Application of Deinterlacing for the Enhancement of Surveillance Video

by

Brian A. Heng

B.S. Electrical Engineering
University of Minnesota, 1999

Submitted to the Department of Electrical Engineering and Computer Science
in Partial Fulfillment of the Requirements for the Degree of

Master of Science
in Electrical Engineering and Computer Science

at the

Massachusetts Institute of Technology

June 2001

Signature of Author ______________________
Department of Electrical Engineering and Computer Science
May 2001

Certified by ______________________
Jae S. Lim
Professor of Electrical Engineering and Computer Science
Thesis Supervisor

Accepted by ______________________
Arthur C. Smith
Chair, Departmental Committee on Graduate Students

Application of Deinterlacing for the Enhancement of Surveillance Video

by

Brian Heng

Submitted to the Department of Electrical Engineering and Computer Science
on May 11, 2001 in Partial Fulfillment of the Requirements for
the Degree of Master of Science in Electrical Engineering

Abstract

As the cost of video technology has fallen, surveillance cameras have become an integral part of a vast number of security systems. However, even with the introduction of progressive video displays, the majority of these systems still use interlaced scanning so that they may be connected to standard television monitors. When law enforcement officials analyze surveillance video, they are often interested in carefully examining a few frames of interest. However, it is impossible to perform frame-by-frame analysis of interlaced surveillance video without performing interlaced to progressive conversion, also known as deinterlacing. In most surveillance systems, very basic techniques are used for deinterlacing, resulting in a number of severe visual artifacts and greatly limiting the intelligibility of surveillance video. This thesis investigates fourteen deinterlacing algorithms to determine methods that will improve the quality and intelligibility of video sequences acquired by surveillance systems. The advantages and disadvantages of each algorithm are discussed followed by both qualitative and quantitative comparisons. Motion adaptive deinterlacing methods are shown to have the most potential for surveillance video, demonstrating the highest performance both visually and in terms of peak signal-to-noise ratio.

Thesis Supervisor: Jae S. Lim

Title: Professor of Electrical Engineering

Dedication

To Mom and Dad,
For Their Endless Support.

To Susanna,
For All Her Love.

Acknowledgements

I would like to take this opportunity to thank to my advisor Professor Jae S. Lim for his guidance and patience. The opportunity to learn from him these past years has been a great honor. I would like to express my gratitude to him for providing me a place in his lab and for making this thesis possible.

I would also like to thank Cindy Leblanc, group secretary for the Advanced Television Research Program. Her kindness has always made me feel welcome, and it is through her efforts that the lab functions so smoothly.

I was lucky to have Dr. Eric Reed as my officemate, who was always willing to provide me with friendship and guidance. I would like to thank him for all the time he spent helping and teaching me.

I am especially grateful to my friend and colleague Wade Wan for reviewing my original manuscript and for doing all he could to make me feel welcome at MIT. I owe him a great debt of gratitude for all he has done for me.

Finally, I would like to thank my family and friends for their love and support. I am privileged to have so many friends and relatives to provide me strength and encouragement. I am especially grateful for the support of my parents, Duane and Mary Jane Heng. This accomplishment would not have been possible without their unending source of love.

Brian Heng
Cambridge, MA
May 11th, 2001

Contents

1 Introduction **15**

2 Surveillance Systems **19**

3 Deinterlacing **23**

4 Deinterlacing Algorithms **27**

4.1 Intraframe Methods **27**

4.1.1 Line Repetition *28*

4.1.2 Linear Interpolation *31*

4.1.3 Parametric Image Modeling *31*

4.2 Interframe Methods **34**

4.2.1 Field Repetition *34*

4.2.2 Bilinear Field Interpolation *36*

4.2.3 Vertical-Temporal Median Filtering *37*

4.2.4 Motion Compensated Deinterlacing *39*

4.2.5 Motion Adaptive Deinterlacing *41*

5 Motion Adaptive Deinterlacing **43**

5.1 Motion Detection in Interlaced Sequences **43**

5.1.1 Two-Field Motion Detection *45*

5.1.2 Three-Field Motion Detection *47*

5.1.3 Four-Field Motion Detection *49*

5.1.4 Five-Field Motion Detection *51*

5.2 Motion Adaptive Deinterlacing **54**

6 **Experiments and Results** 57

6.1 **Experimental Setup** 57

6.2 **Results** 63

7 **Conclusion** 85

7.1 **Summary** 85

7.2 **Future Work** 86

A **Experimental Data** 87

List of Figures

1-1 Scan Modes for Video Sequences 16

2-1 A Typical Security Camera System 20
2-2 Conversion from VHS Storage to NTSC Video 21

