± 1.0 pixel. Algorithms involving models with a one pixel support, such as the Biemond algorithm, are potentially arbitrarily accurate since they can only minimize the expected error if the estimated displacement coincides with the actual displacement.

2. Estimating the coefficients at each iteration based on the current motion estimate is likely to yield useful improvements over other standard gradient based estimators only if the current vector estimate is close to the actual displacement or a large spatial support is used. 'Close' depending on the correlation structure of the image information itself.

These experiments illustrate clearly that the 3D AR model does not necessarily give an accurate estimate for motion. Nevertheless, it can give a better prediction than the Biemond algorithm, and other 'translation only' algorithms. When presented with a textured image sequence the 3D AR model is able to adapt not only to motion but also to the nature of the image contents as well.

3.5 Real Sequences

The previous experiments have been able to highlight, in controlled environments, the shortcomings of 3D AR modelling. The best test as always is with real world data. Therefore, the Biemond algorithm and the 3D AR algorithm were compared by observing the improvement in motion compensation over a a few frames from two real sequences. These were as follows

1. The Miss America Sequence: $256 \times 256 \times 10$ frames.

2. The Cheers sequence: $256 \times 256 \times 10$ frames.

The Cheers sequence is not a standard video sequence. It consists of several objects moving in different directions. It is a short clip from the television series 'Cheers'. It is included in the video supplement to this thesis. For all three sequences, μ was fixed at 10.0 and the iterations at each pixel considered always began at zero displacement. The motion estimation equations were set up using the 7×7 square at the centre of the block used for estimating the AR model coefficients. The iterative process was halted when one of the following criteria was satisfied,

1. The number of iterations > 10.

2. The mean square prediction error (MSE) < 1.0

3. The size of the update vector < 0.01

4. The MSE at the current iteration $>$ the MSE at the previous iteration.

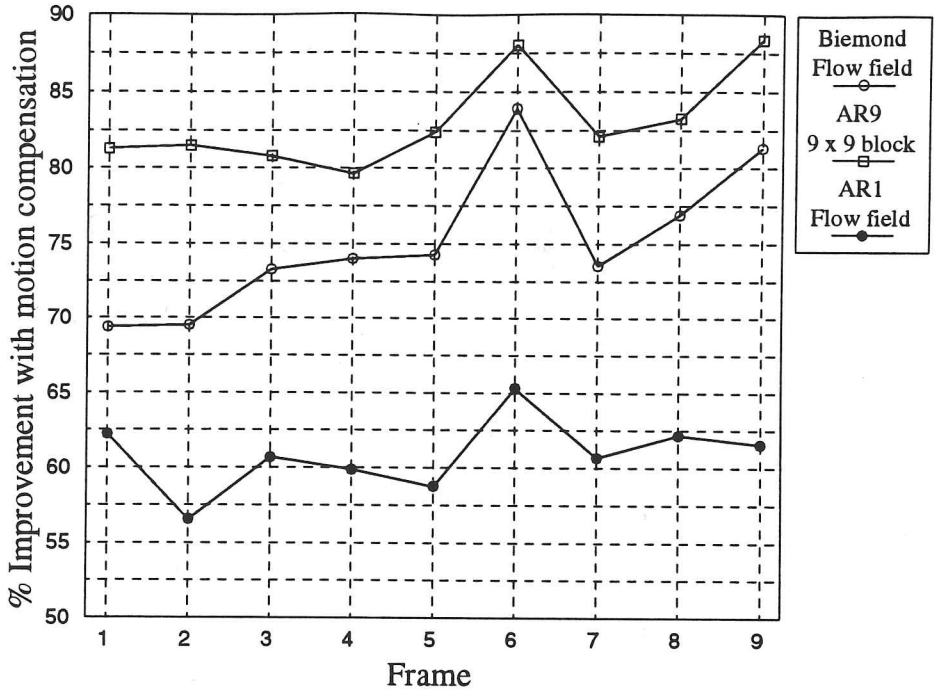

Figure 3.14: Comparing performance on the Cheers sequence.

Because of the size of the images and the heavy computation required for the solution of the Normal equations, a block based method was used for the AR algorithm. One block size of 9×9 was used. The motion estimation equations were set up using the inner 7×7 block of pels and the blocks were overlapped to allow one vector to be estimated for each 7×7 block. The AR estimator was therefore employed to predict tiles of size 7×7. Several different supports for the AR model were tried and they gave the expected behavior, that is, a larger support yielded a lower prediction error. The AR9 model shown in 3.8 gave satisfactory results and was chosen to represent a good compromise between computation and prediction error.

The Biemond algorithm was used to estimate a vector at every pixel in the image. Again a 7×7 block of pixels was used for setting up 49 equations that were then solved for the motion vector.

3.6 Results and Discussion

From both figures 3.14 and 3.15 it is evident that the AR algorithm performs better than the Biemond technique. This is despite the fact that the latter algorithm estimated a motion flow field (i.e. one vector per pel) for the image whereas the former was a block based estimation process. In both examples, the AR9 estimator improves the PIMSE by about 10% over the Biemond estimator.

For comparison there is also a result using the AR1 model as in figure 3.8. A prediction was generated at every pixel, in the same way as for the Biemond estimator (i.e. not block based processing). There is little difference between these two estimators for the Miss America sequence

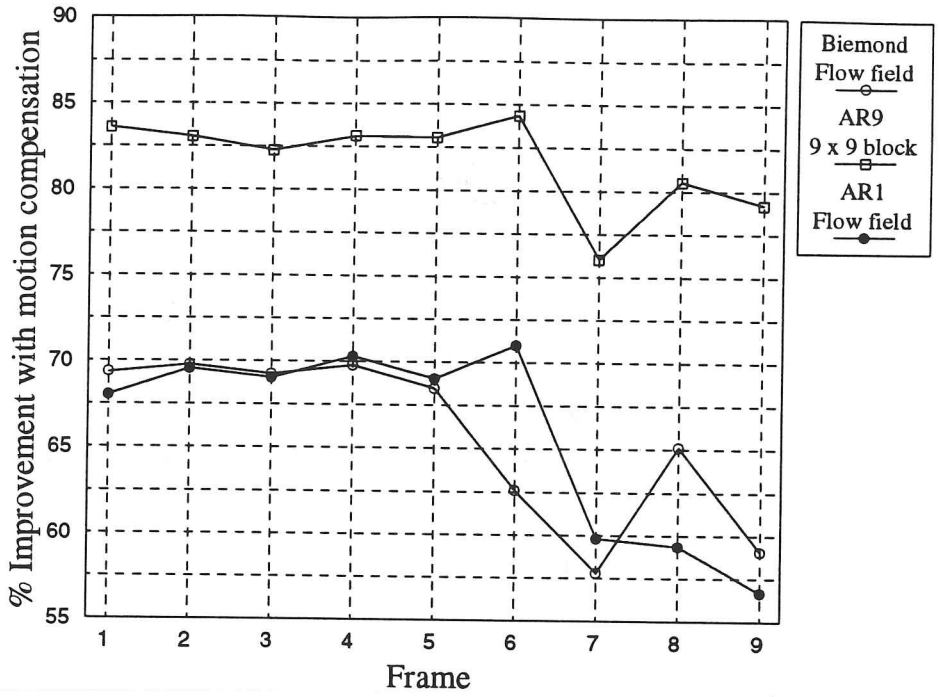

Figure 3.15: Comparing performance on the Miss America sequence.

principally because that sequence is slow moving.

The Cheers sequence shows faster motion therefore the differences are more significant. Recall that the initial start displacement parameter for the AR algorithm is zero. Hence the AR1 model assigns a low value to the one coefficient because the temporal correlation is low at the start of the algorithm. This low temporal correlation biases the motion update towards zero and hence the algorithm does not converge on a useful prediction. This was discussed in section 3.4.2.

Pictures showing the quality of the predictions are not shown at this stage but are left till the algorithm is finalized at the end of this chapter.

3.7 Adaptive estimation

The previous sections have dealt with the basic estimators in order to assess the fundamental differences between the standard methods and this AR approach. The damping factor, μ, has been fixed throughout the investigation. The estimation of the AR coefficients at each iteration is the only adaptive process introduced to the algorithm so far. To propose adaptive schemes that improve the estimator, it is educational to consider the conditions under which the estimator fails. The work by Kearney et al [46] considered gradient based estimators in general. The following sources of error are noteworthy.

1. Measurement error due to noise : Measuring the gradient using simple central differences may yield noisy estimates, especially in a highly textured area. Noise in the image itself will add to this problem. This may cause ill–conditioning in $\mathbf{G}^T\mathbf{G}$.

2. Non–linearity of images : The linear approximation to the image function is not valid everywhere. At edges, for instance, the higher order Taylor series terms are not negligible. This adds to the size of the vector $\mathbf{v}$. Furthermore, this error will be systematic with a definite correlation structure because the underlying image feature producing it is not a random one. The white noise assumption for the contents of $\mathbf{v}$ is therefore violated.

3. Ill–conditioning of $\mathbf{G}^T\mathbf{G}$. : The sources of error above may not directly give rise to ill conditioning. In fact, Martinez [63] has shown that this ill conditioning is directly linked to the directions of maximum and minimum spatial gradients for the Biemond estimator. His analysis is repeated for this estimator in Appendix D. That analysis shows that the 3D AR estimator is ill–conditioned if the points used in setting up the motion equations lie along an edge. In this respect it behaves similarly to the Biemond estimator as discussed previously in Chapter 2. The advantage of the 3D AR estimator seems to be more in its robustness to noise since measurements over a small region are weighted to yield one observation that is used in setting up the matrix solution.

Driessen et al. [38, 37, 39], Böröczky et al [12, 53] have introduced techniques for adapting the regularizing parameter during the standard Wiener based pel–recursive process. Driessen et al. [38] have noted that the techniques which make μ proportional to the ill–conditioning of $\mathbf{G}^T\mathbf{G}$ are useful. Their proposed scheme was introduced in Chapter 2 and is repeated below.

$$\begin{aligned} \mathbf{d} &= [\mathbf{G}^T\mathbf{G} + \mu\mathbf{I}]^{-1}\mathbf{G}^T\mathbf{z} \\ \text{where } \mu &= |\mathbf{z}|\frac{\lambda_{max}}{\lambda_{min}} \end{aligned} \tag{3.19}$$

Martinez [63] recognized that when the solution was ill–conditioned, it was best to align the motion estimate with the direction of maximum spatial gradient. Otherwise, his solution was not regularized.

Although these arguments were introduced for the standard WBME, which in effect uses model AR1 in figure 3.8, they are equally valid for the general AR based estimator. Therefore, a combined adaptive scheme is now presented which incorporates the adaptive estimate for μ as proposed by Driessen et al. and the Martinez 'fallback' mode when the solution is ill–conditioned.

$$\begin{aligned} \mathbf{d} &= \begin{cases} \alpha_{max}\mathbf{e}_{max} & \text{if } \frac{\lambda_{max}}{\lambda_{min}} > \alpha \\ \left[\mathbf{G}_w^T\mathbf{G}_w + \mu\mathbf{I}\right]^{-1}\mathbf{G}_w^T\mathbf{z}_w & \textit{otherwise} \end{cases} \\ \mu &= |\mathbf{z}|\frac{\lambda_{max}}{\lambda_{min}} \textit{ if } \frac{\lambda_{max}}{\lambda_{min}} \leq \alpha \\ \alpha_{max} &= \frac{\mathbf{e}_{max}^T}{\lambda_{max}}\mathbf{G}_w^T\mathbf{z}_w \end{aligned} \tag{3.20}$$

As before, λ, $\mathbf{e}$ refer to the eigen values and eigen vectors of $\mathbf{G}_w^T\mathbf{G}_w$, and α_{max} is a scalar variable introduced to simplify the final expression. This adaptive algorithm, which involves an estimate of the model coefficients at each iteration prior to the motion estimate being generated,

2. Non–linearity of images : The linear approximation to the image function is not valid everywhere. At edges, for instance, the higher order Taylor series terms are not negligible. This adds to the size of the vector $\mathbf{v}$. Furthermore, this error will be systematic with a definite correlation structure because the underlying image feature producing it is not a random one. The white noise assumption for the contents of $\mathbf{v}$ is therefore violated.

3. Ill–conditioning of $\mathbf{G}^T\mathbf{G}$. : The sources of error above may not directly give rise to ill conditioning. In fact, Martinez [63] has shown that this ill conditioning is directly linked to the directions of maximum and minimum spatial gradients for the Biemond estimator. His analysis is repeated for this estimator in Appendix D. That analysis shows that the 3D AR estimator is ill–conditioned if the points used in setting up the motion equations lie along an edge. In this respect it behaves similarly to the Biemond estimator as discussed previously in Chapter 2. The advantage of the 3D AR estimator seems to be more in its robustness to noise since measurements over a small region are weighted to yield one observation that is used in setting up the matrix solution.

Driessen et al. [38, 37, 39], Böröczky et al [12, 53] have introduced techniques for adapting the regularizing parameter during the standard Wiener based pel–recursive process. Driessen et al. [38] have noted that the techniques which make μ proportional to the ill–conditioning of $\mathbf{G}^T\mathbf{G}$ are useful. Their proposed scheme was introduced in Chapter 2 and is repeated below.

$$\begin{aligned} \mathbf{d} &= [\mathbf{G}^T\mathbf{G} + \mu\mathbf{I}]^{-1}\mathbf{G}^T\mathbf{z} \\ \text{where } \mu &= |\mathbf{z}|\frac{\lambda_{max}}{\lambda_{min}} \end{aligned} \tag{3.19}$$

Martinez [63] recognized that when the solution was ill–conditioned, it was best to align the motion estimate with the direction of maximum spatial gradient. Otherwise, his solution was not regularized.

Although these arguments were introduced for the standard WBME, which in effect uses model AR1 in figure 3.8, they are equally valid for the general AR based estimator. Therefore, a combined adaptive scheme is now presented which incorporates the adaptive estimate for μ as proposed by Driessen et al. and the Martinez 'fallback' mode when the solution is ill–conditioned.

$$\begin{aligned} \mathbf{d} &= \begin{cases} \alpha_{max}\mathbf{e}_{max} & \text{if } \frac{\lambda_{max}}{\lambda_{min}} > \alpha \\ \left[\mathbf{G}_w^T\mathbf{G}_w + \mu\mathbf{I}\right]^{-1}\mathbf{G}_w^T\mathbf{z}_w & \textit{otherwise} \end{cases} \\ \mu &= |\mathbf{z}|\frac{\lambda_{max}}{\lambda_{min}} \textit{ if } \frac{\lambda_{max}}{\lambda_{min}} \leq \alpha \\ \alpha_{max} &= \frac{\mathbf{e}_{max}^T}{\lambda_{max}}\mathbf{G}_w^T\mathbf{z}_w \end{aligned} \tag{3.20}$$

As before, λ, $\mathbf{e}$ refer to the eigen values and eigen vectors of $\mathbf{G}_w^T\mathbf{G}_w$, and α_{max} is a scalar variable introduced to simplify the final expression. This adaptive algorithm, which involves an estimate of the model coefficients at each iteration prior to the motion estimate being generated,

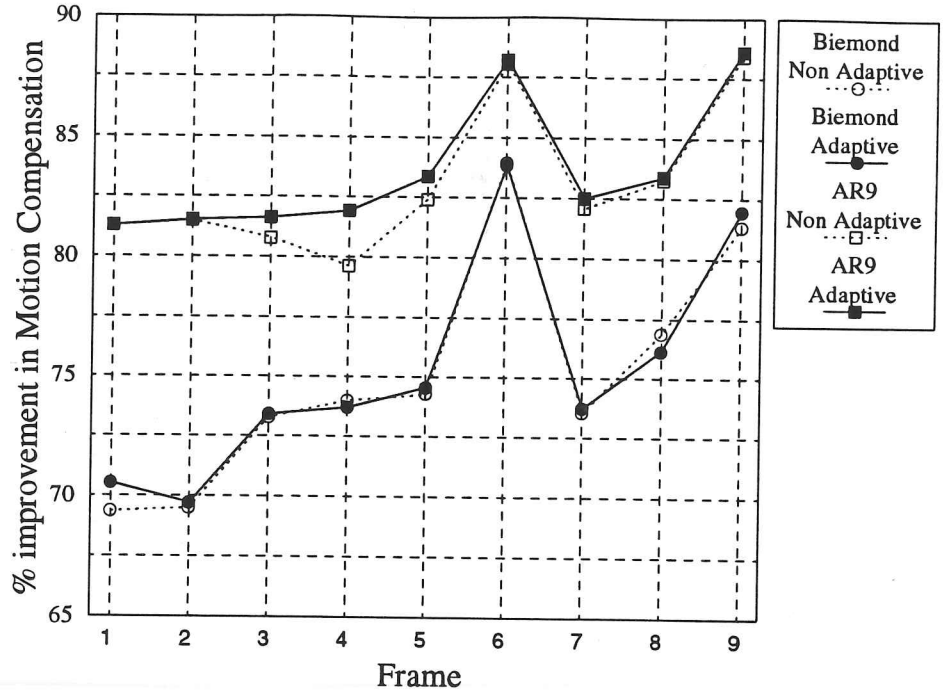

Figure 3.16: Comparing adaptive and non-adaptive algorithms with the Cheers sequence.

will be referred to as the *adaptive 3D AR* (A3DAR) algorithm for image sequence modelling. Incorporating this scheme into the standard WBME yields an algorithm which will be referred to as the adaptive WBME (AWBME).

3.7.1 Experiments

Using the adaptive estimator described in the previous section, two experiments were performed comparing the performance of the AR estimator with and without the adaptation. Two real sequences were used, the Cheers sequence and the Miss America sequence. The results are shown in figures 3.16, 3.17. In these experiments a block size of 9×9 was used for coefficient estimation and motion estimation. The threshold α was set at 25.0. Four remaining criterion were used to halt the iterative process, when any of the following were satisfied, the process was stopped.

1. The number of iterations > 10.
2. The mean square prediction error (MSE) < 1.0
3. The size of the update vector < 0.01
4. The MSE at the current iteration $>$ the MSE at the previous iteration.

Both the Biemond WBME and the 3D AR estimator (employing model AR9 in figure 3.8) were compared. As before, the Biemond technique was used to generate a motion vector for every pixel in the frame. In contrast, the 3D AR estimator was block based, and so generated a

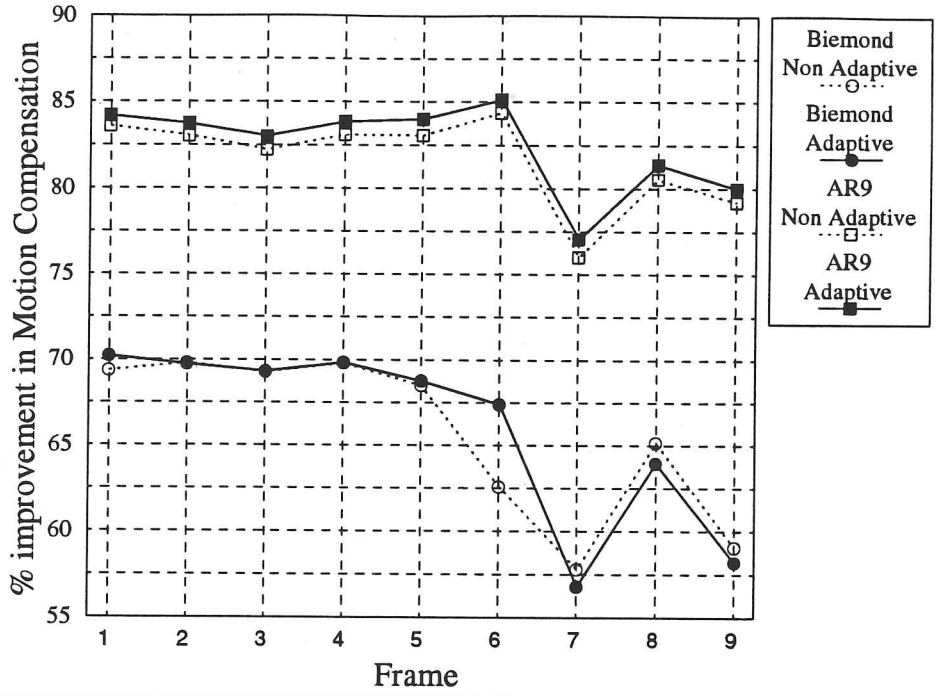

Figure 3.17: Comparing adaptive and non-adaptive algorithms with the Miss America sequence.

prediction for a block of size 9×9 using one motion estimate for that entire block. The blocks were overlapped so that the final predicted tiles were of size 7×7. The adaptive versions of the estimators, AWBME and A3DAR, used the same adaptive algorithm introduced above.

3.7.2 Results and Discussion

In both estimators there is some improvement using the adaptive estimators. Although this improvement is small, the results indicate that it is safe to use the adaptive estimator to remove the guesswork involved in choosing values for μ. It is notable that the adaptive scheme gives a larger improvement for the 3D AR estimator than the WBME. One can argue that there is still guesswork involved in the choice of the Martinez threshold ($\frac{\lambda_{max}}{\lambda_{min}}$), however this threshold operates on the ill–conditioning of the gradient matrix directly. It is therefore further removed from the actual image information than is μ.

3.8 Modelling sequences with large displacements

In real image sequences such as motion pictures, the size of the displacements from frame to frame may be in excess of 10 pixels. Under such conditions, the approximations made to solve for the motion field do not apply and the accuracy of the motion estimates will be severely affected. The main contributing factor to this breakdown is that when there are large displacements the image data at the start point of the iterative gradient based estimator bears little resemblance to the ideal end point corresponding to the correct motion vector. As stated in the motion

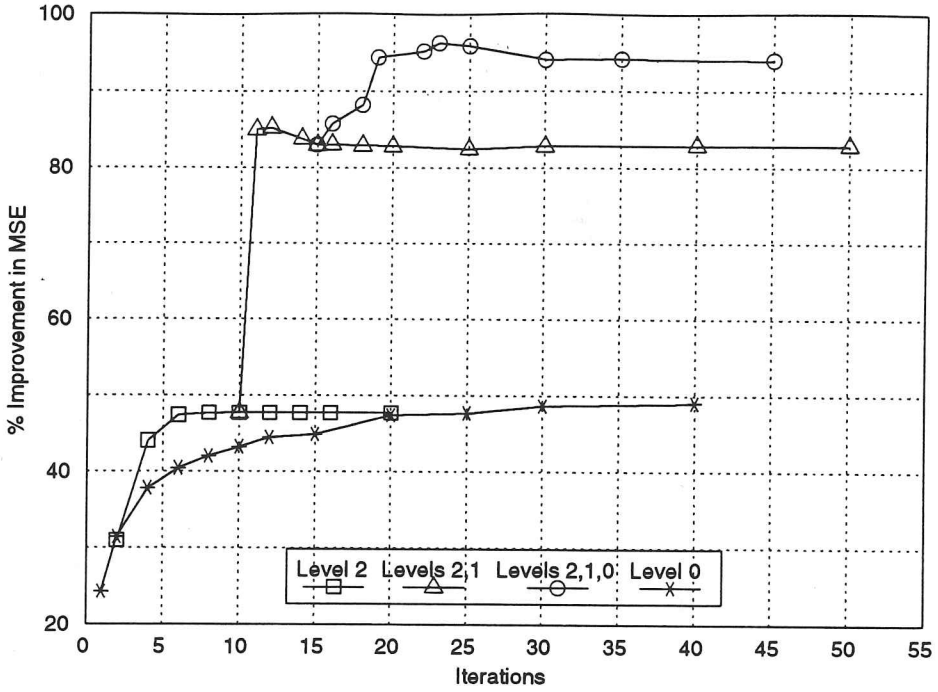

Figure 3.18: Multilevel estimation on Baboon using the Biemond estimator.

estimation review of Chapter 2, a multiresolution scheme is adopted here to solve this problem. The estimator also adopts the motion detection scheme of Bierling [11] to avoid vector haloes.

3.9 Model selection on the pyramid

When considering the performance of the 3D AR predictor for the translating Baboon image earlier in this chapter, it was reported that the spatial extent of the model creates an inherent error in the motion vector estimate. This has extremely serious implications for the multilevel process of modelling a sequence with large displacements.

If a model such as AR9 in figure 3.8 is used, the motion vectors generated have an error of ± 1.0 pixels. This does not necessarily imply that the prediction will suffer, in fact figures 3.10, 3.13 show just the opposite. However, in the case of multilevel estimation, a motion vector error of 1 pixel at the highest level, $L-1$, say, in an L level pyramid, implies an error of 2^{L-1} pixels at the original resolution. If the displacement is large (and one must assume this for real TV sequences), then the prediction at the original level will suffer because of this error. The errors at the higher levels will propagate down the pyramid until the data extracted in compensation will have very little correlation from frame to frame. For this reason, it is important to have accurate motion vector estimation at the higher levels.

'Accurate' motion estimation in this sense means extracting the motion information from the scene that most approximates the true motion of the objects in that scene. The general A3DAR algorithm presented earlier is therefore inadequate because it involves spatial support larger than

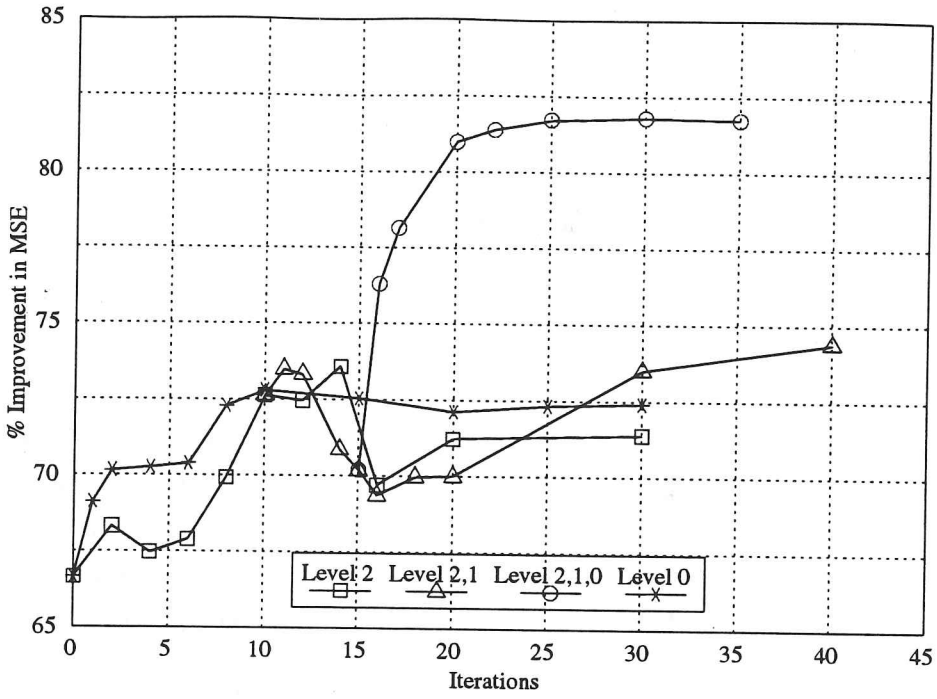

Figure 3.19: Multilevel estimation on Baboon using the AR estimator with model AR9.

1 pixel. At this point it must be stated that there is no reason for the same model to be used at every level in the multiresolution hierarchy. This presents the solution to the problem.

At the higher pyramid levels, an AR model such as AR1 in figure 3.8 can be used to generate accurate motion estimates. At the lower levels, or just the original level only, a more general AR model may be used, for example AR9 in figure 3.8. However, at the higher levels there is now less attention to prediction and more attention to motion estimation, therefore, other algorithms such as Block Matching or the Biemond pel recursive estimator may be used. This has the further advantage of reducing computation.

Figures 3.18, 3.19, 3.20, 3.21 show how important accurate motion estimation can be. The upper 64×64 quadrant of the Baboon image was extracted and translated to yield a translation only sequence with displacement $[-5.0, -5.0]$, see figure 3.7. Multilevel motion estimation was employed using 3 levels and a variance of 1.0 for the Gaussian low pass window. The threshold on ill–conditioning, α, was set at 10.0 but no other thresholds were used. Algorithms were compared on the basis of the PIMSE and figures 3.18, 3.19, 3.20, 3.21 show the progress of the various methods over the second frame in the sequence. The x–axis shows the number of iterations and each differently marked line indicates iterations at a different level in the hierarchy. The lines marked with stars show iterations at the original level only and so this is the progress of the algorithm if no multilevel estimation was employed. The PIMSE was measured at the original level. Therefore the PIMSE for iterations in level 2 only (i.e. none in levels 0,1) was measured by propagating the vectors from level 2 down to level 0 and measuring the PIMSE there. Iterations always begin at level 2 in the pyramid.

Figure 3.18 shows that iterations of the Biemond process at level 2 only can give a PIMSE of about 50% maximum after 20 iterations. But if after 10 iterations, the vectors were refined at level 1 for a further 5 iterations, this improves dramatically to about 80%. Going on to refine vectors at the original resolution results in a PIMSE of about 95%, after a total of 30 iterations. This is much better than the maximum of 50% PIMSE after 40 iterations at the original level alone. This shows that the multilevel approach can provide for useful convergence of the algorithm when displacements are large. Contrast this to the use of the AR9 model at all levels of the pyramid, shown in figure 3.19. The algorithm does not appear to converge at all for iterations at the first two levels, giving about 72% PIMSE. This is very poor compared to the 72% PIMSE after 10 iterations if no multilevel estimation was employed. Only after refinement at level 0 does the PIMSE converge on a more acceptable result of 82% improvement. This pales in comparison to the much better performance of the Biemond multilevel estimator which requires substantially less computation.

If however, the AR1 estimator is used at levels 2,1 and the AR9 model used at the lowest level 0, the situation is improved. Figure 3.20 shows this result. It takes just 20 iterations in total, 10 at level 2, 5 iteration refinement at level 1, and 5 iteration refinement at level 0, to yield a PIMSE that reflects almost perfect compensation at 95%. Compare this to the 70% PIMSE if the AR9 model was used at level 0 alone. Note also that the algorithm appears to converge rapidly as opposed to the almost random behaviour shown in figure 3.19.

Using the Biemond estimator at levels 2,1 and the AR9 model at level 0 gives the best result here. The purely Biemond multilevel estimator is compared to the Biemond/AR9 mixture algorithm in figure 3.21. The figure shows that almost perfect compensation (~~97~~99%) is achieved using the two different algorithms at different levels.

3.9.1 Experiments and Discussion

To highlight the improvement gained with multilevel estimation, experiments were performed on the JUGGLER sequence. This is a sequence showing the torso of a juggler whilst juggling. The motion is very fast, > 20 pixels per frame. The juggling balls give numerous problems to motion estimation since there are large areas which are occluded from frame to frame. The rapidly changing shape of the hands pose additional problems. The sequence is of resolution 256×256.

Figures 3.22 show a comparison between two model based estimators one multilevel and the other single level. The AR1/AR9 multilevel estimator used 4 levels, implementing AR1 on levels 3,2,1 and AR9 on the final level of original resolution. The AR9 estimator used the AR9 model on level 0 alone. The pyramid was generated using $\sigma^2 = 1.0$. A final improvement to the multilevel algorithm was also employed since it is common in multilevel algorithms that stationary areas around moving regions are also estimated as being moving. The phenomenon is known as a vector halo effect and was discussed in the review chapter 2. To reduce this problem a motion

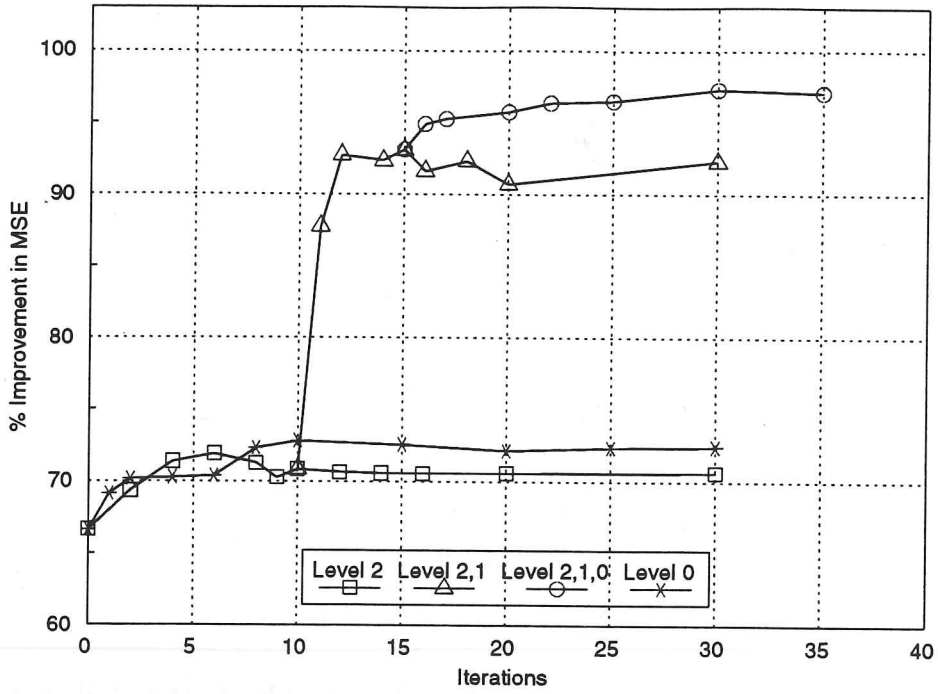

Figure 3.20: Multilevel estimation on Baboon using the AR estimator with AR1 at levels 1,2 and AR9 at level 0.

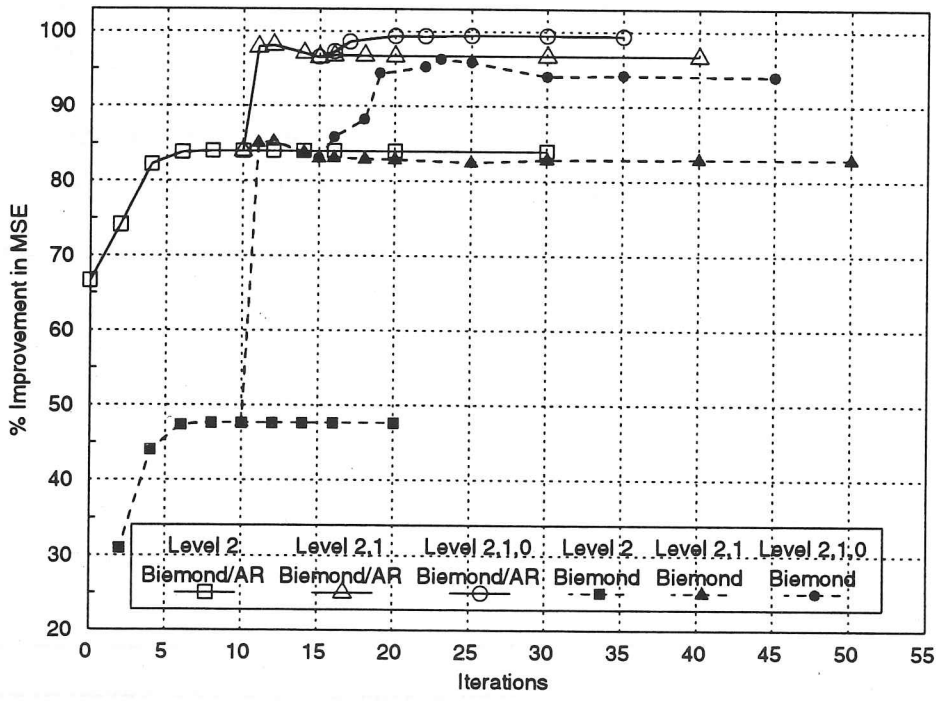

Figure 3.21: Comparing multilevel estimation using a mixture of algorithms at each level versus the Biemond estimator at each level.

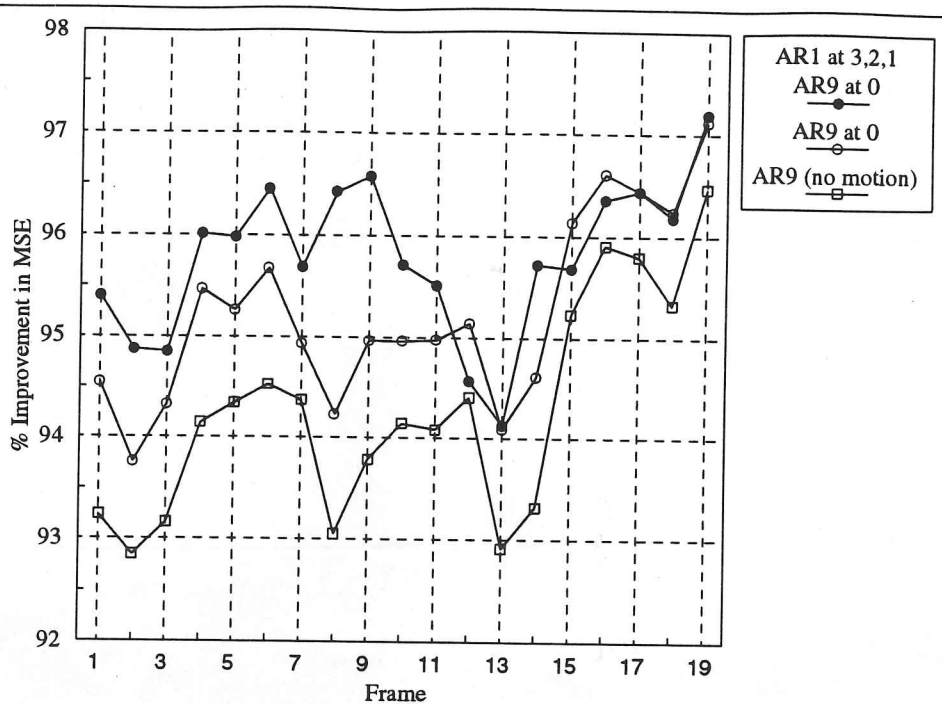

Figure 3.22: 3D AR Modelling for JUGGLER using multilevel 3D AR motion estimation.

detector is applied at the level 0 to verify that a region is moving. This improves the quality of the prediction. The motion detector takes the form of a threshold applied to the non–motion compensated frame difference based on either a MSE or MAE measure, exactly the same as the motion detector applied in standard single level estimation.

The parameters used in the estimation process with the AR1/AR9 system were as follows, (i refers to the iteration index)

1. Modelling block size $= 9 \times 9$, motion estimated using an inner block of 5×5, blocks overlapped to have one motion vector for every 7×7 block.

2. Iterative estimation halted when any one of the following occurred.

$$\begin{aligned} \text{MSE} &\leq 40.0 \text{ At all levels} \\ |\hat{\mathbf{u}}_{n,n-1}| &\leq 0.1 \text{ At all levels} \\ i &> \begin{cases} 10 & \text{For level 3} \\ 5 & \text{Otherwise} \end{cases} \\ \mathbf{z}_i^T \mathbf{z}_i &\geq \mathbf{z}_{i-1}^T \mathbf{z}_{i-1} \end{aligned} \tag{3.21}$$

The parameters used with the single level AR9 system were the same except there were a maximum of 25 iterations allowed at the single level 0.

The curves do not show an impressive improvement with the multilevel estimator, principally because the motion is so large that any reduction in frame difference is emphasized. This is highlighted with the inclusion of the PIMSE for the AR9 model applied with the displacement parameter set to [0.0, 0.0] everywhere in each frame (AR9 (no motion)).

Figure 3.23: Frames 9 (left), 10 (right) of original JUGGLER sequence.

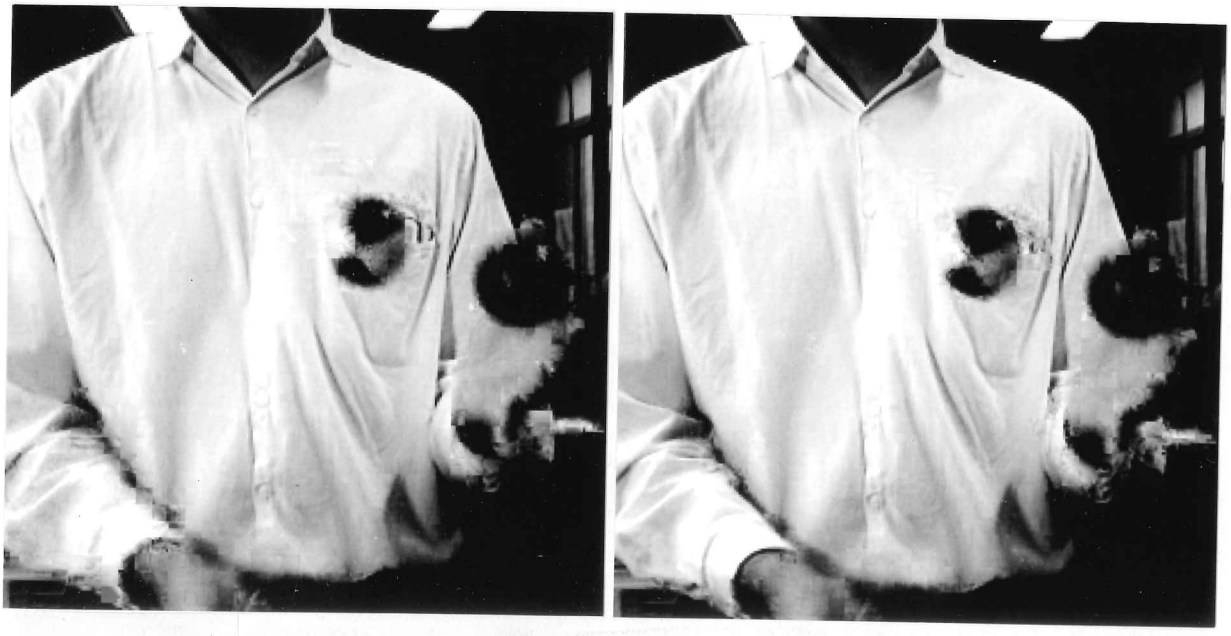

Figure 3.24: Predicted frame 10 using single level AR estimator (left) and Multilevel AR1/AR9 estimator (right).

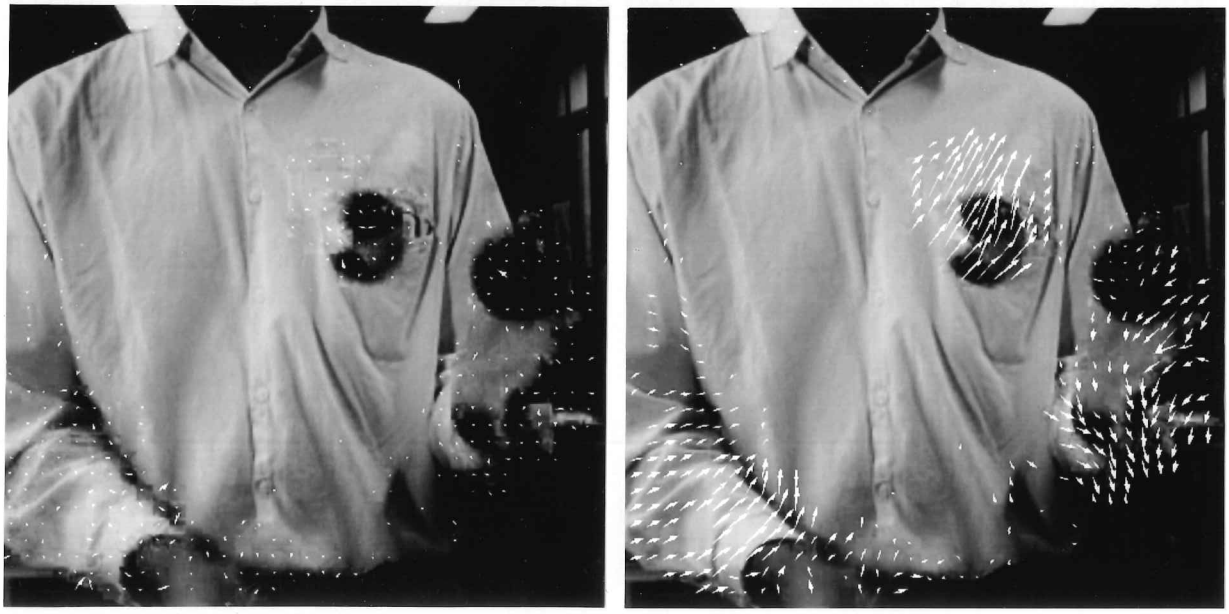

Figure 3.25: Estimated vector field for frame 10 using single level AR estimator (left) and Multilevel AR1/AR9 estimator (right).

Figure 3.26: Frames 14 (left), 15 (right) of original JUGGLER sequence.

Figure 3.27: Predicted frame 15 using single level AR estimator (left) and Multilevel AR1/AR9 estimator (right).

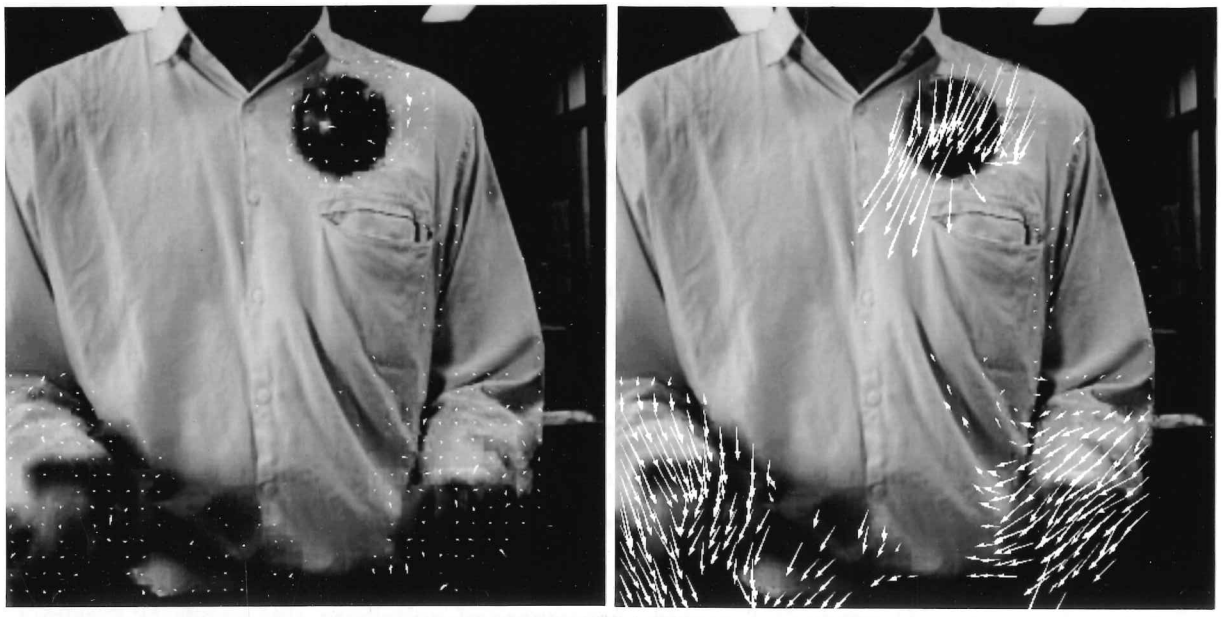

Figure 3.28: Estimated vector field for frame 15 using single level AR estimator (left) and Multilevel AR1/AR9 estimator (right).