3-1 Vertical-Temporal Spectrum of Interlaced Video 25

4-1 Illustration of Line Repetition 28
4-2 Example of Line Repetition 29
4-3 Jitter Caused by Line Repetition 30
4-4 Implementation of Martinez-Lim Spatial Deinterlacing Algorithm 32
4-5 Interpolation by Martinez-Lim Algorithm 33
4-6 Comparison of Intraframe Interpolation Methods 33
4-7 Example of Deficiency in Field Repetition Algorithm 35
4-8 Blurred Frame Caused by Field Repetition 35
4-9 Blurred Frame Caused by Field Interpolation 37
4-10 Three-tap VT Median Filter 38

5-1 Example of Motion Detection Using Progressive Frames 44
5-2 Two-Field Motion Detection Scheme 45
5-3 Three-Field Motion Detection Scheme 47
5-4 Sequence of Frames Causing Three-Field Motion Detection to Fail 48
5-5 Results of Deinterlacing in the Presence of Motion Detection Errors 48
5-6 Four-Field Motion Detection Scheme 49
5-7 Results of Deinterlacing using Four-Field Motion Detection 50
5-8 Sequence of Frames Causing Four-Field Motion Detection to Fail 50
5-9 Five-Field Motion Detection Scheme 51

5-10 Pixel Definitions for Modified VT-Median Filter 53
5-11 General Concept Behind Motion Adaptive Deinterlacing 55
5-12 Transfer Function to generate motion parameter α 55
5-13 Motion Adaptive Deinterlacing Algorithm 56

6-1 Generation of progressive data by downsampling interlaced data 60
6-2 Average PSNR Comparison – News CIF 64
6-3 Average PSNR Comparison – Silent Voice CIF 65
6-4 Average PSNR Comparison – ATRP Lab – Sequence A 66
6-5 Average PSNR Comparison – ATRP Lab – Sequence B 67
6-6 Average PSNR Comparison – ATRP Lab – Sequence C 68
6-7 Average PSNR Comparison – ATRP Lab – Sequence D 69
6-8 Average PSNR Comparison – ATRP Lab – Sequence E 70
6-9 Comparison of Intraframe Interpolation Methods 71
6-10 Bilinear Field Interpolation 72
6-11 Average PSNR Comparison – Combined Results 74
6-12 MPEG Test Sequence – News CIF 75
6-13 Enlarged View of Figure 6-12 76
6-14 MPEG Test Sequence – Silent Voice CIF 77
6-15 Enlarged View of Figure 6-13 78
6-16 Surveillance footage – February 12th, 2001 79
6-17 Comparison of Performance vs. Frame Rate 81
6-17 (continued) 82
6-17 (continued) 83

List of Tables

6-1 Summary of Deinterlacing Algorithms Used for Testing....58

6-2 Summary of MPEG Test Sequences Used for Algorithm Development....59

6-3 Summary of Surveillance Sequences Used for Algorithm Comparison....61

6-3 (continued)....62

6-4 Summary of Surveillance Sequences Used for Frame Rate Comparison....63

A-1 Average PSNR Data Used to Generate Figure 6-2....87

A-2 Average PSNR Data Used to Generate Figure 6-3....87

A-3 Average PSNR Data Used to Generate Figure 6-4....88

A-4 Average PSNR Data Used to Generate Figure 6-5....88

A-5 Average PSNR Data Used to Generate Figure 6-6....88

A-6 Average PSNR Data Used to Generate Figure 6-7....89

A-7 Average PSNR Data Used to Generate Figure 6-8....89

A-8 Frame Rate Comparison Data at 30 Frames per Second....89

A-9 Frame Rate Comparison Data at 10 Frames per Second....90

A-10 Frame Rate Comparison Data at 5 Frames per Second....90

A-11 Frame Rate Comparison Data at 2.5 Frames per Second....90

Chapter 1
Introduction

When the first US television standard was introduced in 1941, interlaced scanning, or interlacing, was used as a compromise between video quality and transmission bandwidth. An interlaced video sequence appears to have the same spatial and temporal resolution as a progressive sequence and yet it only occupies half the bandwidth due to vertical-temporal sampling. This sampling method takes advantage of the human visual system, which tends to be more sensitive to details in stationary regions of a video sequence than in moving regions. Prior to the introduction of the U.S. High Definition Television (HDTV) standard in 1995, interlaced scanning had been adopted in most video standards [24]. For instance, in 1941, the National Television Systems Committee (NTSC) introduced an interlaced-based television standard that was used in the United States exclusively until the introduction of HDTV. As a result, interlacing is still widely used in video systems and is found throughout the video chain, from studio cameras to home television sets.

A video *frame* is a picture made up of a two-dimensional discrete grid of *pixels*. A video *sequence* is a collection of frames, with equal dimensions, displayed at fixed time intervals. The *scan mode* is the method in which the pixels of each frame are displayed. As shown in Figure 1-1, video sequences can have one of two scan modes: *progressive* or *interlaced*. A progressive scan sequence is one in which every line of the video is scanned in every frame. This type of scanning is typically used in computer monitors and high definition television displays. An interlaced sequence is one in which the display alternates between scanning the *even lines* and *odd lines* of the corresponding progressive frames. The standard convention is to start enumeration at zero, i.e. the first line is line 0 and the second line is line 1. Thus, the first line is even and the second line is odd. The term *field* is used (rather than frame) to describe pictures scanned using interlaced scanning, with the *even field* containing all the even lines of one frame and the *odd field* containing the odd lines. The terms *top field* and *bottom field* are also used to denote the even and odd fields, respectively.

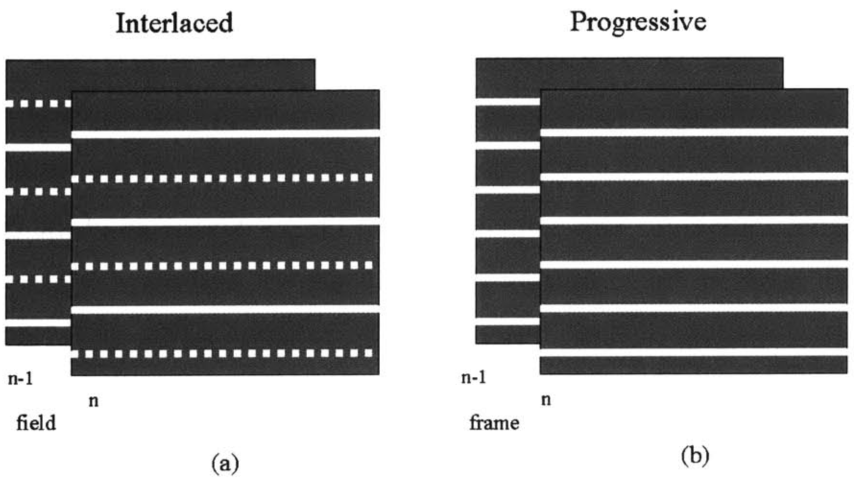


Figure 1-1: Scan modes for video sequences. (a) In interlaced fields either the even or the odd lines are scanned. The solid lines represent the field that is present in the current frame. (b) In progressively scanned frames all lines are scanned in each frame.