Figure 3.29: Predicted frame 15 using single level AR estimator with no motion compensation.

The reason for this is due to the image material itself. JUGGLER consists of fast moving areas; the hands and the juggling balls, but there are large areas of each frame that are not moving with such high velocity; the shoulders and chest. The fast moving areas contribute to a very large frame difference even though these areas account for about 1/3 of the total image area. This is illustrated in figures 3.23, 3.26. These figures are photographs of four frames from two different points in the sequence. Frames 9 and 10 were chosen because they represent typical performance. Frames 14 and 15 were chosen to represent bad performance where the PIMSE for the multilevel motion estimator and the single level method cross over (see figure 3.22). The frames show that the juggling balls rotate as they move and that the hands change shape from frame to frame. The motion estimation algorithm must deal successfully with these problems to be useful.

Because the model can adjust coefficients to compensate for rapid changes in image intensity, the non–compensated model can predict some reasonable grey scale variation in each block even in the fast moving areas. This is shown in figure 3.29. Therefore, the PIMSE is large even for this non–compensated process. The improvement with the single level estimator comes from its partial compensation of the moving areas and uncovered regions. This is well illustrated by frames 9, 10 in the quality of prediction of the moving ball. The multilevel estimator is able to give a very good estimate for the moving arms and so the resulting prediction is much improved. Figure 3.24 shows this prediction improvement. Figure 3.25 shows the estimated vector fields for each frame using the different estimators. The fields are superimposed on the corresponding predicted frame, and that frame is reduced in intensity slightly to allow the vectors to be seen more easily. It is clear that the multilevel estimator copes with the moving arms much better than the single level estimator. Frames 14, 15 show the opposite effect, figures 3.26 and 3.27. The multilevel estimator does not track the white/black[6] ball well in either frame pairs 9,10 or

[6] Yellow and red in a colourful world.

14,15. Unfortunately, in 14,15 neither estimators can track the motion of the right arm well because of blurring and shape changing, therefore no improvement is gained. The vector fields in figures 3.28 verify this observation.

Considering that the AR1/AR9 estimator is computationally lighter and more effective than the AR9 estimator at the single level, it is the better choice for this sequence.

3.10 The motion parameter in the 3D AR model: Using alternative estimators

The previous sections have served to highlight the fact that the modelling process is more effective when the motion of the object is accurately estimated. Therefore, the motion parameter in the model is definitely related to the motion of objects in the scene. It would be advantageous therefore, to consider the modelling process as two processes; a motion estimation process and an image prediction compensation. To this end it is useful to consider the effect of using BM or the WBME to estimate motion and then use this motion parameter in the model framework. The 3D AR model would then act as an interpolator correcting for errors and motion not obeying the pure translational rule. There is also the additional consideration that the adaptive iterative 3D AR process for estimating $\mathbf{d}$ is computationally demanding when compared to either of the two other estimators mentioned.

Figure 3.30 shows the PIMSE for the Juggler sequence using (BBM) Block Matching and WBME to provide the estimate for the motion vector that is then used in the 3D AR modelling process. The adaptive WBME algorithm employed was identical to the previous AR1/AR9 estimator with respect to the number of levels and the parameters used. The MAE criterion was used for the BBM algorithm, with the motion detection threshold set at 10, and the noise ratio[7] threshold, $r = 1.2$ for the level 0 only[8] and 1.0 otherwise. A full search was employed with search space of ± 4 pixels. The block sizes were the same for both algorithms, and the blocks were overlapped to allow one motion vector to be estimated for each inner 7×7 block. There were therefore 36×36 vectors estimated in each frame.

The PIMSE results indicate that the new algorithms perform similarly to the multilevel AR modelling process, with the algorithm employing the adaptive Biemond estimator (Biemond/AR9) performing best overall. However, the figures 3.31 show that the predicted frames 10 and 15 using both the WBME and the BM estimator give much sharper rendering of the moving areas, particularly the juggling balls. The vector fields shown in 3.32 verify that this is due to the better performance of these estimators with respect to motion estimation. The use of these motion estimators therefore improve the performance of the modelling process overall.

[7] Boyce ratio; See Chapter 2.

[8] The noise is greatly reduced in the upper levels of the pyramid due to the low–pass filtering that creates the pyramid.

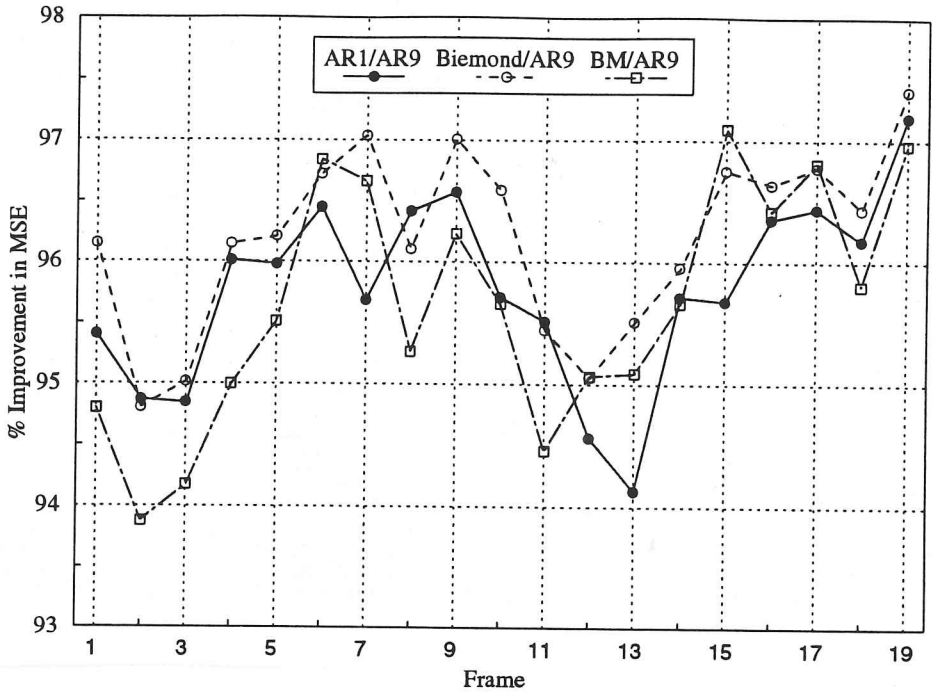

Figure 3.30: 3D AR modelling for JUGGLER using alternative motion estimators.

3.11 Summary

This chapter has considered the use of the 3D AR model for image sequence modelling. The treatment is different from past work in that it allows variation in the model coefficients across the image. The major problem encountered in solving the model equations was the estimation of both the displacement $\mathbf{d}$ and the model coefficients simultaneously.

Two different approaches were investigated. The first was an iterative refinement technique involving estimation of displacement and model coefficients as two separate refinement problems in the iterative scheme. The second approach treated the displacement estimation as a completely separate issue in the modelling process. The displacement estimation was achieved using a standard motion estimation technique and then the modelling was executed using that motion estimate. Both these approaches were implemented finally as part of a multiresolution algorithm for large displacements.

The results show that both approaches can give similar performance with respect to the same sequence when considering PIMSE. Also the modelling approach is superior to standard motion estimation/compensation approaches. This is not surprising. The modelling stage of the proposed algorithms attempts to minimize the squared prediction error directly therefore it is not unexpected that the PIMSE is better than standard 'translation only' estimators.

As a motion estimator, the algorithm employed falls short of standard techniques. This is due in part to the size of the spatial support used in the next frame, and also due to the re–estimation of coefficients at each iteration in the motion estimation process. It is conceivable

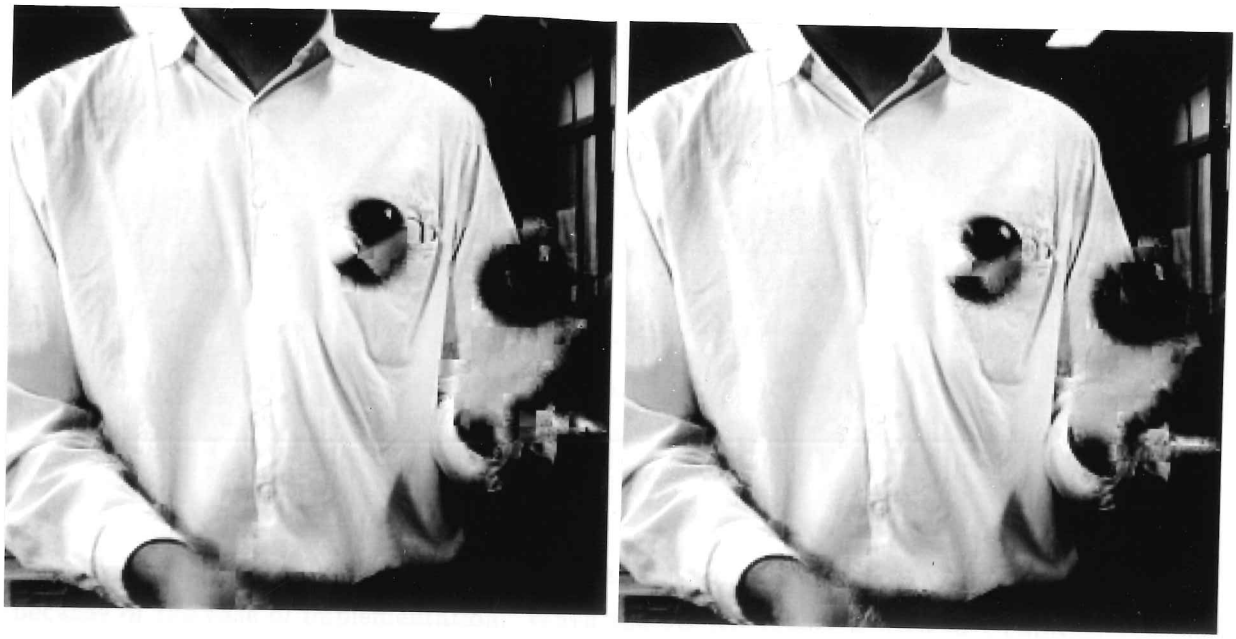

Figure 3.31: Predicted frame 10 using different motion estimators, BM (left) and adaptive WBME (right).

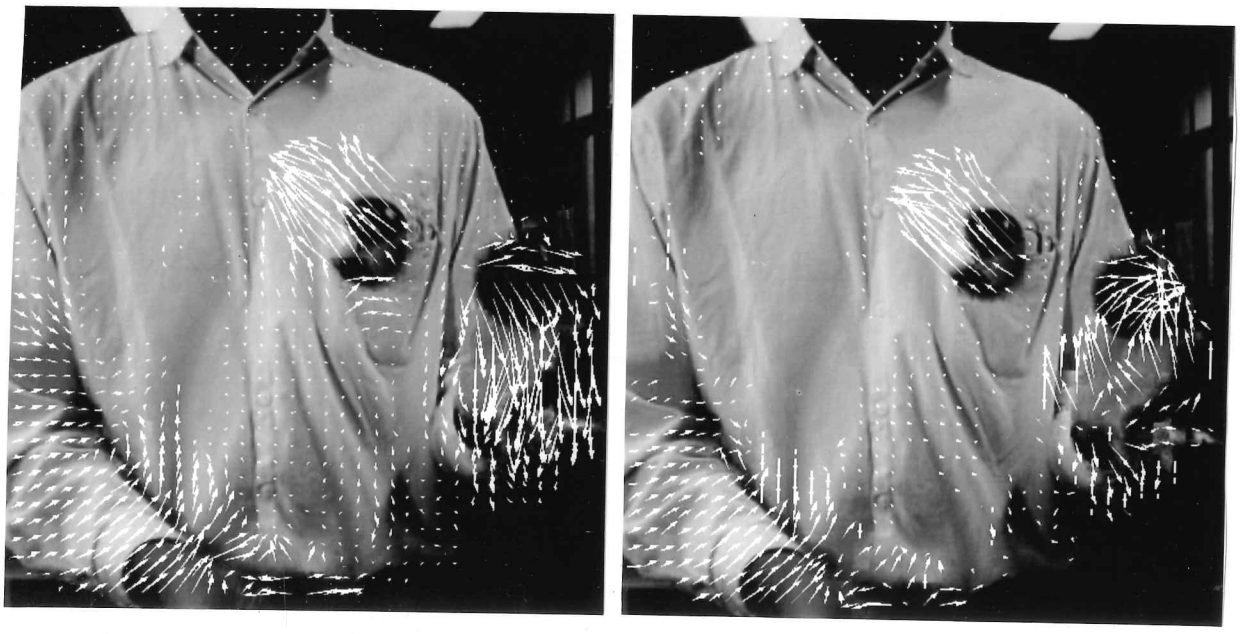

Figure 3.32: Estimated vector fields for frame 10 using different motion estimators, BM (left) and adaptive WBME (right).

that an initial displacement guess would cause the correlation between the current block and the previous displaced block to be small. This may occur due to large motion or the image information itself. In such a case, the model coefficients estimated would be small and prevent motion updates to be made. The motion estimation would therefore halt at that stage despite the large error.

It was shown that the performance of the modelling process depends on the accuracy of the motion estimate. Multilevel motion estimation using the AR1 model was found to be successful when combined with the AR9 predictor as the last stage. The algorithms employing the BM estimator or the adaptive WBME to estimate the motion used in the modelling stage, can perform better compensation than the 3D AR estimator as far as visual perception of the prediction is concerned. In view of the fact that both of these alternatives are less computationally intensive than A3DAR, they are the better choice for 3D AR modelling frameworks. The restoration work that follows uses a multiresolution BM scheme (as discussed previously) for motion estimation because of the ease of implementation. When the 3D AR model is used in Chapter 7 it is used together with this motion estimator.

IMAGE SEQUENCE RESTORATION: A REVIEW

Two dimensional image restoration has been an active research topic since the 1960's. Algorithms have been developed to remove blur, noise and impulsive distortion from single images [55, 84, 42]. In contrast, research concerned with image sequence restoration can be traced back only to about the late 1970's [17] when it was mainly associated with image coding. The subject has only recently become prominent within the last decade [19, 61, 63, 2, 76, 1]. It is during these years that the increasing speed of microprocessors has made the heavy computation practical. That, coupled with the rising importance of video both for communications and entertainment has made the subject grow in importance.

Early work in this field by Dubois and Sabris [19] and Dennis [17], tended to concentrate on the suppression of white noise. The treatment of motion was very simple. The noise suppression filter consisted of some form of weighted frame averaging except where motion was detected. In those areas either little or no averaging was done.

As stated in chapter 2, it is better to consider the processing of image sequences as a full 3D problem rather than a repetitive 2D one. Although the early work done on Image Sequence Restoration (ISR) could achieve satisfactory results, in cases with much motion or heavy noise levels the performance deteriorated. It was clear that explicit treatment of motion would yield improvements.

Huang [35] in 1981, used a simple correspondence motion estimator to direct the application of a 3 tap temporal FIR filter to suppress white noise. He also employed a 3 tap temporal median filter to suppress line dropout and impulsive distortion. His results showed a marked improvement over techniques employing no motion compensation.

More recently, Martinez [63] has studied this exact problem of ISR. His thesis however, was more concerned with the development of a robust motion estimation technique, which was low in computation, to allow rapid sequence processing. Like Huang [35], he implemented 3 tap FIR and median filters to good effect. His work points out that the performance of the motion estimator drastically affects the performance of the restoration system.

It is only recently with the work of Sezan, Tekalp, Özkan, Lagendijk, Katsagellos, Efstratiadis and others that an attempt has been made to use spatio–temporal restoration techniques combined with a more coherent approach to motion estimation. Their work has concentrated on noise suppression [2, 59, 76] and simultaneous blur and noise removal [1]. The work has generally

employed a gradient based approach to motion estimation using both multiple constraint techniques [24] and variants of the WBME [21, 2]. However, it is acknowledged that multiresolution BM gives comparable results and is easier to control than any other technique. These results encourage the development of more 3D algorithms for noise suppression in image sequences. Note however, that the approach to ISR has been to treat the motion estimation and noise suppression separately.

In contrast to the work done in noise suppression, there has been little treatment of the problem of suppressing impulsive noise in image sequences. The problem is not a negligible one since TV engineers are quite familiar with motion pictures showing randomly dispersed flashes of light and dark. The effect is called 'Dirt and Sparkle'. The only work that can be cited here is that due to Arce, Alp et al [7, 5, 4], and of course the motion compensated efforts of Huang [35] and Martinez [63]. Arce and Alp have presented spatio–temporal Multistage Median filter architectures that are robust to motion but still reject impulsive noise. This class of filters had previously been introduced in [75] and the median filter itself has a long history as a good tool for impulsive noise suppression in 2D signals. The efforts of all these authors have been combined and extended in this thesis to introduce new spatio–temporal structures for impulsive noise suppression. Some work is also introduced which proposes a model based approach to the problem. Finally, none of the previous authors have considered using a detector for the impulsive noise before engaging the median operation. The idea is presented in this thesis as an effective way of decreasing computation and increasing output quality. The main philosophy in detection is to restrict the attention of the filtering mechanism only to the distorted areas. The problem is definitely a local one and so must be treated as such.

Having outlined a brief overview of the subject, the review will now continue in three parts. The final two parts discuss the central problems addressed by this thesis and the first discusses early ISR work which gives some basis to the directions chosen for further work.

4.1 Early techniques for noise reduction in image sequences.

Frame averaging has always been recognised as a cheap, effective way of reducing noise in image sequences. The technique has been used extensively and to good effect for electron microscope imagery. The implementation is usually a recursive estimator of the form

$$\mathbf{i}_n = \frac{1}{n}[(n-1)\mathbf{i}_{n-1} + \mathbf{s}_n] \tag{4.1}$$

Here, $\mathbf{i}_n$ represents a vector of the intensities in the current output image, $\mathbf{i}_{n-1}$ the previous output image and $\mathbf{s}_n$ the vector of current noisy image samples that are input to the system. $\mathbf{i}_n$ can be recognised as the running average of all the past n frames. This is successful in microscopy applications because the image sequences observed represent stationary scenes.

It was only natural, therefore, that the first noise reducers for TV imagery attempted to implement frame averaging. Due to motion in real scenes, it was necessary to limit the averaging

effect when motion was detected. The filters implemented were first order recursive of the form

$$\begin{aligned} I(i,j,n) &= I(i,j,n-1) + \alpha(s(i,j,n) - I(i,j,n-1)) \\ &= (1-\alpha)I(i,j,n-1) + \alpha s(i,j,n) \end{aligned} \tag{4.2}$$

When $\alpha = \frac{1}{n}$, the filter shown in 4.1 is the result. The scalar constant α was chosen to respond to the size of the frame difference $|\mathrm{DFD}_0| = (s(i,j,n) - I(i,j,n))$. When this error was large, it implied motion so α was set close to 1.0 to turn off the filtering. A small frame difference implied no motion and so the constant could be set to some small value to allow filtering. A typical example was the strategy used by Dennis [17].

$$\alpha = 1.0 - k_1 e^{-(\frac{|\mathrm{DFD}_0| - k_2}{k_3})^2} \tag{4.3}$$

$$\textit{where } \mathrm{DFD}_0 = s(i,j,n) - I(i,j,n-1) \tag{4.4}$$

k_1, k_2 and k_3 were constants chosen to select the best form of the characteristic.

Although the technique performs well in stationary regions of the image, the final result is not satisfactory because of the following artefacts.

- Moving regions in the output frames would generally have a higher noise level than stationary regions.
- Stationary regions would suddenly become noisy if they began to move.
- There is a smearing effect due to filtering in a direction not close to motion trajectories.

Huang [35] and Dubois et al. [19] presented methods for the *motion compensated* temporal filtering of noisy image sequences. Huang [35] employed a 3 tap averaging operation over 3 frames, for use along an estimated motion trajectory. He performed two experiments using two different motion estimators. In one, the motion estimator used a correspondence matching technique that chose the motion estimate to be along the direction that minimizes the variance in pixel intensity. The other estimator used a gradient based approach. The results for both estimators were similar. The filter is interesting in that it uses not only the previous frame but also the next frame as well. However, the filter can still show smearing effects when the motion is not properly tracked.

To adapt the filter algorithm to this effect, Dubois [19] implemented a recursive filter of the type described by equation 4.3. The pel–recursive scheme of Netravali and Robbins [73] was used for motion estimation and the operation of the filter was directed along the estimated motion trajectories. The filter adapted to occlusion effects and errors in motion estimation by

varying α according to a piecewise linear characteristic repeated below.

$$\alpha = \begin{cases} \alpha_b & \text{for } |\text{DFD}| \leq e_1 \\ (\frac{\alpha_e - \alpha_b}{e_2 - e_1})(|\text{DFD}| - e_1) + \alpha_b & \text{for } e_1 < |\text{ DFD }| \leq e_2 \\ \alpha_e & \text{for } |\text{DFD}| > e_2 \end{cases} \quad (4.5)$$

Note that DFD is the *displaced* frame difference, i.e. the frame difference along an estimated motion trajectory between the current frame and the previous. The characteristic allows for three regions of operation. When motion is tracked effectively, filtering is performed with $\alpha = \alpha_b$. When motion cannot be tracked the filtering is turned off by setting $\alpha_e = 1.0$. The mid–range linear ramp represents a smooth transition in filtering between the tracked and untracked regions.

This process of adapting the filter in addition to motion compensation enabled better image detail preservation as opposed to the strategy used by Huang [35]. In their conclusions, Dubois et al [19] note that improved motion compensation would lead to better motion compensated temporal filtering, and indeed this is so. The lessons from this early work are therefore twofold.

1. Motion compensated filtering of image sequences enables better noise suppression and a better output image quality than non–motion compensated techniques.

2. It is important to adapt the extent of filtering to the accuracy of motion estimation. This would enable a better quality output image by reducing smearing artefacts.

Therefore, although motion compensated filtering is more effective for image sequences, a good algorithm must also be robust to erroneous motion estimation.

An important observation that must be made is that it is often the case when there is rapid motion, this motion cannot be estimated accurately. However this may not affect the subjective quality of the output despite the reduced effectiveness of the filtering operation. This is because the human perception of a fast moving object is much less than a slow moving one. Therefore it is possible to strike a useful compromise between output image quality and the speed of objects in a scene.

The next two sections discuss the development of ISR algorithms that incorporate more spatial information and employ better motion estimators.

4.2 Noise Reduction in Image Sequences

The work by Dubois et al [19] led others [63, 13, 59, 76] to incorporate better motion estimators into a temporal filtering scheme. However workers such as Katsagellos et al. [2] and Sezan et al. [1, 57, 58] recognized that although the driving force behind purely temporal image sequence filtering was the potential for improved image detail preservation, there was no need to ignore the noise rejection capacity of the spatial operation. This led to the use of 3D operators [2, 44].

Recently, in [59, 76, 57], a Wiener filter for the removal of noise and blur has been presented. This operation uses information from several frames at once. The various methods can be considered under the two headings below.

4.2.1 Motion compensated temporal filtering

Purely temporal filtering of image sequences is still popular presumably because of the low computation involved. Boyce [13] has implemented a motion compensated frame averager using BM as the motion estimator. She has made the BM algorithm more robust to noise in the sequence by ensuring that the final match is due to object motion and not to noise. Her BM algorithm was described in Chapter 2. The algorithm used $(N-1)/2$ frames previous to and following from the current frame. Her results showed noise reduction satisfactorily close to the theoretical maximum attenuation of $\frac{\sigma_{nn}^2}{N}$, where σ_{nn}^2 is the noise variance. Using the BBM algorithm implies a robust motion estimator that is able to handle reasonably large motion, therefore the subsequent averaging operation would be better than the approach used by Huang [35]. A multiresolution BM algorithm would be more appropriate for real TV imagery and this is considered in the later chapters of this thesis.

Of particular interest is the use of the 3-D Wiener filter for noise suppression since it can reject more noise given the same number of frames used in a frame averaging operation. Several investigations [61, 63, 59, 76] have employed an N tap Wiener filter temporally to reject noise in sequences. The signal observation model is typically

$$s(i,j,n) = I(i,j,n) + e(i,j,n) \tag{4.6}$$

Here, $s(i,j,n)$ represents the observed, noisy image sequence, $I(i,j,n)$ represents the original, clean sequence and $e(i,j,n)$ the added white Gaussian noise of variance σ_{nn}^2. The single point Wiener filter was used as the estimator, and is stated below.

$$\hat{I}(i,j,n) = \frac{\sigma_{ii}^2}{\sigma_{ii}^2 + \sigma_{nn}^2}(s(i,j,n) - \bar{I}(i,j,n)) + \bar{I}(i,j,n) \tag{4.7}$$

$\hat{I}(i,j,n)$ is the Wiener estimate of $I(i,j,n)$ and $\bar{I}$, σ_{ii}^2 are the mean and variance of the unknown, clean signal, $I(i,j,n)$.

The filter was directed along motion trajectories, and given N frames the following parameter estimates were used, where $[sx_n,\ sy_n]$ represent the components of the motion vector mapping the current frame 0 into the frame n.

$$\bar{I}(i,j,n) = \frac{1}{N+1}\sum_{n=-N/2}^{N/2} s(i+sx_n, j+sy_n, n) \tag{4.8}$$

$$\sigma_{ss}^2 = \frac{1}{N+1}\sum_{n=-N/2}^{N/2} [s(i+sx_n, j+sy_n, n) - \bar{I}(i,j,n)]^2$$

$$\sigma_{ii}^2 = \begin{cases} 0 & \text{if } \sigma_{nn}^2 \geq \sigma_{ss}^2 \\ \sigma_{ss}^2 - \sigma_{nn}^2 & otherwise \end{cases}$$

The filter achieves noise reduction by adding to the averaged signal a fraction of the difference between the noisy and averaged signal. This fraction is proportional to the suspected ratio of original to noisy signal variances. The output signal is forced closer to the averaged signal when the observed signal variance is low.

The filter can be seen to behave in the required manner for robust operation when there is inaccurate motion estimation. When there are errors in motion estimation, then the estimate for σ_{ss}^2 would be large, and $\sigma_{ii}^2 \gg \sigma_{nn}^2$. Therefore less filtering would be done. Conversely, when the motion estimation is more accurate, then the filter would be engaged since the signal variance would then not swamp equation 4.7.

In an earlier work, Martinez and Lim [61] used a cascade of Wiener filters like 4.7 oriented in different directions. No motion estimation was employed so that the cascade of filters would operate as an implicitly compensated filtering operation. The filter directions were not only chosen to be temporal but several directions within the current image plane were also employed. Their results showed good performance in uniform regions but no really improved performance in moving areas. This is because, unfortunately, in many sequences, it is unlikely that among a practical number of 1D filters there will be one whose orientation corresponds to the motion of the scene. That work showed that there is no substitute for an explicit treatment of motion.

Martinez [63] went on to implement an explicitly compensated version of the filter introduced in [61]. Only one filter was used in the direction of the motion estimate and some useful results were reported. The motion estimator employed was a non–regularized version of the WBME discussed previously, in a multiresolution framework.

Sezan et al. [59, 76] employed a much more complex motion estimator to implement the same filter shown in 4.7. Again only one filter was used, and that was directed along the motion trajectory. The algorithm for motion estimation [24] employed a multiresolution scheme involving a multiple constraint[1] gradient based technique. They compared their results with that using the WBME and the previous Martinez et al. algorithm, and found that they were able to achieve superior performance. This demonstrates the importance of good motion estimation in the noise reduction problem.

4.2.2 Motion compensated spatio–temporal filtering.

Purely temporal filtering can provide noise reduction without image detail degradation. Yet the image sequence is a 3D signal and so the spatial information could be used to advantage. The principal flaw in purely temporal filtering is that in cases where the motion is not tracked, the

[1] Oriented smoothness.

image detail can be destroyed. It is generally expected that incorporating spatial information would yield a system with an improved noise suppression capacity together with the added detail preservation of temporal information when this is available.

Katsagellos et al [2] extended a number of 2D noise filtering algorithms to 3D. Their methods involved spatio–temporal filtering. Parameters for the filters were chosen to optimize subjective image quality. The work of Sezan et al [1, 58, 57] represents the first attempts to 'optimally' filter noise and blur in an image sequence. Their approach used 3D Wiener filtering and they have developed a novel form of the filter. Their approach results in a frequency domain image sequence filter that involves 2D FFTs and not 3D FFTs as the case with the full 3D frequency domain Wiener filter. Their derivation of the filter is somewhat involved, but can be arrived at through an alternative route which is introduced in Appendix F. The motion estimator employed was the one presented by Fogel [24]. The filter was applied to images that showed a global displacement only, in particular, aircraft surveillance video. Their results were encouraging and the later chapters of this thesis present the use of the Wiener filter for image sequences that show several moving objects.

The main body of the thesis investigates the various forms of the 3D Wiener filter, such as the 3D FIR spatiotemporal approach and the fully 3D DFT frequency approach. It is employed for noise reduction in TV imagery.

4.3 Removing Impulsive Noise

The problem of impulsive noise suppression in image sequences has only been considered in a few published works. The work of Huang [35] and Martinez [63] represents two attempts to achieve temporal median filtering along motion trajectories using a 3 tap median filter.

The median filter itself was introduced by Tukey [93] in 1977. The technique is one from the generic set of Order Statistic filters [101, 77, 26, 23]. These filters operate on some window of data. The procedure is first to rank the samples in that window and then the output of the filter is chosen to be one of the order statistic values for that ranked distribution. Thus the output can be the Upper Quartile value for instance. The median filter chooses the median value to be the output. For the case of an odd number of samples, $2N \pm 1$, in the window, the median value is the Nth value in the ranked array. But when the number of signal samples is even, $2N$, the median value is defined as the mean of the samples at the two middle positions N and $N + 1$.

The median operation has been quite successful in images since it preserves edges well [77]. As it rejects outliers from a distribution, it is effective in removing impulsive noise such as speckle or Salt and Pepper noise from single images. A typical 2D image filtering application [42, 55, 77] would use a window of size 3×3 and replace the pixel at the centre of the window with the median of the 9 pixels in the window. Every point in the image is treated in this way.

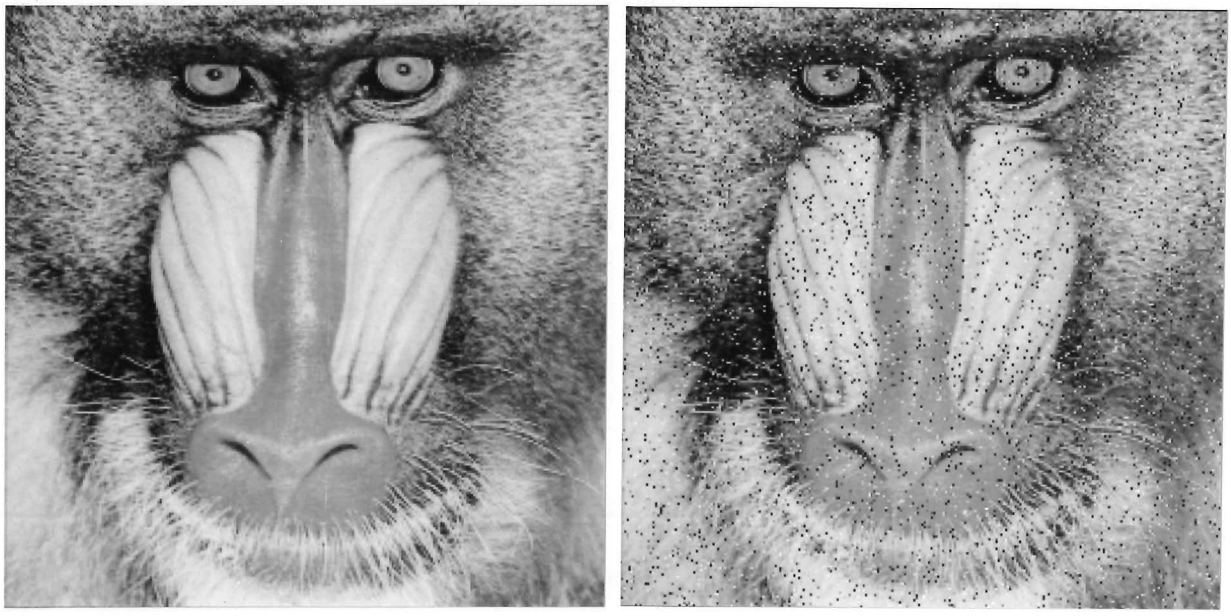

Figure 4.1: Original (left) and distorted (right) images showing impulsive degradation with $P_s = 0.1$ on 8 bit pixel data.

Although the median operation preserves edges, the fact that it rejects isolated impulses implies that it distorts fine detail in the image. Recursive implementation of the filter would reduce image grey scale variation to isolated areas of planar gradations of grey level separated by sharp steps. This is an important shortcoming with respect to median operations and in general it is preferred that the window used for the filter be as small as possible to avoid this distortion. Figures 4.1, 4.2 show the performance of a 3×3 median operation on an image corrupted by uniformly distributed impulsive distortion both in terms of position and grey scale. Note that the distortion is not additive but exclusive, i.e.

$$s(i,j,n) = \begin{cases} S & \text{with probability } P_s \\ I(i,j,n) & \text{Otherwise} \end{cases} \tag{4.9}$$

Here, S represents a spike of uniformly distributed grey scale which *replaces* the value of the undistorted image $I(i,j,n)$ at that same position. The distortion is essentially a local one and its effect depends on how often the distortion occurs i.e. P_s. In the image shown in figure 4.2, $P_s = 0.1$, which is a high rate of distortion. The photographs indicate that although the filter rejects the distortion well, it also blurs the image slightly, cf. figure 4.1 with figure 4.2.

To allow the median operation to preserve more detail in the image sequence, the multilevel median filter (MMF) was introduced by Nieminen et al. [75]. This class of filters employs a hierarchy of median operations that is able to preserve the fine detail that a simple median operation would remove. In [75] the general structure of an MMF was defined. A typical 2D filter structure is shown in figure 4.3 .There it can be seen that the MMF is implemented as the median of the output of median filters with different topologies. Figure 4.3 illustrates a 2 level median operation, but there is no limit to the number of levels. In the figures, the black squares of each window grid indicate the pixels used in the median operation for that window. The centre

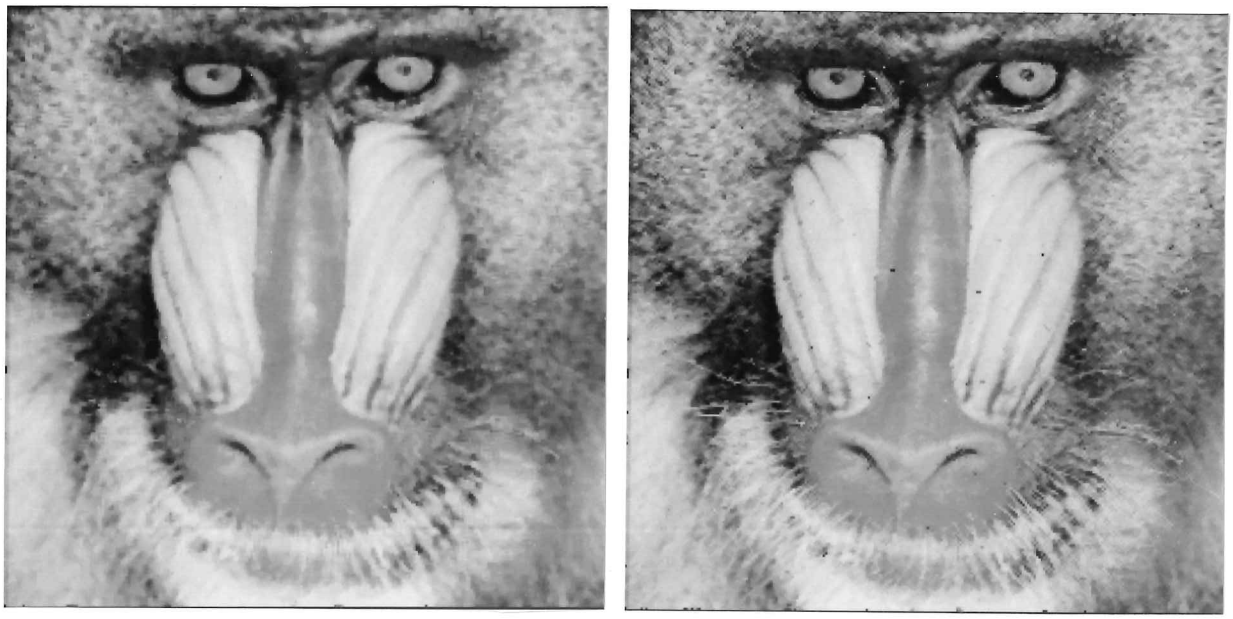

Figure 4.2: Standard 3×3 median operation (left) and Multilevel operation (right) on the distorted image.

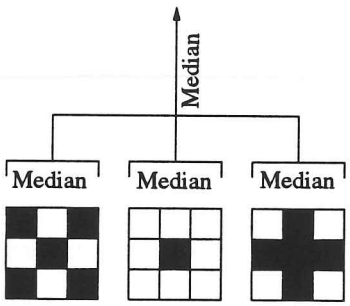

Figure 4.3: An example of a 2D multilevel median filter using 3 sub filter windows.

pixel in each window is the same pixel[2] in the image and it is that pixel which is replaced with the output of the filter.

The filter is able to preserve more detail because of the different orientations of the sub–filter masks and the inclusion of the original sample into the final median level. A typical filtering result is shown in figure 4.2. The figure illustrates the compromise that must be struck between detail preservation and impulsive noise rejection. The whiskers on the Baboon image are better preserved with the MMF than with the standard 3×3 median operation, at the expense of slightly less impulse rejection capacity. Nieminen [75] also introduced the FIR/Median hybrid filter which incorporates FIR filters in the lower hierarchy together with median filters. This class of filters is not considered in this thesis principally because of the additional computation involved.

In 1989 Arce et al. [7] introduced the use of the min/max MMF for image sequence processing. The filters were studied in more detail in [5] and were examined through their ability to reject constant impulse speckle and contaminated gaussian noise. One of the filters presented can be defined with the help of the filter masks in figure 4.4. The bi–directional filter was defined

[2]The windows share the same space on the image with respect to the filtered location.

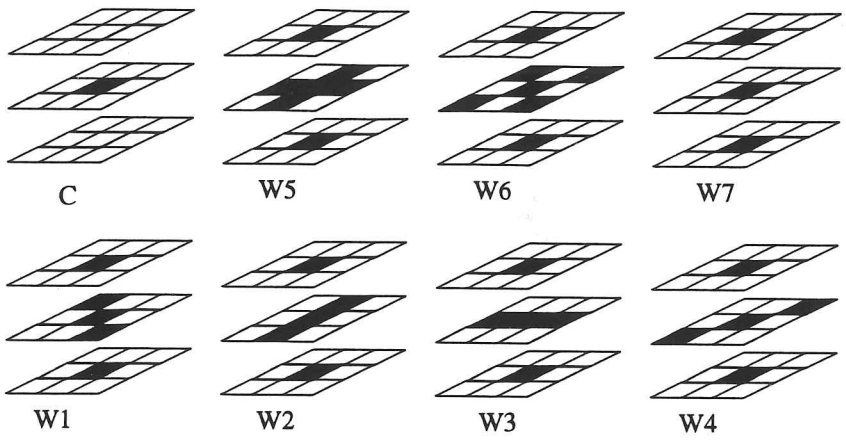

Figure 4.4: Sub–filter masks used for the MMFs presented by Arce and Alp et al.

as follows.

$$\begin{aligned} z_l &= \text{median}[W_l] \; \forall \; 1 \leq l \leq 4 \\ z_{max} &= \text{MAX}_{1 \leq l \leq 4}[z_l] \\ z_{min} &= \text{MIN}_{1 \leq l \leq 4}[z_l] \\ \text{Bi–directional Filter output} &= \text{median}[z_{max}, z_{min}, C] \end{aligned} \tag{4.10}$$

The centre window in each 3 window set refers to a window in the current image. The filters presented used data from three frames to achieve filtering. The Uni–directional filter that was also presented, was the same as the Bi–directional filter except that it used 5 windows, only one of which included information from the surrounding two frames, window 7 in figure 4.4. The other windows were the centre frame masks of windows 1–4.

The motivation for the form of the filters was to incorporate more 3D information into the median operation for image sequences. These filters would be better than the purely temporal implementation in [35, 63] because of the better impulsive noise rejection. The filters were introduced by Arce to be robust to motion, because the computational requirement of motion estimation was seen to be too large. Therefore, the filters were implemented in a direction orthogonal to the frames. When motion occurs, the filters reduce to a purely spatial operation because the pixels in the two surrounding frames would be unlikely to bear any relation to the pixels in the current position.

A similar set of MMF's was introduced by Alp et al. The windows are also indicated in figure 4.4. They utilize median operations only (see equation below). These were also not motion compensated implementations. However, because they incorporate more spatial information in each window, their impulse noise rejection capacity is better than those proposed by Arce. Their detail preservation capacity is reduced from those of Arce's for the same reason. The ML3D MMF introduced by Alp et al. is defined below.

$$z_l = \text{median}[W_l] \;\; 5 \leq l \leq 6$$

$$\text{ML3D Filter output} \quad = \quad \text{median}[z_5, z_6, C] \tag{4.11}$$

The form of the filters show that when motion occurs, the MMF again reduces to a purely spatial operation in the current frame. It would be an improvement of course to implement these filters along a motion trajectory, and that is pursued later in the thesis. Further, the topology of the MMF's introduced means that impulsive distortion greater in size than 3×3 cannot be completely removed after one pass. Further passes (i.e. a recursive implementation) would remove such distortion but adversely affect the output image quality. 'Dirt and Sparkle' is typically larger than this and so this thesis also spends some effort in introducing new MMF's that can solve this problem.

It is possible to quantify the performance of median filters if suitable simplifying assumptions are made [6, 5, 23, 26, 75]. However these analyses are of limited use in practice. Nevertheless there is growing interest in optimal adaptive order statistic filtering which quantifies the effects of this general class of filters [56, 16]. This thesis does not investigate the prospect of these optimal schemes, preferring instead to concentrate on low complexity heuristics and a linear model based approach which is outlined in the next section.

4.3.1 Model based impulsive noise removal

There has been little work in the area of impulsive noise suppression in images that does not involve median filtering, the notable exception being the work of Geman [28] and Morris [64]. For the 1D (audio) case two authors, Veldhuis [96] and Vaseghi [94], have presented practical alternatives which have been quite successful. Recently, Godsill [31] has been improving on these techniques. They consider the problem of scratch and click removal in degraded audio such as archived Gramophone recordings. The methods are similar so only the ideas of Vaseghi will be considered.

Vaseghi argues that given an appropriate model for the audio signal (he used an AR model), the signal error between the prediction and the observed signal value can give useful information. At an impulsive distortion, the error signal has an amplitude which is virtually unchanged from the observed, distorted signal, but otherwise the prediction error is quite small. Essentially, the impulsive distortion cannot be handled by the model. Therefore, the error signal can be used as a detector for impulsive noise. He goes on to show that the AR model can be used to interpolate the missing information at the detected location by minimizing the squared error of the predicted samples that would be used to fill missing gap. His algorithm shows impressive performance when used to correct degraded audio.

Veldhuis also developed a 2D AR version of the audio 'click' interpolator which is based on the same ideas as used by Vaseghi. The important point about these ideas is that by detecting the impulsive distortion, attention may be concentrated only at the degraded signal portions. This would isolate any artefacts that may be introduced in a global operation. This has impor-

tant implications for median filtering in particular since, as discussed earlier, the global median operation invariably blurs fine texture in images.

Chapters 3, 7 investigate further the 3D AR work of Efstratiadis et al [20] discussed in chapter 2. The work of Veldhuis and Vaseghi is then extended to deal with the problem of detection and interpolation of Dirt and Sparkle in motion pictures.

4.4 Summary

The work in ISR has been so sparse that it has been possible to consider in this chapter almost all the published work on this topic. Most of the work done to date involves white noise suppression. Only recently has Özkan et al. [58] introduced techniques for the optimal 3D filtering of image sequences. These techniques are superior to the purely temporal operations that have been implemented previously.

Very little treatment has been given to the problem of impulsive noise removal in image sequences. Arce [7, 5] and Alp at al. [4] represent two works that have developed 3D median filters for impulsive noise suppression in image sequences. Those filters were implemented without motion compensation and so the results did not realize the full potential of these structures. Further, the median operation, although quite successful in the suppression of impulsive noise in images, invariably introduces distortion. This distortion primarily takes the form of blurring fine image detail. The methods of Veldhuis and Vaseghi can be extended to deal with this difficulty. The basis of the ideas of Vaseghi, that of detecting the distortion, can be used to introduce some heuristics for control of the median operation. This is considered in chapter 6. Building on the work of Efstratiadis [20], it becomes possible to treat this 3D problem within a model based framework that represents an alternative to median filtering. See chapter 7.

Additional work on blotch removal is given in [104, 105].

REGISTERING THE LINES FROM NOISY DIGITIZED TV IMAGERY.

The first stage of any digital restoration process must be the conversion of the analogue video signal into a digital form. For PAL signals this implies a maximum sampling rate of 768 pixels horizontally and about 576 lines vertically per frame. To perform this sampling, the digitizing equipment[1] must synchronize to the incoming lines and frames. If the incoming video signal is a noisy one or is affected by timing distortion, then the digitizing equipment may be unable to locate the start and end of lines and frames. The result is a digitized image in which the lines are displaced relative to each other due to the bad line synchronization. The lack of accurate frame timing would cause some sort of vertical rolling effect in the digitized video. In practice, the line *jitter* effect is more common, since the line synchronization pulses are of smaller amplitude than the frame pulses and so can be affected by smaller levels of noise.

This line jitter is most frequently encountered when digitizing frames from a home video recorder. This may be due to the physical tracking mechanism of the machine playing the tape. A noisy environment causing degradation of the video signal itself as it is transmitted through the cabling between the source and the digitizer may also yield this distortion. Finally, loss of information on the video tape as it wears with age also results in the lack of synchronization. It is accepted that there now exist digital video tape recorders and players which store digital image information directly on video tape. These machines show much less jitter effects since the digital information itself is stored on tape and timing information can be recovered with accuracy. However, the tape is still subject to wear and tear and the digital video format has not yet filtered down to the non–professional consumer.

Whatever the source of the degradations, the line jitter effect is disturbing to the viewer. The jitter is typically within ± 1 pixel for a good quality frame grabber. Figures 5.6 and 5.7 illustrate some typical distortion. At present, possible solutions to the problem involve either searching for the shift between lines that maximizes the correlation between them or averaging lines. Correlation matching tends to cause a drift in the estimated shifts since it has a limited 'memory'. That is to say, such a method involves only two lines in the image and good matches across these lines may not reflect the overall image structure. Averaging does remove much of the perceived jitter, provided the jitter is within $\pm$ 1 pel, but this is at the expense of half of the vertical resolution.

What is needed therefore, is a method capable of retaining longer range vertical structure

[1] Frame grabber.

in the restored image without a loss of resolution. Such a method is developed here through a model based approach. Although there exist a variety of image models, it is thought that a simple linear autoregressive structure is adequate for this problem, especially in view of the low computational complexity of the modelling equations.

The method introduced in this chapter to provide an estimate for the relative shifts between lines is closely related to the 3D AR pel-recursive modelling framework discussed in Chapter 2. The method presented here uses a non-homogeneous 2-D AR model in a similar way that [20] uses a homogeneous model for the considered pel. The chapter will begin with a model description and then explain the registration algorithm. This work has been published in [49].

5.1 The Model

The assumption is made that the original image[2] obeys a non–homogeneous 2D AR model. This model is exactly the same form as the 3D AR model described in chapter 2, with the important exception that it uses support pixels only in the same frame. The model used is also explicitly non–homogeneous in that it is acknowledged that different areas of the image are modelled by different AR coefficients.

The situation in the original image can be defined following some of the notation introduced in Chapter 2. The model support is defined by a set of N spatial vectors $\mathbf{q_k} = [q_k^h,\ q_k^v]$. The corresponding 2D AR coefficients at a particular location[3] are given by $a(i,l)$. Therefore the predicted grey level $\hat{I}(i,l)$ at the location (i,l) is given in terms of N surrounding pixel values $I(i+q_k^h, l+q_k^v)$ as follows. The image is assumed to be zero mean.

$$\hat{I}(i,l) = \sum_{k=1}^{N} a_k(i,l) I(i+q_k^h, l+q_k^v) \tag{5.1}$$

The corresponding error between the predicted value and the actual grey level of the pixel at (i,l) is then given by

$$e(i,l) = I(i,l) - \sum_{k=1}^{N} a_k(i,l) I(i+q_k^h, l+q_k^v) \tag{5.2}$$

Changing the sign of the coefficients and defining $a_0(i,l) = 1.0$, allows a more useful expression for the prediction error.

$$e(i,l) = \sum_{k=0}^{N} a_k(i,l) I(i+q_k^h, l+q_k^v) \tag{5.3}$$

After the image frame is jittered so that the lines are displaced relative to each other, the prediction equations must incorporate this displacement. Denoting the relative displacement[4]

[2] That is the image prior to the jitter distortion.

[3] i for horizontal component, l for line number or vertical component.

[4] Always purely horizontal

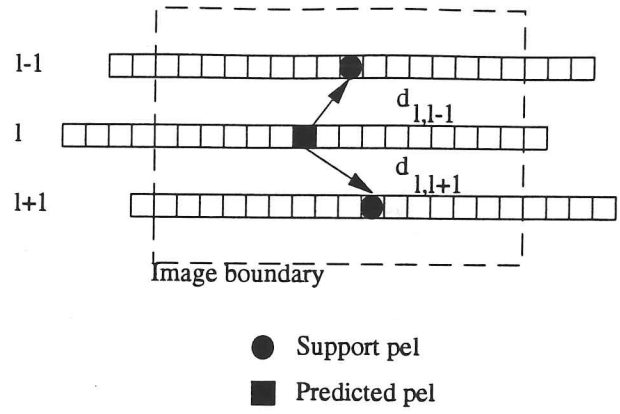

Figure 5.1: A simple 2DAR model applied to an image of 3 jittered lines.

between lines l and $l+1$ as $d_{l,l+1}$, the displacement between lines l and $l+n$ is the sum of all the displacements of the lines between, represented by

$$d_{l,l+n} = d_{l,l+1} + \sum_{k=0}^{k=n-2} d_{l+1+k,l+1+k+1} \tag{5.4}$$

In order to predict the intensity of the ith pixel at line l in the degraded image, the offset vectors $\mathbf{q}_k$ must be modified by the displacement between the current line and the line on which the support point is to be found. Thus the model of the degraded image can be written using equation 5.2 as

$$I(x,l) = \sum_{k=1}^{N} a_k(x,l) I(x + q_k^h + d_{l,l+q_k^v}, l + q_k^v) + e(x,l) \tag{5.5}$$

The model arrangement as regards the jittered lines is shown diagrammatically in figure 5.1

The object is now to estimate the displacements $d_{l,l+1}$ such that by displacing the line $l+1$ by $-d_{l,l+1}$, the original image is the result.

5.2 Displacement estimation

As has been stated, to achieve a restoration of the jittered image, the relative displacement between lines must be found. Unfortunately, using the model structure above requires knowledge of the model coefficients for the unknown original image. Therefore, the solution proposed uses an iterative process which estimates a set of coefficients given a current estimate for the displacement and then estimates a displacement based on the estimated coefficients and so on.

Each line is considered separately. It is assumed that displacement estimates for all lines are available except the estimate for the pair $l, l+1$. Therefore an estimate is required for $d_{l,l+1}$. The line to be registered (or un-jittered) is line $l+1$. To make the equations less cumbersome, a specific 2D AR model is dealt with. The model is shown in figure 5.1. It consists of two support vectors pointing to the previous and next lines. The prediction that this model gives for the pixel

at (i,l) is indicated below as equation 5.6. Note that the sign of the coefficients is changed to allow the simpler error equation 5.3.

$$\hat{I}(i,l) = -a_1 I(i + d_{l,l+1}, l+1) - a_2 I(i + d_{l,l-1}, l-1) \tag{5.6}$$

The general formulation can be derived using a straightforward extension of these ideas.

Given an initial estimate, $d^0_{l,l+1}$ for the actual displacement $d_{l,l+1}$ the current prediction error, $e_0(i,l)$ can be written (by equation 5.3),

$$e_0(i,l) = I(i,l) + a_1(i,l) I(i + d^0_{l,l+1}, l+1) + a_2(i,l) I(i + d_{l,l-1}, l-1) \tag{5.7}$$

Note that it is assumed that $d_{l,l-1}$ and the model coefficients are known. A substitution may be made for $I(i,l)$ given the model equation 5.2. This yields

$$\begin{aligned} e_0(i,l) &= e(i,l) - a_1(i,l) I(i + d_{l,l+1}, l+1) - a_2(i,l) I(i + d_{l,l-1}, l-1) \\ &\quad + a_1(i,l) I(i + d^0_{l,l+1}, l+1) + a_2(i,l) I(i + d_{l,l-1}, l-1) \end{aligned} \tag{5.8}$$

In order to estimate an update displacement for the current estimate, $I(i + d_{l,l+1}, l)$ is expanded about $d^0_{l,l+1}$ using a first order Taylor series approximation. The update displacement, $u_{l,l+1}$, is defined such that $d_{l,l+1} = d^0_{l,l+1} + u_{l,l+1}$. The expansion is then substituted in equation 5.8 to give the following relation.

$$\begin{aligned} e_0(i,l) &= e(i,l) - a_1(i,l) I(i + d^0_{l,l+1}, l+1) - u_{l,l+1} a_1 \frac{\partial}{\partial x} I(i + d^0_{l,l+1}, l+1) \\ &\quad + a_1(i,l) \nu(i + d^0_{l,l+1}, l+1) + a_1(i,l) I(i + d^0_{l,l+1}, l+1) \end{aligned} \tag{5.9}$$

$\nu(i,l)$ represents the higher order terms of the Taylor series expansion and it is considered to behave like additive white noise with variance $\sigma_{\nu\nu^2}$.

Since every pixel along line l is shifted by the same amount relative to $l+1$, the equation 5.9 at every pixel may be set up into a set of simultaneous equations to be solved for the relevant displacement. Assuming that there are M pixels along a line the observations may be arranged as follows.

$$\begin{bmatrix} e_0(0,l) \\ \vdots \\ e_0(M,l) \end{bmatrix} = \begin{bmatrix} -a_1(0,l)\frac{\partial}{\partial x}I(0+d^0_{l,l+1},l+1) \\ \vdots \\ -a_1(M,l)\frac{\partial}{\partial x}I(M+d^0_{l,l+1},l+1) \end{bmatrix} u_{l,l+1}$$
$$+ \begin{bmatrix} e(0,l)+a_1(0,l)\nu(0+d^0_{l,l+1},l+1) \\ \vdots \\ e(M,l)+a_1(M,l)\nu(M+d^0_{l,l+1},l+1) \end{bmatrix} \tag{5.10}$$

Note the important difference between $e_0(i,l)$ and $e(i,l)$. The former is the observed prediction error given the current displacement estimate $d^0_{l,l+1}$ and the latter is the actual model error given the actual displacement $d_{l,l+1}$.

Equation 5.10 can be written in a more compact form as below.

$$\mathbf{z} = \mathbf{g}u_{l,l+1} + \mathbf{v} \tag{5.11}$$

Following Efstratiadis et al. [20] and Biemond et al. [10], a Wiener solution may be derived for an estimate of u defined $\hat{u}$. The update estimate is assumed to be uncorrelated with the error signals and has variance σ^2_{uu}.

$$\hat{u} = [\mathbf{g}^T\mathbf{R}^{-1}_{vv}\mathbf{g} + \frac{1}{\sigma^2_{uu}}]^{-1}\mathbf{g}^T\mathbf{R}^{-1}_{vv}\mathbf{z} \tag{5.12}$$

This solution is similar to that considered in appendix C.

This pel–recursive solution is virtually identical to the motion estimation solution of Efstratiadis et al. [20] and Biemond et al. [10]. Nevertheless, there are two important differences.

1. Within a frame an image is highly non–stationary. Hence it is necessary to use a different model at different positions along the line. Therefore the correlation matrix R_{vv} cannot be represented by a scaled identity matrix as was the case in [20, 10]. The matrix is diagonal if the model error $e(i,l)$ is assumed to be white noise of variance σ^2_{ee}. The matrix can then be described as below.

$$\mathbf{R}_{vv} = \begin{bmatrix} \sigma^2_{ee}(0,l)+\sigma^2_{\nu\nu}(0,l)a^2_1(0,l) & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & \sigma^2_{ee}(M,l)+\sigma^2_{\nu\nu}(M,l)a^2_1(M,l) \end{bmatrix} \tag{5.13}$$

2. The model is two–dimensional and the displacement parameter required is a scalar.

Of course it is not necessary to use a different set of coefficients at every pixel. In practice the line is divided into blocks which are then assigned different model coefficients.

It is acknowledged that it may be possible to use the 2DAR model itself to treat the line l as missing and so interpolate some meaningful information[5]. However the form of the distortion is known to be the relative displacement between the lines, therefore it is thought that a method that explicitly attacks this problem would be superior.

[5]See Chapter 7

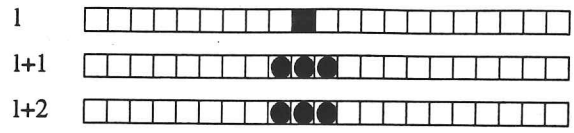

Figure 5.2: The model used in the experiments. A darkened circle represents a support pixel and a blackened square a predicted pixel.

5.3 Implementation

First of all the support used in the model is restricted by the assumption made about the correlation structure of the model error $e(i,l)$. The correlation terms used in $\mathbf{R}_{vv}$ all concern horizontal offsets. The assumption about the error implies that all the pure horizontal offset correlation coefficients are zero. These terms are only zero if the support of the model is causal or if not, does not include taps in the current line. This is analogous to the situation with regard to 3D AR modelling in chapter 2 and it is due to the fact that the error from a non–causal AR model is not white. (See appendix B). These statements are justified by considering the correlation function of the residual (or error) from a non–causal AR process stated below

$$r_{ee}(\mathbf{x},\mathbf{q}_n) = \begin{cases} \sigma_{ee}^2 & \text{for } \mathbf{q}_n = [0,\ 0] \\ a_n\sigma_{ee}^2 & \forall n = 1\ldots N \end{cases} \tag{5.14}$$

It is assumed that the other correlation terms not defined by the above equation are zero. The terms in the correlation matrix $\mathbf{R}_{vv}$ that refer to the model error are of the form $r_{ee}(\mathbf{x},\mathbf{x}+[k,\ 0])$. They are terms that apply to the horizontal lags only since the observations are assembled from line l only. Therefore, as long as the model does not contain taps in this line, there would be no contribution from the term $a_n\sigma_{ee}^2$ in the expression 5.14.

Because the model error variances vary from block to block across a line, the simple regularized solution of Biemond [10] is unavailable here. However, in practice it is possible to use the variance of the current prediction error as an estimate for variance of the actual model prediction error, especially since the line jitter is typically ± 1 pixel.

The problem remains of estimating the AR coefficients. The proposed iterative procedure estimates the coefficients at each iteration using the Normal Equations prior to the displacement estimation. The equations are solved exactly, as is described in Appendix A for the 3D AR case.

During the iterative procedure, it is necessary to evaluate pixel intensities at fractional displacements. There are a variety of techniques available to do this. Bilinear interpolation is found to cause too much blurring and instead 3rd order piecewise polynomial interpolation[6] is used.

The algorithm may be enumerated as follows

1. Fragment the current line l into a set of segments of size N pixels.

[6] A crude but effective interpolator that fits an nth order polynomial to n points around the interpolated position.

2. Associate with each segment a surrounding block from which to estimate the model parameters: size $N \times N$.

3. Set the current displacement to be 0 pixels.

4. Estimate the model parameters for each block using the Normal equations.

5. Solve for the update using equation 5.12. The gradients are calculated using a difference technique as indicated in Chapter 2. An estimate for the unknown model error variance can be gained by measuring the variance of the prediction error over the blocks associated with each line segment. A fixed value for σ_{uu}^2 is assumed.

6. If $\sum |\mathbf{z}| \leq z_t$ where z_t is some previously set threshold, then halt the process. The correct displacement has been found. Alternatively the MSE may be used. The Sum Absolute Error was used in the experiments because of the simpler computation.

7. If $|u| \leq |u|_t$ then assume that the algorithm has converged and halt the process.

8. Update the displacement: $\hat{d}_{l,l+1}^{i+1} = \hat{d}_{l,l+1}^{i} + u_{l,l+1}$

9. If the number of iterations has exceeded a maximum allowed limit then halt the process.

10. Goto 4.

After the algorithm has terminated, line $l+1$ is shifted by $-\hat{d}_{l,l+1}$. Then the next line is considered until all the lines in the image have been treated. In practice it is better to limit the maximum estimated displacement and perform several passes over the entire frame. This will displace all the lines by small amounts with each pass, eventually converging on a smooth image. Limiting this displacement prevents the reconstructed image showing gross lateral drift over parts of the image.

5.4 Results

Figure 5.3 shows the results from a restoration of an artificially jittered Lena subimage of resolution 128×128. Ten blocks of size 10×10 samples were used across the image, six estimation points were used per block. Each block had independently estimated AR coefficients using a model support as shown in figure 5.2. The jitter was made to occur with a probability of 0.5 at each line and at each jitter occurrence, the displacement was uniformly randomly distributed within ± 1.0 pel. Three iterations over the entire image were needed, at most 4 iterations per line. The values used in the respective iterations were $\sigma_{uu}^2 = 1.0, 1.0, 1.0$, maximum allowed $d_{l,l+1}^{\circ} = 1.0, 1.0, 1.0$ pels, $|u|_t = .02$ and $z_t = 480, 600, 530$. The process was started at the 10th line and terminated at the 123rd line. Third order piecewise polynomial interpolation was used. The quality of the final image is superior to the artificially jittered image although there is some distortion in the angle of some of the features in the lower quarter of Lena's hair and in the

Figure 5.3: Left: Artificially jittered lena. Right: Restored using 2DAR model.

forehead. The discontinuity at the lowest part is where the processing was stopped. This jitter is much more severe than encountered in practice, but it serves to highlight the effects of the process. The original Lena image is shown in figure 5.4.

The results for Lena are compared to two alternative techniques. One involves vertically weighted averaging of pels in the image such that the intensity on line l, $I(l)$, is replaced by the intensity $0.25I(l-1) + 0.5I(l) + 0.25I(l+1)$. The other chooses the displacement that gives the maximum correlation between each pair of lines. The search was done in steps of 0.1 pel to a displacement of ± 2.5 pels. Bilinear interpolation was used. The correlation is measured as $\sum_i I(i,l)I(i+d_{l,l+1}, l+1)$. The averaging method does remove the jitter but causes blurring, see figure 5.5. The correlation method is unable to choose the correct displacements and is biased toward vertical structures. See figure 5.5. The Lena image here shows bad distortion in the hat.

Figures 5.6 and 5.7 show the result from restoration of two actual 128×128 subimages taken from a frame captured with an EPIX framestore into an IBM PC AT. The video signal was input from a Umatic VTR in PAL format and the line 5 to 125 were registered. The improvement is noticeable and the effect pleasing to the eye. Attention is drawn to the eyes and chin of figure 5.6. Figure 5.8 shows the detail around the eye in a zoomed section. Figure 5.7 shows a cartoon image again with jitter. The method is less successful here and cannot remove all the jitter. A zoomed portion is shown in figure 5.8. The algorithm used the model as in figure 5.2 for both these images. The first image required 1 pass over the whole image while the second required 3. In the first case the displacement was limited to ± 1.0 while in the second case this was set to 0.5 for all 3 passes. $z_t = 350, 550$ respectively. In both cases the update threshold was set to $|u|_t = .02$.

Figure 5.4: Original Lena

Figure 5.5: Left: Restoration by correlation. Right: Restoration by averaging.

Figure 5.6: Left: Real jittered image 1. Right: Restoration by AR modelling.

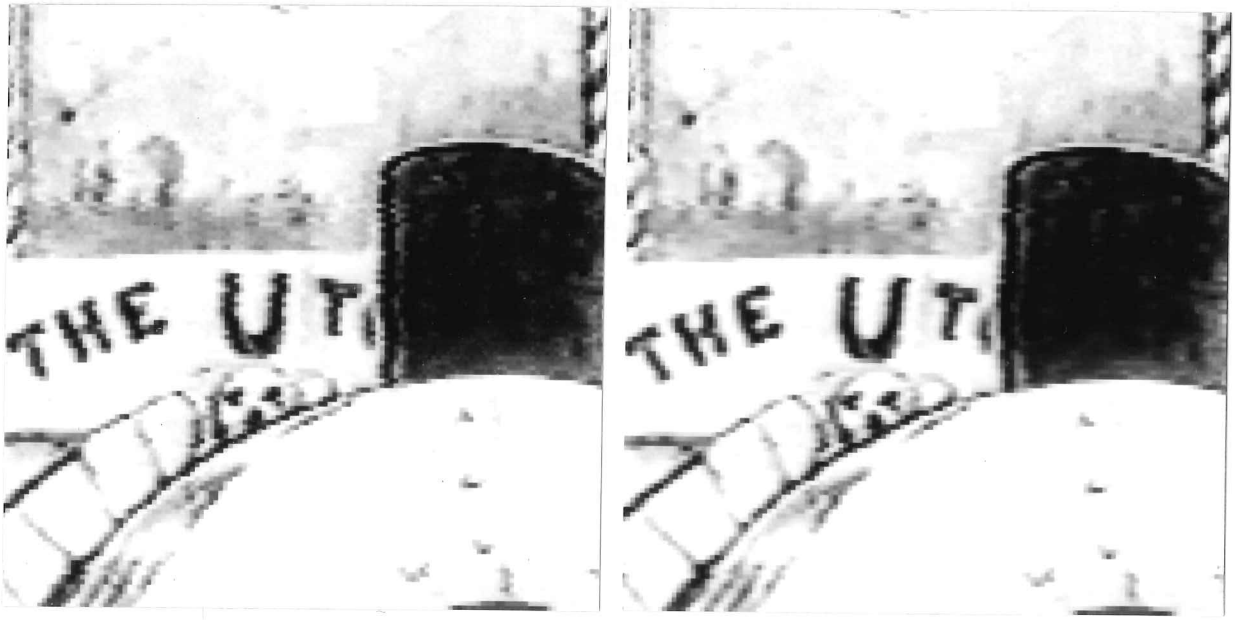

Figure 5.7: Left: Real jittered image 2. Right: Restoration by AR modelling.

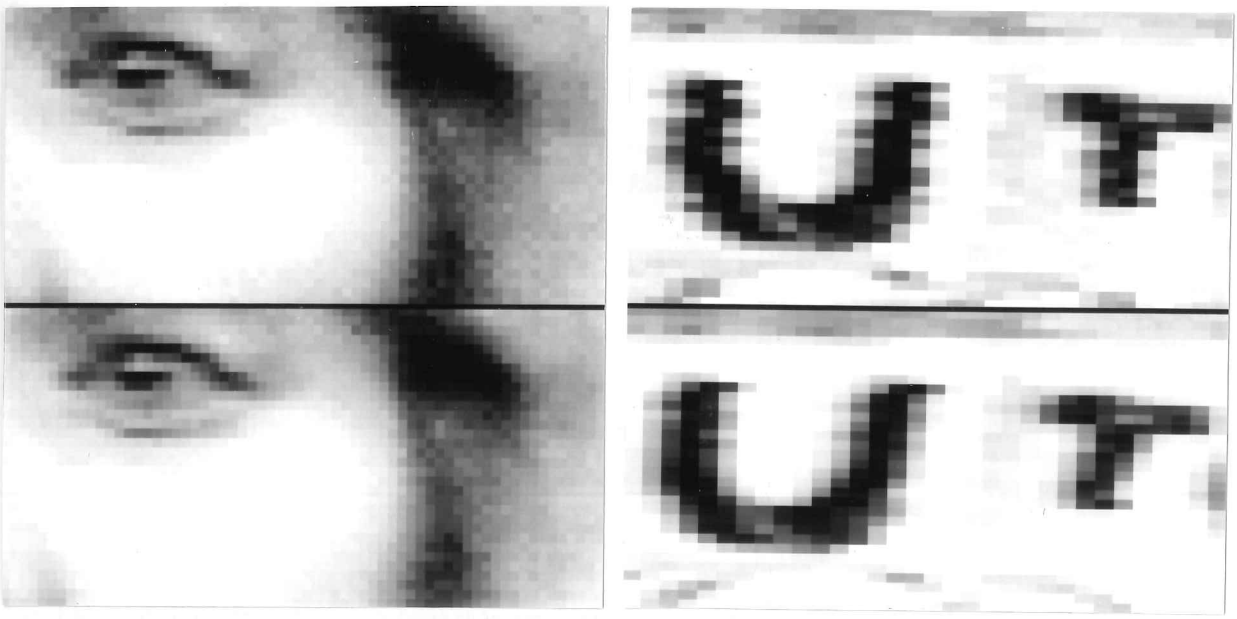

Figure 5.8: Zoomed portions of images 1,2. Top: Original; Bottom: Restored.

5.5 Summary

An algorithm has been presented for solving a common problem that occurs when capturing frames. The algorithm has been demonstrated to be quite effective in removing the disturbing inter-line jitter that is often observed in captured frames. The success of the method lies primarily with the relatively good modelling capacity of the non-homogeneous 2D AR process and that the jitter between lines causes only relative horizontal shifts and no other distortion. Unfortunately, the algorithm requires several parameters for successful operation. It would be an improvement to consider estimating these parameters from the image data. Further, there is still some tendency for the registered lines to drift slightly as the algorithm proceeds. This is evidenced by the not quite perfect restoration of the Lena image. The last chapter in this thesis considers possible modifications that may prevent this.

SPATIO–TEMPORAL MEDIAN FILTERING FOR SUPPRESSING LOCAL DISTORTION IN IMAGE SEQUENCES

One of the most common and striking distortions in archived motion picture film is the presence of scratches. These scratches may take the form of regions of high contrast which appear at random positions in the frame. This chapter presents one technique for removing the random blotches (hereafter referred to as Dirt and Sparkle) from such degraded image sequences. The problem of line scratches is a separate problem and is left for future work which is discussed in Chapter 9. This is due to the fact that line scratches tend to be correlated from frame to frame.

An important consideration in the treatment of any real distortion is an appreciation of the agents that cause the problem and the manifestation of the degradation. Dirt and Sparkle on film is caused by the physical interaction of the film material with the projecting equipment as it is transported through the mechanism. These distortions will occur also in Telecine equipment since the mechanism of film transport is the same. Dirt and Sparkle would then occur on the video sequences as well. The abrasion is not a normal side–effect of the equipment, it can be caused by foreign particles caught in the mechanism. Because the film material is abraded, bright flashes of light (in the case of accumulation of particles, flashes of dark) are seen because of the projector light source. The distortion is clearly a local one and there are large areas of each frame which are not affected at all.

Dirt and Sparkle can be effectively modelled as randomly distributed Impulsive distortion which *replaces* the image information in selected regions. These distortions are not limited to single pixel impulses but can also occur as variable sized patches of uniform grey level which are highly contrasted with the surrounding image. These patches represent regions of missing information in the frame. They will be referred to as *Blotches* in the text.

As a first step toward a low cost treatment of this distortion, Median filters are considered. Chapter 4 has already outlined the advantages of three dimensional motion compensated operations for video processing. This chapter serves not only to illustrate this point but also to introduce new 3D Median filter structures which deal effectively with Dirt and Sparkle.

The discussion begins with a treatment of single pixel impulsive distortion and then moves on to consider the more realistic problem of removing randomly distributed blotches.

Parts of the work reported here have already been presented in[51, 50].

6.1 Motion Compensated Median Filtering

Chapter 4 has already outlined the work of Alp [4] and Arce [7, 5] who have both previously introduced 3D median filter structures for removing impulsive noise. The structures were introduced without a motion compensated implementation. Both Huang [35] and Martinez [63] implemented a 3 tap motion compensated median operation with good results.

In order to illustrate the advantages of motion compensated median operations the case of a perfectly compensated artificial sequence is considered. Figure 6.1 shows a 3 frame test sequence of a moving damped two dimensional cosine function. It is the same form of image used in experiments performed by Netravali et al [73] and will be referred to as the *eye* sequence. The background in the *eye* sequence was set at 20 and the maximum value at the centre of the *eye* set at 60. The equation below was used to generate the *eye* in each frame, the coordinates of the centre of the eye are (cx, cy).

$$\begin{aligned} R &= |(i-cx)^2 + (j-cy)^2| \\ I(x,y) &= 60.0\exp(-0.05R)\cos(0.2\pi R) \end{aligned} \tag{6.1}$$

The generated shape was limited to a radius of 12 pixels and was displaced by 5 pixels, in both spatial directions, in each frame. The frames were of size 64×64 and 6 bit grey scale resolution.

For this experiment, the sequence was degraded with single pixel impulsive distortion following the manner stated in Chapter 4, and repeated here as

$$\begin{aligned} g(\mathbf{x}) &= \begin{cases} I(\mathbf{x}) & \text{with Probability } 1-P_s \\ s & \text{with Probability } P_s \end{cases} \\ s &= \text{Uniformly Distributed Grey level} \end{aligned} \tag{6.2}$$

In this equation, $I(\mathbf{x})$ is the intensity at position $\mathbf{x}$ in the original image, $g(\mathbf{x})$ the intensity at the same position in the degraded image, and s the uniformly distributed grey scale value.

The probability of a spike was increased from 0 to 0.5 in steps of .05 and the accuracy of the filtered output was measured as the percentage square error, PSE.

$$\text{PSE} = 100\frac{\sum[\hat{I}(\mathbf{x}) - I(\mathbf{x})]^2}{\sum[g(\mathbf{x}) - I(\mathbf{x})]^2} \tag{6.3}$$

Where $\hat{I}(\mathbf{x})$ is the filtered output. Each PSE value calculated for the graphs was taken as the average of 50 runs of various noise realizations at each probability of impulsive distortion, P_s.

Since this work eventually uses a BM motion estimator, the experiments illustrating perfect motion compensation simulated the best performance of a Block Matching estimator. Hence the frames were split into 8×8 sub-blocks and each block was assigned a motion vector depending on the known motion of the object within it. Blocks containing background were assigned a zero

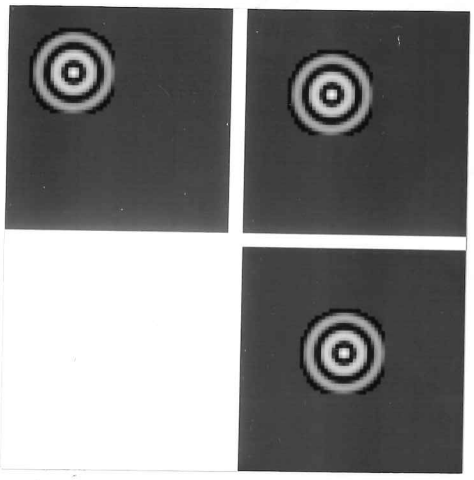

Figure 6.1: The original EYE sequence.

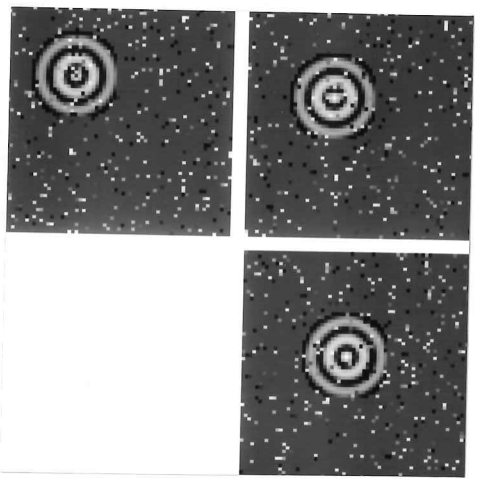

Figure 6.2: A degraded EYE sequence, $P_s = 0.1$.

motion vector and those with all or part of the object within them were assigned the vector $[5, 5]$. This obviously causes some conflict with the actual performance of a BM system. In uncovered or occluded regions, the BM estimator can yield a good match, which does not represent true motion, if the search space allows it to match these regions with other background areas. As far as this uniform background EYE is concerned, an actual BM estimator may provide slightly better results by being able to find good matches for the occluded areas. Nevertheless one would expect an excellent motion estimator to assign NULL vectors to the occluded regions or at least flag them as not having a valid motion component.

Figure 6.1 shows the original sequence used. The centre of the EYE begins at coordinates $19, 19$ in the first frame. Figures 6.2 shows the sequence degraded with an impulse probability of 0.1.

6.1.1 The Filters

For convenience, the filters that were reviewed in Chapter 4 are defined again here with the help of the windows in Figure 6.3. Arce's bi–directional filter (BI) was defined as follows.

$$\begin{aligned} z_l &= \text{median}[W_l] \; \forall \; 1 \leq l \leq 4 \\ z_{max} &= \text{MAX}_{1 \leq l \leq 4}[z_l] \\ z_{min} &= \text{MIN}_{1 \leq l \leq 4}[z_l] \\ \text{Bi–directional Filter output} &= \text{median}[z_{max}, z_{min}, C] \end{aligned} \tag{6.4}$$

The centre window in each 3 window set refers to a window in the current image. The filters presented used data from three frames to achieve filtering.

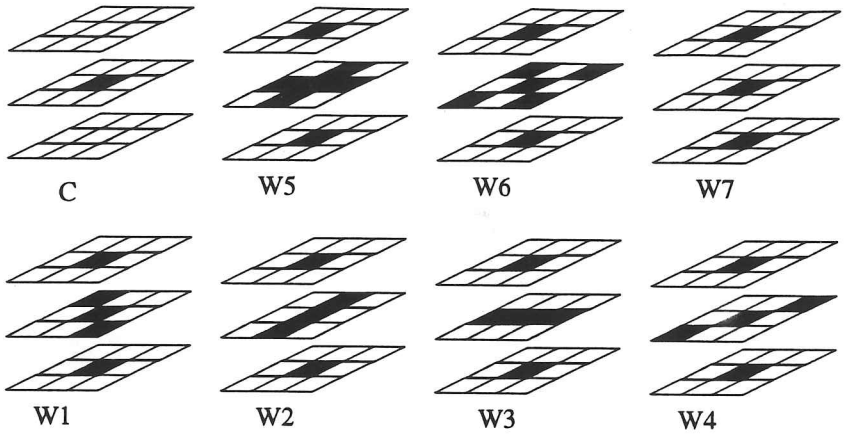

Figure 6.3: Sub–filter masks used for the MMFs presented by Arce and Alp et al.

The ML3D MMF introduced by Alp et al. is defined below.

$$\begin{aligned} z_l &= \text{median}[W_l] \quad 5 \leq l \leq 6 \\ \text{ML3D Filter output} &= \text{median}[z_5, z_6, C] \end{aligned} \tag{6.5}$$

The filter used by Huang and Martinez was a motion compensated three tap filter (LIN), and is the median of the samples shown in window 7 of figure 6.3.

A fourth filter, the CUBE, is also considered. Its output is the median of all the pixels in a $3\times 3\times 3$ volume centred on the pixel to be considered. The filter is used to compare a simple full 3D median operation to the multilevel filters, and hence to demonstrate that the 3D multilevel filters are more effective than the simple median operation even in a motion compensated application.

The motion compensated application of these filters involves selecting the pixels required for each window as directed by the motion between the frames. Hence if the pixel at $\mathbf{x} = (i, j)$ in frame n is to be filtered using the LIN filter, the output is defined as below,

$$\hat{I}(\mathbf{x}, n) = \text{median}[\, g(\mathbf{x}, n),\ g(\mathbf{x} + \mathbf{d}_{n,n-1},\ n-1),\ g(\mathbf{x} + \mathbf{d}_{n,n+1},\ n+1)] \tag{6.6}$$

where $\mathbf{d}_{n,n-1}$, $\mathbf{d}_{n,n+1}$ are the motion vectors mapping information at $\mathbf{x}$ in the current frame into the previous and next frames. Similarly for the MMF's that were defined above. The filters are applied to every pixel in the image.

6.1.2 Results and Discussion

Figure 6.4[1] shows the PSE using the CUBE, LIN, BI and ML3D filters with and without motion compensation. These experiments were done on the EYE sequence with 8×8 block sizes. Also plotted are the results using a BM scheme on the degraded images which used the same block sizes. The motion threshold varied linearly with the spike probability from MAE = 0.0 to MAE = 10.0. The search space for the BM scheme was ±6 pixels and the estimate was integer accurate. This

[1]The comparison with figure ML3dex is referred to later in the text.

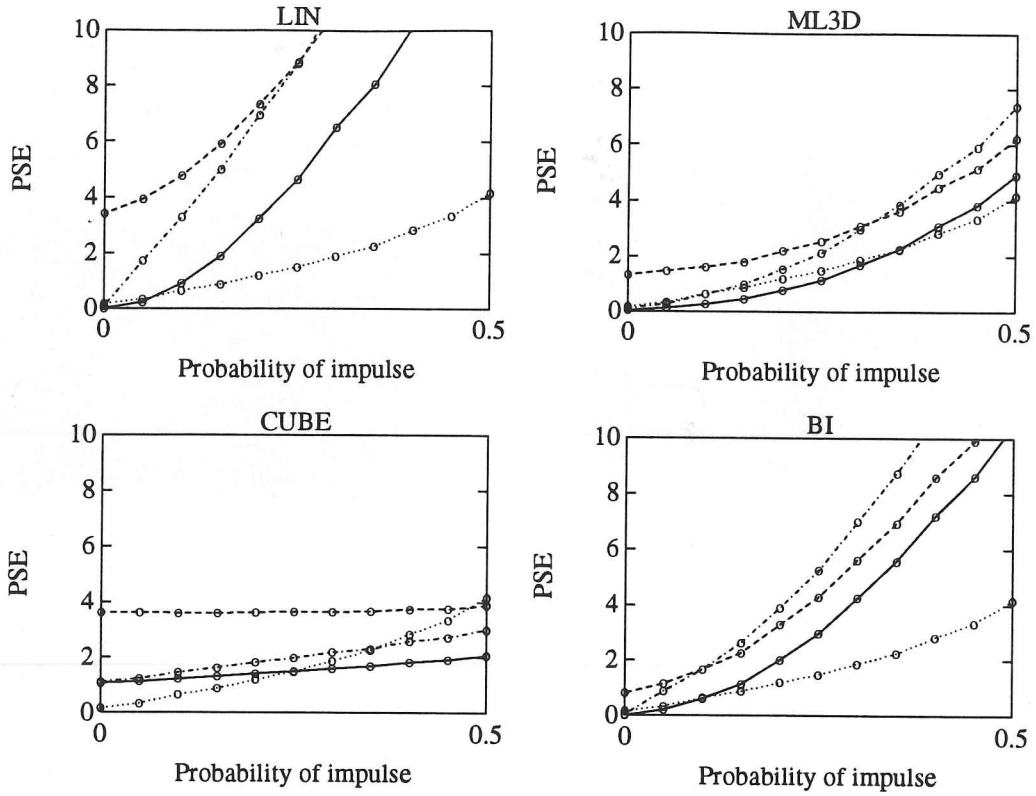

Figure 6.4: Filter performances for EYE: Dashed–no compensation, Solid–perfectly compensated, Dash dotted–Real compensation with Block Matching, Dotted ML3Dex–Real compensation with BM

was done to show how a real motion estimator affects the filter output. It is certain that all the objects in this sequence are moving and so the only purpose of the motion threshold in this experiment is to make the BM method robust to the impulsive degradations. For a spike probability near zero, very little distortion is found and so a low motion threshold is acceptable. The opposite is true at a high spike probability. It may be argued that the thresholds actually used are not sufficient for robustness to impulsive noise. However, this experiment is intended only to show that motion compensated application of the MMF's introduced is useful.

It is evident from figure 6.4 that there is much to be gained with motion compensation. The reduction in the PSE with perfect motion compensation is as much as 50%. Of course, as the impulse densities increase, the restorations are adversely affected since more and more of the pixels in each median subfilter become corrupted.

With the BM estimator, the performance at first is better than without motion compensation but gradually gets worse with higher spike probabilities. This is due to the block matching process becoming increasingly affected by the noise, eventually yielding erroneous motion vectors. The BM process can match the noise in two frames giving a vector not representing true motion. In such cases, the median filter operation is less likely to be successful since the faulty matching process effectively causes the noise correlation between the compensated blocks to become more significant than the underlying image portions. Although there is the potential for these filters

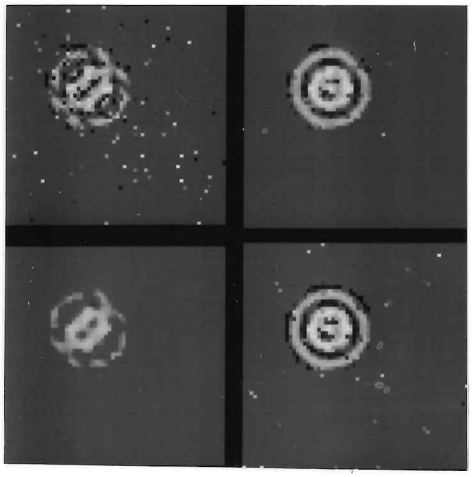

Figure 6.5: No Motion Compensation, Clockwise from top left: LIN, ML3D, BI, CUBE : $P_s = 0.1$.

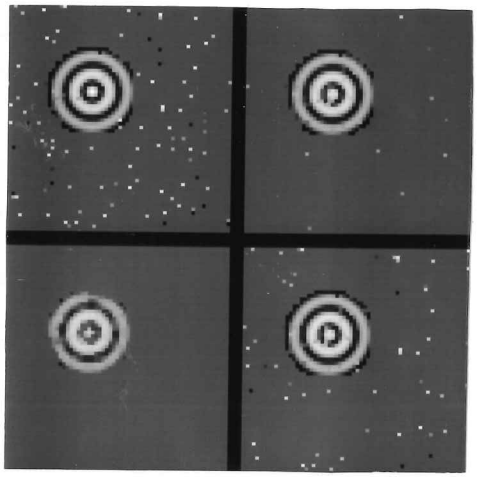

Figure 6.6: Compensation using BM, Clockwise from top left: LIN, ML3D, BI, CUBE : $P_s = 0.1$.

to perform very well under a motion compensation scheme they are sensitive to incorrect motion estimation. The BM curve on each plot therefore moves away from the perfect curve to the no motion compensation curve with increasing impulse probabilities.

Figures 6.5 and 6.6 show typical output frames when $P_s = 0.1$. They represent the filtered second frame of the sequence shown in figure 6.2. The pictures show that without motion compensation the object features are blurred. Figure 6.6 shows the result using the BM motion estimator (i.e. not perfectly compensated). Note that in figure 6.6, the impulse rejection capacity of the ML3D filter is greater than that of the BI filter although it distorts more of the outer ring of pixels. With motion compensation, the improvement in image quality is appreciable although more impulses remain. This result is due to the practical shortcomings of the motion estimator since it can match impulses in two frames and as explained before, reduce the effectiveness of the filters.

The LIN filter distorts the image heavily without motion compensation. This is expected since it is based purely on temporal information. Without compensation for motion the two pixels in the outer frames used by this filter bear little relation to the central filtered position. Therefore the output data is error prone. The CUBE filter is included to show that the simple median operation does not perform well here. Although the CUBE can reject many more impulses, the degradation in detail is too heavy. This is reflected in the PSE curves which show that at low impulse probabilities the CUBE is worse than the other filters because it cannot preserve detail well, but at high probabilities it is better because it rejects more distortion.

The ML3D and BI filters do not fail without motion compensation because in such conditions[2],

[2]Where the motion is larger than the size of the spatial median window

the pixels in the two outer frames would be outliers in the distribution of the samples used in the median windows. Therefore, the 3D multilevel median operation reduces to a 2D multilevel median operation. This window topology implicitly rejects the information in frames $n+1$ and $n-1$ if the data is not sufficiently compensated. As can be seen in figures 6.5 and 6.6, this implies that they perform quite well in the regions of the image which are not moving, and when implemented in a motion compensated application, they are robust to motion estimation errors. In general, the ML3D filter appears to offer the best compromise between noise rejection and detail preservation.

Of course, this sequence represents an artificial situation. The next section presents results for real sequences where the situation is seen to remain the same.

6.2 An Improved spatio–temporal MMF

The previous section has shown that motion compensated MMF implementations can provide an appreciable improvement in restored image quality. Despite the extra computation involved, it is a worthwhile pursuit. This is important in the case of TV imagery in which the motion is not small, therefore causing the filters above to tend to a purely spatial operation and reduce the detail in the output. However, if motion estimation is being performed, then it would be advantageous to incorporate more spatial information from the outer frames into the sub–filter windows. This would help increase the probability of impulse rejection and potentially improve the detail preservation capacity of the filter.

Furthermore, the MMF's introduced so far cannot remove an impulsive distortion which is larger than the size of the spatial median window in the current frame. Consider the ML3D filter in a situation where it operates on some occurrence of Dirt or Sparkle which is of size 3×3 pixels in the current frame. Assume for simplicity that the image itself is just a uniform grey level covering a large area. When the filter is applied to the centre pixel in the distorted region, the portion of the median subfilter windows in the current frame would be filled with 'Blotched' pixels. In each of the two spatio–temporal windows used in ML3D, there are 7 pixels involved, 5 of which are in the current frame. The output of the filter is the 4th sample in the ranked distribution of these 7 pixels. If 5 are corrupted, then it is impossible to yield a median output which is not one of these corrupted pixels. Since Dirt and Sparkle usually occurs as a region of high contrast which is uniform, the output pixel would be of the same intensity.

However, this would only occur inside the distortion. At the edges of the degradation, where less of the current frame window is occupied by distorted pixels, the output is more likely to be some image value. Therefore, both BI and ML3D will partially remove large sized distortions. It would take a recursive[3] implementation of these filters to completely remove artefacts such as these. Such a recursive implementation is likely to affect the detail in the rest of the frame adversely. Incorporating more information from the surrounding frames would be more successful

[3] Apply the filter again to the output image.

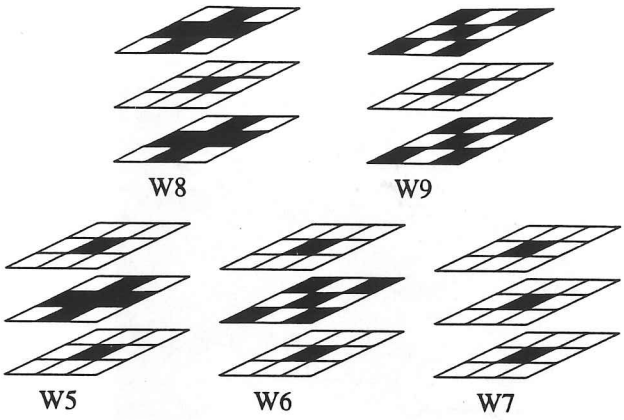

Figure 6.7: Sub–filter masks used for the new MMF: ML3Dex.

in removing distortion provided the Blotch does not occur in the same place in successive frames. In general this is true for Dirt and Sparkle.

The proposed filter structure is therefore defined with the help of the sub–filter windows shown in figure 6.7. The MMF is similar to that of Alp et al. [4] and is called ML3Dex for Extended ML3D. The output of the filter is defined below.

$$\begin{aligned} z_l &= \text{median}[W_l] \quad 5 \leq l \leq 9 \\ \text{ML3Dex Filter output} &= \text{median}[z_5, z_6, z_7, z_8, z_9] \end{aligned} \tag{6.7}$$

Two additional windows have been incorporated which contain minimal information from the current frame and extensive information from the outer frames. Consider once more the situation with a large 'Blotch'. It can be seen that although the windows, $W5$ and $W6$ would output Blotch values inside the degraded area, the three windows $W7$, $W8$, $W9$ would still be able to yield a correction using image data. Therefore, provided that the Blotch does not occur at the same position in the three frames, the 5 value 2nd level median is not dominated by scratch data. Furthermore, the additional information improves the scratch rejection capacity of the filter. This is accompanied by a subsequent loss of detail preservation.

6.2.1 Results and Discussion

Figure 6.8 shows the result using a motion compensated ML3Dex to filter frame 2 of the 3 frame sequence shown in 6.2. The output using BI, ML3D and CUBE is also shown for comparison. It is clear from the output of the CUBE, that even in a motion compensated application, taking the simple median of a volume of pixels is not as useful as a 3D MMF. The CUBE blurs the EYE badly. However, the ML3Dex filter, not only rejects the impulses well, but also preserves the detail in the frame. Note however, that the outer ring of the EYE is less complete than in the output of the BI or ML3D filters. This emphasizes that improved impulse suppression comes at the expense of detail preservation. It is interesting to note that ML3Dex uses the same set of pixels as CUBE, yet its output is superior. Figure 6.4 also includes a curve showing how the

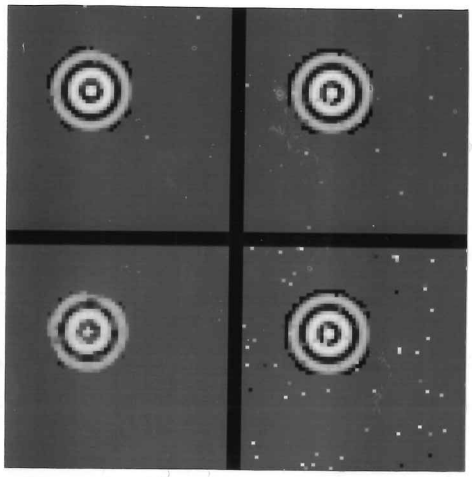

Figure 6.8: Motion Compensation with BM, Clockwise from top left: ML3Dex, ML3D, BI, CUBE : $P_s = 0.1$.