While interlacing does succeed in reducing the transmission bandwidth, it also introduces a number of high frequency spatio-temporal artifacts that can be distracting to the human eye, such as line crawl and interline flicker. In addition, there are a number of applications where interlaced scanning is unacceptable. For instance, freeze frame displays and the photographic capture of a television image both require a whole frame to be displayed at once. With advances in technology, it is also becoming more popular to view video on a progressively scanned computer monitor or high definition television set. These applications all require interlaced to progressive conversion. A simple conversion method is to combine the even and odd fields to form a new frame. However, this results in severe blurring in regions of significant motion. The desire to eliminate interlacing artifacts provides motivation for developing high quality algorithms for interlaced to progressive conversion, also known as *deinterlacing.*

One particular application where deinterlacing is very useful is in video surveillance systems. As video recording equipment has become more affordable, the use of video surveillance has become an integral part of security systems worldwide. Nearly every retail store, restaurant, bank, and gas station has some form of video surveillance installed. For the same reasons that interlacing is found throughout broadcast television systems, interlaced scanning is used in the vast majority of security systems as well. The bulk of all existing displays use interlacing in order to facilitate easy interoperability with other interlaced video devices such as cameras, VCRs, or DVD players. Since the rest of the security system is coupled to the display, interlacing is found throughout the video chain. While it is possible that progressively scanned systems may become more commonplace in the future, especially with the recent introduction of HDTV, at the present time nearly every surveillance system uses interlaced scanning.

Typically, when criminal investigators view surveillance footage, they only need to analyze a few frames of interest. They often want to view the frames one at a time, looking for one good image that can help to identify a suspect or analyze a region of interest. Since the video source is interlaced, this type of frame-by-frame analysis is difficult without deinterlacing. Video sequences should be deinterlaced and then analyzed on a high quality, progressively scanned display to improve the intelligibility of the video.

The overall goal of this thesis is to investigate different deinterlacing algorithms to determine methods that will improve the quality and intelligibility of video sequences acquired by surveillance systems. The deinterlacing problem has been studied since interlacing was first introduced, and many different algorithms have been proposed to deinterlace video sequences in general. However, security video sequences have certain attributes that differentiate them from other sequences. Deinterlacing algorithms for surveillance video sequences do not need to consider a number of characteristics that occur in general video sequences, such as scene changes, global camera motion, or zooming. This greatly simplifies the deinterlacing problem. Secondly, surveillance systems often record video at temporal rates as low as 5 to 10 fields per second. This is significantly lower than the 60 fields per second used in the NTSC standard. As a result of these low temporal rates, adjacent fields are less temporally correlated. Deinterlacing algorithms that rely on this temporal correlation for effectiveness will not work well on this type of video. This thesis will test a number of different deinterlacing algorithms in order to determine which algorithm best fits these characteristics and helps aid in the identification of individuals present in these sequences.

Chapter 2 provides an in depth look at a typical surveillance system. The video analysis methods currently used in practice are also discussed. Formal definitions of interlacing and deinterlacing are provided in Chapter 3. Chapter 4 discusses the deinterlacing algorithms that were examined in this thesis. The advantages and disadvantages of each algorithm are also discussed. One particular class of deinterlacing algorithms showed excellent potential throughout this research. These methods, known as motion adaptive deinterlacing algorithms, are examined in detail in Chapter 5. In addition to the details of the experiments conducted for this thesis, results and analysis are provided in Chapter 6. Finally, Chapter 7 provides the final conclusions and suggestions for further research.

Chapter 2
Surveillance Systems

As video technology has improved, the expense of installing a surveillance system has dropped significantly, leading to an exponential increase in the use of security cameras. Today it is nearly impossible to enter a shopping mall or to eat at a restaurant and not be video taped.

As shown in Figure 2-1, a typical security camera system is made up of three components: the video camera itself, a time-lapse recorder, and a monitor. The video signal flows from the camera to the time-lapse recorder where it is recorded onto a VHS tape. The signal also passes through the recorder to the monitor for possible real-time viewing. Often black-and-white cameras are used to reduce cost, but color cameras are not uncommon. Time-lapse videocassette recorders are essentially high quality VCRs, as can be found in any consumer entertainment system. The only difference is that time-lapse recorders have the ability to record at extremely low temporal rates. The monitor used in this type of system can be any video display, such as a standard television. If a black-and-white camera is used, the monitor is often black-and-white as well, again to help reduce costs. At a given location, there may be many cameras multiplexed into one or more different recorders, but the overall system will be similar to the one depicted in Figure 2-1.