PSE of the ML3Dex restoration compares with the filters discussed previously. It indicates that the impulse noise suppression of the filter is much better than that of the other filters, even when the motion field estimated is not ideal.

Figures 6.9, 6.10 show results comparing the performance of various motion compensated filtering systems using one of the LIN, CUBE, BI, ML3D or ML3Dex filters on the Cheers sequence with artificial distortion. The measure used is the root mean square (RMS) error between the filtered and original sequence in each frame. A line showing the RMS of the corrupted sequence is drawn for reference. The sequence was corrupted with uniformly distributed impulses occurring with probabilities, $P_s = 0.1$, and 0.005.

The motion estimation algorithm used a multiresolution scheme employing 3 levels. The BBM algorithm was used for motion estimation, as discussed in Chapter 2. Integer accurate estimates were generated using a ± 4 pixel search space at each level. A block size of 5×5 was used at the lowest resolution level, and 9×9 at the other levels. The noise ratio[4], r, required for the BBM was 1.2 for the 2 lower resolution levels when $P_s = 0.1$ and 1.0 otherwise. For the highest resolution level, when $P_s = 0.1$, $r = 1.5$ and when $P_s = 0.005$, $r = 1.2$. The motion estimates were generated from the corrupted sequence. The application of the filters requires a motion vector at every pixel. Therefore, the motion vectors generated from the block based algorithm were interpolated using Bilinear interpolation to give the larger vector set required. It is agreed that the pixels could have been assigned vectors also on a block basis, but it was found that generating a smooth vector field using Bilinear interpolation, gave improved results. A border of 10 pixels around each frame was ignored in making measurements of the RMS error between the original and filtered images.

[4]Boyce ratio

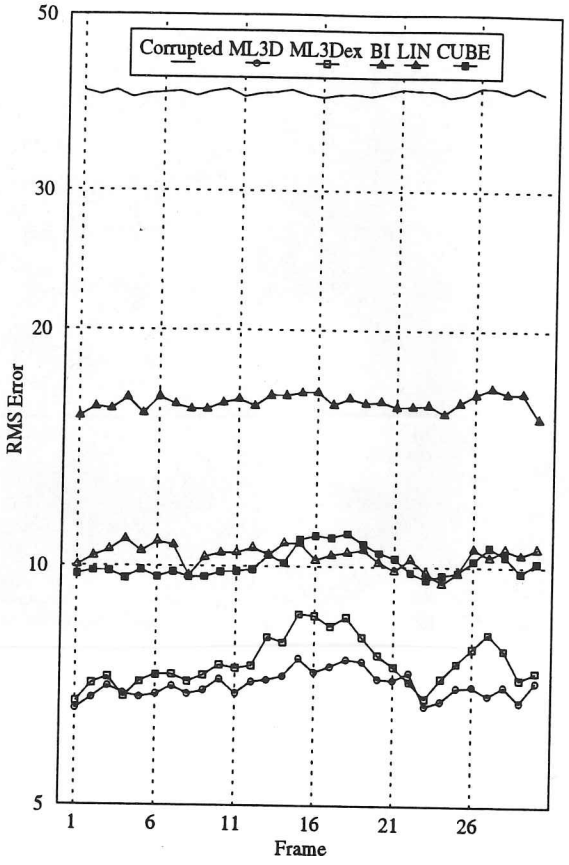

Figure 6.9: RMS Error of filtered frames from CHEERS, $P_s = 0.1$.

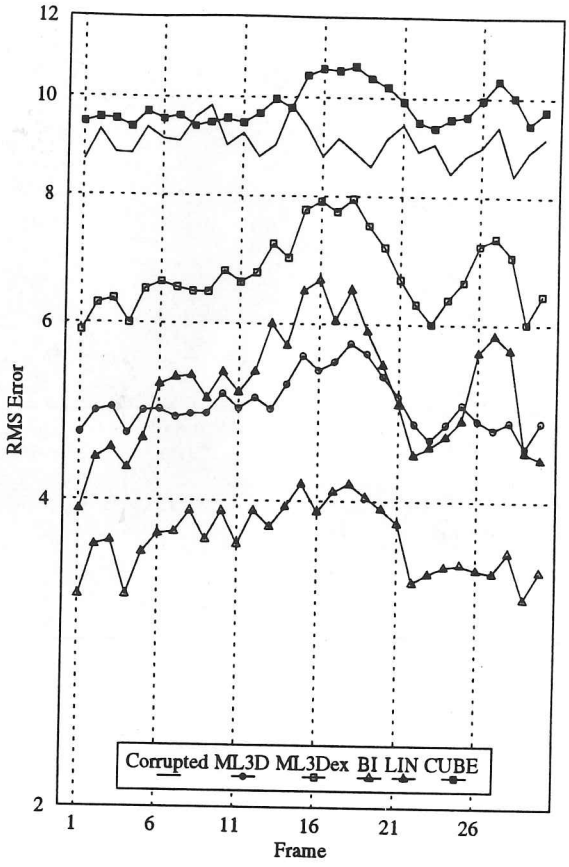

Figure 6.10: RMS Error of filtered frames from CHEERS, $P_s = 0.005$.

Figure 6.11: Frame 12 of CHEERS (original)

Figure 6.12: Zoomed Frame 11 of CHEERS (original)

Figure 6.13: Zoomed Frame 12 of CHEERS (original)

Figure 6.14: Zoomed Frame 13 of CHEERS (original)

Figure 6.15: Zoomed Frame 12 of CHEERS $P_s = 0.1$

Figure 6.16: Zoomed Frame 12: ML3D

Figure 6.17: Zoomed Frame 12: BI

Figure 6.18: Zoomed Frame 12: ML3Dex

Figure 6.19: Zoomed Frame 12: CUBE

Figure 6.20: Zoomed Frame 12: LIN

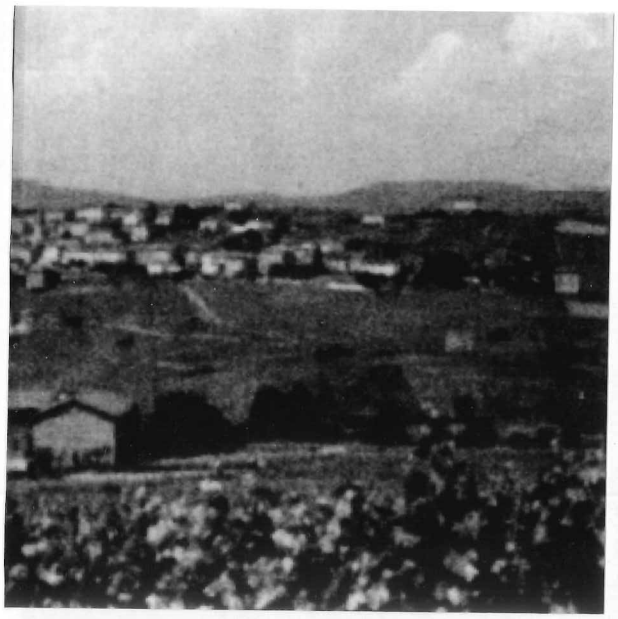

Figure 6.21: Frame 1 of CLOCHE

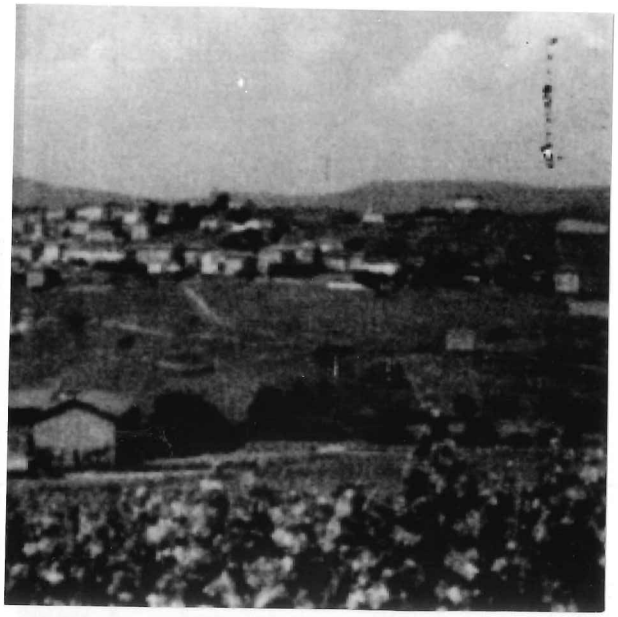

Figure 6.22: Frame 2 of CLOCHE

The figures 6.9, 6.10 indicate that the corruption level influences which filter performs well in terms of RMS. When the level of corruption is low, i.e at 0.005, the BI filter performs best overall because it can preserve detail well. However, when the corruption level is high, preserving detail is not enough and ML3D and ML3Dex become best performers because of their good compromise between detail preservation and impulse suppression. Even in terms of RMS the CUBE filter is seen to perform very badly, managing to increase the RMS above the level of corruption when $P_s = 0.005$. The LIN filter performs well only when the level of corruption is low. When the impulses occur with a high rate there is an increased occurrence of multiple corruption along a motion trajectory and so this filter is less effective.

The photos in figures 6.11 to 6.20 complement the plots in demonstrating the quality of the output from these filters. Zoomed portions of frame 12 are used to show the effects more clearly. From these figures it can be seen that the ML3D (figure 6.16) and ML3Dex (figure 6.18) filters are the best filters overall in terms of the compromise they strike between noise removal and detail preservation. The ML3D filter in particular is well suited to this type of pixel sized degradation.

Figures 6.21 to 6.25 illustrate the improvement gained with respect to rejection of typical TV distortion; Dirt and Sparkle. Three frames from a sequence (figures 6.21, 6.22, 6.23) grabbed from a live TV broadcast are shown. The frames are corrupted with Blotches[5]. The frame in figure 6.22 was filtered using the ML3D and ML3dex filters (figures 6.25, 6.24) respectively. It is clear that the new filter does remove all the distortion at the first pass whereas the other can only do so partially. Nevertheless the new filter removes image detail to a larger extent than the ML3D filter. Of course the CUBE filter would also remove the distortion, but at the expense of much of the image detail. The LIN filter can also perform very well here, but it is not robust

[5]Note the severe corruption at the top right hand edge of frame 2

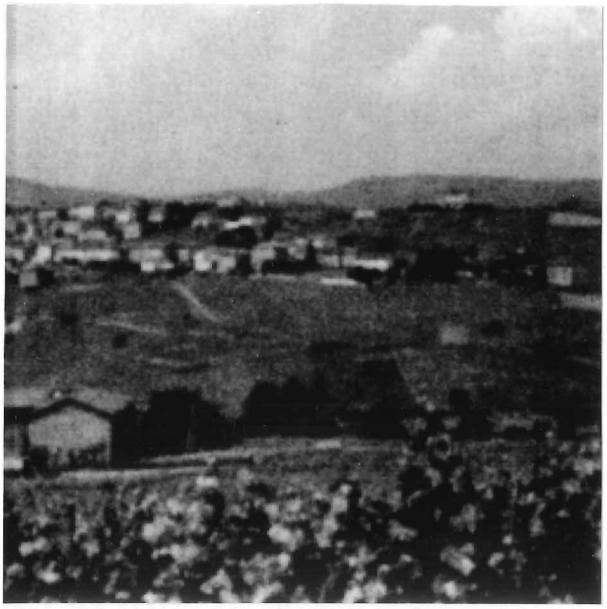

Figure 6.23: Frame 3 of CLOCHE

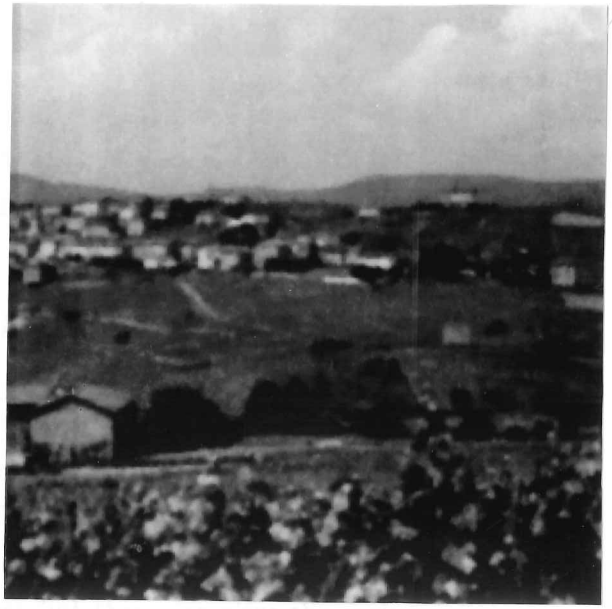

Figure 6.24: Restored frame 2: ML3Dex

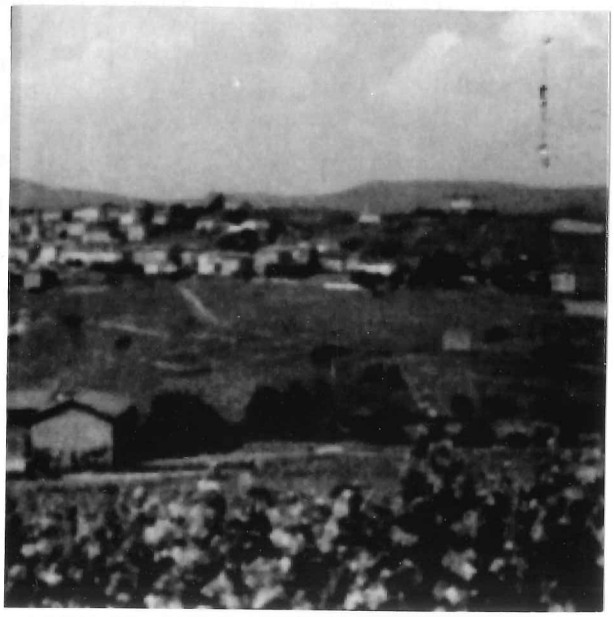

Figure 6.25: Restored frame 2: ML3D

to erroneous motion estimation, as explained earlier, and so can only be used if the distortion is low and there is a large confidence that the motion estimator would perform well[6]. For this particular sequence the LIN filter performs well because there is little motion.

In general, for typical distortion found in TV sequences, the ML3dex filter is appropriate since it can remove Blotches more effectively than the other filters, but is not as sensitive to motion estimation errors as the LIN or CUBE filters.

6.3 A heuristic for detection of Impulsive Distortion in image sequences

So far the application of these median filters has been standard. The filter is applied to all the regions in the image regardless of the position of the impulses. This represents a global filtering operation applied to a problem which is essentially a local one. The previous results have shown that the MMFs are effective in removing impulses but they also alter the regions of the image which are not corrupted. In order to limit this effect, it would be useful to be able to detect the position of distortion such as Dirt and Sparkle. The MMF may then be engaged only at the suspect locations so limiting the blur that is introduced in regions that are not corrupted.

In order to detect impulses and still deal with the possibilities of occlusion, uncovering, etc., in the image sequence, a number is generated (the Spike Detection Index – SDI) which gives a useful index to the presence or absence of any local distortion in an image sequence.

Define p to be the intensity at the current pel, f (for forward) to be the intensity at the next pel along the motion trajectory from the present pel into the next frame, and b (for backward) the intensity at the previous pel along the motion trajectory from the present pel into the previous frame. Then the SDI is defined as follows,

$$\begin{aligned} d_1 &= |p - f| \\ d_2 &= |p - b| \\ SDI &= 1 - \left|\frac{d_1 - d_2}{d_1 + d_2}\right| \quad \text{for } d_1 > t_1 \text{ or } d_2 > t_1 \\ SDI &= 0 \quad \text{otherwise} \end{aligned} \tag{6.8}$$

where t_1 is a low threshold which overcomes problems when d_1 and d_2 tend to zero. The SDI is limited to values between 0 and 1 and the decision that a spike is present is taken when the SDI at the tested pel is greater than some predefined threshold.

To understand how this approach works, assume that the motion is purely translational. Now consider the following points,

- Occlusion: $|p - f|$ will be large and $|p - b|$ will be zero. Therefore $SDI = 0$.

[6]Or if there is very little motion in the sequence.

- Uncovering: $|p - f|$ will be zero and $|p - b|$ will be large. Therefore $SDI = 0$.
- Normal Motion: Both $|p - f|$ and $|p - b|$ will be zero. As both $p - f$ and $p - b$ tend to 0 the SDI is not well behaved. However, when this happens, it means that the motion estimator has found a good match in both directions hence the current pel is not likely to be a scratch. Therefore in this case the SDI is set to zero.
- A spike at the current pel but in the position of an object showing normal motion: Both $|p - f|$ and $|p - b|$ will be large and the same and so $SDI = 1$. They would be the same since f, b would both be the same pels on the object at different times thus having the same intensity, provided the assumption of pure translation holds.
- A spike at the current pel but in a position on an object image showing occlusion or uncovering: It is difficult to say how the SDI behaves here. The SDI will take some undefined value, not necessarily zero or one. This value would depend on the actual intensities of the occluding regions.
- A spike at the current pel but f and/or b represent pels at which spikes have also occurred. Again the SDI is not defined, but if the spikes are fairly constant valued the index tends to 0.

The general rule is that when the SDI is 0 the current pel is uncorrupted, else when it is 1 the current pel is corrupted. In order to allow for the cases where occlusion and multiple corruptions along the motion trajectory are possible, there must be some threshold to make the decision. The threshold also allows some tolerance in the case of real sequences where motion is not purely translational and one has to deal with slight lighting changes not due to motion.

For real sequences there must be some lower threshold, t_1, for the forward and backward differences which will indicate that the match found is sufficiently low so that the current pel is uncorrupted. This is necessary because in real sequences the motion is not translational and due to lighting effects the intensities of corresponding areas do not necessarily match. Further, there will be errors from the motion estimator.

6.3.1 Performance

Figures 6.26 and 6.27 show two graphs representing the performance of the detector for two impulsive noise probabilities, 0.1 and 0.005. Ten frames[7] at the start of the Cheers sequence were used for this experiment. The block matching algorithm used was identical in form and parameter values to the algorithm described previously. The block based motion vector field was interpolated for use by the detector. The degraded sequences were identical to the sequences used previously. The lower threshold for the SDI, t_1 was set at 25.0.

[7]That is frame 1 to 10 where the first frame is frame 0

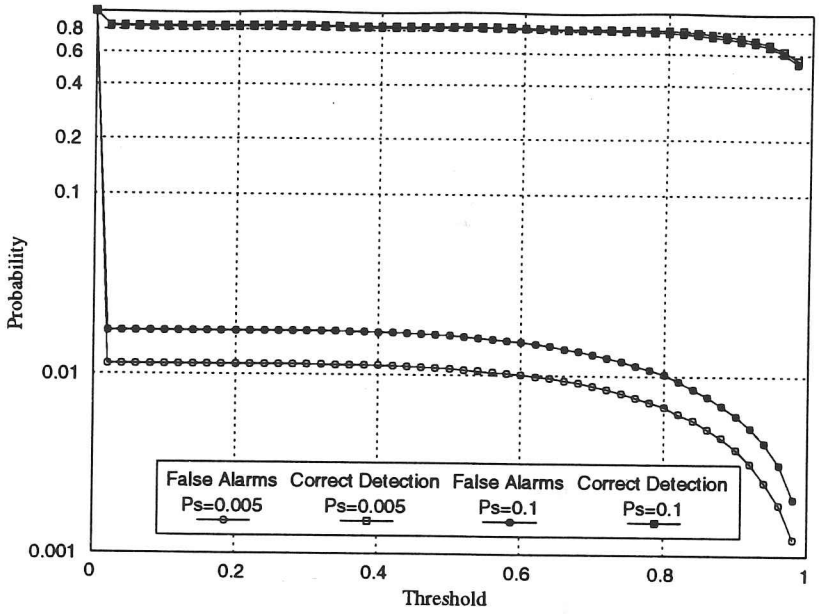

Figure 6.26: Performance of the SDI with respect to threshold.

Figure 6.26 shows the probability of correct detection and probability of false alarm vs SDI threshold. The curve plotted is the average performance of the detector over the 10 frames. Figure 6.27 is a receiver operating characteristic for the system, the point at False alarm = 1.0 and Correct Detection = 1.0 is not shown. There is an improvement of about 10% in correct detection when $P_s = 0.005$ as compared with $P_s = 0.1$.

The graphs indicate that the detector can achieve about 80% accuracy with a false alarm rate of 0.01 for $P_s = 0.005$. It is interesting that the operating characteristics plotted in figure 6.27 show that there are limited regions of activity for the detector. For instance, with $P_s = 0.1$, there is a maximum false alarm rate of about 0.03 and a corresponding maximum detection rate, 0.8. This is due in part to repeated corruptions along a motion trajectory. Once these occur the detector is unable to make the correct decision. Also, the use of the threshold t_1 automatically excludes a set of points from consideration. The form of the graphs indicate that the distribution of the SDI is top and bottom heavy. There are a finite number of occurrences of $SDI = 1$ and $SDI = 0$, hence the limits of operation.

6.3.2 An adaptive detector

The detector performance can be improved by allowing for the variation of movement and image information across the image. Instead of thresholding the SDI with a constant value across the image, it is better to use as the threshold some multiple of the local mean of the SDI. In effect, this makes the thresholding operation adapt to the local spatial information, thus improving upon the purely temporal nature of the SDI itself.

To avoid instances where large distortions cause the local mean to be biased away from a value which will give a good detection rate, it is necessary to filter the local means of the SDI

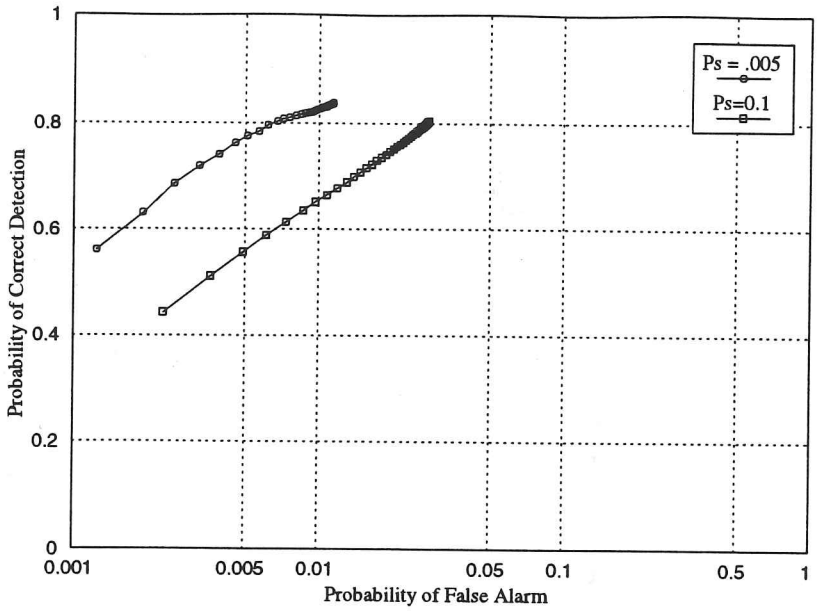

Figure 6.27: Receiver operating characteristic for the SDI.

to remove impulses due to large size or frequently occurring distortion. The local means are calculated for each block of some fixed size and then filtered with a multistage median filter. The filter chosen is the ML3D filter introduced by Alp without the temporal taps. This topology was chosen to preserve directional information in the local means due to motion edges or spatial image edges.

The performance of this detector as compared to the previous detector is shown in fig 6.28. The comparison is done for the same spike probability of 0.005. The block size used for the calculation of the means was 11×11 with a two pixel overlap. The threshold for the new detector was varied from 1 to 40 times the local mean in unit intervals. The improvement gained with the adaptive detector is appreciable, especially at false alarm rates below 0.005.

Figure 6.30 shows the performance of the adaptive detector with respect to the 12th frame in the Cheers sequence with $P_s = 0.005$, shown in figure 6.29. The threshold value was set to 2.0 times the local mean. The corresponding false alarm rate was about .01. The detector is less efficient in detecting spikes in moving areas, e.g the moving arm. Most of the false alarms are concentrated in these regions because of occlusion and uncovering. The probability of correct detection overall is about 80%. Note that the missed spikes are difficult to see in the degraded sequence because their intensity is not very different from the surrounding region.

The next section introduces a full controlled impulse suppression system and finally illustrates the performance of the adaptive detector with respect to real degradation.

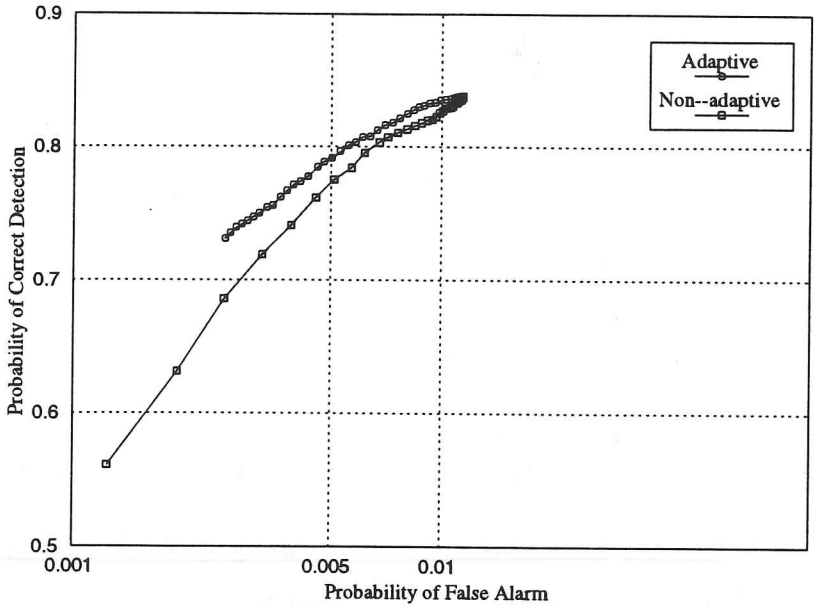

Figure 6.28: Comparing the performance of the adaptive and non-adaptive detectors.

Figure 6.29: Degraded Frame 12 (zoom), $P_s = 0.005$

Figure 6.30: The accuracy of the adaptive detector. Red: Missed Impulse, Brown: False Alarms, Green: Correctly detected Impulses.

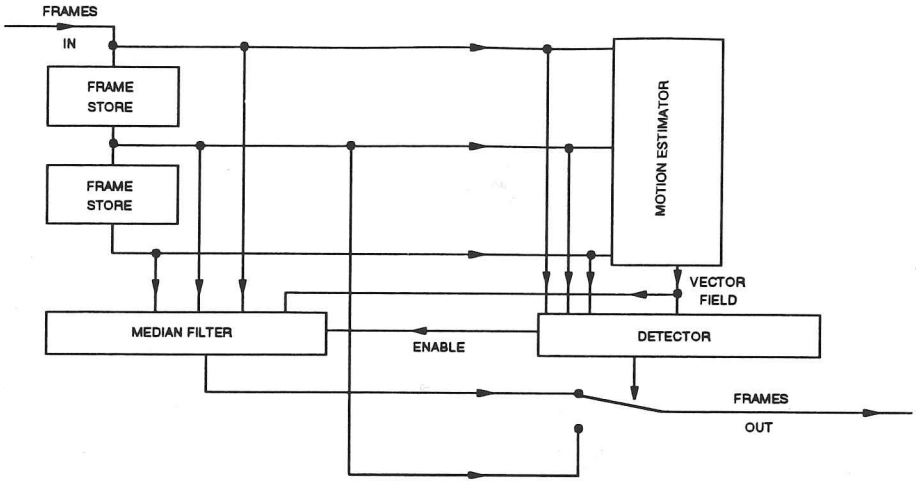

Figure 6.31: The SDI controlled median filtering system

6.4 *SDI* controlled median filtering for image sequences

Figure 6.31 shows a block diagram of a controlled median filtering system that engages the median filter only when an impulse is detected. The system was applied to the degraded Cheers sequence as presented above, with $P_s = 0.005$. Figure 6.32, shows the RMS error when the controlled system is used compared with the same error when the normal global operation is engaged. Two filters are considered, and it is clear that a controlled operation provides useful improvements since in both cases the controlled operation has limited the blurring effect of the median operation.

Figures 6.33 and 6.34 compare visually the ML3D filter output when a global operation is used as opposed to a controlled operation. The same pair of results for ML3Dex is shown as figures 6.35 and 6.36. The photos show that the blur has been reduced. This is particularly easily seen in the head of the central figure. The result is not unexpected. What is particularly encouraging is that the detector has a very low complexity. This implies that the controlled operation is potentially faster than the global operation since it reduces the number of times that the relatively high complexity median operation must be applied.

Figures 6.37 to 6.41 illustrate the performance of the system on real data as supplied by Channel 4 television. The three frame sequence is a subset of a 5 second sequence of resolution 512×512 and 8 bit grey scale. The frames are badly corrupted by noise and Dirt and Sparkle. The system discussed above was applied using a 3 level multiresolution BBM motion estimator and the result is shown on the supplementary video tape. Figures 6.40 and 6.41 show the result of controlled filtering using ML3Dex and ML3D filters respectively. The size of the distortion is too large to be treated successfully by the ML3D filter. It should now be obvious that the controlled filter will preserve more detail than the global operation therefore no examples are given of these operations.

The video supplement for this thesis gives more examples of restorations achieved using this

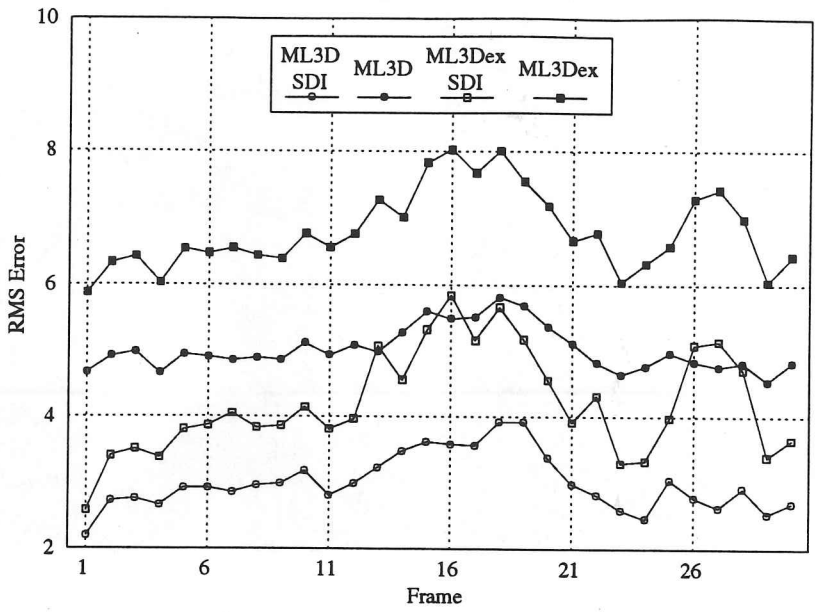

Figure 6.32: Comparing the performance of the controlled and global median operation.

system and a global application. Both the Cheers sequence and a real sequence are shown. The Cheers sequence was corrupted with binary Blotches of varying size. This sequence will be mentioned in the next chapter. The reader is referred to the supplement for a visual validation of the various points raised in this section. Section 7.2.3 describes the motion estimation parameters used for this Cheers sequence. The adaptive detector used a threshold of twice the local mean SDI value and t_1 was set to 10.0.

6.5 Summary

Motion compensated multilevel median filtering has been investigated for suppressing Dirt and Sparkle in motion pictures. The filters previously introduced by Arce [7, 5], Alp [4] did not incorporate motion estimation[8], and this chapter has shown that there is much to gain from such a process. The MMF's introduced by Arce, Alp also cannot remove the typical distortion found and so a new filter, ML3Dex was introduced.

A detector for Dirt and Sparkle was presented which was shown to improve the quality of the filtered output. The detector limits the operation of the filter to only a fraction of the image, depending on the distortion suspected. The detector itself requires just 6 operations, therefore the median filter (ML3Dex) is the more computationally complex. The controlled operation is potentially faster than the global median operation with the added advantage of improved image quality.

[8]Note that Huang [35] and Martinez [63] did employ motion estimation with the simple LIN median filter. However, the LIN filter is sensitive to motion estimation errors as indicated earlier in this chapter.

Figure 6.33: Frame 12 of CHEERS $P_s = 0.005$: ML3D (global)

Figure 6.34: Frame 12 of CHEERS $P_s = 0.005$: ML3D (SDI)

Figure 6.35: Restored Frame 12 of CHEERS: ML3Dex (global)

Figure 6.36: Restored Frame 12 of CHEERS: ML3Dex (SDI)

Figure 6.37: Frame 1 of BIPLANE

Figure 6.38: Frame 2 of BIPLANE:

Figure 6.39: Frame 3 of BIPLANE

Figure 6.40: Restored frame 2: ML3Dex (SDI)

Figure 6.41: Restored frame 2: ML3D (SDI)

Note that in all cases the block based motion field generated by the BBM algorithm was interpolated to give one vector per pixel for use with the algorithms presented. This did give improved results because it suppressed blocking artefacts in the case of the global filtering process. However the two major disadvantages are that the interpolated vectors would be erroneous at motion discontinuities and when a Blotch is encountered, the vectors around it would be detrimentally affected. While it is difficult to maintain motion discontiuities with this interpolation process, a marked improvement in performance could be gained if the motion estimator was made explicitly robust to Blotches. This could also be augmented by a process for correcting vectors at suspected Blotch locations. Chapter 9 discusses these points in terms of future prospects for research.

Finally some results for a real TV sequence were presented. The SDI controlled median system is effective in suppressing typical distortion. The reader is referred to the supplementary video tape for moving examples of the results from some of the algorithms discussed.

THREE DIMENSIONAL AR MODELLING FOR SUPPRESSING LOCAL DISTORTION IN IMAGE SEQUENCES

The 3D AR model has been previously introduced as an effective model of the image sequence in Chapter 3. Chapter 6 has investigated controlled median filtering as an efficient technique for removing Dirt and Sparkle in video. This chapter uses the 3D AR model to provide alternative solutions to the problems introduced in that chapter.

The 1D AR model has been used with considerable success for restoring audio signals degraded by scratches, noise and 'Wow'. Both Vaseghi [94] and Veldhuis [96] have introduced similar techniques to suppress impulsive distortion in audio. Their techniques involve using the AR model to interpolate the missing information. Vaseghi [94] has also used the model structure as a basis for detecting the position of the distortion. Recently, Godsill [31] has introduced a Bayesian approach for detection and interpolation of this artefact. Such techniques may be developed for image sequences but they are not treated here.

Although the extension of the techniques, used by Veldhuis and Vaseghi, to 3 dimensions is feasible, the resulting algorithms do not give similar performance. However modifications can be made to improve the result. The problems with using the 3D AR model in this manner are due primarily to the highly non–stationary nature of images. This encourages the use of small data block sizes. Consequently, there is a bias in the estimates for the model coefficients.

The discussion begins with the use of the 3D AR model as a detector for local distortion like Dirt and Sparkle. An algorithm for interpolating missing data in video signals is then developed from the model.

7.1 The Model

For convenience, the 3D AR model of the image sequence is repeated below. The model tries to make the best prediction of a pel in the current frame based on a weighted linear combination of intensities at pels in a predefined support region. This support region may occupy pels in the current frame as well as previous and past frames. It is defined as

$$I(i,j,n) = \sum_{k=1}^{N} a_k I(i + q_k(x) + sx_{n,n+q_k(n)}, j + q_k(y) + sy_{n,n+q_k(n)}, n + q_k(n)) + \epsilon(i,j,n) \quad (7.1)$$

The image is 'normalized' prior to modelling by subtracting a 3D trend from the data volume. (See Chapter 3). The terms are as follows

- $I(i,j,n)$: The intensity of the pixel at the position $\mathbf{x} = (i,j,n)$; where (i,j) is the horizontal and vertical position in frame n.
- a_k: The model coefficients.
- $\mathbf{q}_k = [q_k(x),\ q_k(y),\ q_k(n)]$: The vector offset to the kth support pixel.
- $\mathbf{d} = [sx_{n,n+q_k(n)},\ sy_{n,n+q_k(n)}]$: The motion vector mapping an object in frame n into frame $n + q_k(n)$.
- $\epsilon(i,j,n)$: The error, residual or innovations sequence representing the difference between the model prediction and the actual pixel intensity.

For the purposes of parameter estimation, the model is considered in its prediction mode. In this mode the model is reconfigured as a predictor of images in the sequence. The predicted intensity is as follows.

$$\hat{I}(i,j,n) = \sum_{k=1}^{N} a_k I(i + q_k(x) + sx_{n,n+q_k(n)}, j + q_k(y) + sy_{n,n+q_k(n)}, n + q_k(n)) \tag{7.2}$$

The task then becomes to choose the parameters in order to minimize some function of the prediction error, or residual,

$$\epsilon(i,j,n) = I(i,j,n) - \hat{I}(i,j,n) \tag{7.3}$$

The equation 7.3 is just a rearrangement of the model equation 7.1 with the emphasis placed on the prediction error, $\epsilon(i,j,n)$.

For the purposes of this chapter, it will be assumed that some motion estimation technique is used to generate an estimate for the motion vector, $\mathbf{d}$ for each block in a frame. The chapter uses a Multiresolution BBM technique as discussed in Chapter 2 and Chapter 3. These vectors can then be used to extract a volume of data from the sequence which is compensated for motion. To simplify matters therefore, it will be assumed that the data modelled has already been so compensated, in which case the displacement parameters can be dropped. The simplified prediction equation, which acts on data compensated for motion, is as follows.

$$\hat{I}(i,j,n) = \sum_{k=1}^{N} a_k I(\mathbf{x} + \mathbf{q}_k) \tag{7.4}$$

The prediction error can then be simplified accordingly.

7.2 Detecting local distortion

Assume that the image is corrupted according to the following equation.

$$g(\mathbf{x}) = I(\mathbf{x}) + b(\mathbf{x}) \tag{7.5}$$

$$\text{where } b(\mathbf{x}) = \begin{cases} 0 & \text{With probability } (1 - P_s) \\ s & \text{With probability } P_s \end{cases} \tag{7.6}$$

Here, s is a randomly distributed grey level representing a Blotch or impulse, and it occurs at random intervals in the frame with probability P_s. As in the previous chapter, it is required to detect the likely occurrences of $b(\mathbf{x}) \neq 0$ in order to isolate the distortion for further examination. The key to the solution is to recognize that the undistorted image $I(i,j,n)$ obeys the AR model whereas $b(\mathbf{x})$ does not.

Suppose that the model coefficients for a particular image sequence were known. The prediction error could then be considered to be noise with some correlation structure as defined in Appendix B. If $g(\mathbf{x})$ was filtered with the model prediction filter the output could be written as below

$$\epsilon'(\mathbf{x}) = g(\mathbf{x}) - \sum_{k=1}^{N} a_k g(\mathbf{x}+\mathbf{q}_k) \tag{7.7}$$

$$= I(\mathbf{x}) + b(\mathbf{x}) \tag{7.8}$$

$$- \sum_{k=1}^{N} a_k I(\mathbf{x}+\mathbf{q}_k) - \sum_{k=1}^{N} a_k b(\mathbf{x}+\mathbf{q}_k)$$

$$= \epsilon(\mathbf{x}) + b(\mathbf{x}) - \sum_{k=1}^{N} a_k b(\mathbf{x}+\mathbf{q}_k) \tag{7.9}$$

Equation 7.9 shows that the undistorted samples in the degraded signal are reduced to the scale of the error or residual sequence. The distorted samples can reduced in amplitude if there are other distorted samples nearby but this does not occur often. Therefore thresholding the prediction error is a useful method of detecting local distortion.

However, the support of the model affects the detectability of distortion. In the case of Dirt and Sparkle, because the distortion occupies a large spatial area, a model with spatial support in the current frame would only give large prediction errors at the edges of a Blotch. Inside the Blotch the residual would be reduced in magnitude due to the growing importance of the last term in equation 7.9. In practice, models with supports not in the current frame are more effective since the distortion is always local (even impulsive) in time but not necessarily as local in space.

7.2.1 Parameter Estimation

For a real sequence, the model coefficients are unknown. In this work they are estimated from the degraded sequence using the Normal Equations. A motion compensated volume of data is extracted and then 'centralized' by subtracting a 3D trend following Chapter 3. The coefficients are then estimated according to Appendix A. The choice of the spatial extent of the volume used is important. If the size of a block is comparable to the size of a particular Blotch then the coefficients are heavily biased toward that distortion and the resulting detection efficiency

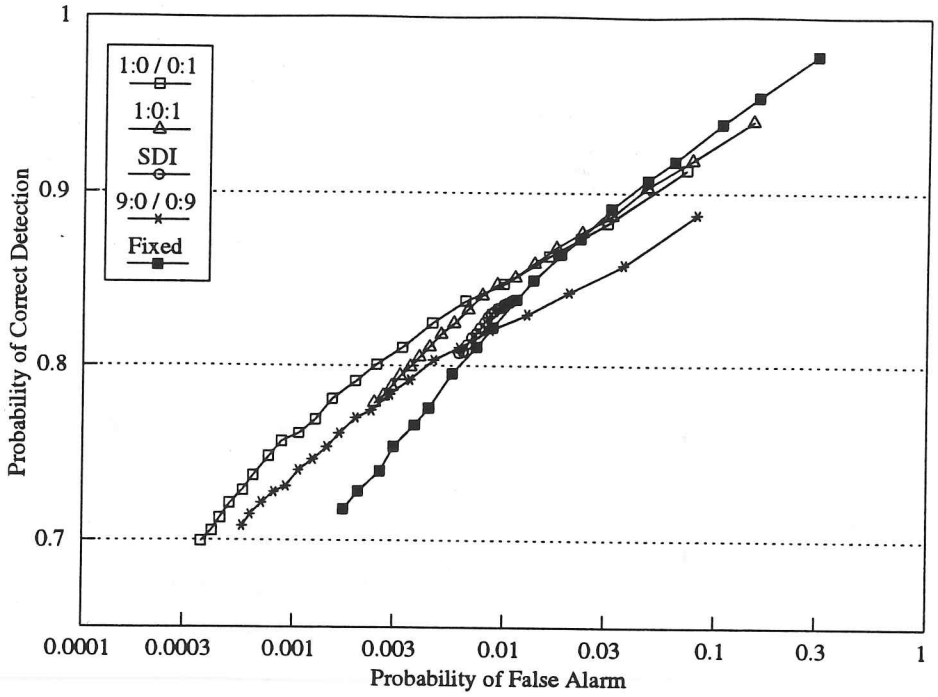

Figure 7.1: Comparing the performance of different models.

is poor. This effect is enhanced when the model has spatial support in the current frame. In practice, models with spatial support in the current frame must be avoided.

7.2.2 Results and Discussion

The Detection algorithm adopted is similar to the one introduced in Chapter 6 except that the squared prediction error is thresholded instead of the SDI. Multilevel BBM is used to estimate motion from the current frame into the past and next frames. The algorithm used is therefore the same as that previously discussed at the end of Chapter 3. Note that the AR model error was evaluated on a block basis using one model per 9×9 block. The receiver operating characteristics (ROC) were measured by averaging the ROC of a detector with respect to one frame, over all the considered frames.

Ten frames from the start of the Cheers sequence were again used. It was corrupted using single pixel sized impulses of probability 0.005 and uniformly distributed grey scale. The same motion estimation parameters as in Chapter 6 (see section 6.2.1) were used and several different models were then employed for detection. It was found that the best performance was given by thresholding the error from two differently oriented detectors with little spatial support.

Figure 7.1 compares the receiver operating characteristics when four different 3D AR models are used. The models are described by the number of pixels of support in each frame, there is no reason for using asymmetric supports and so the support is always symmetric. A support of 9 pixels therefore refers to a square of 3×3. The models are described in the order *past frame : current frame : next frame*. Therefore a model of the form 1:0:1 is a non–causal model with 2

pixels support. The prediction given by that model may be given as

$$\hat{I}(i,j,n) = a_{-1}I(i,j,n-1) + a_1 I(i,j,n+1) \tag{7.10}$$

There were two types of model based detection systems investigated. The first thresholds the prediction error given a single model. Therefore, an impulse is detected when

$$[\epsilon(\mathbf{x})]^2 \geq t_\epsilon \bar{\epsilon^2} \tag{7.11}$$

where $\bar{\epsilon^2}$ is the filtered local mean of the squared prediction error and t_ϵ some threshold that is a multiple of this mean. This local mean filtering process is performed exactly as in Chapter 6; the means of each $N \times N$ block are filtered using a multistage median filter to prevent bias caused by large Blotches.

The other detection system used two temporally different models, a forward predictor, 1:0, and a backward predictor, 0:1. The two prediction error fields, ϵ_1 and ϵ_2, are then thresholded to yield a detected distortion when

$$([\epsilon_1(\mathbf{x})]^2 \geq t_\epsilon \bar{\epsilon_1^2}) \text{ AND } ([\epsilon_2(\mathbf{x})]^2 \geq t_\epsilon \bar{\epsilon_2^2}) \tag{7.12}$$

Therefore, a Blotch is located when both predictors agree that a match cannot be found in either of the two frames. Such a system is denoted here by 1:0/0:1. Block sizes of 9×9 were used in generating the prediction error, and a border of 10 pixels around the edge of the image frame was ignored in taking measurements.

Figure 7.1 shows that the best model performance is actually given by the 1:0/0:1 system. This can be explained by the fact that in the coefficient estimation stage, the model coefficients can adjust themselves to compensate for the presence of low contrast distortion. This means that the detection capability of models with larger supports in the surrounding frames, such as 9:0/0:9, is reduced. The figure also shows the non–causal model is worse than the forward/backward model combination. This performance is due to the small set of data samples from which an estimate of the model coefficients must be made. The impulses can bias the model coefficients adversely.

Figure 7.1 also compares the performance of the model based systems with two other systems. One of these is the SDI from Chapter 6. The other is again 1:0/0:1, but with the coefficient values fixed at 1.0. This second system, (denoted by *Fixed*), effectively uses the displaced frame difference at each pixel for an error measure. The block based motion field yielded by the BBM estimator was Bilinearly interpolated for use with this detector. The decision rule used was not adaptive and was based on the absolute forward and backward displaced frame differences. Therefore a spike was detected when

$$(|\epsilon_1(\mathbf{x})| \geq t_\epsilon) \text{ AND } (|\epsilon_2(\mathbf{x})| \geq t_\epsilon) \tag{7.13}$$

The results show that the 1:0/0:1 detector, is the best overall performer. The SDI detector is limited by the low threshold used (25), as discussed earlier in Chapter 6. In practice, a useful

operating point, is found to be $[0.01, 0.85]$, and here, all the detectors perform roughly similarly. The simplest detector, Fixed, would appear to be a good choice in terms of computation. No photographs of the detection accuracy are shown because again the main difficulty is false alarms in regions of occlusion and uncovering, the same problem encountered before with the SDI.

7.2.3 Large size distortion

Figure 7.2 shows the Receiver Operating Characteristics of three detectors with respect to the first 10 frames of the Cheers sequence degraded by large sized distortion. The Blotches took the form of uniformly distributed square areas of uniform intensity of sizes uniformly randomly distributed from 2×2 pixels to 6×6 pixels. The Blotches occurred with probability 0.002. The Blotches were of two intensities, 0 and 255, i.e. black and bright white, which occured with equal probability when a Blotch was made present. This is a more realistic distortion with respect to typical TV degradation than single pixel impulses.

The same motion estimation technique (BBM) as above was used. The algorithm used 3 levels, with a block size of 17×17 in the lower two levels of resolution 256×256 and 128×128, and a block size of 5×5 at the highest level. This large block size was necessary to prevent the relatively large Blotches from biasing the model coefficient estimates. The motion vectors were generated to integer accuracy using a ± 4 pel search space at each level. The noise ratio required for the BBM was 1.5, 1.2 for levels 0, 1 and 1.0 at level 2.

The figure shows that the Fixed detector is the best overall, with the 1:0/0:1 system being much better than this for high false alarm rates, and worse for false alarm rates less than ≈ 0.01. The SDI does not perform well with respect to either of these detectors. However, in practice Dirt and Sparkle do not occur with this high probability and the SDI is found to be adequate. In this particular sequence almost all of the missed Blotches were due to occurrences of distortion at the same position in consecutive frames. In such situations the distortion can no longer be characterized by a purely temporal discontinuity and so cannot be located by the detectors.

The main problem presented by Dirt and Sparkle is that the distortion can be so large that it severely affects the motion estimator and subsequently biases the AR model coefficients. This problem would not arise if there were many data samples being used, but here only blocks of 17×17 pixels in three[1] frames of data are used. This implies that the two detectors, *Fixed* (referred to as SDIa in the text to come) and the SDI are safer to use in an actual situation.

[1] Only a few frames are used because occlusion and uncovering effects become more influential the further one moves from the central considered frame.

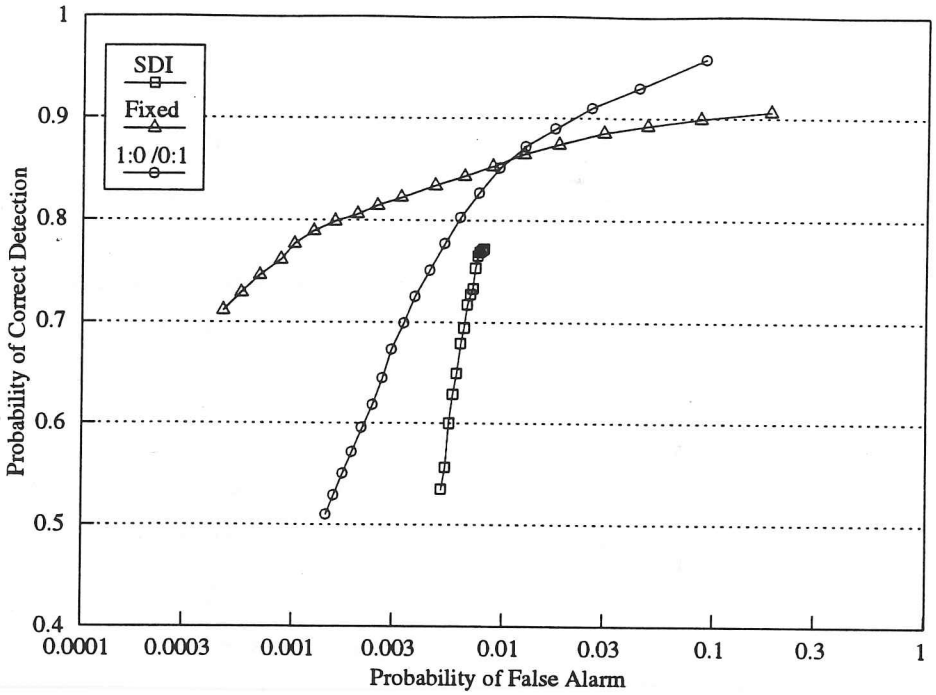

Figure 7.2: Comparing the performance of different models.

7.3 Removing Dirt and Sparkle

Having detected the position of some local distortion, the task is then to remove the distortion by filling in the missing information. This section presents an interpolator for missing information based on a 3D AR model. The method developed by Vaseghi [94] has been very successful for audio signals and it is this method which is extended to three dimensions for use in this image sequence interpolation problem.

It is assumed that the statistics of the information in a block of $M \times M$ pixels is sufficiently stationary for a single model to be used for that block. Within this block, assume that there are m pixel locations that are corrupted. These locations are arbitrary; they may be connected together as in a Blotch, or they may be a set of dispersed impulses. The locations are given as $\mathbf{p}_i = \mathbf{l}_i$, $(i = 1 \ldots M)$, where $\mathbf{p}_i$ is the location of the ith point in the block of pixels.

Further, it is assumed that the block can be modelled by a 3D AR model incorporating $M \times M$ blocks in the surrounding frames that is compensated for motion. Again, the intensity of the pixels in the block in the current frame may be written as

$$I(\mathbf{x}) = \sum_{k=1}^{N} a_k I(\mathbf{x} + \mathbf{q}_k) + \epsilon(\mathbf{x}) \tag{7.14}$$

Where again, $I()$ represents pixel grey level, a_k are model coefficients, and $\epsilon(\mathbf{x})$ the model excitation (or ideal prediction error). It is assumed that a set of N frames are considered. Allowing for a border of pixels at the edge of the $M \times M$ block in the current frame, n say, (so that $(\mathbf{x} + \mathbf{q}_k)$ will never result in a location outside the $M \times M$ block), an equation for the error at every pixel

within a centred $B \times B$ block in that frame can be written as below.

$$\mathbf{e} = \mathbf{A}\mathbf{i} \tag{7.15}$$

where $\mathbf{i}$ represents an $NM^2 \times 1$ column vector of row ordered pixels from the N $M \times M$ blocks, $\mathbf{e}$ is a $B^2 \times 1$ column vector of errors, and $\mathbf{A}$ a matrix of coefficients satisfying the model equation at all the considered points. This coefficient matrix is of size $B^2 \times NM^2$. The vector $\mathbf{i}$ contains intensities of both known and unknown pixels. If this vector is separated into two vectors $\mathbf{i}_u$ (u for unknown) and $\mathbf{i}_k$ (k for known), which represent the known and unknown pixel intensities, then equation 7.15 can be written as

$$\mathbf{e} = \mathbf{A}_k\mathbf{i}_k + \mathbf{A}_u\mathbf{i}_u \tag{7.16}$$

Here, $\mathbf{A}_k$, $\mathbf{A}_u$ are the coefficient matrices corresponding to the known and unknown data vectors. They are submatrices of the $\mathbf{A}$ matrix, made by extracting the relevant columns.

To derive an interpolation, $\mathbf{i}_u$ must be found. Following Vaseghi [94], this is done by minimizing the squared error $\mathbf{e}^T\mathbf{e}$ with respect to $\mathbf{i}_u$ as follows.

$$\begin{aligned}
\mathbf{e}^T\mathbf{e} &= [\mathbf{A}_k\mathbf{i}_k + \mathbf{A}_u\mathbf{i}_u]^T[\mathbf{A}_k\mathbf{i}_k + \mathbf{A}_u\mathbf{i}_u] \\
&= \mathbf{i}_k^T\mathbf{A}_k^T\mathbf{A}_k\mathbf{i}_k + \mathbf{i}_k^T\mathbf{A}_k^T\mathbf{A}_u\mathbf{i}_u + \mathbf{i}_u^T\mathbf{A}_u^T\mathbf{A}_k\mathbf{i}_k + \mathbf{i}_u^T\mathbf{A}_u^T\mathbf{A}_u\mathbf{i}_u \\
\frac{\partial \mathbf{e}^T\mathbf{e}}{\partial \mathbf{i}_u} &= 2\mathbf{A}_u^T\mathbf{A}_k\mathbf{i}_k + 2\mathbf{A}_u^T\mathbf{A}_u\mathbf{i}_u \\
\Rightarrow \mathbf{i}_u &= -[\mathbf{A}_u^T\mathbf{A}_u]^{-1}\mathbf{A}_u^T\mathbf{A}_k\mathbf{i}_k
\end{aligned} \tag{7.17}$$

Therefore, the solution for the interpolated pixels is given by equation 7.17. Of course this solution implies knowledge of the model coefficients. These must be estimated from the corrupt data, and this estimation process is discussed next.

An important point to recognize is that equation 7.15 can be made up of error observations not only from the block in the current frame. In fact, it is sensible to incorporate as many observations as possible that incorporate data in the missing regions so as to maximize the information used. This implies that for a causal model with support in the previous frame only, observations in the $B \times B$ block in the next frame also incorporate the missing pixel information $\mathbf{i}_u$ in the current frame. Therefore, the resulting interpolator incorporates information from one frame both previous to and following the current frame. For a non–causal model incorporating one frame of support in the previous and next frames, a total of 5 frames can be incorporated into the equations (two frames previous and following the considered one). In practice, however, this extra information does not yield significant improvements over using just observations from the current frame.

A useful practical point to be noted is that for a given set of missing pixels, there is a maximum area of observations around the missing region beyond which no improvement in interpolation quality is gained. This region depends on the model support. The reason is that the observation equations (i.e the equations for the model errors) are only useful for interpolation

if they contain at least one missing pixel. Therefore, if there was a region of missing pixels that was of size $S \times S$ then given a 9:0 causal 3D AR model, the spatial extent of the observation equations need only be $(S+2)\times(S+2)$. This result follows from examination of the interpolation equation 7.17. Of course, the observation equations may be set up for such a block in both the current frame and the next since the pixels in the motion compensated block in the next frames also involve missing pixels in the current frame.

7.3.1 Estimating the model coefficients

To solve equation 7.17, the model coefficients are required. In a real case however, these values are unavailable. They must be estimated from the degraded image sequence data. However, as has been stated before, the block sizes that must be used are small. This is forced because of the highly non–stationary nature of image sequences, both in terms of space and time (due to errors in motion estimation). This means that the distortion can bias the model coefficients adversely. Because a detector is used to isolate a suspected distorted area, this information can be used to suppress the bias that the distorted area would cause.

The Normal equations are altered to solve for the AR parameters using weighted coefficient estimation. Recall that normally, the model coefficients are chosen to minimize the expected value of the squared prediction error at all points in the block considered. Because some of this data is now known to be missing, the prediction error at these points may be weighted to zero so that this data does not affect the estimation process. The general approach is to weight the prediction error by some function $w(\mathbf{x})$, prior to minimizing the squared weighted error.

For the purposes of coefficient estimation, the new prediction equation may be written as

$$\epsilon_w(\mathbf{x}) = w(\mathbf{x}) \sum_{k=0}^{N} a_k I(\mathbf{x} + \mathbf{q}_k) \tag{7.18}$$

Where all the symbols have their usual meaning, $a_0 = 1.0$, and $\epsilon_w(\mathbf{x})$ is the weighted error at position $\mathbf{x}$. Recall that this chapter assumes that the data volume being used has already been compensated for motion and so the motion parameter has been omitted from the 3D AR model. Further, recall that a 3D trend is subtracted from the data prior to modelling to improve the prediction. The estimation of the trend coefficients is also weighted in an identical manner to that shown here.

Minimizing the squared error $[\epsilon_w(\mathbf{x})]^2$ with respect to the coefficients, then yields the following set of N+1 equations.

$$\sum_{k=0}^{N} a_k E[(w(\mathbf{x}))^2 I(\mathbf{x} + \mathbf{q}_k) I(\mathbf{x} + \mathbf{q}_m)] = 0 \text{ for } m = 0 \ldots N. \tag{7.19}$$

Recall that $a_0 = 1.0$. Therefore, these equations may be written in matrix form as

$$\mathbf{R}_w \mathbf{a} = -\mathbf{r}_w \tag{7.20}$$

Figure 7.3: Original Frame 2 of *GORN*.

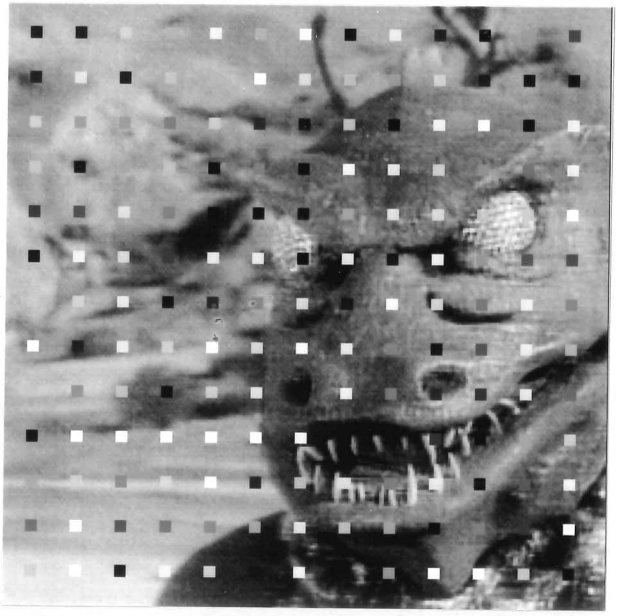

Figure 7.4: Corrupted Frame 2 of *GORN*

where $\mathbf{R}_w$ is an $N \times N$ matrix of correlation coefficients, and $\mathbf{r}_w$ an $N \times 1$ vector of correlation coefficients. Equation 7.20 is the weighted solution for the N model coefficients. The most obvious choice for the weighting function is a binary field set to 0 for all the Blotch positions and 1 otherwise. This is found to be extremely effective in practice.

7.3.2 Results

The figure 7.4 shows a photograph of one frame of the *GORN* sequence artificially corrupted by regularly occurring square patches (5×5 pels) of uniform intensity. The sequence consists of 3 frames in which a main figure (the *GORN*) undergoes a small rotation, zoom and camera pan, amounting to a motion of less than 5 pixels per frame. The three frames are not shown since it is difficult to see this motion from still pictures. A single level BBM algorithm was used for motion estimation. The details of the parameters used for the motion estimation algorithm are unimportant, it is only necessary to ensure that all the algorithms employed used the same set of vectors from the same motion estimator. The motion estimator generates a vector for each block of the image and this vector field is then interpolated using bilinear interpolation to give a vector at a particular pixel if required by the interpolation algorithm. Perfect detection was assumed. This experiment therefore illustrates the best performance of the algorithms employed. Only the second frame of the sequence was corrupted.

The figures 7.5 and 7.6 compare the results of interpolating the missing data in the corrupted frame using the two processes discussed so far. Figure 7.5 shows the result using ML3Dex median filtering and figure 7.6 the result using 3D AR interpolation as just discussed. The 3D AR algorithm used a 9:8 causal model with equations set up via the prediction error in the current and next frame. A block size of 17×17 pixels was used to generate the model coefficients

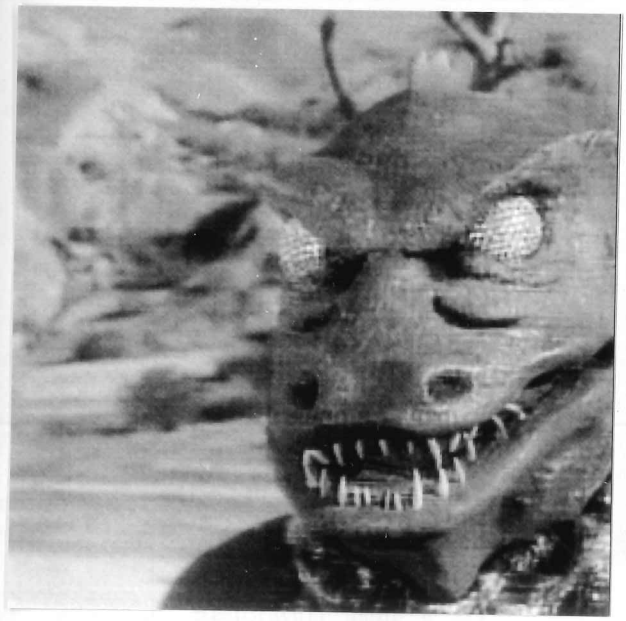

Figure 7.5: Restored using ML3Dex.

Figure 7.6: Restored using 9:8 3D AR model.

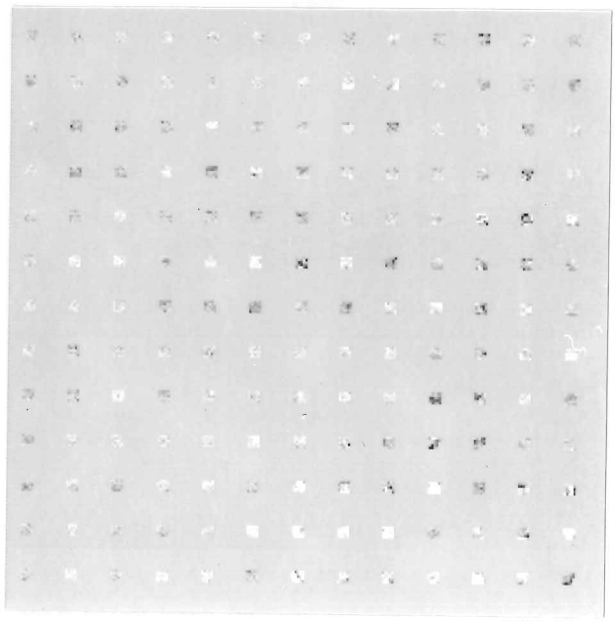

Figure 7.7: Error in median filtering.

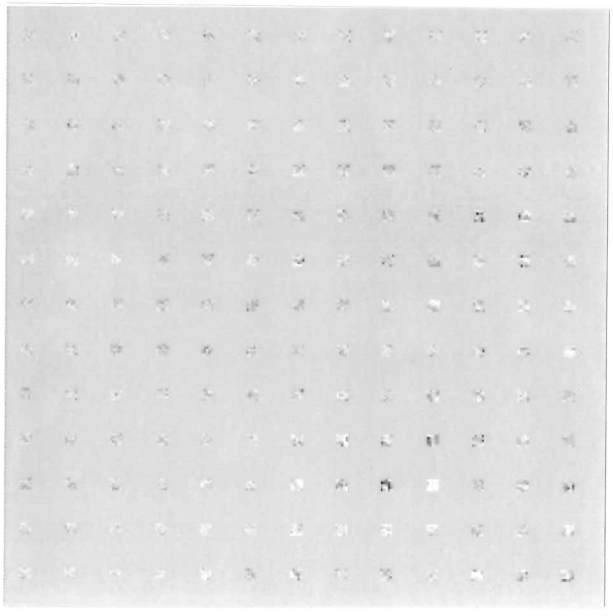

Figure 7.8: Error in 3D AR filtering.

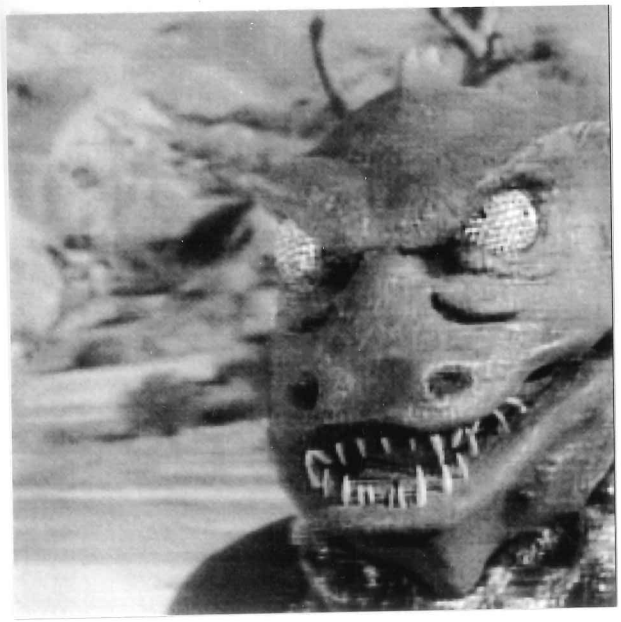

Figure 7.9: Restoration using a 2D model.

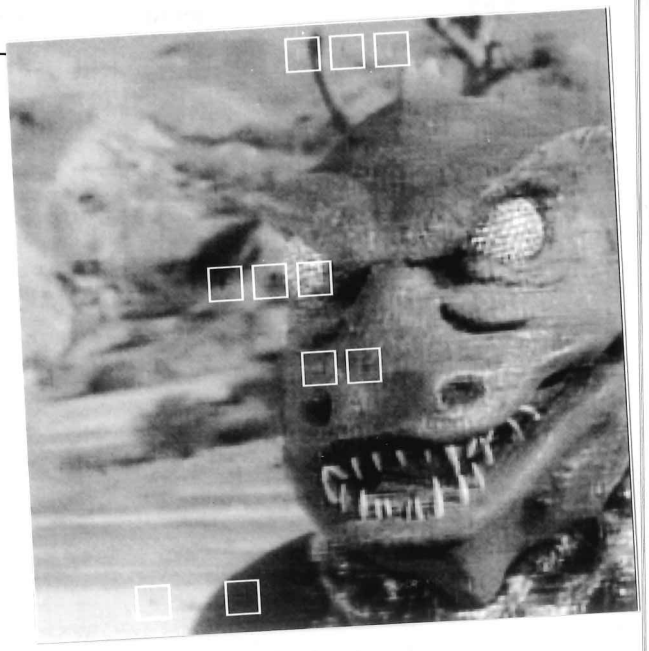

Figure 7.10: Restoration using unweighted coefficient estimation. (9:8 Model)

through weighted estimation. The restorations are virtually identical except for the right eye of the creature. The median operation replaces two blocks there with a fairly uniform intensity, but the AR interpolator restores the texture almost perfectly. Figures 7.7 and 7.8 show the error signals between the interpolated and original frames. The error was added to a uniform grey background of value 32 and the result clipped between 0 and 255. It is clear from these images that the median filter is not as accurate as the model based interpolator for textured regions. The MSE for each restoration is 5.2 and 1.86 for the median and 3D AR processes respectively. This further reinforces the visual comparisons.

Figure 7.9 shows the result of 3D AR interpolation using a 2D model (8 nearest neighbours in the current frame only) instead of the 3D model used above. It can be seen that two regions in the right eye are not interpolated correctly. They are too large for the 2D model to replace the missing data with sensible information toward the centre of the block. Figure 7.10 illustrates the importance of weighted coefficient estimation by applying the 3D AR interpolator without weighting the estimation process. Artefacts can be seen in the background. They are highlighted with white borders. The defects in the corrupted data have biased the coefficients to yield a badly contrasted interpolation. Weighted coefficient estimation is a necessity when dealing with such small data blocks.

The results indicate that given the same motion estimates the 3D AR model can yield a higher fidelity restoration than the median operation. However the quality of the interpolations may not be vastly different visually.

7.4 Dirt and Sparkle suppression in real degraded sequences.

For a real degraded sequence there are two major complications. The first is that the sequence is usually corrupted by white noise as well as Dirt and Sparkle. The second is that the Blotch sizes can be very variable and they can extend over huge areas.

The problem with white noise is not very severe as far as the model based interpolator is concerned. However, it affects the motion estimation process despite the protective measures that are taken by the BBM algorithm. Of course, this problem is intensified when large Blotches occur. This can adversely affect coefficient estimation. This work chooses block sizes that are sufficiently large to suppress the effect of the Blotches encountered.

Large Blotch sizes pose no extra problem to the Median based Blotch suppression algorithms, but as far as the model based technique is concerned this situation can only be handled by a non–homogeneous model. As stated before, the image is highly non–stationary and a large Blotch will traverse statistically differing regions of the image. One might assume that the use of a 3D model would circumvent this problem by taking advantage of the high temporal correlation in the image sequence. Unfortunately, in practice this is not so. The use of a spatially varying set of models for interpolation is not investigated in this thesis although it represents a relatively small modification to the current model based interpolator. It is not considered due to the increased implementational cost of such a technique.

Another problem that the model based interpolator faces in practice is that it is necessary to centre the Blotch in the middle of a modelling area. This would ensure that the area to be interpolated is surrounded by useful information. If this was not done, then data at the edge of a block may be missing and so the interpolated data would not be bounded at one side. The Blotches must therefore be measured and a block assigned which encompasses the entire Blotch. To map a path around each Blotch, a standard outline tracing algorithm is used [79]. This path then defines a bounding box around the Blotch. The data used to estimate the model coefficients is taken from a block which is centred on the bounding box around the Blotch. This *modelling* block is made to be of sufficient size that less than 10% of the block is corrupted. This fraction was chosen fairly arbitrarily as one that gave suitable model coefficients.

In many cases of real distortion, it is found that the detectors discussed often do not detect the extremities of a Blotch since that artefact is often not of uniform brightness. This can be a catastrophic situation as far as the model based interpolator is concerned since it means that the weighted coefficient estimation process would consider some portion of the Blotch to be undistorted. This leads once more to a bias in the coefficients and the interpolated data tends to take on the intensity of the extremities of the Blotch. This interpolated data is then badly contrasted with the immediate vicinity of the image and is often easily visible as a less intense Blotch. This problem may be solved by ensuring a pessimistic estimate for the existence of a Blotch. The most straightforward way of making the detected Blotches larger than originally

detected is to use Morphological operations [84]. Of course an alternative is to reduce the threshold for detection, but this will increase the false alarm rate as a side effect.

In practice therefore, the detection operation is followed by a Morphological *Closing* operation to smooth the detection field. Then a Dilation operation to grow the detected Blotches by one pixel around their edges. The operations used a 3×3 window. Dilation can be described using set theory but is better understood through the following operations.

1. Consider each pixel in the detection field. This field contains pixels in a binary state: 1 for Blotch, 0 Otherwise.
2. If the current pixel is a 1 then output a 1 and move on to the next pixel.
3. If the centre pixel is a 0 then consider a 3×3 square around the current pixel.
4. If there is at least one pixel in this square that is set to 1, then set the current pixel in the corresponding output square to 1.
5. Move on to the next pixel.

The net effect is to grow an isolated 1 to a square of 3×3 ones. A line of $N \times 1$ pixels is grown to a size of $(N+2) \times 3$ pixels. The exact opposite operation is called erosion, that operation will remove all isolated 1's and single pixel width lines. The Closing operation is a Dilation followed by an Erosion. These operations are effective in giving spatially smooth, pessimistic estimates for the presence of a Blotch in an image. They are useful post processing operators.

Figures 7.12 to 7.18 illustrate the operation of a 3D AR interpolator combined with the SDIa detector on a real distorted sequence (CLOCHE) with typical distortion. The three frames of the sequence show very small motion, about 4 pixels at the most. The second frame of the sequence is to be restored. The resulting detection using SDIa with a threshold set at 20.0 is shown in figure 7.15 superimposed on the image. It is certain that the distortion that is most often encountered is of a multiple pixel size. Therefore, another postprocessing stage has been used which rejects all detected Blotches which are flagged by a singe isolated[2] pixel. This is applied before Closing and Dilation. Figure 7.16 shows the post processed detection field and also delineates the Bounding box of each detected blotch indicated by the boundary finding algorithm discussed earlier. Note that there are a few false alarms. The post processing stage has been able to effectively delineate the Blotched regions even though they were not completely detected. Figures 7.17 and 7.18 show restorations using the median operation and the 3D AR operation with a 9:8 model. Note that visually the difference, if any, is small.

Figures 7.19 to 7.21 shows another real degraded sequence with more motion (CLOCHE2). A 2 level BBM algorithm was used for motion estimation with a search space of ± 4 pixels at each level. The Boyce ratio (see Chapter 2) was set at 1.2 for the first level and 1.0 for the second.

[2] All 8 nearest neighbors are 0.

The MAE motion threshold was set at 15.0 and 10.0 for each level. A Block size of 11×11 was used for motion estimation. As with the GORN, the vector field was interpolated for use by the 3D AR interpolator at a particular position, using Bilinear interpolation. The SDIa was used for Blotch detection with the threshold (t_ϵ) set at 35.0. Figures 7.22 and 7.23 show the detected Blotches before and after post processing. Figure 7.25 shows the result of Median filtering and figure 7.24 the result of AR modelling using a 9:8 model. Both the restorations have removed the two Botches satisfactorily. Again, the results are comparable. Note however, that due to the limitations of the hardware used, the larger detected Blotches could not be interpolated by the 3D AR interpolator. Figures 7.26 and 7.27 show the error between the respective restorations and the original degraded frame. They are included to indicate to the reader where the model based interpolator was actually used. The error was added to 128 and then displayed, hence the grey background corresponds to zero error. The error was also clipped so that the minimum and maximum values displayed were 0 and 255. In this case the false alarms are not badly treated by the model based interpolator.

The results do show however, that false alarms are most serious in moving regions, the arm of the figure in the pictures shown. Had the model based interpolator attempted to interpolate the larger suspected missing regions, it is doubtful whether the interpolated data would be of a useful quality. The large size of the distortion implies that it may traverse statistically different regions in space and time; this will affect the model based interpolator adversely. The median operation would be less affected since it is applied to each pixel separately. Note however, that the median filter used is more affected by errors in motion estimation than the model based interpolator (when the model based method can be safely applied). This can be seen in some areas along the arm. This is because the model based interpolator used incorporates more information from the current frame explicitly into its estimate. This robustness to motion estimation errors is a useful quality of the model based technique.

In order to assess objectively the difference between median and 3D AR processes, figure 7.11 shows a graph of the MSE of 3 different restorations of the first 10 frames of the Cheers sequence. The sequence was degraded by square patches of uniform intensity (either value 0 or 255 of equal probability) and uniformly distributed size (maximum 6 pixels). The sequence is the same as that used in generating figure 7.2 which shows the performance of various detectors on the degraded sequence. The detector and post–processing operations discussed above were used to detect Blotches with a SDIa threshold of 55.0, and the same motion estimation parameters were used. The detection fields were the same for all cases, with the AR method involving the necessary sizing operations discussed previously. Note that the detected scratches were of a sufficient size so that all necessary locations were interpolated, therefore the comparison is valid. The figure shows that the 3D AR interpolator can perform consistently better than median filtering. The 9:8 model did not work as well as expected because in some instances only part of a Blotch was detected due to multiple occurrences along a motion trajectory. Therefore some Blotch information would be incorporated into the coefficient estimation. Since the 9:8 model

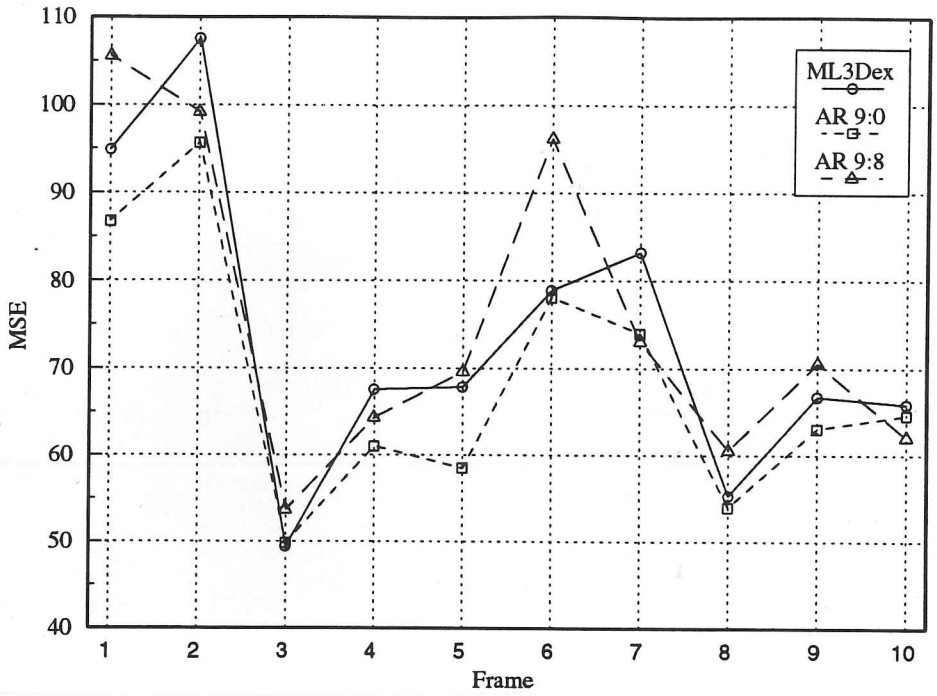

Figure 7.11: Comparing the performance of different interpolation schemes on CHEERS.

incorporates points in the current frame for a prediction, this model is badly affected by that shortcoming.

The video supplement shows more of the same Cheers sequence used here and allows a visual comparison of the restorations obtained using the ML3Dex and 3D AR interpolation processes. The Clochemerle sequence is also shown in its entirety.

7.5 Summary

A model based technique has been presented for detecting and removing Blotches in image sequences. It is found that a simple technique for detection using the motion compensated frame difference, the SDIa, is as effective as the model based technique. In fact it is found to be better in some cases. This is due primarily to the fact that the model based approach can be biased by large distortion since the block size that is used is relatively small.

The model based interpolator has been demonstrated to give results of a potentially higher quality than Median filtering. In practice however, the results are visually comparable. This, in conjunction with the lower complexity of Median filtering and the current speed of microprocessors, implies that the median operation is a better choice of interpolator in the short term.

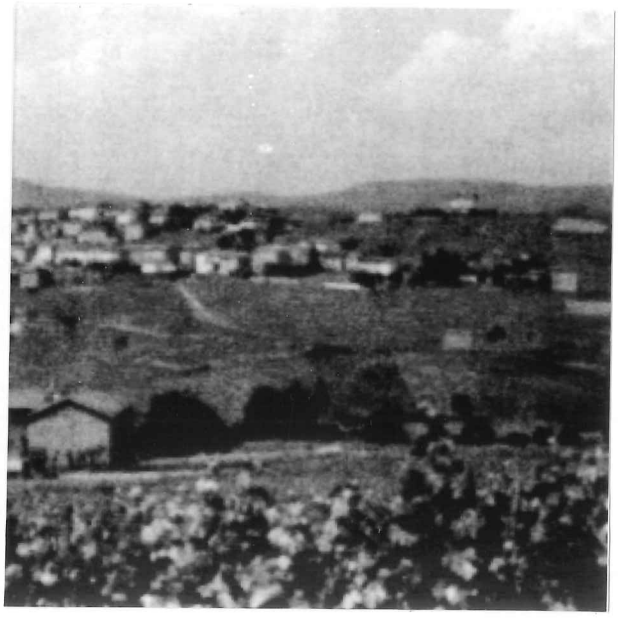

Figure 7.12: Frame 1 of CLOCHE.

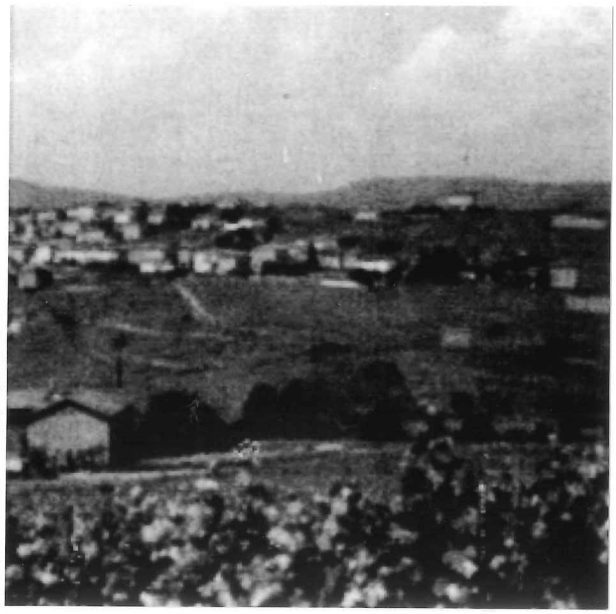

Figure 7.13: Frame 2 of CLOCHE

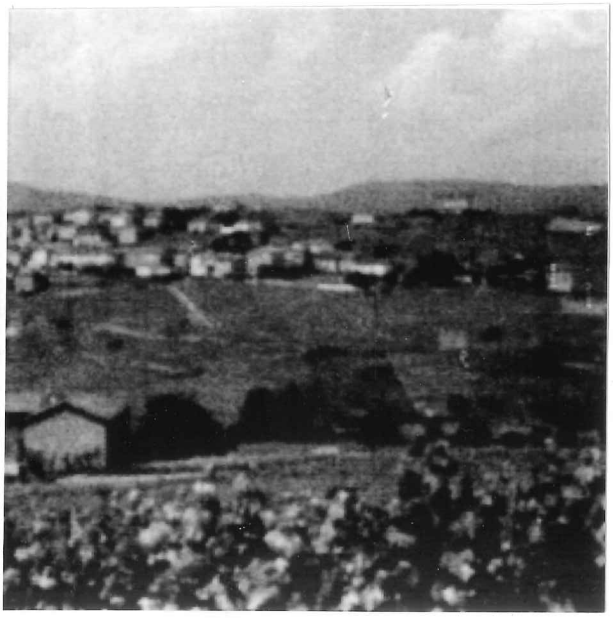

Figure 7.14: Frame 3 of CLOCHE.

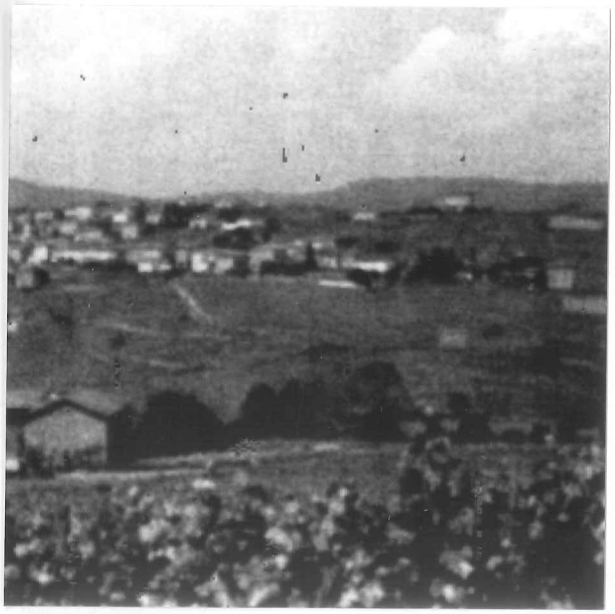

Figure 7.15: Detected Blotches (red) using SDIa.

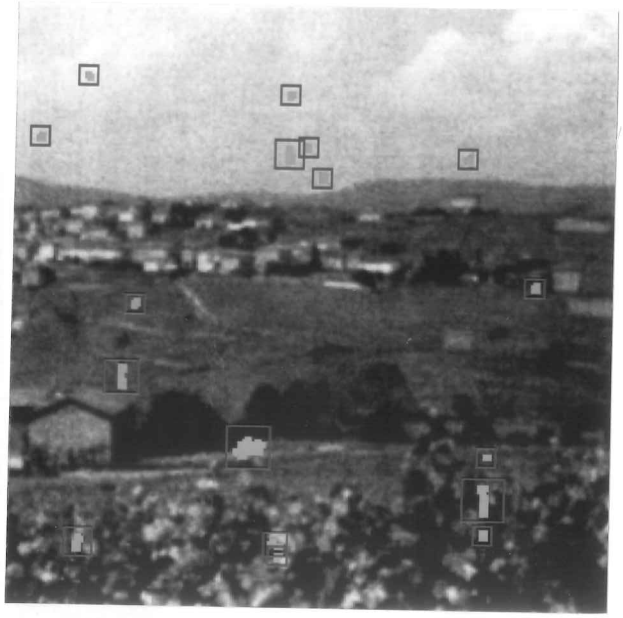

Figure 7.16: Post Processed Detection field: Yellow = Detected Blotches, Red = Bounding Boxes.

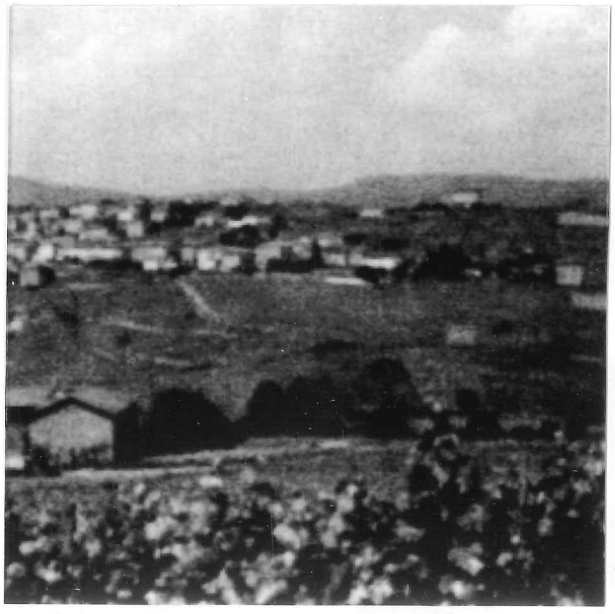

Figure 7.17: Restored Frame 2 of CLOCHE using ML3Dex.

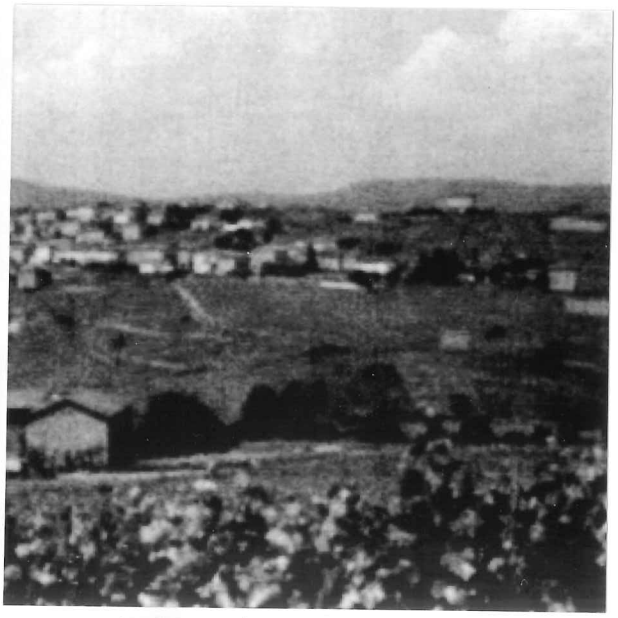

Figure 7.18: Restored Frame 2 of CLOCHE using 3D AR (9:8).

Figure 7.19: Frame 1 of CLOCHE2.

Figure 7.20: Frame 2 of CLOCHE2

Figure 7.21: Frame 3 of CLOCHE2.

Figure 7.22: Detected Blotches (red) using SDIa.

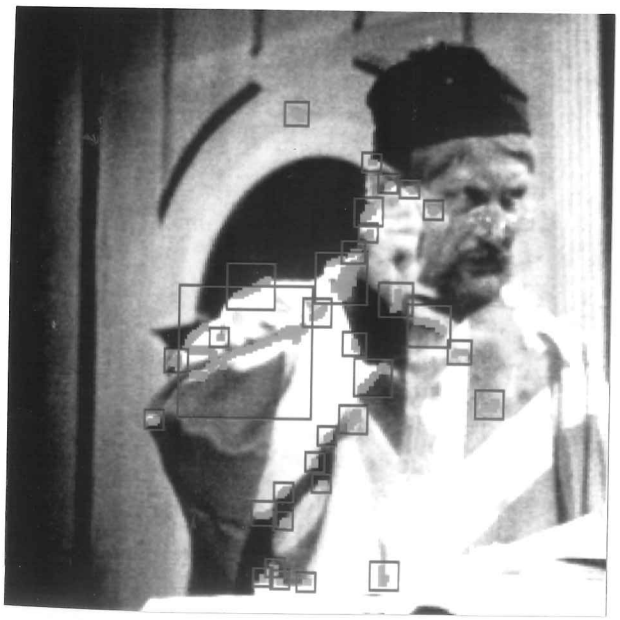

Figure 7.23: Post Processed Detection field: Yellow = Detected Blotches, Red = Bounding Boxes.

Figure 7.24: Restored Frame 2 of CLOCHE2 using 3D AR (9:8).

Figure 7.25: Restored Frame 2 of CLOCHE2 using ML3Dex.

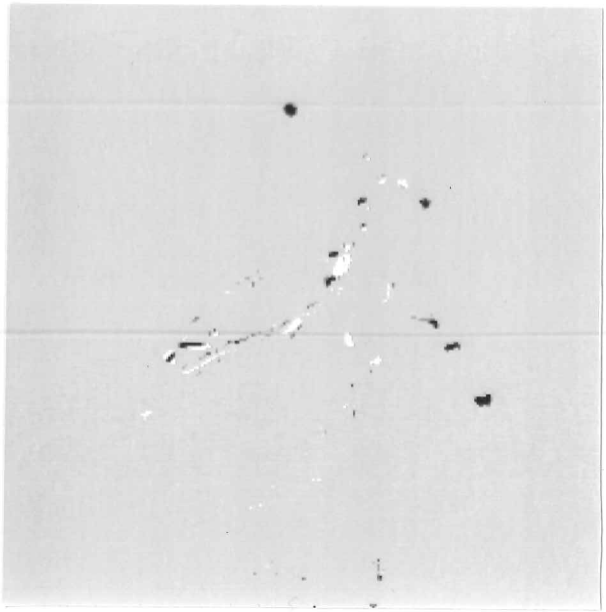

Figure 7.26: Error field: Frame 2 - Restoration using ML3Dex.

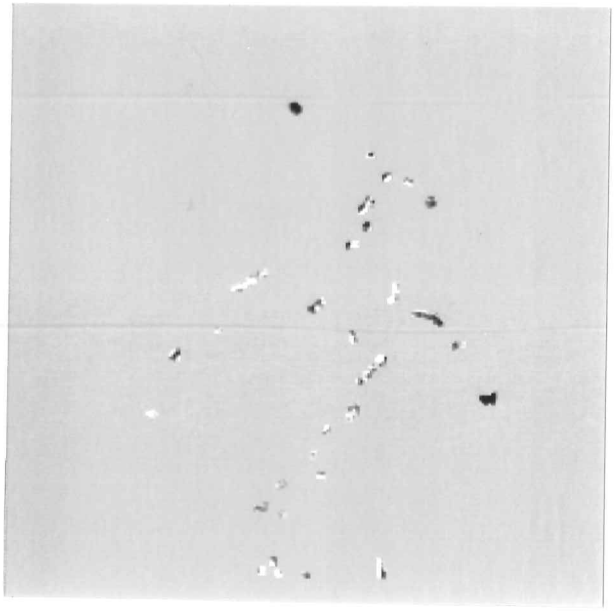

Figure 7.27: Error Field: Frame 2 - Restoration using 3D AR (9:8).

NOISE REDUCTION FOR IMAGE SEQUENCES

Noise is one of the most common forms of distortion observed in image sequences. It can be encountered both due to degradation of the original film material or due to transmission or receiver noise in the case of live video streams. A particularly severe level of noise is observed in video sources such as Electron Microscopes and cameras that are sensitive to non–visible radiation.

As outlined in the introduction to this thesis, there have been many proposed solutions to the problem of noise reduction in image sequences. The trend has been toward motion compensated temporal filtering techniques and recently optimal 3D filtering has been applied. This chapter considers the use of three dimensional motion compensated Wiener filters both in the spatio–temporal domain and in the frequency domain. These filters are seen to be a viable alternative to the approaches discussed by previous authors. One optimal filter that was presented in [58] has not been compared but it is discussed in Appendix F.

8.1 Motion Compensated Wiener Filtering

Wiener filtering is a well known technique for the reduction of noise in degraded signals. It has been used to good effect by Vaseghi [94] for reducing the noise in archived Gramophone recordings. The filter can be applied both in the frequency domain, as an approximation to the IIR filter, or in the spatio–temporal domain as an FIR filter.

At this stage it should be clear to the reader that provided good motion estimates can be obtained, a 3D implementation of a filter would provide substantial improvement over a 2D implementation. Therefore, the approach considered here uses a 3D filtering operation on data from three motion compensated frames in a sequence. The data in a block of $N \times N$ pixels in each frame is extracted allowing for motion, to give a data volume which is filtered to suppress noise in the central frame. As is typical for motion compensated filtering techniques, therefore, the algorithm begins by estimating the motion vectors that map the data in the current frame n into the next frame $n+1$ and the previous frame $n-1$. These vectors are then used to compensate the necessary data.

The observed image sequence model is defined by the following equation.

$$g(i,j,n) = I(i,j,n) + \eta(i,j,n) \tag{8.1}$$

Where $g(i,j,n)$ is the observed signal grey scale value at position (i,j) in the nth frame, $I(i,j,n)$ is the actual non–degraded signal and $\eta(i,j,n)$ the added gaussian noise of variance $\sigma_{\eta\eta}$. Appendix E outlines the derivation of the Wiener filter for noise reduction in a signal which is degraded according to that equation. The filter is derived by first of all deciding on a filter structure, either FIR or IIR, and then finding the coefficients which minimize the expected value of the squared error between the filter output and the original, clean signal.

8.1.1 The 3D IIR/3D Frequency Domain filter

The IIR form of the filter can be expressed as a frequency domain operation which requires an estimate of the Power Spectrum of the original, clean data to yield the attenuation factor required for each frequency bin. The 3D Wiener filter in the frequency domain is then given as below.

$$A(\omega_1,\omega_2,\omega_3) = \frac{P_{gg}(\omega_1,\omega_2,\omega_3) - P_{\eta\eta}(\omega_1,\omega_2,\omega_3)}{P_{gg}(\omega_1,\omega_2,\omega_3)} \tag{8.2}$$

Where $A(\omega_1,\omega_2,\omega_3)$ defines the frequency response of the filter, and P_{gg}, $P_{\eta\eta}$ refer to the Power spectral densities (PSD's) of the degraded and noise signals respectively. The arguments to the functions refer to *discrete* frequency bins as gained via the 3D DFT[1].

It is important to recognise that although the 3D IIR filter can be implemented in the 3D frequency domain using equation 8.2, the resulting filter is no longer IIR in practice. In practice, via the 3D DFT, the 3D Frequency domain Wiener filter operates only on a finite input data volume whereas a 3D IIR filter (as it is normally implemented) would additionally operate on previous outputs.

The noise is assumed to be uncorrelated with the original signal, and so the following equation applies.

$$P_{gg} = P_{ii} + P_{\eta\eta} \tag{8.3}$$

With this in mind it follows that the effect of the Wiener filter is to attenuate the frequency components of $g(i,j,k)$ according to the observed power of that particular component. When this is high, $P_{gg} >> P_{\eta\eta}$ less filtering is done, and when this power is low, the component is heavily attenuated. This is a useful property for images in particular [59, 1]. In regions of high image activity, such as highly textured areas and edges, there is less attenuation of the signal. In areas of low activity such as uniform areas, more attenuation is achieved. The signal detail is therefore less attenuated than in the uniform regions. The human visual system is known to be less sensitive to noise in regions of high activity. This is a useful bonus since the areas in which the filter will attenuate less noise corresponds to the areas in which noise is less easily observed.

The situation is exactly the same along the temporal axis. The activity along the temporal axis depends on the accuracy of motion estimation. When the motion estimation is not accurate,

[1]The n–dimensional DFT is a separable operation and is implemented as a recursive DFT operation in orthogonal directions.

the filter will automatically reduce the contribution of the temporal information to the noise reduction. This property makes the filter robust to even bad errors in motion estimation. Of course the amount by which it reduces the attenuation when this problem occurs may not correspond to an optimal result in human visual terms. Nevertheless the filter has an advantage in this respect in comparison with motion compensated frame averaging [13] which tends to blur the image when the motion is not correctly estimated.

Practical Considerations

The derivation of the 3D Frequency domain Wiener filter is a simple extension of the standard 1D framework. However, in a practical implementation there are outstanding considerations.

1. Estimating the PSD:

   As indicated by equation 8.2, this Wiener filter can be defined in terms of the PSD of the observed degraded signal and the noise PSD. The work presented here does not automatically estimate the noise PSD, rather, it is a user selectable parameter which is altered to suit the tastes of the viewer. This is due to the fact that the noise PSD is assumed to be a constant level across all the frequency components.

   The PSD of the degraded signal is estimated by the magnitude of the 3D DFT. To prevent spectral leakage effects, the signal must be windowed prior to the DFT operation. Many window functions exist for this purpose. The half–cosine window is chosen here, following [102]. For a block size of $N \times N$ the window is defined as below.

$$\begin{aligned} w(n_1, n_2) &= \cos(\frac{n_1\pi}{2N}) \cos(\frac{n_2\pi}{2N}) \\ \text{for } n_1, n_2 &= -(N - \frac{1}{2}), -(N - \frac{1}{2}) + 1, \ldots, (N - \frac{1}{2}) \end{aligned} \tag{8.4}$$

   This window is called the *analysis* window since it is used to assist in acquiring an estimate of the Fourier Transform of the image data.

   The image information in each frame is therefore windowed with this 2D half–cosine window prior to taking the DFT. There is an argument for implementing a similar analysis window along the temporal axis since again there would be problems in taking the DFT in this direction. In practice the visible effect of using a half–cosine window across the 3 frames[2] is found to be small.

2. Overlapped Processing:

   The entire image cannot be treated with the same Wiener filter since the image information changes across the frame. Therefore, the image is broken into blocks of size $N \times N$. Each block is compensated for motion across the three frames used so that in restoring the information in a block in the current frame n a data volume of size $N \times N \times 3$ is used.

[2] Essentially, a 3 tap half–cosine window.

It is common in such a block based algorithm that the non–stationarity of the image would cause blocking artefacts across the edges of the blocks. To suppress this effect the processed blocks can be overlapped [102]. In this implementation the blocks are overlapped by half their horizontal and vertical dimensions, an overlap of 2 : 1. If the processed blocks were merely placed such that one half of the next block replaced one half of the current block then artefacts could still occur. Overlapped processing implies windowing the output data such that when the overlapped blocks are summed, the effective signal gain is 1. Therefore any output pixel at the edge of a block has a contribution from several blocks around it.

The output or *synthesis* window must be chosen with regard to the analysis window. The two windows complement each other and taken together must not change the net gain through the noise reduction system. In this case, using a half–cosine window (and 2:1 overlap) as synthesis and analysis windows yields a net gain of unity as required. The net windowing effect is that of a raised cosine function defined below.

$$\begin{aligned} w(n_1, n_2) &= \cos^2(\frac{n_1\pi}{2N})\cos^2(\frac{n_2\pi}{2N}) \\ \text{for } n_1, n_2 &= -(N-\frac{1}{2}), -(N-\frac{1}{2})+1, \ldots, (N-\frac{1}{2}) \end{aligned} \tag{8.5}$$

Overlapped processing has not been necessary so far in the work of the earlier chapters. This is primarily due to the fact that the solution to the noise suppression problem involves filtering the entire image as opposed to selective treatment. Chapter 9 considers this point.

3. Noise margin:

The frequency domain Wiener filter essentially involves estimating an attenuation for each frequency component in the degraded signal. In the calculation of each factor it is possible that $P_{gg}(\omega_1, \omega_2, \omega_3) < P_{\eta\eta}(\omega_1, \omega_2, \omega_3)$ in the numerator of the expression 8.2. This would result a negative attenuation which is impractical. The most common method for dealing with this problem is to set $P_G - P_N = 0$ when $P_G < P_N$ [59, 61]. Note that the frequency arguments have been dropped and the notation simplified.

However, this solution is somewhat drastic and can lead to ringing or patterned noise artefacts in the output frame. To avoid this difficulty, an alternative mapping may be used which reduces the attenuation of the filter for low values of P_G. This implies that more noise is left in the image, but this is preferable to the artefacts that would otherwise occur. The new mapping is as follows, where the Wiener filter is defined as $\frac{\alpha}{P_G}$.

$$\alpha = \begin{cases} P_G - P_N & \text{for } P_G > \beta P_N \\ \frac{\beta-1}{\beta} P_G & \text{otherwise} \end{cases} \tag{8.6}$$

When $\beta = 1$ the criterion becomes the same as that used previously in [59, 61]. β is a user defined parameter that governs the amount of noise left in the restoration, and is therefore called the Noise Margin. The mapping is illustrated in figure 8.1. The modified Wiener

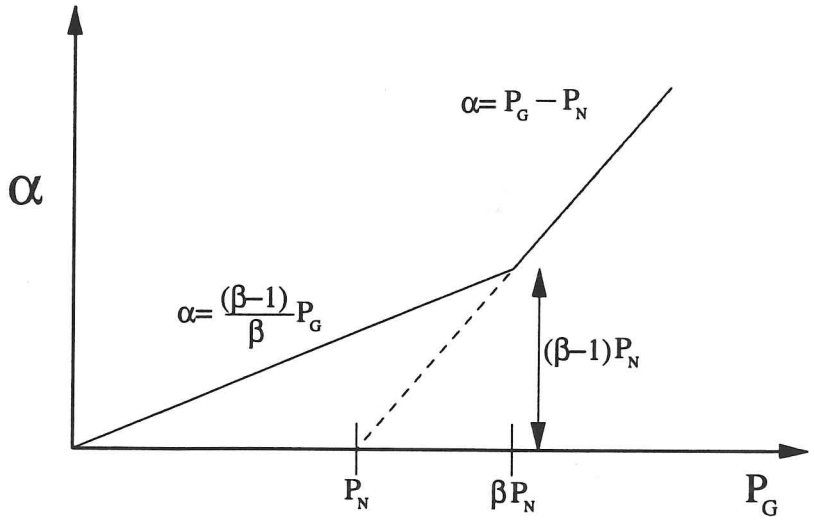

Figure 8.1: The mapping (solid line) for the numerator of the Wiener filter, α.

filter attenuation is given below.

$$H(\omega_1,\omega_2,\omega_3) = \text{MAX}\left\{ (\frac{\beta-1}{\beta}),\ \frac{P_{gg}(\omega_1,\omega_2,\omega_3) - P_{\eta\eta}(\omega_1,\omega_2,\omega_3)}{P_{gg}(\omega_1,\omega_2,\omega_3)} \right\} \tag{8.7}$$

Note that since $\left(\frac{\beta-1}{\beta}\right)$ is a constant, the effect is to define a minimum, non–zero, value of the filter coefficient at frequency bins where α would normally be set to zero.

8.1.2 The 3D FIR filter

This filter is defined in terms of the autocorrelation sequence of the degraded signal and the cross correlation of the degraded and original signals. If the spatial and temporal extent of the FIR filter is such that the filter solution includes all the significant terms in the relevant correlation sequences, then there is little difference between the FIR and IIR filters.

The filter solution is derived in appendix E, and the result is stated here. The filter is of the form shown below.

$$\hat{I}(i,j,n) = \sum_{k_1=-N_1}^{N_1} \sum_{k_2=-N_2}^{N_2} \sum_{k_3=-N_3}^{N_3} a(k_1,k_2,k_3) g(i+k_1, j+k_2, n+k_3) \tag{8.8}$$

The filter can be defined with the same notation that was used to define the AR model equations in chapter 3. The support of the filter can then be defined more generally as below.

$$\hat{I}(\mathbf{x}) = \sum_{k=0}^{N-1} a_k g(\mathbf{x}+\mathbf{q}_k) \tag{8.9}$$

Note that in this case, a_0, which corresponds to the support vector $\mathbf{q}_k = [0\ 0\ 0]$ is not necessarily unity. The filter coefficients can be arranged into a vector $\mathbf{a}$, and the solution for the coefficients is given by the following equation.

$$\mathbf{a} = [\mathbf{R}_{gg}]^{-1}\mathbf{r}_{ig} \tag{8.10}$$

The correlation terms in the matrices used in the above expression are defined as below.

$$\mathbf{R}_{gg}(r,c) = E[g(\mathbf{x}+\mathbf{q}_r)g(\mathbf{x}+\mathbf{q}_c)]\ r,c = 0...N-1$$

$$\mathbf{r}_{ig}(c) = \begin{cases} E[(g(\mathbf{x}))^2] - \sigma_{\eta\eta}^2 & \text{for } c = 0 \\ E[g(\mathbf{x})g(\mathbf{x}+\mathbf{q}_c)] & \text{otherwise} \end{cases}$$

The filter therefore operates on a 3 block data volume to give a filtered output for the centre frame.

Practical Considerations

In a similar manner to the previous discussion regarding the frequency domain Wiener filter, the following points are notable.

1. Data Windowing:
   No analysis window is used prior to estimating the correlation terms required since this would detrimentally affect the values measured. To prevent blocking artefacts, the output is windowed and overlapped with a 2:1 ratio in the same way as previously discussed. Because the analysis window is effectively a rectangular function, a raised cosine window is needed as the synthesis window prior to overlapping. This window is defined by equation 8.5.

2. The Noise Margin:
   In a similar fashion to the mapping used for the IIR Wiener filtering, the value for $r_{ig}(0) = \alpha = r_{gg}^2(0) - \sigma_{nn}^2$ that is used, is defined by a mapping shown below.

$$\alpha = \text{MAX}((\frac{\beta-1}{\beta})r_{gg}(0),\ (r_{gg}(0) - \sigma_{\eta\eta}^2)) \tag{8.11}$$

3. Ill Conditioning:
   It is possible that the solution to equation 8.10 in a real situation becomes ill conditioned. It is best to detect this condition using some form of eigen analysis. To keep computation low, a less effective yet satisfactory solution is adopted. After finding **a**, equation 8.10 is evaluated, this yields a calculated value, $\hat{\mathbf{r}}_{ig}$ for the same observed quantity. If any term in this calculated vector differs from the observed term by more than 5% then the solution is labelled as ill conditioned. This situation does not occur often but when it does occur it is usually in uniform regions. Therefore, the filter coefficients are all set to $\frac{1}{N}$ when this condition is detected. The output of the filter is then the average of all the pixels in its support. It would be useful to consider the effect of using eigen analysis to detect ill–conditioning in future work.

8.2 Results

The two filters presented above were compared with several filters that have been presented previously for noise reduction. The filters were compared with respect to noise reduction of 20 frames of the Cheers sequence degraded with Gaussian noise of variance 100. This gave an SNR of 20db. However, the MSE of the degraded signal was measured at about 70 due to quantization and clipping effects. A 3 level multiresolution BBM motion estimator (as described in chapter 2) was used in all cases with the parameters kept the same as follows: Blocksize 9×9 at levels 0,1 and 5×5 at level 2; threshold MAE = 10 at each level, $r = 1.5,\ 1.2,\ 1.0$ for the highest to lowest resolution level. A variance of 0.5 was used for the Gaussian low pass filter (to create the pyramid) which was of size 9×9. The various filters that were compared are described below. The block based vector field that was generated by the algorithm was interpolated using bilinear interpolation to provide the filters with motion vectors where required.

1. The Temporal Wiener filter: Presented in [59], and discussed earlier in an implicit motion compensated scheme [61], this filter is essentially a one tap Wiener filter. The filter was reviewed in Chapter 4 and was defined as

$$\hat{I}(i,j,n) = \frac{\sigma_{gg}^2 - \sigma_{nn}^2}{\sigma_{gg}^2}(g(i,j,n) - \bar{g}(i,j,n)) + \bar{g}(i,j,n) \tag{8.12}$$

   $\hat{I}(i,j,n)$ is the Wiener estimate of $I(i,j,n)$ and $\bar{g}$, σ_{gg}^2 are the mean and variance of the observed, degraded signal, $g(i,j,n)$. The filter does not explicitly incorporate information from more than one frame, in fact it operates by scaling the current pixel. The multi-frame information comes from the parameter estimates which are calculated from 3 motion compensated pixels[3]. The mean of the original signal, $\bar{I}(i,j,n)$ is assumed to be well approximated by the mean of the observed degraded signal, given that the noise is white. A noise margin, β can also be presented for this filter. The margin is set to unity for these experiments since the noise variance is known.

2. Frame Averaging: The output of this filter is the average of the 3 motion compensated pixels from 3 frames. The particular implementation used was presented by Boyce in [13]. The output of the filter is as follows

$$\hat{I}(i,j,k) = \frac{1}{3}[g(i,j,k) + g(i + sx_{k,k-1}, j + sy_{k,k-1}, k-1) + g(i + sx_{k,k+1}, j + sy_{k,k+1}, k+1)] \tag{8.13}$$

3. Temporally Recursive Filtering: Presented in [19], this filter is of the form

$$\hat{I}(i,j,k) = \alpha g(i,j,k) + (1-\alpha)\hat{I}(i + sx_{k,k-1}, j + sy_{k,k-1}, k-1) \tag{8.14}$$

[3]Of course these estimates can be calculated using a data volume, but to avoid problems with blurring it was decided to use the temporal estimate.

It was discussed in Chapter 4. The attenuation of the filter is regulated by the scalar α. This is varied according to the magnitude of the backward displaced frame difference at the current pixel to be filtered. This difference is defined as

$$e = g(i,j,k) - g(i + sx_{k,k-1}, j + sy_{k,k-1}, k-1) \tag{8.15}$$

When this error is large, α is set closer to unity so that the filtering is reduced. The opposite is the case when the error is small. α is varied according to the piecewise characteristic defined below.

$$\alpha = \begin{cases} \alpha_b & \text{for } |e| \leq P_b \\ (\frac{\alpha_e - \alpha_b}{P_e - P_b})(|e| - P_b) + \alpha_b & \text{for } P_b < |e| \leq P_e \\ \alpha_e & \text{for } |e| > P_e \end{cases} \tag{8.16}$$

The parameters used were as follows: $\alpha_e = 1.0$, $\alpha_b = 0.5$, $P_e = 40.0$, $P_b = 20.0$.

4. Recursive Frame Averaging: This filter is the same as that discussed above except that $\alpha_b = \frac{1}{N}$, where N is the number of frames processed. Therefore, when $|e|$ is small, and the motion estimation has been effective, the output is the average of all the past frames.

5. The 3D IIR/3D Frequency Domain Wiener Filter: As defined earlier, with $\beta = 1.0$, $\sigma_{nn}^2 = 100.0$. A Block size of 16×16 was used.

6. The 3D FIR Wiener filter: As defined earlier with $\beta = 1.0$, $\sigma_{nn}^2 = 100.0$. A Block size of 16×16 was used. The filter was non–causal using the support of a square of 3×3 pixels in each of 3 frames.

Figure 8.2 shows the MSE of the restored frames with respect to the original frames for each of the filters used. The 3D Frequency Domain Wiener filter is seen to give the best performance overall in terms of MSE. The frame averaging filters perform the worst. The 3 frame averaging filter is sensitive to erroneous motion estimation, causing blurring. The recursive frame averager does not perform well because it can consider too many frames in its current filter output again causing blurring. It is difficult to effectively control the operation of this filter with the piecewise linear characteristic used. There is an increase in the observed MSE after frame 10 because from that frame onwards the motion of the figures in the scene increases substantially. Predictably, the filters with less spatial support are more sensitive to this because they are more affected by motion estimation error.

The MSE is however, a bad measure of image quality and figures 8.7 to 8.12 show photographs of a zoom on the 12th frame of the restored sequences. Figures 8.3, 8.4, 8.5 show the zoomed degraded sequence and figure 8.6 shows a zoom on the original 12th frame. The frames resulting from the temporal Wiener filter are sharper to some extent than the 3D DFT version, but less noise suppression is observed. The photographs illustrate well the compromise that must be struck. Note that although the MSE plots indicate that the Temporal Wiener filter is better

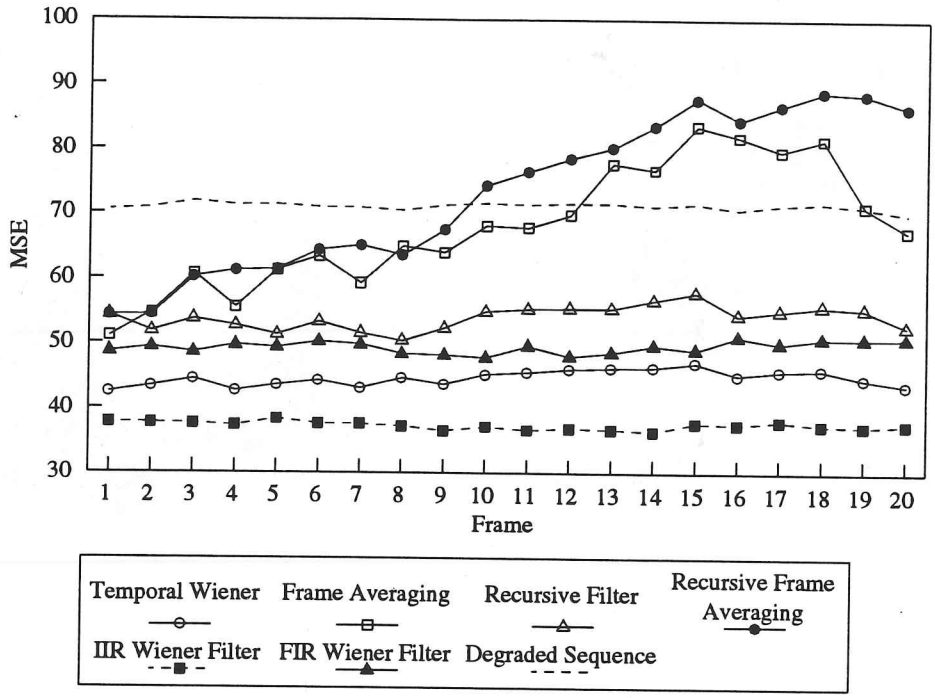

Figure 8.2: Mean Squared Error of various restorations on Cheers. (Note that the IIR filter refers to the 3D frequency domain implementation.)

Figure 8.3: Frame 11 of Cheers, $\sigma_{nn}^2 = 100.0$.

Figure 8.4: Frame 12 of Cheers, $\sigma_{nn}^2 = 100.0$.

Figure 8.5: Frame 13 of Cheers, $\sigma_{nn}^2 = 100.0$.

Figure 8.6: Original frame 12.

Figure 8.7: 3D IIR/3D Frequency Domain Wiener Filter, $\sigma_{nn}^2 = 100.0$, $\beta = 1.1$.

Figure 8.8: 3D FIR Wiener Filter, $\sigma_{nn}^2 = 100.0$, $\beta = 1.1$.

Figure 8.9: Temporal Wiener filter, $\sigma^2_{nn} = 100.0$.

Figure 8.10: Frame Averaging.

Figure 8.11: Recursive frame averaging, $\alpha_b = \frac{1}{N}$.

Figure 8.12: Recursive filtering, $\alpha_b = 0.5$.

than the 3D FIR Wiener filter, the photographs show otherwise. The 3D FIR filter retains more residual noise than the 3D Frequency Domain filter around high contrast edges. This is not well illustrated by the photographs because the grey scale reproduced represents only 6 bits, and the images are all 8 bit grey scale. The reader is referred to the accompanying video tape for a demonstration of this observation. The MSE of the 3D FIR filter output therefore reflects the effects of a low level residual noise. Visually there is a difference between the Frequency Domain Wiener filter result and the FIR filter, however the FIR filter result is still better than that corresponding to the temporal Wiener filter. The results show that the Wiener filters are consistently better than those previously introduced.

The 3 types of Wiener filters compared can remove more noise by increasing the noise variance parameter used. However for the purposes of this comparison this was not done. The Wiener filter works well because it is able to implicitly vary the attenuation of the filter with motion estimation accuracy. When there is an error in motion estimation, the DFT in the temporal direction would result in components of a high amplitude. The 3D frequency domain filter would then reduce the attenuation of these components resulting in more residual noise at the output. This is a desirable effect which avoids signal degradation in regions where the noise is not usually noticeable since the motion estimation errors would typically occur in areas of high temporal activity e.g. fast motion.

For the FIR filter, motion estimation errors would reduce the strength of the temporal correlation of the blocks and hence reduce the contribution of the corresponding taps, yielding the same behaviour. For the temporal one tap Wiener filter, the variance of the observed pixels would increase with motion estimation error and so the attenuation would be reduced.

It is reasonable that the filter performances in order of merit (best first) are 3D Frequency Domain, FIR, and Temporal because the Temporal filter has no 2D fall back mode. The FIR filter does not perform as well as the frequency domain filter in terms of MSE because it is not certain that all the significant terms in the correlation function of the signal have been considered with the FIR filter. Visually, however, the distinction between the frequency domain and FIR filters is small.

8.3 Real Sequences

Figures 8.13 to 8.15 show 3 frames of a cartoon sequence which are degraded by noise. The frames are of resolution 256×256 and 8 bit grey scale. The motion clearly poses problems to any of the standard motion estimators since some shapes change drastically from frame to frame. A 4 level pyramid was used with a 9×9 block size and ± 8 pixels search space at each level.

Four filter outputs are shown as figures 8.16 to 8.19. They show that the visual quality of the 3D Frequency Domain Wiener filter is superior. The performance of that filter is better in uniform regions because it is easier to generate a uniform region by attenuating AC frequency

components than by using a spatio–temporal domain weighted average approach[4]. Both Wiener filters used a 16×16 blocksize, $\beta = 1.1$ and $\sigma_{nn}^2 = 40.0$. The Temporal Wiener filter also used the same parameters. A border of 16 pels all around the image has been omitted in the restoration, hence the dark borders.

Figures 8.13 and 8.14 show clearly that the Wiener filter attenuation is reduced in regions of fast motion. Around the legs of the cow there is a higher level of residual noise. The residual noise manifests itself in different ways depending on the filter used. The residual from the FIR filter appears more correlated than that left by the Frequency Domain filter. In the latter case it may be difficult to see this residual noise from the photographs. The reader is referred to the video supplement for a more accurate reproduction. These effects are less visible in the case of the Temporal Wiener filter because it leaves more noise in the image overall. The noise in the stationary regions of the image has been effectively reduced by all the Wiener filters, with the best performance being given by the Frequency Domain filter.

The averaging process shown in figure 8.19 does not yield a good result. The restoration is blurred because of errors in motion estimation. This cartoon is an extreme case where it is impossible for a BBM algorithm to give an accurate estimate for motion all over the moving object. The example illustrates well the need for a noise reduction algorithm to be robust to motion estimation errors. Only in the case of a slow moving or purely translational sequence would the frame averaging process be successful.

It is important to note that the Wiener filters have been designed to be optimal with regard to mean squared error. This error measure is certainly not optimal as far as Human Visual characteristics are concerned and this explains why the filtered output may show artefacts.

These results are illustrated on the supplementary video tape. The reader is directed to the recording to gain a visual appreciation of the relative performance of these filters. Both the experiments with the Cheers sequence and this cartoon have been recorded on the tape. The video shows that the Frequency Domain Wiener filter is the best performer overall.

8.4 Summary

The Wiener filter for spatio–temporal signals has been presented. It has been shown to compare favourably with other temporal only filters. Overlapped processing has been used to suppress blocking artefacts. The Wiener filter performs well because it is robust to motion estimation errors. When this error is large, the attenuation is reduced. The FIR Wiener filter is the most computationally intensive followed by the frequency domain filter, the Temporal Wiener filter and the others which are roughly similar in this respect. This must be considered when choosing a filter for a particular application. Although the Frequency Domain filter can attenuate noise

[4]The FIR filter

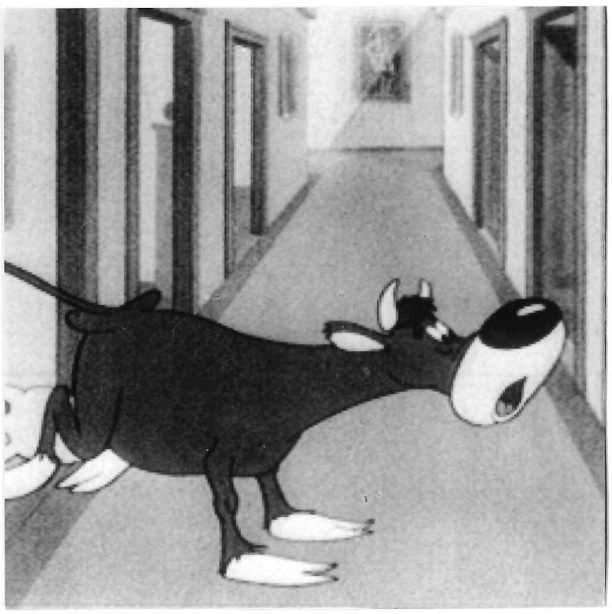

Figure 8.13: Frame 1 of Cartoon.

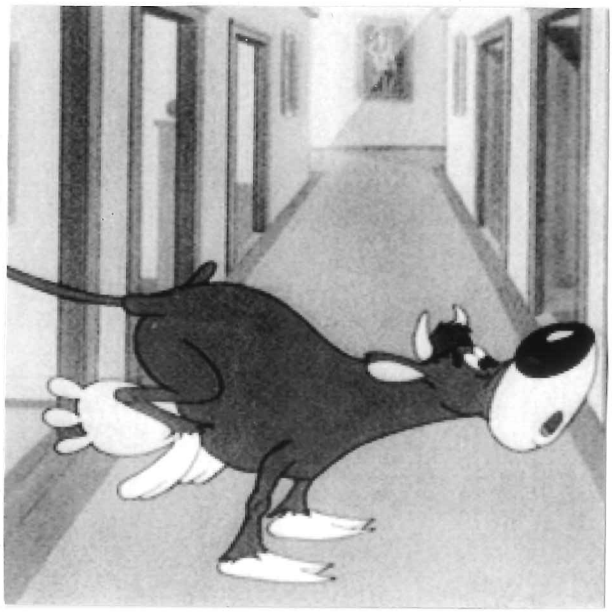

Figure 8.14: Frame 2 of Cartoon.

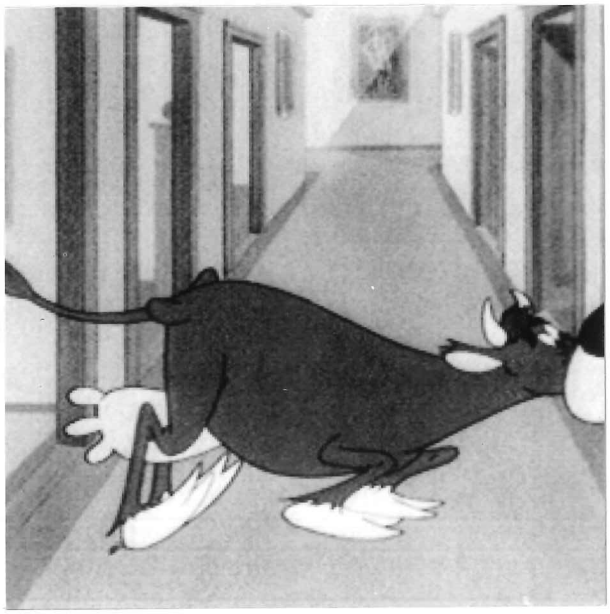

Figure 8.15: Frame 3 of Cartoon.

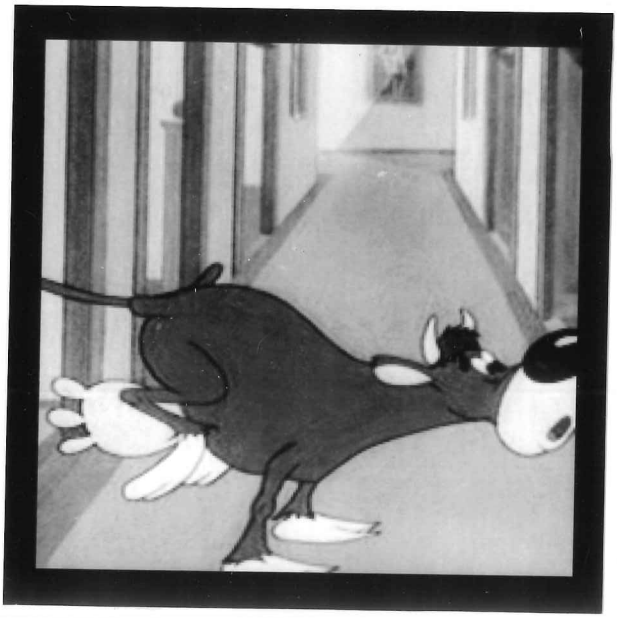

Figure 8.16: 3D IIR Wiener Filter, $\sigma_{nn}^2 = 40.0$, $\beta = 1.1$.

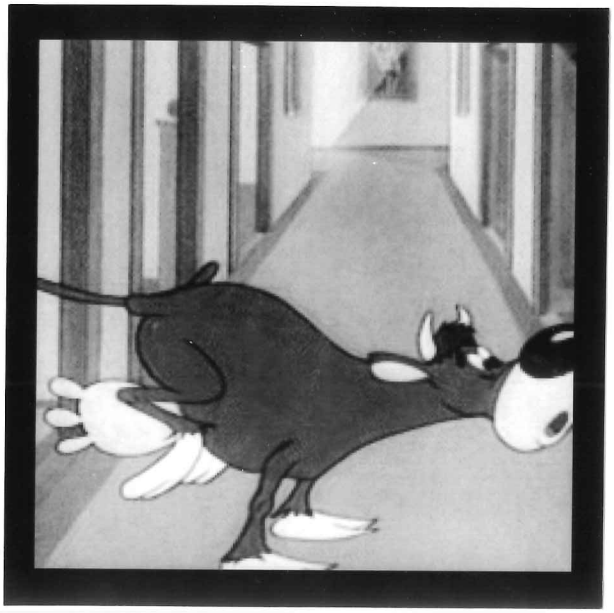

Figure 8.17: 3D FIR Wiener Filter, $\sigma_{nn}^2 = 30.0$, $\beta = 1.3$.

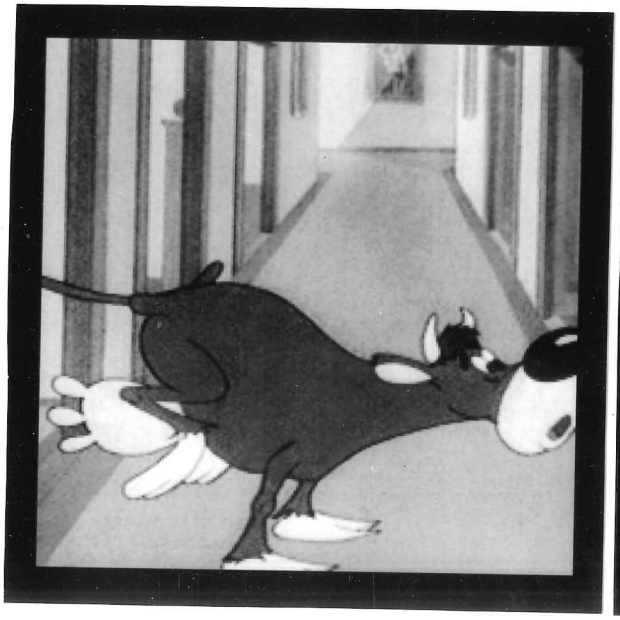

Figure 8.18: Temporal Wiener filter, $\sigma_{nn}^2 = 40.0$, $\beta = 1.1$.

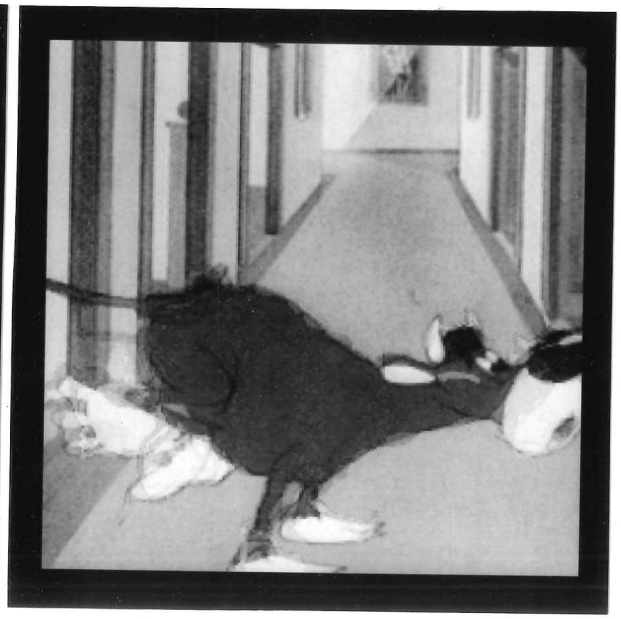

Figure 8.19: Frame Averaging.

more effectively than the Temporal filter, it requires more computation. Therefore, in cases where the noise levels are relatively low, the Temporal filter may be the better choice. As far as output quality is concerned, the Frequency Domain filter performs best out of the those examined.

more effectively than the Temporal filter, it requires more computation. Therefore, in cases where the noise levels are relatively low, the Temporal filter may be the better choice. As far as output quality is concerned, the Frequency Domain filter performs best out of the those examined.

CONCLUSIONS AND FURTHER WORK

This thesis has presented algorithms for removing two different types of distortion, Dirt and Sparkle and additive white noise. The distortions were treated as separate issues. The work has highlighted the usefulness of motion compensated algorithms in the treatment of image sequences. All the methods presented operate on 3 motion compensated frames with the goal being the restoration of the central frame. There exist many Motion Estimation algorithms, and a Multiresolution Block Matching technique was chosen for use with the restoration algorithms that were developed. The Block Matching technique has been found to be robust to the typical distortion found. Multiresolution motion estimation was found to be invaluable for estimating the large displacements in a typical TV scene.

A solution was also presented for the problem of removing inter–line jitter in digitized TV frames. The method involves a 2D AR model which is used for estimating the relative displacement between the lines. The final algorithm was able to reduce the jitter but many parameters needed to be set, including a maximum allowed estimated displacement. The reason for this is that the reconstructed frame would otherwise show a drift in the image features. Preliminary results have indicated that if that drift were removed then an almost perfect reconstruction results. This image drift has been found to be directly related to an estimated displacement signal which has a varying mean. If this local mean is subtracted from the estimated displacement then the image drift is removed. The method has been found to be quite effective and also improves the robustness of the algorithms by releasing the need for a careful choice of parameters.

Dirt and Sparkle was shown to be a localized distortion. As such it is best treated by isolating the site of the distortion and then interpolating the missing information. Several detectors were considered for this distortion involving a heuristic and a model based approach. The simplest detector which thresholds the forward and backward displaced frame difference at a pixel was found to be the most effective. However, it is conceivable that the model based detector can perform better if further constraints on the model coefficients could be incorporated. One point to be considered is that in the case of a one tap causal model, the coefficient must always be close to unity since the translational approximation is very effective for many image sequences. Therefore, perhaps constraining the central tap in the surrounding frames to be close to unity could yield improved performance.

To interpolate the missing information that is occupied by the Dirt and Sparkle, both median filters and 3D AR models were considered. The AR model based method was found to give the

higher fidelity result. However, for large sized distortion, using one AR model is inadequate due to the non–stationarity of the image. Further work needs to be done in order to develop a non–stationary model based interpolator for this purpose. The model based interpolator is further complicated by the need to explicitly measure the size of the Dirt and Sparkle distortion. This together with the necessary complexity of coefficient estimation and finally interpolation, makes the method unwieldy in practice. The median filter system is therefore found to be a practical solution.

The work presented did not consider that at the site of the distortion, motion vectors would be inaccurate. It was presumed that the Block size of the motion estimation process used would be large enough to suppress the effect of Dirt and Sparkle. While that assumption is satisfactory in practice, it may be possible to improve on the quality of the interpolation by explicitly incorporating a solution. This may take the form of interpolating a motion vector for the missing region using the vectors estimated for the good data around the distortion. The vector median operation [8] may be useful here.

There have been several algorithms presented for noise reduction in image sequences, however none have considered the use of the standard 3D Wiener filter, be it IIR or FIR. The thesis presented these filters and showed that their performance compared favourably with the previously presented methods. Overlapped processing was used to prevent blocking artefacts in the output images. This was not an issue with the removal of Dirt and Sparkle since in that case the problem was a local one and not all the image was treated. Further, the blocking artefacts that may arise in measuring the displaced frame difference used for detecting the distortion were much smaller than the discrepancies caused by the distortion itself. Therefore, for the detection of Dirt and Sparkle, blocking is not as important and overlapped processing is unnecessary.

The noise reduction filters which are effective are those which are robust to motion estimation errors. The Wiener filter is implicitly robust to erroneous motion vectors because an error in motion estimation implies a large magnitude in the AC frequency components that are directed along the temporal axis. When the magnitude of a component is large, then the attenuation is small. The filter is therefore easier to control than the Recursive filter used in which a user-specified piecewise linear characteristic defines how the filter responds to motion estimation errors. However, the extent to which the the attenuation is reduced by the Wiener filter, when bad motion estimation occurs, is not governed by any human visual criterion. If this were the case, one could be assured of a certain visual output quality. Future work in this area could involve consideration of such a constrained Wiener filter.

It is clear that the mean squared error criterion (which is the basis of Wiener filtering) does not necessarily reflect human visual criterion in the appreciation of image degadation. Future work in this area would benefit from the development of a tractable measure for image artefacts. Such a measure would allow the design of algorithms that are optimal with respect to the human visual system. Techniques such as those employed by Karunasekera et al [45] could prove valuable

for more effective image restoration algorithms.

To reduce artefacts in the filter output a noise margin, β, was introduced. It was based on a linear mapping for the numerator of the Wiener filter. Alternative mappings involving a non–linear function, such as a quadratic, may be more useful in this respect and this can be investigated in further work. As a whole, there needs to be more work directed towards noise reduction which takes advantage of the fact that there can be large areas of each frame that are not moving. In these areas a recursive filter can be very effective. In the area where there is motion, the 'memory' of the filter would need to be less. The human is more likely to notice noise in stationary regions and so it would be beneficial to attenuate the signal heavily in those areas.

There is also a need to develop an algorithm that can automatically generate an estimate for the noise variance of a degraded image as the sequence progresses. This is best done in stationary uniform regions and this involves detecting such regions. One approach may be to isolate blocks of small displaced frame difference and subtract the average image grey scale value to give a noise field from which the noise estimate may be found.

Although two major problems occurring in image sequences have been addressed, two obvious problems remain. One is the phenomenon of line scratches. These cannot be detected in the same way as Dirt and Sparkle because they often occur in the same place in several frames. When they do move, the motion is regular and can be tracked by a motion estimator. Therefore, line scratches cannot be characterized by temporal discontinuities in image sequences. However, the movement of these lines is not correlated with the motion of objects around it. Therefore, they may be detected using a combination of motion and contrast clues. Line scratches represent a serious problem in archived film and must be considered for further research.

The second problem is that of shake or roll. The effect is often encountered in video recorded with a hand held video camera. The problem also occurs in telecine transfer mechanisms. To correct this problem an estimate of the global displacement between frames is needed. This displacement must be distinguished from local object motion and purposefully shot pan motion. An algorithm for solving the problem in hand held video cameras has been presented in [43]. The algorithm involves estimating the shake displacement as the vector median of the motion estimates from 4 large blocks. This is combined with a number of heuristics to allow for image pan. However, this problem is definitely a mechanical one, with the motion of the camera showing resonances due to the system being shaken. It may be advantageous to model the evolution of the shake displacement with time using an AR model. Constraints on the global displacement may then be easily incorporated to result in a more efficient algorithm.

The current algorithms, however do represent a useful set of techniques. They have been shown to be effective. The low computational cost of the scratch detector and Wiener filter in particular makes them suitable for a real time implementation. The work has also illustrated the effectiveness of motion compensated spatio–temporal techniques in general. As the speed of

processing continues to increase it will become more appropriate to study the usefulness of the non–linear Bayesian techniques introduced by Geman and Geman [27, 92], which have become quite popular in recent years for 2D image processing. Those techniques are likely to surpass the performance of the current algorithms for scratch detection in particular since they are based on the use of probabilistic estimation.

The reader is referred to the last 10 minutes of the video supplement which gives several examples of the performance of a complete system. The sequences shown were all digitized from actual degraded television movies, and the system used the SDI controlled median filter to remove Blotches followed by a 3D Frequency Domain Wiener filter for noise suppression. There are various split screen displays which allow fair assessment of the results. The restorations represent useful improvements over the degraded originals.

BIBLIOGRAPHY

[1] M. Sezan A. Erdem and M. Özkan. Motion–compensated multiframe wiener restoration of blurred and noisy image sequences. In *IEEE ICASSP*, volume 3, pages 293–296, March 1992.

[2] S. Efstratiadis A. Katsagellos, J. Driessen and R. Lagendijk. Spatio–temporal motion compensated noise filtering if image sequences. In *SPIE VCIP*, pages 61–70, 1989.

[3] I. Abdelqader and S. Rajala. Energy minimisation approach to motion estimation. *Signal Processing*, 28:291–309, 1992.

[4] Bilge Alp, Petri Haavisto, Tiina Jarske, Kai Öistämö, and Yrjö Neuvo. Median-based algorithms for image sequence processing. In *SPIE Visual Communications and Image Processing*, pages 122–133, 1990.

[5] G.R. Arce. Multistage order statistic filters for image sequence processing. *IEEE Transactions on Signal Processing*, 39:1146–1161, May 1991.

[6] G.R. Arce and R.E. Foster. Multilevel median filters, properties and efficacy. In *International Conference on Acoustics Speech and Signal Processing*, pages 824–826, 1988.

[7] G.R. Arce and E. Malaret. Motion preserving ranked-order filters for image sequence processing. In *IEEE Int. Conference Circuits and Systems*, pages 983–986, 1989.

[8] J. Astola, P. Haavisto, and Y. Nuevo. Vector median filters. *Proceedings of the IEEE*, 78:678–689, April 1990.

[9] M. Adams B. Levy and A. Willsky. Solution and linear estimation of 2–d nearest neighbour models. *Proceedings of the IEEE.*, 78:627–641, April 1990.

[10] J. Biemond, L. Looijenga, D. E. Boekee, and R.H.J.M. Plompen. A pel–recursive wiener based displacement estimation algorithm. *Signal Processing*, 13:399–412, 1987.

[11] M. Bierling. Displacement estimation by heirarchical block matching. In *SPIE VCIP*, pages 942–951, 1988.

[12] L. Böröczky, J. N. Driessen, and J. Biemond. Adaptive algorithms for pel–recursive displacement estimation. In *Proceedings SPIE VCIP*, pages 1210–1221, 1990.

[13] J. Boyce. Noise reduction of image sequences using adaptive motion compensaed frame averaging. In *IEEE ICASSP*, volume 3, pages 461–464, 1992.

[14] P. Burt and E. Adelson. The laplacian pyramid as a compact image code. *IEEE transactions on Communications*, 31:532–540, April 1983.

[15] C. Cafforio and F. Rocca. Methods for measuring small displacements of television images. *IEEE transactions on Information Theory*, 22:573–579, 1976.

[16] E. Coyle, J. Lin, and M. Gabboui. Optimal stack filtering and the estimation and structural approaches to image processing. *IEEE Trans on Acoustics, Speecha and Signal Processing.*, 37:2037–2065, December 1989.

[17] T. Dennis. Nonlinear temoral filter for television picture noise reduction. *IEE Proceedings*, 127:52–56, April 1980.

[18] H. Derin and P. Kelly. Discrete–index markov–type random processes. *Proceedings of the IEEE.*, 77:1485–1509, October 1989.

[19] E. Dubois and S. Sabri. Noise reduction in image sequnces using motion compensated temoral filtering. *IEEE Transactions on Communications*, 32:826–831, July 1984.

[20] S. Efstratiadis and A. Katsagellos. A model based, pel–recursive motion estimation algorithm. In *Proceedings IEEE ICASSP*, pages 1973–1976, 1990.

[21] S. Efstratiadis and A. Katsaggelos. A multiple–frame pel–recursive wiener–based displacement estimation algorithm. In *SPIE VCIP IV*, pages 51–60, 1989.

[22] W. Enkelmann. Investigations of multigrid algorithms for the estimation of optical flow fields in image sequences. *Computer Vision graphics and Image Processing*, 43:150–177, 1988.

[23] J. Patrick Fitch, Edward J. Coyle, and Neal C. Gallagher. Median filtering by threshold decomposition. *IEEE Trans. Acoustics and Signal Processing*, 32:1183–1188, December 1984.

[24] S. Fogel. The estimation of velocity vector fields from time–varying image sequences. *Computer Vision Graphics and Image Processing: Image Understanding.*, 53:253–287, May 1991.

[25] N. Galatsanos and R. Chin. Digital restoration of multichannel images. *IEEE Transactions on ASSP*, 37:415–421, March 1989.

[26] Neal C. Gallagher and Gary L. Wise. A theoretical analysis of the properties of median filters. *IEEE Trans. Acoustics and Signal Processing*, 29:1136–1141, December 1981.

[27] S. Geman and D. Geman. Stochastic relaxation, gibbs distributions and the bayesian restoration of images. *IEEE PAMI*, 6:721–741, 1984.

[28] S. Geman and D. McClure. A nonlinear filter for film restoration and other problems in image processing. *CVGIP, Graphical Models and Image Processing*, 54:281–289, July 1992.

[29] M. Ghanbari. The cross–search algorithm for motion estimation. *IEEE Transactions on Communications*, 38:950–953, July 1990.

[30] B. Girod. Motion–compensating prediction with fractional–pel accuracy. *IEEE Transactions on Communications.*, Accepted March 1991.

[31] S. Godsill and P. Rayner. A bayesian approach to the detection and correction of error bursts in audio signals. In *IEEE ICASSP*, volume 2, pages 261–264, 1992.

[32] H. Higgins. The interpretation of a moving retinal image. *Proceedings of the Royal Society, London*, B 208:385–397, 1980.

[33] H. Higgins. The visual ambiguity of a moving plane. *Proceedings of the Royal Society, London*, B 223:165–175, 1984.

[34] B. Horn and B. Schunck. Determining optical flow. *Artificial Intelligence*, 17:185–203, 1981.

[35] T. Huang. *Image Sequence Analysis.* Springer–Verlag, 1981.

[36] B. Hunt. The application of constrained least squares estimation to image resoration by digital computer. *IEEE Transactions on Computers.*, 22:805–812, September 1973.

[37] J. Biemond J. Driessen and D. Boekee. A pel–recursive segmentation and estimation algorithm for motion compensated image sequence coding. In *IEEE ICASSP*, pages 1901–1904, 1989.

[38] L. Boroczky J. Driessen and J. Biemond. Pel–recursive motion field estimation from image sequences. *Visual Communication and Image Representation*, 2:259–280, 1991.

[39] R. Belfor J. Driessen and J. Biemond. Backward predictive motion compensated image sequence coding. In *Signal Processing V: Theories and Applications*, pages 757–760, 1990.

[40] A. Jain. Partial differential equations and finite-difference methods in image representing, part 1: Image representation. *Journal of Optimization Theory and Application*, pages 65–91, September 1977.

[41] A. Jain and J. Jain. Partial differential equations and finite difference in image procesing-part 2: Image restoration. *IEEE Transactions on Automatic Control*, pages 817–834, October 1978.

[42] A.K. Jain. *Fundamentals of Digital Image Processing.* Prentice Hall, 1989.

[43] A. Morimura K. Uomori and H. Ishii. Electronic image stabilisation system for video cameras and VCR's. *SMPTE Journal*, pages 66–75, February 1992.

[44] D. Kalivas and A. Sawchuk. Motion compensated enhancement of noisy image sequences. In *IEEE ICASSP*, volume 1, pages 2121–2124, 1990.

[45] Shanika Karunasekera and N. Kingsbury. A distortion measure for blocking artifacts in images based on human visual sensitivity. *Submitted to IEEE Transactions on Image Processing*, 1993.

[46] J. Kearney, W.B. Thompson, and D. L. Boley. Optical flow estimation: An error analysis of gradient based methods with local optimisation. *IEEE PAMI*, pages 229–243, March 1987.

[47] A. Kokaram. An investigation into frog species classification by call pattern analysis. In *Fourth annual technical conference of the Association of Professional Engineers of Trinidad and Tobago.*, November 1990.

[48] A. Kokaram and W. J. Fitzgerald. Image processing applied to ancient manuscripts. In *20th International Congress of Papyrologists.*, August 1992.

[49] A. Kokaram and P. Rayner. An algorithm for line registration of TV images based on a 2–D AR model. In *Signal Processing VI, Theories and Applications*, pages 1283–1286, August 1992.

[50] A. Kokaram and P. Rayner. Removal of impulsive noise in image sequences. In *Singapore International Conference on Image Processing.*, pages 629–633, September 1992.

[51] A. Kokaram and P. Rayner. A system for the removal of impulsive noise in image sequences. In *SPIE Visual Communications and Image Processing*, pages 322–331, November 1992.

[52] A. Kokaram, A. Stark, and W. J. Fitzgerald. Enhancement and restoration of ancient manuscripts. In *SPIE Conference on Applications of Digital Image Processing XV.*, pages 322–331, July 1992.

[53] K. Fazekas L. Böröczky and T. Szabados. Convergence analysis of a pel–recursive wiener based motion estimation algorithm. In *Time varying image processing and moving object recognition 2.*, pages 38–45. Elsevier, 1990.

[54] J. S. Lim. Image restoration by short space spectral subtraction. *IEEE ASSP*, 28:191–197, April 1980.

[55] Jae S. Lim. *Two-Dimensional Signal and Image Processing.* Prentice-Hall, 1990.

[56] J. Lin, T. Sellke, and E. Coyle. Adaptive stack filtering under the mean absolute error criterion. *IEEE Transactions on Acoustics, Speech and Signal Processing*, 38:938–954, June 1991.

[57] M. Sezan M. Özkan and A. Erdem. LMMSE restoration of blurred and noisy image sequences. In *SPIE VCIP*, pages 743–754, 1991.

[58] M. Sezan M. Özkan, A. Erdem and A. Tekalp. Efficient multiframe wiener restoration of blurred and noisy image sequences. *IEEE Transactions on Image Processing*, 1:453–476, October 1992.

[59] M. Özkan M. Sezan and S. Fogel. Temporally adaptive filtering of noisy image sequences using a robust motion estimation algorithm. In *IEEE ICASSP*, volume 3, pages 2429–2431, May 1991.

[60] S.G. Mallat. Multifrequency channel decompositions of images and wavelet models. *IEEE Transactions on Acoustics, Speech and Signal Processing*, pages 2091–2110, December 1989.

[61] D. Martinez and J. Lim. Implicit motion compensated noise reduction of motion video scenes. In *IEEE ICASSP*, pages 375–378, 1985.

[62] D. Martinez and J. Lim. Spatial interpolation of interlaced television pictures. In *IEEE ICASSP*, volume M9.21, pages 1886–1889, 1989.

[63] D. M. Martinez. *Model–based motion estimation and its application to restoration and interpolation of motion pictures.* PhD thesis, Massachusetts Institute of Technology, 1986.

[64] R. D. Morris and W. J. Fitzgerald. Detection and correction of speckle degradation in image sequences using a 3d markov random field. In *In Proceedings of the International Conference on Image Processing: Theory and Applcations (IPTA '93)*, June 1993.

[65] H. Nagel. Constraints for the estimation of displacement vector fields from image sequences. In *International joint conference on Artificial Intelligence.*, pages 945–951, 1983.

[66] H. Nagel. Displacement vectors derived from second order intensity variations in image sequences. *Computer Vision, Graphics and Image Processing.*, 21:85–117, 1983.

[67] H. Nagel. Recent advances in image sequence analysis. In *Premiere coloque image traitment, synthese, technologie et applications.*, pages 545–558, May 1984.

[68] H. Nagel. Spatio–temporal modeling based on image sequences. In *Int. symposium on imag processing and its applications.*, pages 222–252, 1984.

[69] H. Nagel. Towards the estimation of displacement vector fields by 'oriented smoothness' constraints. In *International joint conference on Pattern Recognition.*, pages 6–8, 1984.

[70] H. Nagel. Image sequences – ten (octal) years – from phenomenology towards a theoretical foundation. In *IEEE ICASSP*, pages 1174–1185, 1986.

[71] H. Nagel. On a constraint equation for the estimation of displacement rates in image sequences. *IEEE PAMI.*, 11:13–30, January 1989.

[72] H. Nagel and W. Enkelmann. An investigation of smoothness constraints for the estimation of displacement vector field from image sequences. *IEEE PAMI*, 8:565–592, September 1986.

[73] A. Netravali and J. Robbins. Motion–compensated television coding: Part 1. *The Bell system technical journal.*, 58:631–670, March 1978.

[74] A. Netravali and J. Robbins. Motion–compensated coding: Some new results. *The Bell system technical journal.*, 59:1735–1745, November 1980.

[75] A. Nieminen, P. Heinonen, and Y. Nuevo. A new class of detail–preserving filters for image processing. *IEEE Trans. PAMI*, 9:74–90, January 1987.

[76] M. Özkan M. Sezan and A. Tekalp. Motion–adaptive weighted averaging for temporal filtering of noisy image sequences. In *SPIE Image Processing Algorithms and Techniques III*, pages 201–212, February 1992.

[77] I. Pitas and A.N. Venetsanopoulos. *Nonlinear Digital Filters Principles and Applications.* Kulwer Academic Publishers, 1990.

[78] W. K. Pratt. *Digital Image Processing.* Wiley, 1978.

[79] T. Reiss. *Recognizing Objects using Invariant Image Features.* PhD thesis, Cambridge University, England, 1992.

[80] J. Riveros and K. Jabbour. Review of motion analysis techniques. *IEE Proceedings*, 136:397–404, December 1989.

[81] J. Robbins and A. Netravali. *Image sequence processing and Dynamic scene analysis.*, chapter Recursive motion compensation: A review., pages 76–103. Springer–Verlag, 1983.

[82] A. Rosenfeld, editor. *Univariate and Multivariate random fields for images.*, pages 245–258. Academic Press, 1981.

[83] A. Rosenfeld and A. Kak. *Digital Picture Processing.* Academic Press, 1982.

[84] R. Schalkoff. *Digital Image Processing and Computer Vision.* Wiley, 1989.

[85] B. Schunck. Image flow: fundamentals and future research. In *IEEE ICASSP*, pages 560–571, 1985.

[86] B. Schunck. The image flow constraint equation. *Computer Vision, Graphics and Image Processing.*, 35:20–46, 1986.

[87] P. S. Spencer. *System identification with application to the restoration of archived gramophone recordings.* PhD thesis, Cambridge University, England, 1990.

[88] C. Srinivas. A stochastic model–based approach for simultaneous restoration of multiple misregistered images. In *SPIE VCIP*, pages 1416–1427, 1990.

[89] R. Srinivasan. Image restoration by spatial filter design. In *SPIE VCIP*, pages 193–197, 1986.

[90] C. Stiller. Motion–estimation for coding of moving video at 8kbit/sec with gibbs modeled vectorfield smoothing. In *SPIE VCIP.*, volume 1360, pages 468–476, 1990.

[91] P. Strobach. Quadtree–structured linear prediction models for image sequence processing. *IEEE PAMI*, 11:742–747, July 1989.

[92] C. Therrien. Statistical model–based algorithms for image analysis. *IEEE Proceedings.*, 74:532–551, April 1986.

[93] J. Tukey. *Exploratory Data Analysis.* Addison–Wesley, 1977.

[94] S. V. Vaseghi. *Algorithms for the restoration of archived gramophone recordings.* PhD thesis, Cambridge University, England, 1988.

[95] S. V. Vaseghi and P. J. W. Rayner. Detection and suppression of impulsive noise in speech communication systems. *Proceedings IEE*, 137:38–46, 1990.

[96] R. Veldhuis. *Restoration of lost samples in Digital Signals.* Prentice Hall, 1980.

[97] K. M. Mutch W. B. Thompson and V. A. Berzins. Dynamic occlusion analysis in optical flow fields. *IEEE PAMI*, 7:374–383, July 1985.

[98] D. Walker and K. Rao. Improved pel–recursive motion compensation. *IEEE transactions on communications*, 32:1128–1134, October 1984.

[99] J. Woods. Two-dimensional discrete markovian fields. *IEEE Transactions on Information Theory*, pages 232–240, March 1972.

[100] J. Woods. Markov image modelling. *IEEE Transactions on Automatic Control*, pages 846–850, October 1978.

[101] Olli Yli-Harja. *Median Filters: Extensions, Analysis and Design.* PhD thesis, Lappeenranta University of Technology, 1989.

[102] R. Young and N. Kingsbury. Video compression using lapped transforms for motion estimation/compensation and coding. In *SPIE VCIP*, pages 276–288, 1992.

[103] A. Zaccarin and B. Liu. Fast algorithms for block motion estimation. In *IEEE ICASSP*, volume 3, pages 449–452, 1992.

[104] R. Storey. Electronic Detection and concealment of film dirt. UK patent spec no. 2139039, 1984.

[105] R. Storey: Electronic detection and concealment of film dirt. SMPTE Journal, June 1985, pp 642-647.

[106] Thomas, G A: TV picture motion measurement. UK patent appl. GB 2188510A, 1987.

[107] Thomas, G A: Distorting the time axis: motion compensated image processing in the studio. IBC'88 (Brighton UK), 24-27, Sept. 1988, IEE Conference Pub. no 293, pp256-259.

[108] Dabner S C: Real-time motion vector measurement hardware. 3rd Intl. Workshop on HDTV (Turin, Italy) 30th Aug - 1 Sept 1989.

[109] Thomas G A, Lau H: Generation of high quality slow-motion replay using motion compensation. IBC'90 (Brighton, U.K), 22-24 Sept 1990, IEE Conference Pub. no. 327, pp 121-125.

ESTIMATING THE AR COEFFICIENTS FOR THE 3DAR MODEL

This appendix presents the least squared solution (the Maximum Likelihood solution may be found in [82]) for the coefficients of the three dimensional autoregressive model as outlined in Chapter 2. The model is best discussed in its prediction mode. The prediction equation is as below where $\hat{I}(i,j,n)$ is the predicted value of the pixel at (i,j,n).

$$\hat{I}(i,j,n) = \sum_{k=1}^{N} a_k I(i + q_k(x) + sx_{n,n+q_k(n)}, j + q_k(y) + sy_{n,n+q_k(n)}, n + q_k(n)) \tag{A.1}$$

The task then becomes to choose the parameters in order to minimize some function of the error, or residual,

$$\epsilon(i,j,n) = I(i,j,n) - \hat{I}(i,j,n) \tag{A.2}$$

The parameters of the model are both the AR coefficients $\mathbf{a} = [a_1, a_2, a_3..a_N]$, and the displacement $\mathbf{d}_{k,l} = [sx_{k,l}\ sy_{k,l}\ 0]$. This section is concerned only with coefficient estimation given an estimate for the displacement.

The coefficients are chosen to minimize the squared error, $\epsilon()$, above. This leads to the Normal equations [9, 18, 42]. The derivation is the same as the one dimensional case and the solution can be determined by invoking the principle of orthogonality. $E[\epsilon^2(i,j,n)]$ is minimized by making the error $\epsilon(i,j,n)$ orthogonal to the signal values used in its generation [42]. Therefore,

$$E[\epsilon(i,j,n) I(i + q_m(x) + sx_{n,n+q_m(n)}, j + q_m(y) + sy_{n,n+q_m(n)}, n + q_m(n))] = 0 \tag{A.3}$$

$$\text{for } m = 1..N \tag{A.4}$$

Defining $\mathbf{q_0} = [\mathbf{0}, \mathbf{0}, \mathbf{0}]$ and $a_0 = 1.0$, then

$$\epsilon(i,j,n) = \sum_{k=0}^{N} a_k I(i + q_k(x) + sx_{n,n+q_k(n)}, j + q_k(y) + sy_{n,n+q_k(n)}, n + q_k(n)) \tag{A.5}$$

Note that the a_k are now reversed in sign to allow for this simpler formulation. To continue, the following notation is introduced.

$$\mathbf{x} = [i\ j\ n] \tag{A.6}$$

$$\mathbf{q}_k = [q_k(x)\ q_k(y)\ q_k(n)] \tag{A.7}$$

$$\mathbf{d}_{\mathbf{x},\mathbf{x}+\mathbf{q}_k} = [sx_{n,n+q_k(n)}\ sy_{n,n+q_k(n)}\ 0] \tag{A.8}$$

Substituting for $\epsilon()$ in equation A.3 gives,

$$\sum_{k=0}^{N} a_k E[I(\mathbf{x}+\mathbf{q}_k+\mathbf{d}_{\mathbf{x},\mathbf{x}+\mathbf{q}_k})I(\mathbf{x}+\mathbf{q}_m+\mathbf{d}_{\mathbf{x},\mathbf{x}+\mathbf{q}_m})] = 0 \qquad (A.9)$$

$$\forall m = 1..N$$

The expectation can be recognised as a term from the autocorrelation function of the 3-D signal $I(\mathbf{x})$. Matters may be simplified therefore by redefining the equation as

$$\sum_{k=0}^{N} a_k C(\mathbf{q}'_k, \mathbf{q}'_m) = 0 \qquad (A.10)$$

Where $\mathbf{q}'_k, \mathbf{q}'_m$ are both motion compensated vector offsets as defined implicitly in the previous equation. However, a_0 has already been defined to be 1.0. Therefore, letting

$$\mathbf{a} = [a_1\ a_2\ ..\ a_N]^T \qquad (A.11)$$

$$\mathbf{C} = \begin{bmatrix} C(\mathbf{q}'_1,\mathbf{q}'_1) & C(\mathbf{q}'_1,\mathbf{q}'_2) & ... & C(\mathbf{q}'_1,\mathbf{q}'_N) \\ C(\mathbf{q}'_2,\mathbf{q}'_1) & C(\mathbf{q}'_2,\mathbf{q}'_2) & ... & C(\mathbf{q}'_2,\mathbf{q}'_N) \\ C(\mathbf{q}'_3,\mathbf{q}_1{}') & C(\mathbf{q}'_3,\mathbf{q}'_2) & ... & C(\mathbf{q}'_3,\mathbf{q}'_N) \\ \vdots & \vdots & \vdots & \vdots \\ C(\mathbf{q}'_N,\mathbf{q}'_1) & C(\mathbf{q}'_N,\mathbf{q}'_2) & ... & C(\mathbf{q}'_N,\mathbf{q}'_N) \end{bmatrix} \qquad (A.12)$$

$$\mathbf{c} = [C(\mathbf{q}_0,\mathbf{q}'_1)\ C(\mathbf{q}_0,\mathbf{q}'_2)...C(\mathbf{q}_0,\mathbf{q}'_N)] \qquad (A.13)$$

$$\mathbf{q}_0 = [0\ 0\ 0] \qquad (A.14)$$

the parameters, $\mathbf{a}$ can be determined by solving

$$\mathbf{Ca} = -\mathbf{c} \qquad (A.15)$$

It must be pointed out that although $\mathbf{C}$ is symmetric, it is not Toeplitz in the multidimensional case. This is due to the fact that along a diagonal, the differences between the offset vectors that define each correlation term are not necessarily parallel or the same magnitude. Consider the diagonal of matrix $\mathbf{C}$, consisting of terms at locations $[2,1][3,2][4,3]\ldots[N,N-1]$, where the top left element of $\mathbf{C}$ is at position $[1,1]$. Then vector $\mathbf{v}_1 = [q'_2 - q'_1]$ is not necessarily equal to $\mathbf{v}_2 = [q'_3 - q'_2]$ or $\mathbf{v}_3 = [q'_4 - q'_3]$ or any other such difference vector along the diagonal. The support vectors $\mathbf{q}$ may be chosen to allow this to occur by choosing vectors that lie along a line in the support volume. In general, however, when the support set delineates some volume, the vectors do not allow $\mathbf{C}$ to be Toeplitz. Therefore, it is difficult to exploit the structure of this matrix for computational purposes.

In the thesis, the equation A.15 is solved exactly. That is to say that no approximations about the autocorrelation function are made in estimating $\mathbf{C}$ or $\mathbf{c}$. The expectation operator in equation A.10 is taken to be the mean operation. Note that in order to calculate the required autocorrelation terms from a block of data of size $N \times N$ in the current frame n say, the offset

vectors $\mathbf{q}$ require that data outside this block is necessary. The extent of this extra data is explained next.

Figure 2.5 shows a support set of 5 vectors. Calculation of $C(\mathbf{q}_0, \mathbf{q}_2)$, say, requires the following sum of products, where $\mathbf{q}_2 = [-1, 0, -1]$.

$$\sum_{\mathbf{x} \in \mathrm{B1}} I(\mathbf{x} + \mathbf{q}_0) I(\mathbf{x} + \mathbf{q}_2) \tag{A.16}$$

Block B1 is of size $N \times N$ as stated before, and this yields data for $I(\mathbf{x} + \mathbf{q}_0)$. The term $I(\mathbf{x} + \mathbf{q}_2)$ requires data from a block, B2, which is in the previous frame and the same size, but offset by $\mathbf{q}_2$ in that frame. In this case therefore, to solve for the AR coefficients exactly in blocks of size $N \times N$ involves data from a block of size $(N + 2) \times (N + 2)$ in the previous frame centred at the same position.

THE RESIDUAL FROM A NON–CAUSAL AR MODEL IS NOT WHITE.

This section investigates the nature of the residual sequence from an AR model given a least squared estimate for the coefficients of the model. The analysis shows that unlike the causal AR model, the error or residual sequence of a non–causal model is not white but coloured (See [42, 82, 100, 99]). The model is considered in its 3D form as introduced in Chapter 2.

The model equation is as follows (see Chapter 2).

$$I(\mathbf{x}) = \sum_{k=1}^{N} a_k I(\mathbf{x} + \mathbf{q}_k) + \epsilon(\mathbf{x}) \tag{B.1}$$

This form of the model does not allow for any motion of objects between frames. Incorporation of this movement makes the expressions more cumbersome but does not affect the result. Typical support sets of $N = 9$ and $N = 1$ vectors defined by different $\mathbf{q}_k$ are shown in figure 3.8.

In solving for the coefficients using the least squared approach (see Appendix A), the error, $\epsilon(\mathbf{x})$ is made orthogonal to the data at the locations pointed to by the support vectors, $\mathbf{q}_k$. This implies that

$$E[\epsilon(\mathbf{x})I(\mathbf{x}+\mathbf{q}_n)] = 0 \quad \text{for } n = 1 \ldots N \tag{B.2}$$

The goal of this analysis is to find an expression for the correlation function of $\epsilon(\mathbf{x})$. That is

$$R_{\epsilon\epsilon}(\mathbf{x}, \mathbf{q}_n) = E[\epsilon(\mathbf{x})\epsilon(\mathbf{x}+\mathbf{q}_n)] \tag{B.3}$$

Multiplying equation B.1 by $\epsilon(\mathbf{x}+\mathbf{q}_n)$ and taking expectations gives

$$E[I(\mathbf{x})\epsilon(\mathbf{x}+\mathbf{q}_n)] = \sum_{k=1}^{N} a_k E[I(\mathbf{x}+\mathbf{q}_k)\epsilon(\mathbf{x}+\mathbf{q}_n)] + E[\epsilon(\mathbf{x})\epsilon(\mathbf{x}+\mathbf{q}_n)] \tag{B.4}$$

Let the variance of $\epsilon(\mathbf{x})$ be $\sigma^2_{\epsilon\epsilon}$. Then from B.4, when $\mathbf{q}_n = [0\ 0\ 0]$,

$$E[I(\mathbf{x})\epsilon(\mathbf{x})] = \sigma^2_{\epsilon\epsilon} \tag{B.5}$$

The summation term disappears because of equation B.2, since $\mathbf{x} \neq (\mathbf{x}+\mathbf{q}_k)$. When the $\mathbf{q}_n$ refer to other positions within the support of the model then the following simplifications may be made

$$E[I(\mathbf{x})\epsilon(\mathbf{x}+\mathbf{q}_n)] = 0 \text{ by B.2} \tag{B.6}$$

$$\sum_{k=1}^{N} a_k E[I(\mathbf{x}+\mathbf{q}_k)\epsilon(\mathbf{x}+\mathbf{q}_n)] = a_n \sigma^2_{\epsilon\epsilon} \text{ by B.2 and B.5} \tag{B.7}$$

These simplifications can be substituted into B.4 to give the correlation term for non–zero vector lags. From this substitution it can be seen that the correlation structure of $\epsilon(\mathbf{x})$ is not white and it depends on the model coefficients. The final result then is

$$R_{\epsilon\epsilon}(\mathbf{x}, \mathbf{q}_n) = \begin{cases} \sigma^2_{\epsilon\epsilon} & \text{for } \mathbf{q}_n = [0\ 0\ 0] \\ a_n \sigma^2_{\epsilon\epsilon} & \text{for } n = 1 \ldots N \end{cases} \tag{B.8}$$

ESTIMATING THE DISPLACEMENT PARAMETER IN THE 3D AR MODEL.

The three dimensional autoregressive model incorporating motion is defined as below (from Chapter 2).

$$I(i,j,n) = \sum_{k=1}^{N} a_k I(i + q_k(x) + sx_{n,n+q_k(n)}, j + q_k(y) + sy_{n,n+q_x(n)}, n + q_k(n)) + \epsilon(i,j,n) \quad \text{(C.1)}$$

The parameters of the model are both the AR coefficients $\mathbf{a} = [a_1, a_2, a_3..a_N]$, and the displacement $\mathbf{d}_{k,l} = [sx_{k,l}\ sy_{k,l}\ 0]$. This section is concerned only with displacement estimation given an estimate for the coefficients.

In order to gain an explicit relation for $\epsilon()$, in terms of $\mathbf{d}$, the approach used by Biemond [10] and Efstratiadis [20], was to expand the image function, $I()$, in the previous frames, as a Taylor series about the current displacement guess. This effectively linearizes the equation for $\epsilon()$ and allows a closed form estimate for $\mathbf{d}$. It is this solution that is used for estimating $\mathbf{d}$ in this work. The derivation given here is for a general non–causal model which involves support in both the past and future frames as well as the current frame.

It is necessary first of all to separate the support region for the AR model into three parts.

1. The support in the frames previous to the current one, i.e. $\mathbf{q}_k(n) < 0$. This is the temporally causal support.
2. The support in the current frame, $\mathbf{q}_k(n) = 0$
3. The support in the frames to come, $\mathbf{q}_k(n) > 0$. The temporally anti-causal support

Further, given the displacement $\mathbf{d}_{l,l+1}$ from frame l to frame $l+1$ and $\mathbf{d}_{l,l-1}$ defined similarly, the displacement $d_{l,l+k}$ is defined as the linear sum of the displacements form frame l through to frame $l+k$. That is

$$\mathbf{d}_{l,l+k} = \mathbf{d}_{l,l+1} + \sum_{m=l+1}^{k-1} \mathbf{d}_{m,m+1} \quad \text{(C.2)}$$

(Similarly for $\mathbf{d}_{l,l-k}$).

The notation for the modelling equations is now improved slightly to allow a more condensed derivation of the estimation equations.

- $\mathbf{q}_k(f)$ is the spatial support vector in the (n+f)th frame

- n is the current frame
- $\mathbf{x}$ is a **spatial** position vector
- $I(\mathbf{x}, n)$ is the grey level at the position $\mathbf{x}$ in the nth frame
- $N(f)$ is the number of points in the support of the 3D AR model in frame $n + f$.
- N_0 is the number of points in the support of the 3D AR model in the current frame.
- F^- is the maximum frame offset in the causal support (a negative number for the number of causal frames)
- F^+ is the maximum frame offset in the anti causal support (a positive number for the number of anti causal frames)
- a^- the coefficients for the temporally causal support
- a the coefficients for the support in the current frame
- a^+ the coefficients for the temporally anti-causal support

The modelling equation can now be broken up into

$$\begin{aligned} I(\mathbf{x}, n) &= \sum_{k=0}^{N_0} a_k I(\mathbf{x} + \mathbf{q}_k, n) \\ &+ \sum_{f=-1}^{F^-} \sum_{k=1}^{N(f)} a_k^- I(\mathbf{x} + \mathbf{q}_k(f) + \mathbf{d}_{n,n+f}, n + f) \\ &+ \sum_{f=1}^{F^+} \sum_{k=1}^{N(f)} a_k^+ I(\mathbf{x} + \mathbf{q}_k(f) + \mathbf{d}_{n,n+f}, n + f) \\ &+ \epsilon(\mathbf{x}, n) \end{aligned} \tag{C.3}$$

If the various support is then expressed in terms of the displacement into the next and previous frames, the following equation results after using C.2.

$$\begin{aligned} I(\mathbf{x}, n) &= \sum_{k=0}^{N_0} a_k I(\mathbf{x} + \mathbf{q}_k, n) \\ &+ \sum_{f=-1}^{F^-} \sum_{k=1}^{N(f)} a_k^- I(\mathbf{x} + \mathbf{q}_k(f) + \mathbf{d}_{n,n-1} + \sum_{m=n-1}^{n+f+1} \mathbf{d}_{m,m-1}, n + f) \\ &+ \sum_{f=1}^{F^+} \sum_{k=1}^{N(f)} a_k^+ I(\mathbf{x} + \mathbf{q}_k(f) + \mathbf{d}_{n,n+1} + \sum_{m=n+1}^{n+f-1} \mathbf{d}_{m,m+1}, n + f) \\ &+ \epsilon(\mathbf{x}, n) \end{aligned} \tag{C.4}$$

It is assumed that there exist already estimates for $\mathbf{d}_{n,n+1}$ and $\mathbf{d}_{n,n-1}$. What is required is therefore an update for each value. Let the current estimates be $\mathbf{d}^0_{n,n+1}$ and $\mathbf{d}^0_{n,n-1}$. Further, let the updates required be such that

$$\mathbf{d}_{n,n+1} = \mathbf{d}^0_{n,n+1} + \mathbf{u}_{n,n+1} \tag{C.5}$$

$$\mathbf{d}_{n,n-1} = \mathbf{d}^0_{n,n-1} + \mathbf{u}_{n,n-1} \tag{C.6}$$

Where $\mathbf{u}$ represents the update to be found.

Equation C.4 can now be written as

$$\begin{aligned} I(\mathbf{x},n) &= \sum_{k=0}^{N_0} a_k I(\mathbf{x}+\mathbf{q}_k, n) \\ &+ \sum_{f=-1}^{F^-}\sum_{k=1}^{N(f)} a_k^- I(\mathbf{x}+\mathbf{q}_k(f)+\mathbf{d}^0_{n,n-1}+\mathbf{u}_{n,n-1}+\sum_{m=n-1}^{n+f+1}\mathbf{d}_{m,m-1}, n+f) \\ &+ \sum_{f=1}^{F^+}\sum_{k=1}^{N(f)} a_k^+ I(\mathbf{x}+\mathbf{q}_k(f)+\mathbf{d}^0_{n,n+1}+\mathbf{u}_{n,n+1}+\sum_{m=n+1}^{n+f-1}\mathbf{d}_{m,m+1}, n+f) \\ &+ \epsilon(\mathbf{x},n) \end{aligned} \tag{C.7}$$

The function for $I()$ given in equation C.7 can then be linearised using a Taylor expansion[1] about $(\mathbf{x}+\mathbf{q}_k(f)+\mathbf{d}^0_{n,n+1}+\sum\mathbf{d}_{m,m+1}, n+f)$, which represents the current displacement in both previous and next frames[2]. The form of the next expression is unwieldy unless the following definition is made.

$$\mathbf{D}(f,k,n) = \mathbf{x}+\mathbf{q}_k(f)+\mathbf{d}^0_{n,n+1}+\sum\mathbf{d}_{m,m+1} \tag{C.8}$$

The Taylor series expansion then yields the following expression.

$$\begin{aligned} I(\mathbf{x},n) &= \sum_{k=0}^{N_0} a_k I(\mathbf{x}+\mathbf{q}_k, n) \\ &+ \sum_{f=-1}^{F^-}\sum_{k=1}^{N(f)} a_k^- I(\mathbf{D}(f,k,n), n+f) \\ &+ \mathbf{u}^T_{n,n-1}\sum_{f=-1}^{F^-}\sum_{k=1}^{N(f)} a_k^- \nabla I(\mathbf{D}(f,k,n), n+f) \\ &+ \sum_{f=-1}^{F^-}\sum_{k=1}^{N(f)} a_k^- \nu(\mathbf{D}(f,k,n), n+f) \\ &+ \sum_{f=1}^{F^+}\sum_{k=1}^{N(f)} a_k^+ I(\mathbf{D}(f,k,n), n+f) \end{aligned}$$

[1] Note that this is not the only expansion that can be employed, an alternative is to use a Bilinear interpolation function. However, the first order Taylor expansion gives a simpler solution.

[2] The limits on the summation of displacement vectors are intentionally left out to allow the same expression to be used for the forward and backward displacement depending on the equation context.

$$
\begin{aligned}
&+ \sum_{f=1}^{F^+} \sum_{k=1}^{N(f)} a_k^+ \nu(\mathbf{D}(f,k,n), n+f) \\
&+ \mathbf{u}_{n,n+1}^T \sum_{f=1}^{F^+} \sum_{k=1}^{N(f)} a_k^+ \nabla I(\mathbf{D}(f,k,n), n+f) \\
&+ \epsilon(\mathbf{x}, n)
\end{aligned} \tag{C.9}
$$

$\nu()$ represents the higher order terms in the Taylor series expansions.

For the current set of estimated parameters, $\mathbf{a}$ and $\mathbf{d}_0$ there will be some observed error ϵ_0. This error is defined as (following A.1),

$$
\begin{aligned}
\epsilon_0(\mathbf{x}, n) &= I(\mathbf{x}, n) - \sum_{k=0}^{N_0} a_k I(\mathbf{x} + \mathbf{q}_k, n) \\
&- \sum_{f=-1}^{F^-} \sum_{k=1}^{N(f)} a_k^- I(\mathbf{D}(f,k,n), n+f) \\
&- \sum_{f=1}^{F^+} \sum_{k=1}^{N(f)} a_k^+ I(\mathbf{D}(f,k,n), n+f)
\end{aligned} \tag{C.10}
$$

Therefore substituting C.9, an expression involving $\mathbf{u}$ in terms of observables follows (where the limits on the sums have been dropped).

$$
\begin{aligned}
\epsilon_0(\mathbf{x}, n) &= \mathbf{u}_{n,n-1}^T \sum\sum a_k^- \nabla I(\mathbf{D}(f,k,n), n+f) \\
&+ \sum\sum a_k^- \nu(\mathbf{D}(f,k,n), n+f) \\
&+ \mathbf{u}_{n,n+1}^T \sum\sum a_k^+ \nabla I(\mathbf{D}(f,k,n), n+f) \\
&+ \sum\sum a_k^+ \nu(\mathbf{D}(f,k,n), n+f) \\
&+ \epsilon(\mathbf{x}, n)
\end{aligned} \tag{C.11}
$$

The spatial, two component, update vectors, $\mathbf{u}$ are now required, but there is only one equation. Collecting observations of $\epsilon_0(\mathbf{x}, n), \nabla I()$ at each position in some predefined region, an overdetermined system of equations results. These equations can be written as follows.

$$
\mathbf{z}_w = \mathbf{G}_w \mathbf{u} + \mathbf{v}_w \tag{C.12}
$$

The quantities are defined as follows, given a set of equations made by observing a block of $N_1 \times N_2$ pixels.

- $\mathbf{z}_w$, $(N_1 \times N_2 \times 1)$ is a column vector of current errors at all the points in the region used

for estimation.

$$\mathbf{z}_w = \begin{bmatrix} \epsilon_0(\mathbf{x}_1, n) \\ \epsilon_0(\mathbf{x}_2, n) \\ \vdots \\ \epsilon_0(\mathbf{x}_{(N_1 N_2)}, n) \end{bmatrix} \tag{C.13}$$

- $\mathbf{G}_w$, $(N_1 \times N_2 \times 4)$ is a matrix of gradients at the past and future support positions.

$$\mathbf{G}_w = \begin{bmatrix} \sum\sum a_k^- \frac{\partial I_1(\mathbf{D}(f,k,n))}{\partial x} & \sum\sum a_k^- \frac{\partial I_1(\mathbf{D}(f,k,n))}{\partial y} \\ \sum\sum a_k^- \frac{\partial I_2(\mathbf{D}(f,k,n))}{\partial x} & \sum\sum a_k^- \frac{\partial I_2(\mathbf{D}(f,k,n))}{\partial y} \\ \vdots & \vdots \\ \sum\sum a_k^- \frac{\partial I_{(N_1 N_2)}(\mathbf{D}(f,k,n))}{\partial x} & \sum\sum a_k^- \frac{\partial I_{(N_1 N_2)}(\mathbf{D}(f,k,n))}{\partial y} \\ \sum\sum a_k^+ \frac{\partial I_1(\mathbf{D}(f,k,n))}{\partial x} & \sum\sum a_k^+ \frac{\partial I_1(\mathbf{D}(f,k,n))}{\partial y} \\ \sum\sum a_k^+ \frac{\partial I_2(\mathbf{D}(f,k,n))}{\partial x} & \sum\sum a_k^+ \frac{\partial I_2(\mathbf{D}(f,k,n))}{\partial y} \\ \vdots & \vdots \\ \sum\sum a_k^+ \frac{\partial I_{(N_1 N_2)}(\mathbf{D}(f,k,n))}{\partial x} & \sum\sum a_k^+ \frac{\partial I_{(N_1 N_2)}(\mathbf{D}(f,k,n))}{\partial y} \end{bmatrix} \tag{C.14}$$

- $\mathbf{u}$ is the (4×1) vector of updates defined as

$$\mathbf{u} = \begin{bmatrix} \mathbf{u}_{n,n-1} \\ \mathbf{u}_{n,n+1} \end{bmatrix} \tag{C.15}$$

- $\mathbf{v}_w$ is the collection of all the error terms ϵ and ν.

$$\mathbf{v}_w = \begin{bmatrix} \sum\sum a_k^- \nu_1(\ldots) + \sum\sum a_k^+ \nu_1(\ldots) + \epsilon(\mathbf{x}_1, n) \\ \sum\sum a_k^- \nu_2(\ldots) + \sum\sum a_k^+ \nu_2(\ldots) + \epsilon(\mathbf{x}_2, n) \\ \vdots \\ \sum\sum a_k^- \nu_{(N_1 N_2)}(\ldots) + \sum\sum a_k^+ \nu_{(N_1 N_2)}(\ldots) + \epsilon(\mathbf{x}_{(N_1 N_2)}, n) \end{bmatrix} \tag{C.16}$$

So far the derivation for the parameter estimates has placed no restriction on the spatial or temporal nature of the model support. However, the work in the thesis is concerned with causal modelling primarily due to the decreased computation necessary.

Solving for the updates

It is possible to estimate the displacement update vector in C.12 directly via the pseudo inverse of $\mathbf{G}_w$ as follows.

$$\begin{aligned} \mathbf{G}_w^T \mathbf{z}_w &= \mathbf{G}_w^T \mathbf{G}_w \mathbf{u} + \mathbf{G}_w^T \mathbf{v}_w \\ \mathbf{u} &= [\mathbf{G}_w^T \mathbf{G}_w]^{-1} [\mathbf{G}_w^T \mathbf{z}_w - \mathbf{G}_w^T \mathbf{v}_w] \end{aligned} \tag{C.17}$$

To arrive at a more robust solution, the approach adopted by [20] has been to derive a Wiener estimate for $\mathbf{u}$ [20, 10]. The method was initially presented by Biemond, and it attempts to find the estimate $\hat{\mathbf{u}}$ for $\mathbf{u}$ which minimizes the error $E[|\mathbf{u}-\hat{\mathbf{u}}|^2]$. Therefore,

$$\begin{aligned} E[|\mathbf{u}-\hat{\mathbf{u}}|^2] &= E[(\mathbf{u}^T-\hat{\mathbf{u}}^T)(\mathbf{u}-\hat{\mathbf{u}})] \\ &= E[\mathbf{u}^T\mathbf{u}-\mathbf{u}^T\hat{\mathbf{u}}-\hat{\mathbf{u}}^T\mathbf{u}+\hat{\mathbf{u}}^T\hat{\mathbf{u}}] \end{aligned} \tag{C.18}$$

The estimate, $\hat{\mathbf{u}}$, is found from a linear transformation of the observed error vector $\mathbf{z}$ such that

$$\hat{\mathbf{u}} = \mathbf{L}\mathbf{z}_w \tag{C.19}$$

Substituting this expression for $\hat{\mathbf{u}}$ in C.18 and differentiating with respect to the required unknown, $\mathbf{L}$, to find the minimum squared error, yields the following equation.

$$\mathbf{L}E[(\mathbf{G}_w\mathbf{u}+\mathbf{v})(\mathbf{G}_w\mathbf{u}+\mathbf{v}_w)^T] = E[\mathbf{u}(\mathbf{G}_w\mathbf{u}+\mathbf{v}_w)^T] \tag{C.20}$$

Therefore, assuming that, $\mathbf{v}_w$, which involves higher order terms, is uncorrelated with the actual update, $\mathbf{u}$, an explicit expression for $\mathbf{L}$ results.

$$\mathbf{L} = \mathbf{R}_{uu}\mathbf{G}_w^T[\mathbf{G}_w\mathbf{R}_{uu}\mathbf{G}_w^T+\mathbf{R}_{vv}]^{-1} \tag{C.21}$$

This solution for $\mathbf{L}$ involves the inverse of a large matrix. If the number of positions at which observations are taken is P, then it involves the inverse of a $P^2 \times P^2$ matrix. Biemond [10] has employed a matrix identity which simplifies this solution considerably.

$$\mathbf{R}_{uu}\mathbf{G}_w^T[\mathbf{G}_w\mathbf{R}_{uu}\mathbf{G}_w^T+\mathbf{R}_{vv}]^{-1} = [\mathbf{G}_w^T\mathbf{R}_{vv}^{-1}\mathbf{G}_w+\mathbf{R}_{uu}^{-1}]^{-1}\mathbf{G}_w^T\mathbf{R}_{vv}^{-1} \tag{C.22}$$

Using C.22, therefore,

$$\mathbf{L} = [\mathbf{G}_w^T\mathbf{R}_{vv}^{-1}\mathbf{G}_w+\mathbf{R}_{uu}^{-1}]^{-1}\mathbf{G}_w^T\mathbf{R}_{vv}^{-1} \tag{C.23}$$

Assuming that the vector $\mathbf{v}$ represents white noise and that the components of $\mathbf{u}$ are uncorrelated, ie $\mathbf{R}_{vv} = \sigma_{vv}^2\mathbf{I}$ and $\mathbf{R}_{uu} = \sigma_{uu}^2\mathbf{I}$, $\mathbf{L}$ is given by

$$\mathbf{L} = [\mathbf{G}_w^T\mathbf{G}_w+\mu\mathbf{I}]^{-1}\mathbf{G}_w^T \tag{C.24}$$

where $\mu = \frac{\sigma_{vv}^2}{\sigma_{uu}^2}$. Due to the identity in C.22 and this assumption, the matrix inverse is reduced to the inverse of a 2×2 matrix , regardless of the number of equations.

It is important to recognize that the validity of the assumption regarding $\mathbf{R}_{vv}$ is affected by the causality of the model support. This is because part of $\mathbf{v}$ consists of the model error $\epsilon()$. It has been shown in [55, 99, 100, 42], that this error is not white when the model support is non–causal. This implies that if the support for the model consists of points in the current frame that represent a non–causal region in that frame, the assumption is not valid. To ensure the validity of the white noise assumption, the support for the AR model in the current frame must be limited to a causal region, i.e. to the left and above the predicted location.

The Wiener estimate for $\mathbf{u}$ is therefore given by

$$\hat{\mathbf{u}} = [\mathbf{G}_w^T \mathbf{G}_w + \mu \mathbf{I}]^{-1} \mathbf{G}_w^T \mathbf{z}_w \tag{C.25}$$

This solution for the update for the current displacement is incorporated into an iterative refinement scheme. A guess for the displacement, which may be zero, is iteratively refined using the above equation until some convergence criterion is satisfied. Two main criterion are used in this thesis, a threshold on the magnitude of the error vector, $|\mathbf{z}_w|_t$ and a threshold on the size of the update vector, $|\mathbf{u}|_t$. The iterative refinement process is halted if the magnitude of the current error is less than $|\mathbf{z}_w|_t$ or the magnitude of the update is less than $|\mathbf{u}|_t$. A final criterion is the most harsh, and simply halts iteration if no other criterion has been fulfilled when a certain number of iterations have completed. These criterion are necessary to limit the computational load of the algorithm.

C.1 Summary

The approach taken by Biemond [10] can be used to generate a solution for the motion update in the case of the general 3DAR model. The solution is linear only if the model coefficients are known beforehand. These are not available in practice but it is possible to estimate the coefficients and displacement successively in the iterative process. The motion equations reduce to approximately the same form as the standard WBME.

It is important to recognize that the Taylor series expansion is not the only expansion which can be used to linearize the model equation. The purpose of the expansion is to make the displacement parameter available explicitly. To this end any interpolator would suffice. The compromise is one of interpolator quality versus computation. Sinc interpolation is a possibility but it would yield a non–linear solution. Bilinear interpolation is also an alternative which may prove better than the Taylor expansion. This thesis uses the Taylor expansion to facilitate a simple linear solution but other interpolators would be useful to consider in further work.

EXAMINING ILL–CONDITIONING IN G^TG

Several workers[1] all recognised that when gradient based algorithms failed they did so primarily because of the ill conditioning of the Gradient matrix that was set up. This ill–conditioning can be quantified through the eigen values of the matrix. The analysis presented here considers this phenomenon and gives some basis for the final algorithm presented by Martinez [63] and reviewed in Chapter 2.

D.1 Condition for singularity

In the standard WBME the solution for the update vector is given as

$$\hat{\mathbf{u}} = [\mathbf{G}^T\mathbf{G} + \mu]^{-1}\mathbf{G}^T\mathbf{z} \tag{D.1}$$

(see Chapter 2). The gradient matrix $\mathbf{G}$ contains elements as follows.

$$\mathbf{G} = \begin{bmatrix} \frac{\partial I_{n-1}(\mathbf{x}_1)}{\partial x} & \frac{\partial I_{n-1}(\mathbf{x}_1)}{\partial y} \\ \frac{\partial I_{n-1}(\mathbf{x}_2)}{\partial x} & \frac{\partial I_{n-1}(\mathbf{x}_2)}{\partial y} \\ \vdots & \vdots \\ \frac{\partial I_{n-1}(\mathbf{x}_{N_1 \times N_2})}{\partial x} & \frac{\partial I_{n-1}(\mathbf{x}_{N_1 \times N_2})}{\partial y} \end{bmatrix} \tag{D.2}$$

Figure 2.3 illustrates the situation when solving for the motion update using this technique. Note how this solution follows from using a one point temporal AR model as described in Appendix C, when the single coefficient used has a value of unity.

To facilitate further analysis, $\mathbf{G}$ can be split into two vector components, $\mathbf{G} = [\mathbf{g}_x, \mathbf{g}_y]$. The matrix of importance, $\mathbf{G}^T\mathbf{G}$ can then be written

$$\mathbf{G}^T\mathbf{G} = \begin{bmatrix} \mathbf{g}_x^T\mathbf{g}_x & \mathbf{g}_x^T\mathbf{g}_y \\ \mathbf{g}_y^T\mathbf{g}_x & \mathbf{g}_y^T\mathbf{g}_y \end{bmatrix} \tag{D.3}$$

This matrix is singular iff

$$(\mathbf{g}_x^T\mathbf{g}_x)(\mathbf{g}_y^T\mathbf{g}_y) = (\mathbf{g}_y^T\mathbf{g}_x)(\mathbf{g}_x^T\mathbf{g}_y) \tag{D.4}$$

The form of this equation is $(\mathbf{a}^T\mathbf{a})(\mathbf{b}^T\mathbf{b}) = (\mathbf{a}^T\mathbf{b})^2$. According to the Schwartz inequality, this is satisfied iff $\mathbf{a} = \alpha\mathbf{b}$, where α is some scalar constant. The equality therefore implies that $\mathbf{g}_x = \alpha\mathbf{g}_y$. This situation,

$$\frac{\partial I_{n-1}(\mathbf{x}_n)}{\partial x} = \alpha\frac{\partial I_{n-1}(\mathbf{x}_n)}{\partial y} \tag{D.5}$$

occurs when all the points used to set up the gradient matrix lie along an edge.

[1] For instance, Martinez [63], Kearney et al. [46] and Böröczky et al. [12, 53, 38, 37].

D.2 Relating ill–conditioning to the spatial contrast

Consider the problem of finding the directions of maximum and minimum average contrast. First of all, define a directional gradient at position $\mathbf{x}$ in a frame of the sequence with the following expression, $\mathbf{v}^T\nabla I(\mathbf{x})$, where $I(\mathbf{x})$ represents the grey level, and the direction is along the vector $\mathbf{v}$. The average magnitude of the spatial gradient (contrast) along $\mathbf{v}$ can then be defined as

$$\bar{C}(\mathbf{x}) = \frac{1}{N}\sum_{i=1}^{N}[\mathbf{v}^T\nabla I(\mathbf{x})]^2 \tag{D.6}$$

Minimizing and maximizing $C(\mathbf{x})$ with respect to $\mathbf{v}$ will yield the direction of minimum and maximum contrast. Of course, to yield a direction only, an additional constraint is needed to ensure the resulting vector is a unit vector. The constraint is

$$\mathbf{v}^T\mathbf{v} = 1 \tag{D.7}$$

Note that the definition of directional gradient introduced, which is intuitively reasonable, now allows the sum[2] magnitude of contrast to be expressed in terms of $\mathbf{G}$. It becomes easier to see how $\mathbf{G}^T\mathbf{G}$ is related to a measure of directional contrast.

$$\begin{aligned} C(\mathbf{x}) &= \mathbf{v}^T\mathbf{G}^T\mathbf{G}\mathbf{v} \\ &= \mathbf{v}^T\mathbf{W}\mathbf{v} \end{aligned} \tag{D.8}$$

Another simplification is introduced, letting $\mathbf{W} = \mathbf{G}^T\mathbf{G}$.

Combining equation D.7 and D.9 and a Lagrange multiplier λ, gives the following expression to optimize.

$$O(\lambda, \mathbf{v}) = \mathbf{v}^T[\mathbf{W} - \lambda\mathbf{I}]\mathbf{v} + \lambda \tag{D.9}$$

Differentiating D.9 with respect to $\mathbf{v}$ and setting the result to zero, yields D.10.

$$[\mathbf{W} - \lambda\mathbf{I}]\mathbf{v} + [\mathbf{W} - \lambda\mathbf{I}]^T\mathbf{v} = 0 \tag{D.10}$$

But $\mathbf{W}$ is symmetric, therefore,

$$[\mathbf{W} - \lambda\mathbf{I}]\mathbf{v} = 0 \tag{D.11}$$

This is an eigenvalue problem. Therefore, the directions of maximum and minimum contrast lie along the eigen vectors of $\mathbf{W}$. Similarly, the magnitudes of maximum and minimum contrast are given by eigenvalues of $\mathbf{W}$. The larger eigenvalue and corresponding vector refer to the magnitude and direction of maximum contrast.

When $\mathbf{W}$ is ill–conditioned, $\lambda_{\max} \gg \lambda_{\min}$. Hence $\mathbf{G}^T\mathbf{G}$ is ill–conditioned when the directions of averaged spatial gradient are not of equal importance; at an edge for instance.The solution for

[2]The average value differs only by a constant which is not of significance for this analysis.

the motion estimate is therefore well conditioned when the image portion consists of two clearly defined gradient directions. A good example is at a corner [72, 66].

This agrees well with intuition. As discussed in Chapter 2, vertical motion at a vertical (step) edge cannot be estimated. In that case all the information in the vertical direction would be the same. Any vertical motion would yield no change in information. Horizontal motion would of course yield a change in information because of travel across the vertical step. Therefore this motion can be readily estimated. This phenomenon can be summed up by stating that the motion estimate is most accurate in the direction of maximum contrast. This observation would reflect itself in the ill–conditioning of $\mathbf{W}$ by giving meaning to only one of the two equations for motion in equation D.1 when the equations are set up at an edge. This cannot be unravelled in the straightforward inverse operation required in equation D.1, the only clue is that the solution becomes ill–conditioned.

In the light of this analysis it seems natural to detect ill–conditioning in the usual way, by thresholding the eigenvalue ratio, then proceed to stabilize the solution using eigen analysis. This analysis can then yield a direction in which the motion estimation is more accurate. This is the solution that Martinez proposed, and his solution is repeated here.

$$\mathbf{d} = \begin{cases} \alpha_{\max}\,\mathbf{e}_{\max} & \text{if } \lambda_{\max} \gg \lambda_{\min} \\ [\mathbf{G}^T\mathbf{G}]^{-1}\,\mathbf{G}^T\mathbf{z} & \textit{otherwise} \end{cases} \tag{D.12}$$

$$\alpha_{\max} = \frac{\mathbf{e}_{\max}^T}{\lambda_{\max}}\mathbf{G}^T\mathbf{z}$$

Here, λ, $\mathbf{e}$ refer to the eigen values and eigen vectors of $\mathbf{G}^T\mathbf{G}$, and α is a scalar variable introduced to simplify the form of the final solution.

Note that in the work of Martinez [63], the solution was neither pel–recursive nor was any consideration given to the use of the regularising μ used in D.1.

D.3 Ill–conditioning in the general 3DAR solution.

Following a similar approach to that of Martinez, it is possible to examine the relation between the WBME and the general AR estimator with respect to conditioning of the solution.

Consider a temporally causal AR model in which the support region consists of points in the previous frame only, for example, AR9 in figure 3.8 in Chapter 3. Then, using the definition of $\mathbf{G}_w$ in equation 3.7, it is possible to express the general matrix in terms of $\mathbf{g}_x$ and $\mathbf{g}_y$, defined above, as follows.

$$\mathbf{G}_w = [\mathbf{A}\mathbf{g}_x \;\; \mathbf{A}\mathbf{g}_y] \tag{D.13}$$

$\mathbf{A}$ is a matrix of AR coefficients that would weight the relevant gradient terms of $\mathbf{g}$ to yield the general form indicated by equation 3.7.

From D.13 $\mathbf{G}_w^T\mathbf{G}_w$ can be expanded to

$$\mathbf{G}_w^T\mathbf{G}_w = \begin{bmatrix} \mathbf{g}_x^T\mathbf{A}^T\mathbf{A}\mathbf{g}_x & \mathbf{g}_x^T\mathbf{A}^T\mathbf{A}\mathbf{g}_y \\ \mathbf{g}_y^T\mathbf{A}^T\mathbf{A}\mathbf{g}_x & \mathbf{g}_y^T\mathbf{A}^T\mathbf{A}\mathbf{g}_y \end{bmatrix} \tag{D.14}$$

The matrix is therefore singular when

$$(\mathbf{g}_x^T\mathbf{A}^T\mathbf{A}\mathbf{g}_x)(\mathbf{g}_y^T\mathbf{A}^T\mathbf{A}\mathbf{g}_y) = (\mathbf{g}_y^T\mathbf{A}^T\mathbf{A}\mathbf{g}_x)(\mathbf{g}_x^T\mathbf{A}^T\mathbf{A}\mathbf{g}_y) \tag{D.15}$$

As before, the Schwartz inequality may be applied to yield the condition for singularity as

$$\mathbf{A}\mathbf{g}_x = \alpha\mathbf{A}\mathbf{g}_y \tag{D.16}$$

Provided that $\mathbf{A}$ has an inverse, the condition for a singular weighted gradient matrix, $\mathbf{G}_w^T\mathbf{G}_w$, is the same as for the matrix $\mathbf{G}^T\mathbf{G}$, as discussed previously. Therefore, the general 3DAR solution for motion is also ill–conditioned at an edge in the image.

An eigen analysis would yield similar conclusions to that discussed previously. The weighting matrix, $\mathbf{A}$ would change the eigen values and vectors of the un–weighted matrix, $\mathbf{G}^T\mathbf{G}$. Nevertheless, in view of the similar conditions for singularity, it would appear that 3DAR solution is not any better conditioned in this respect. In practice however, it is noticed that the solution is better behaved, and this must be due to the noise reducing effect of the gradient weighting.

D.4 Summary

An eigen analysis of the simple gradient based motion solution has related ill–conditioning to the presence of an edge in the image. The eigen values and vectors of $\mathbf{G}^T\mathbf{G}$ have been shown to represent the value and directions of maximum and minimum contrast in the image. Hence, the ill–conditioned solution can be related to the contrast orientation in the image, a standard problem.

In this respect, the 3DAR solution carries no advantage. It is also singular at edges in the image. However, it uses a weighted gradient matrix and each term therefore represents a weighted combination of gradient observations in a small region. This results in a less noisy gradient observation and so the solution is more stable.

The eigen analysis of both $\mathbf{G}^T\mathbf{G}$ and $\mathbf{G}_w^T\mathbf{G}_w$ can be used to propose a strategy when the solution is detected as ill–conditioned. This strategy is used to good effect in Chapter 3.

THE WIENER FILTER FOR IMAGE SEQUENCE RESTORATION

The Wiener filter has been used frequently in the past as an effective method for noise and blur reduction in degraded signals. The form of the filter for image sequences is a direct extension of the 1–D result, however the limited amount of temporal information available implies different considerations in the implementation of the method. This appendix makes explicit the various forms of the filter that can be used and in so doing lays the foundation for the ideas presented in appendix F. The theory presented here ignores motion between frames. Motion is considered in the main body of the text.

This thesis concerns itself with the problem of noise reduction in image sequences, blur is not considered. The observed signal model is then

$$g(i,j,n) = I(i,j,n) + \eta(i,j,n) \tag{E.1}$$

where $g(i,j,n)$ is the observed signal grey scale value at position (i,j) in the nth frame, $I(i,j,n)$ is the actual non–degraded signal and $\eta(i,j,n)$ the added white noise of variance $\sigma_{\eta\eta}$.

E.1 The 3D Frequency Domain/3D IIR Wiener filter

The Wiener filter attempts to produce the best estimate for $I(i,j,n)$, denoted $\hat{I}(i,j,n)$. It does so by minimizing the expected squared error given by

$$E[(e(i,j,n))^2] = E[(I(i,j,n) - \hat{I}(i,j,n))^2] \tag{E.2}$$

This estimate may be achieved using either an IIR or FIR Wiener filter, which operates on the observed noisy signal. The following two expressions show respectively IIR and FIR estimators.

$$\hat{I}(i,j,n) = \sum_{k_1}\sum_{k_2}\sum_{k_3} a(k_1,k_2,k_3)g(i+k_1, j+k_2, n+k_3) \tag{E.3}$$

$$\hat{I}(i,j,n) = \sum_{k_1=-N_1}^{N_1}\sum_{k_2=-N_2}^{N_2}\sum_{k_3=-N_3}^{N_3} a(k_1,k_2,k_3)g(i+k_1, j+k_2, n+k_3) \tag{E.4}$$

In equation E.3 there are no limits on the summations, hence IIR, and in equation E.4 the filter mask (or support) is a symmetric volume around the filtered location of size $(2N_1+1)\times(2N_2+1)\times(2N_3+1)$.

Proceeding with the IIR filter leads here to a frequency domain expression for the Wiener filter. Using the principle of orthogonality to bypass some lines of differential algebra, $E[(e(\ldots))^2]$

is minimized if the error is made orthogonal to the data samples used in the estimate equation E.3.

$$E[e(i,j,n)g(i+k_1,j+k_2,n+k_3)] = 0 \ \forall \ k_1,k_2,k_3 \tag{E.5}$$

Substituting for $e(i,j,n)$ in equation E.5 gives

$$E[\hat{I}(i,j,n)g(i+k_1,j+k_2,n+k_3)] = E[I(i,j,n)g(i+k_1,j+k_2,n+k_3)] \ \forall \ k_1,k_2,k_3 \tag{E.6}$$

The filter coefficients are then chosen to satisfy the condition of equation E.6. Substituting for $\hat{I}(i,j,n)$ in E.6 yields,

$$\sum_{l_1,l_2,l_3} a(l_1,l_2,l_3)E[g(i+l_1,j+l_2,n+l_3)g(i+k_1,j+k_2,n+k_3)]$$
$$= E[I(i,j,n)g(i+k_1,j+k_2,n+k_3)] \ \forall \ k_1,k_2,k_3,l_1,l_2,l_3 \tag{E.7}$$

The expectations can be recognized as terms from the autocorrelation and crosscorrelation sequences of the observed image sequence $g(i,j,n)$ and the actual sequence $I(i,j,n)$. The solution involves a separate equation for each coefficient, i.e. an infinite set of equations. However, using the 3D DFT gives a tractable result in terms of the power spectra concerned. From equation E.6 the following expressions result assuming stationary statistics.

$$\sum_{l_1}\sum_{l_2}\sum_{l_3} a(l_1,l_2,l_3)R_{gg}(l_1-k_1,l_2-k_2,l_3-k_3)] \ = \ R_{ig}(k_1,k_2,k_3) \tag{E.8}$$

Taking Fourier transforms yields

$$A(\omega_1,\omega_2,\omega_3)P_{gg}(\omega_1,\omega_2,\omega_3) \ = \ P_{ig}(\omega_1,\omega_2,\omega_3) \tag{E.9}$$

The only unknown quantity here is the cross power spectrum P_{ig}. However, using the original assumption of uncorrelated noise and image frames, implies that

$$P_{gg} \ = \ P_{ii} + P_{\eta\eta} \tag{E.10}$$

$$P_{ig} \ = \ P_{ii} \tag{E.11}$$

From these two equations, an expression for the 3D frequency domain Wiener filter, $A(\omega_1,\omega_2,\omega_3)$ is as follows.

$$A(\omega_1,\omega_2,\omega_3) = \frac{P_{gg}(\omega_1,\omega_2,\omega_3) - P_{\eta\eta}(\omega_1,\omega_2,\omega_3)}{P_{gg}(\omega_1,\omega_2,\omega_3)} \tag{E.12}$$

The estimated signal, $\hat{I}(i,j,n)$ is then given by the expression below.

$$\hat{I}(i,j,n) = \ \mathrm{IDFT}[A(\omega_1,\omega_2,\omega_3)G(\omega_1,\omega_2,\omega_3)] \tag{E.13}$$

The IDFT is the inverse 3D DFT, and $G(\omega_1,\omega_2,\omega_3)$ is the 3D DFT of the observed signal, $g(i,j,n)$. The filter is therefore defined by the signal power spectrum and the noise variance.

Using the 3D DFT in this way makes the Wiener filter computationally attractive. However in the temporal direction there are often not many frames involved. Therefore, the temporal

Fourier component is less meaningful, having been derived from only a few samples. In the thesis, 3 frames are involved, hence the DFT in the temporal direction involves just 3 samples. More frames can be included but at the expense of stationarity since effects such as occlusion and uncovering are more likely to appear. The assumption of an infinite support volume is violated. This phenomenon is also applicable to the spatial components since the image is only stationary over small areas. Therefore, in a practical situation, the 3D IIR filter implemented in the frequency domain in this way, is no longer IIR since it operates on a finite volume of input data only and not on any past outputs.

The problem may be overcome by considering two further forms of the filter which allow for finite support. One form follows from the FIR filter expression in equation E.3, the other results from a matrix formulation of the filter.

E.2 The 3D FIR Wiener filter

The FIR filter result follows again by using the theorem of orthogonality. The expression E.4 for the filter is substituted into the error term in the orthogonality expression E.5 using equation E.2, except allowing for the finite support. This yields

$$\sum_{l_1,l_2,l_3} a(l_1,l_2,l_3)E[g(i+l_1,j+l_2,n+l_3)g(i+k_1,j+k_2,n+k_3)] = E[I(i,j,n)g(i+k_1,j+k_2,n+k_3)] \quad \text{(E.14)}$$

$$\text{for} \begin{cases} -N_1 \leq k_1,\ l_1 \leq N_1 \\ -N_3 \leq k_2,\ l_2 \leq N_2 \\ -N_3 \leq k_3,\ l_3 \leq N_3 \end{cases} \quad \text{(E.15)}$$

Because the support of the FIR filter is finite, it is better to describe the filter expression in terms of offset vectors $\mathbf{q}_k$ as was done in the case of the 3DAR framework described in Chapter 2 and Appendix A. This results in a much simpler formulation[1] .

$$\begin{aligned} E[\hat{I}(\mathbf{x})g(\mathbf{q}_k)] &= \sum_{l=0}^{N} a_l E[g(\mathbf{x}+\mathbf{q}_l)g(\mathbf{x}+\mathbf{q}_k)] \\ &= E[I(\mathbf{x})g(\mathbf{x}+\mathbf{q}_k)] \text{ for } 0 \leq k \leq N \end{aligned} \quad \text{(E.16)}$$

Here N vectors, $\mathbf{q}_i$ define the support volume of the filter, and so the expression above defines a set of N equations each for one filter coefficient. The solution for the coefficients can then follow from the matrix equation below.

$$\mathbf{R}_{gg}\mathbf{a} = \mathbf{r}_{ig} \quad \text{(E.17)}$$

[1]Note that the 3D FIR filter described here uses a mask volume of size $N_1 \times N_2 \times N_3$.

The terms in each matrix are defined below.

$$\mathbf{a} = [a_0\ a_1\ a_2\ \ldots\ a_N] \tag{E.18}$$

$$\mathbf{R}_{gg} = \begin{bmatrix} r_{gg}(\mathbf{q}_0,\mathbf{q}_0) & r_{gg}(\mathbf{q}_0,\mathbf{q}_1) & \cdots & r_{gg}(\mathbf{q}_1,\mathbf{q}_N) \\ r_{gg}(\mathbf{q}_2,\mathbf{q}_1) & r_{gg}(\mathbf{q}_2,\mathbf{q}_2) & \cdots & r_{gg}(\mathbf{q}_2,\mathbf{q}_N) \\ r_{gg}(\mathbf{q}_3,\mathbf{q}_1) & r_{gg}(\mathbf{q}_3,\mathbf{q}_2) & \cdots & r_{gg}(\mathbf{q}_3,\mathbf{q}_N) \\ \vdots & \vdots & \vdots & \vdots \\ r_{gg}(\mathbf{q}_N,\mathbf{q}_1) & r_{gg}(\mathbf{q}_N,\mathbf{q}_2) & \cdots & r_{gg}(\mathbf{q}_N,\mathbf{q}_N) \end{bmatrix} \tag{E.19}$$

$$\mathbf{r}_{ig}^T = [r_{ig}(\mathbf{q}_0,\mathbf{q}_0)\ldots r_{ig}(\mathbf{q}_0,\mathbf{q}_1)\ldots r_{ig}(\mathbf{q}_0,\mathbf{q}_N)] \tag{E.20}$$

$$\mathbf{q}_0 = [0\ 0\ 0] \tag{E.21}$$

Although this matrix solution is virtually identical to the solution for 3D AR model coefficients (See Appendix A), note the important difference that this solution includes an estimate for the coefficient at the zero lag tap, $\mathbf{q}_0$. This explains why the 3D AR formulation is called a *predictor* and this formulation a *filter*.

Again the only information not available for the solution of equation E.17 is the cross correlation vector, $\mathbf{r}_{ig}$. This vector involves correlations between the unknown original signal and the observed noisy samples. However, the assumption that the image and noise frames are uncorrelated allows a series of useful equalities[2].

$$\begin{aligned} r_{ig}(\mathbf{q}_0,\mathbf{q}_h) &= E[I(\mathbf{x})g(\mathbf{x}+\mathbf{q}_h)] \\ &= E[I(\mathbf{x})\{I(\mathbf{x}+\mathbf{q}_h)+\eta(\mathbf{x}+\mathbf{q}_h)\}] \\ &= \begin{cases} E[I(\mathbf{x})I(\mathbf{x}+\mathbf{q}_h)] & \text{for } \mathbf{q}_h \neq [0\ 0] \\ E[I(\mathbf{x})I(\mathbf{x})] & \text{for } \mathbf{q}_h = [0\ 0] \end{cases} \end{aligned} \tag{E.22}$$

$$\tag{E.23}$$

Multiplying equation E.1 by $g(\mathbf{x}+\mathbf{q}_h)$ and taking expectations gives

$$\begin{aligned} E[g(\mathbf{x})g(\mathbf{x}+\mathbf{q}_h)] &= E[I(\mathbf{x})I(\mathbf{x}+\mathbf{q}_h)] \\ &\quad +E[I(\mathbf{x})\eta(\mathbf{x}+\mathbf{q}_h)]+E[\eta(\mathbf{x})I(\mathbf{x}+\mathbf{q}_h)] \\ &\quad +E[\eta(\mathbf{x})\eta(\mathbf{x}+\mathbf{q}_h)] \end{aligned} \tag{E.24}$$

Substituting for $E[I(\mathbf{x})I(\mathbf{x}+\mathbf{q}_k)]$ from equation E.24 into equation E.23, gives a usable expression for the required cross correlation term using the observed signal and an estimate of the noise variance.

$$r_{ig}(\mathbf{q}_0,\mathbf{q}_h) = \begin{cases} E[g(\mathbf{x})g(\mathbf{x}+\mathbf{q}_h)] & \text{for } \mathbf{q}_h \neq [0,\ 0] \\ \sigma_{gg}^2 - \sigma_{\eta\eta}^2 & \text{for } \mathbf{q}_h = [0,\ 0] \end{cases} \tag{E.25}$$

[2]In practice the data volume is finite and so a sufficiently large data volume is necessary to make measurements of correlation terms.

σ^2_{gg} is the observed signal variance, and $\sigma^2_{\eta\eta}$ the noise variance. For non–zero lags, it is intuitively reasonable to equate the crosscorrelation of the clean signal and the observed signal by the autocorrelation of the observed signal since the effect of the noise on the terms would be small. A solution for equation E.17 then follows using the inverse of $\mathbf{R}_{gg}$.

Having found the coefficients of the filter, the filter can then be applied following equation E.4 to achieve noise reduction.

E.3 The matrix formulation of the 3D Wiener filter

The last approach to be considered is also motivated by the need to deal with a finite volume of sequence data. In this form the signals are ordered into column vectors. The observation equation for each frame, n is then

$$\mathbf{g}_n = \mathbf{s}_n + \eta_n \tag{E.26}$$

Here $\mathbf{s}_n$ is used to represent the ordered values from the unknown clean signal $I(i, j, n)$. The vectors for each frame may then be stacked upon each other in another larger column vector. If the size of each frame is $M \times M$ pixels and there are N frames, then the new vectors are NM^2 long and the new equation is as follows.

$$\mathbf{g} = \mathbf{s} + \eta \tag{E.27}$$

The task of the Wiener filter is to operate on the observed signal, $\mathbf{g}$, to suppress noise. The 3D FIR filter above gives one framework for operating on finite data, but for all possible taps to be taken into account, the matrix solution, given below, is useful.

$$\hat{\mathbf{s}} = \mathbf{H}\mathbf{g} \tag{E.28}$$

$\mathbf{H}$ is a matrix operator of size $NM^2 \times NM^2$. It is necessary to find the coefficients of the matrix operator, $\mathbf{H}$, to give the estimated signal. Again, the idea is to choose this matrix to minimize the expectation of the squared error as below.

$$E[\mathbf{e}^T\mathbf{e}] = E[(\mathbf{s} - \hat{\mathbf{s}})^T(\mathbf{s} - \hat{\mathbf{s}})] \tag{E.29}$$

A substitution from equation E.28 can then be made into E.29 which is then minimized with respect to $\mathbf{H}$. Again making the assumption that the image and noise frames are uncorrelated, finally allows the estimate for $\mathbf{H}$ to be derived as

$$\begin{aligned} \mathbf{H} &= E[\mathbf{s}\mathbf{s}^T](E[\mathbf{s}\mathbf{s}^T] + E[\eta\eta^T])^{-1} \\ &= \mathbf{R}_{ss}(\mathbf{R}_{ss} + \mathbf{R}_{\eta\eta})^{-1} \end{aligned} \tag{E.30}$$

The complete derivation is not shown here but can be found in [42, 55, 88, 25]. The important thing about this expression is that it involves the inverse of very large matrices. The correlation matrices are all of size $NM^2 \times NM^2$, for a typical block size of about 16×16, and 3 frames,

this is 768×768. This is not a practical solution and so several authors have introduced approximations to reduce the computation necessary to calculate this inverse, [58, 88, 25, 36]. One of these approximations is discussed in Appendix F.

REDUCING THE COMPLEXITY OF WIENER FILTERING

Although the frequency domain Wiener filter is computationally simple, requiring only 3 separable DFT implementations, the matrix formulation of the filter is computationally quite heavy. This formulation is potentially more accurate than the frequency domain implementation since the form takes into account the finite temporal size of the data volume involved. Özkan et al. [58, 1] have developed an efficient solution for this filter form which takes advantage of some approximations about the circulant nature of the correlation matrix of the signals. It is possible to derive their result by an alternative route which is described here after a brief overview of their result.

F.1 Efficient Wiener filtering via 2D DFT diagonalisation

The signal model is as stated in the previous appendix on Wiener Filtering.

$$\mathbf{g} = \mathbf{s} + \eta \tag{F.1}$$

The column vectors are made up of the data from N frames ordered and stacked. Therefore,

$$\mathbf{g} = [\mathbf{g}_1 \mathbf{g}_2 \cdots \mathbf{g}_N]^T \tag{F.2}$$

and so on for the other data vectors, where the frames are indexed from 1 to N.

The matrix Wiener solution was given in the previous appendix. Following Galatsanos et al. [25], Özkan et al [58] noted that the correlation matrices involved were block circulant. The solution for the transformation matrix, $\mathbf{H}$ is

$$\mathbf{H} = \mathbf{R}_{ss}(\mathbf{R}_{ss} + \mathbf{R}_{\eta\eta})^{-1} \tag{F.3}$$

The Wiener estimate for $\mathbf{s}$ is then

$$\hat{\mathbf{s}} = \mathbf{R}_{ss}(\mathbf{R}_{ss} + \mathbf{R}_{\eta\eta})^{-1}\mathbf{g} \tag{F.4}$$

The structure of each matrix can be represented as below,

$$\mathbf{R}_{ss} = \begin{bmatrix} \mathbf{R}_{ss:11} & \cdots & \mathbf{R}_{ss:1N} \\ \vdots & \ddots & \vdots \\ \mathbf{R}_{ss:N1} & \cdots & \mathbf{R}_{ss:NN} \end{bmatrix} \tag{F.5}$$

$$\mathbf{R}_{\eta\eta} = \begin{bmatrix} \mathbf{R}_{\eta\eta:11} & \dots & \mathbf{R}_{\eta\eta:1N} \\ \vdots & \ddots & \vdots \\ \mathbf{R}_{\eta\eta:N1} & \dots & \mathbf{R}_{\eta\eta:NN} \end{bmatrix} \tag{F.6}$$

$$\text{Where } \mathbf{R}_{ss:hk} = E[\mathbf{s}_h \mathbf{s}_k^T] \tag{F.7}$$

$$\text{and } \mathbf{R}_{\eta\eta:hk} = E[\eta_h \eta_k^T] \tag{F.8}$$

$$\tag{F.9}$$

Assuming that the image and noise frames are statistically homogeneous, the sub–matrices in each correlation matrix are block Toeplitz. As stated in [58, 1], the submatrices may be diagonalized [36] using the 2–D DFT. A matrix $\mathbf{W}$ may be defined such that $\mathbf{W}^{-1}\mathbf{g}$ stacks the 2–D DFTs of the ordered data vector $\mathbf{g}$. Following Özkan, premultiplying both sides of equation F.4 by $\mathbf{W}^{-1}$ yields

$$\mathbf{W}^{-1}\hat{\mathbf{s}} = \mathbf{W}^{-1}\mathbf{R}_{ss}\mathbf{W}[\mathbf{W}^{-1}\mathbf{R}_{ss}\mathbf{W} + \mathbf{W}^{-1}\mathbf{R}_{\eta\eta}\mathbf{W}]^{-1}\mathbf{W}^{-1}\mathbf{g} \tag{F.10}$$

The submatrices in the equation have now been diagonalized via the 2–D DFT. The equation may be written as

$$\hat{\mathbf{F}}_{\hat{s}} = \mathbf{P}_{ss}[\mathbf{P}_{ss} + \mathbf{P}_{\eta\eta}]^{-1}\mathbf{F}_g \tag{F.11}$$

$$\hat{\mathbf{F}}_{\hat{s}} = \mathbf{P}_{ss}\mathbf{Q}^{-1}\mathbf{F}_g \tag{F.12}$$

$$\tag{F.13}$$

Here, $\mathbf{F}_{\hat{s}}$, $\mathbf{F}_g$ are column vectors containing the stacked 2D DFTs of the signal estimate and observed noisy signal respectively. The matrices $\mathbf{P}_{ss}$, $\mathbf{P}_{\eta\eta}$ are the 2D cross Power spectral densities of the various frames involved. Note that P_{ss} is not diagonal but its sub matrices, $P_{ss:hk}$, are diagonal. They can be defined as below.

$$\mathbf{P}_{ss} = \begin{bmatrix} \mathbf{P}_{ss:11} & \dots & \mathbf{P}_{ss:1N} \\ \vdots & \ddots & \vdots \\ \mathbf{P}_{ss:N1} & \dots & \mathbf{P}_{ss:NN} \end{bmatrix} \tag{F.14}$$

$$\text{where } \mathbf{P}_{ss:hk} = \begin{bmatrix} P_{ss:hk,1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & P_{ss:hk,M^2} \end{bmatrix} \tag{F.15}$$

$$\mathbf{Q} = \begin{bmatrix} \mathbf{Q}_{11} & \dots & \mathbf{Q}_{1N} \\ \vdots & \ddots & \vdots \\ \mathbf{Q}_{N1} & \dots & \mathbf{Q}_{NN} \end{bmatrix} \tag{F.16}$$

$$\text{where } \mathbf{Q}_{hk} = \begin{bmatrix} Q_{hk,1} & \dots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \dots & Q_{hk,M^2} \end{bmatrix} \tag{F.17}$$

$$\text{and } \mathbf{Q}_{hk} = \mathbf{P}_{ss:hk} + \delta_{hk}\mathbf{P}_{\eta\eta:hk} \tag{F.18}$$

$$\forall\, h,\ k \ = \ 1 \ldots N \tag{F.19}$$

Note that the noise frames are assumed uncorrelated, hence the inclusion of the Kronecker delta function[1] in the expression for $\mathbf{Q}_{hk}$ above.

The expressions are somewhat cumbersome, but the result has been to diagonalize the submatrices of the correlation matrices used in the Wiener estimate. The elements along the main diagonal of the submatrices $\mathbf{Q}_{hk}$ and $\mathbf{P}_{\mathbf{ss:hk}}$ are the ordered 2D DFT bins of the relevant correlation functions. Therefore, the element $P_{ss:hk,l}$ is the amplitude of the lth ordered frequency in the 2D Cross Power spectral density of the original, clean signal in frame h, and k.

The matrix $\mathbf{Q}$ is very large, but because of the diagonalization transformation, it is a block matrix with diagonal blocks. Therefore, Özkan et al. proposed an efficient solution by invoking a rule that implies that the inverse of a block matrix with diagonal blocks is itself a block matrix with diagonal blocks. The inverse of the $NM^2 \times NM^2$ matrix $\mathbf{Q}$ was then derived from the inverse of M^2 smaller matrices of size $N \times N$. This in a sense is the crux of the solution presented by Özkan et al.

Define the inverse of $\mathbf{Q}$ to be $\mathbf{Z}$. Then define submatrices $\mathbf{Q}_k$ and $\mathbf{Z}_k$ as follows.

$$\text{Let } \mathbf{Z} \ = \ \begin{bmatrix} Z_{11} & \ldots & Z_{1N} \\ \vdots & \ddots & \vdots \\ Z_{N1} & \ldots & Z_{NN} \end{bmatrix}$$

$$\text{and } \mathbf{Z}_{hk} \ = \ \begin{bmatrix} Z_{hk:1} & \ldots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \ldots & Z_{hk:M^2} \end{bmatrix}$$

$$\mathbf{Q}_k \ = \ \begin{bmatrix} Q_{11:k} & \ldots & Q_{1N:k} \\ \vdots & \ddots & \vdots \\ Q_{N1:k} & \ldots & Q_{NN:k} \end{bmatrix} \tag{F.20}$$

$$\mathbf{Z}_k \ = \ \begin{bmatrix} Z_{11:k} & \ldots & Z_{1N:k} \\ \vdots & \ddots & \vdots \\ Z_{N1:k} & \ldots & Z_{NN:k} \end{bmatrix} \tag{F.21}$$

(F.22)

Therefore the two sets of $N \times N$ submatrices, $\mathbf{Q}_k$ and $\mathbf{Z}_k$, are subsampled versions of the major matrices. Then the lemma invoked by Özkan at al. shows that the components of $\mathbf{Z}_k$ are equal to the inverse of the $N \times N$ submatrix, $\mathbf{Q}_k$. There are M^2 such sub matrices hence the inverse operation is much reduced in complexity. In the case of $N = 2$, or 3, an analytic solution can easily be formulated.

[1] $\delta_{hk} = 1$ for $h = k$

The Wiener estimate for the ith frame then results from substituting this inverse into equation F.12 as follows

$$\mathbf{F}_{\hat{s}:i,k} = \sum_{p=1}^{N} P_{ss:ip,k} \sum_{q=1}^{N} Z_{pq,k} F_{g:q,k} \tag{F.23}$$

for each of the M^2 frequencies involved indexed by k.

F.2 An alternative derivation.

The situation above can be viewed in a different light to give the same result. Consider again that there are $2N+1$ frames of size $(2M+1) \times (2M+1)$ in the observation equation F.2. Construct the required estimate in the spatiotemporal domain such that

$$\hat{I}(x_1, x_2, x_3) = \sum_{i,j,l} a(i,j,l) g(x_1+i, x_2+j, x_3+l) \tag{F.24}$$

In this expression, (x_1, x_2, x_3) and (i, j, l) are the spatial and temporal coordinates and offsets of a pixel, where $-M \leq i, j \leq M$ and $-N \leq l \leq N$.

The equation F.24 may be written explicitly as the sum of separate convolutions of the observed frames g_i with a set of different coefficients for each frame. If there were 3 frames required for the FIR filter the equation would read

$$\begin{aligned} \hat{I}(x_1, x_2, x_3) &= \sum_{i,j} a(i,j,-1) g(x_1+i, x_2+j, x_3-1) \\ &\quad + \sum_{i,j} a(i,j,0) g(x_1+i, x_2+j, x_3) \\ &\quad + \sum_{i,j} a(i,j,+1) g(x_1+i, x_2+j, x_3+1) \end{aligned} \tag{F.25}$$

for a non–causal filter using a frame previous to and after the current frame x_3. This equation can then be written in the frequency domain, using the 2D DFT, as

$$\begin{aligned} F_{\hat{s}:x_3}(\omega_1, \omega_2) &= A_{-1}(\omega_1, \omega_2) F_{g:x_3-1}(\omega_1, \omega_2) \\ &\quad + A_0(\omega_1, \omega_2) F_{g:x_3}(\omega_1, \omega_2) \\ &\quad + A_1(\omega_1, \omega_2) F_{g:x_3+1}(\omega_1, \omega_2) \end{aligned} \tag{F.26}$$

Here, $F_{\hat{s}:h}(\omega_1, \omega_2)$ and $F_{g:h}(\omega_1, \omega_2)$ represent the 2D DFT of the hth frame estimate and noisy observed frame respectively. Similarly $A_q(\omega_1, \omega_2)$ represents the 2D DFT of the coefficient mask in the qth offset frame, so that it is applied to frame $x_3 + q$ in this case. The transformed coefficient masks, $A_k(\omega_1, \omega_2)$ etc, are required.

The 2D DFT of the estimate of the clean signal in frame v can then be represented by

$$F_{\hat{s},v}(\omega_1, \omega_2) = \sum_{q=-N}^{N} A_q(\omega_1, \omega_2) F_{g,v+q}(\omega_1, \omega_2) \tag{F.27}$$

Returning to the spatiotemporal domain, the MMSE (Minimum Mean Squared Error) estimate for the coefficients, $a(i,j,l)$ is found by solving the following set of equations (from equation E.15).

$$\sum_{i,j,l} a(i,j,l)E[g(x_1+i,x_2+j,x_3+l)g(x_1+k_1,x_2+k_2,x_3+k_3)] = E[I(x_1,x_2,x_3)g(x_1+k_1,x_2+k_2,x_3+k_3)] \quad \text{(F.28)}$$

$$\text{for} \begin{cases} -N \leq k_3 \leq N \\ -M \leq k_1,\ k_2 \leq M \end{cases} \quad \text{(F.29)}$$

This equation can then be simplified to make clear the correlation measurements that are required between frames.

$$\sum_{i,j,l} a(i,j,l)r_{gg:x_3+l,x_3+k_3}(i-k_1,j-k_2) = r_{sg:x_3,x_3+k_3}(k_1,k_2) \quad \text{(F.30)}$$

$$\text{for} \begin{cases} -N \leq k_3 \leq N \\ -M \leq k_1,\ k_2 \leq M \end{cases} \quad \text{(F.31)}$$

Two assumptions are made, spatial homogeneity and that the noise and image frames are not correlated with each other. The solution may be simplified by taking the 2D DFT of the expression to yield

$$P_{sg:x_3,x_3+k_3}(\omega_1,\omega_2) = \sum_{q=-N}^{N} A_q(\omega_1,\omega_2)P_{gg:x_3+q,x_3+k_3}(\omega_1,\omega_2) \quad \text{(F.32)}$$

$$\text{for } -N \leq k_3 \leq N \quad \text{(F.33)}$$

Here, $P_{gg:x_3+q,x_3+k_3}(\omega_1,\omega_2)$ represents the Cross Power Spectral Density of the noisy frames x_3+q and x_3+k_3, and $P_{sg:x_3,x_3+k_3}(\omega_1,\omega_2)$ is the required cross power spectral density of the noisy and original frames. Here s is used instead of i as the index to avoid confusion with the use of i as the horizontal offset used in the filter expression.

Equation F.32 represents a set of N equations for the N coefficients, $A_q(\omega_1,\omega_2)$, at each spatial frequency (ω_1,ω_2). The set of equations may be written as, (using a shortened form for the Power Spectral Densities concerned)

$$\begin{bmatrix} P_{gg:-N,-N}(\omega_1,\omega_2) & \cdots & P_{gg:-N,N}(\omega_1,\omega_2) \\ \vdots & \ddots & \vdots \\ P_{gg:N,-N}(\omega_1,\omega_2) & \cdots & P_{gg:N,N}(\omega_1,\omega_2) \end{bmatrix} \begin{bmatrix} A_{-N}(\omega_1,\omega_2) \\ \vdots \\ A_N(\omega_1,\omega_2) \end{bmatrix} = \begin{bmatrix} P_{sg:-N}(\omega_1,\omega_2) \\ \vdots \\ P_{sg:N}(\omega_1,\omega_2) \end{bmatrix}$$

(F.34)

$$\begin{aligned} \text{where } P_{gg:hk}(\omega_1,\omega_2) &= P_{gg:x_3+h,x_3+k}(\omega_1,\omega_2) \\ &= P_{ss:x_3+h,x_3+k}(\omega_1,\omega_2) \\ &\quad +\delta_{x_3+h,x_3+k}P_{\eta\eta}(\omega_1,\omega_2) \\ \text{and } P_{sg:h}(\omega_1,\omega_2) &= P_{sg:x_3,x_3+h}(\omega_1,\omega_2) \\ &= P_{ss:x_3,x_3+h} \end{aligned}$$

$P_{ss:hk}(\omega_1,\omega_2)$ is the Cross Power Spectrum of the frames in the clean original signal $I(i,j,n)$, similarly for $P_{\eta\eta}(\omega_1,\omega_2)$). The solution to this matrix equation at each spatial frequency would then yield the 2D DFT of the coefficients in each frame. This result can then be substituted into the frequency domain expression for the filter, equation F.27, to yield the 2D DFT of the estimate for the required frame.

The equation F.34 may be expressed in a more compact form.

$$\tilde{\mathbf{Q}}\mathbf{a} = \mathbf{p} \tag{F.35}$$

From this equation the estimate of the clean signal at spatial frequency (ω_1,ω_2) in the vth frame may be written as

$$F_{\hat{s},v}(\omega_1,\omega_2) = \mathbf{a}^T\mathbf{F}_g \tag{F.36}$$

where $\mathbf{F}_g$ is as defined by the terms required in F.27.

Letting $\tilde{\mathbf{Z}}(\omega_1,\omega_2)$ represent the required inverse matrix in equation F.34, $\tilde{\mathbf{Q}}^{-1}$, the following definition is helpful.

$$\tilde{\mathbf{Z}}(\omega_1,\omega_2) = \begin{bmatrix} Z_{-N,-N}(\omega_1,\omega_2) & \dots & Z_{-N,N}(\omega_1,\omega_2) \\ \vdots & \ddots & \vdots \\ Z_{N,-N}(\omega_1,\omega_2) & \dots & Z_{N,N}(\omega_1,\omega_2) \end{bmatrix} \tag{F.37}$$

Note that the inverses defined in equation F.37 and F.21 are identical matrices because the $\mathbf{Q}_k$ matrix introduced previously in equation F.20 is the same as the left hand matrix introduced in equation F.34. There is an offset in the indices, but this is due to the definition of the filter.

Given this matrix inverse then, it is possible using F.36 to write the 2D DFT of the estimate in the vth frame as

$$F_{\hat{s}:v}(\omega_1,\omega_2) = \sum_{p=-N}^{N} P_{ss:v,v+p}(\omega_1,\omega_2) \sum_{q=-N}^{N} Z_{pq}(\omega_1,\omega_2)F_{g:q}(\omega_1,\omega_2) \tag{F.38}$$

Here the 2D DFT of the required estimate of the original clean signal is denoted by $F_{\hat{s}:v}(\omega_1,\omega_2)$. Note that again $\hat{s}$ is used as a synonym for $\hat{I}$. This solution is identical to the one given previously in F.23 except for the range of the indices. This depends on the definition of the the filtering operation but does not affect the outcome.

This alternative approach highlights the basic difference between this efficient Wiener solution and the matrix solution discussed previously in Appendix E. It strikes a balance between the fully 3D Frequency approach, which depends on the validity of the DFT of a small number of samples in the temporal direction, and the fully spatiotemporal approach which is computationally intensive. Özkan et al. presented this efficient method with respect to global interframe motion, they went on to incorporate the motion itself into the filter solution.

F.3 A final refinement

Özkan et al. recognized that the inverse operation that was required for this new solution could also be computed efficiently. This was allowed if the approximation was made that the power spectra for the noise frames was identical for each frame and that

$$P_{ss:p,q}(\omega_1, \omega_2) = F_{s:p}(\omega_1, \omega_2) F^*_{s:q}(\omega_1, \omega_2) \tag{F.39}$$

Which means that the cross power spectral densities of frames p and q can be calculated by the magnitude of the product of the 2D DFT's of the frames. They show that under these conditions, the inverse of the matrix $\mathbf{Q}_k$, defined in equation F.20, can be solved analytically. The final result for the 2D DFT of the required estimate of frame v is stated below. A complete derivation can be found in [58]. The equation stated here uses their notation for the implementation of the filter with respect to the indices used to the image frames. This was defined at the start of this appendix.

$$F_{\hat{s}:v}(\omega_1, \omega_2) = \frac{F_{s:v}(\omega_1, \omega_2) \sum_{q=1}^{N} F^*_{s:q}(\omega_1, \omega_2) F_{g:q}(\omega_1, \omega_2)}{\sum_{l=1}^{N} |F_{s:l}(\omega_1, \omega_2)|^2 + P_{\eta\eta}(\omega_1, \omega_2)} \tag{F.40}$$