Since the video camera and monitor are both self-explanatory, the component that is really of interest is the time-lapse recorder. These recorders take the video signal from the camera, and record it onto standard VHS tapes. They record the video at very low temporal rates so that the recording time per tape is maximized and fewer tapes are needed. For instance, a standard VHS tape can hold about 8 hours of video using a standard VCR. However, by using a time-lapse recorder up to 40 hours of video can be recorded on the same tape by recording only 12 fields per second compared to the 60 fields per second recorded by a standard VCR. Requiring the end-user to change videotapes every 8 hours is unacceptable for many surveillance situations, and thus, time-lapse recording makes these surveillance systems feasible. Time-lapse recorders give users the ability to select temporal rates much lower than the NTSC standard of 60 fields per second. Rates such as 30 fields per second, 20 fields per second, and 12 fields per second are common.

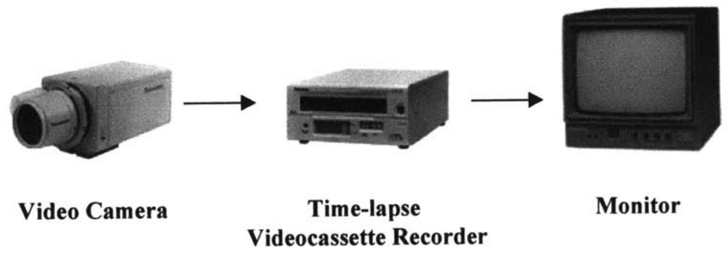


Figure 2-1: A typical security camera system. The video signal is taken from a video camera typically mounted on a wall or ceiling and recorded with a time-lapse videocassette recorder. The recorded tape can then be viewed later on a monitor.

Once video is recorded onto the VHS tape, it can be played back by the time-lapse recorder and viewed on a monitor. The recorder can play the tape back at the appropriate speed to match the rate at which the video was originally recorded. Since the monitor requires a NTSC signal with a temporal rate of 60 fields per second, the recorder must adjust the output signal. To accomplish this, many recorders use line repetition, the simplest of all deinterlacing algorithms. Line repetition is discussed in more detail in Chapter 4. In the line repetition algorithm, the lines of one field are simply repeated to generate the missing lines and create a whole frame. Figure 2-2 shows an example of using line repetition to convert a 20 field per second signal into a 60 field per second NTSC signal. The display of a single frame on an interlaced monitor is done in the same manner, however in this case, a single field is continuously repeated for as long as desired.

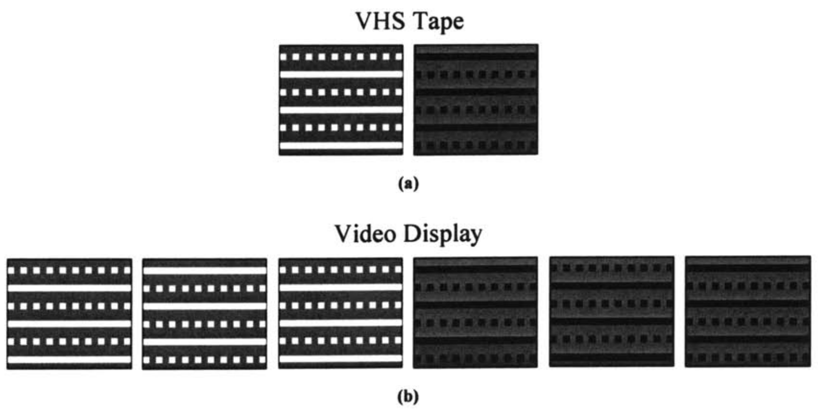


Figure 2-2: Conversion from VHS storage to NTSC video. (a) Two consecutive fields recorded at 20 fields per second as stored on a VHS tape. (b) Display of the same two fields as an NTSC signal at 60 fields per second. In this example, each field is displayed three times in order to generate a valid NTSC signal. This has the same effect as line repetition, which is discussed in Chapter 4.

Line repetition is one of the simplest methods of deinterlacing and will be shown in later chapters to result in very poor performance. However, this simple algorithm leads to a simple, low-cost implementation. Therefore, line repetition is the method implemented on most time-lapse recorders. For this reason, in most surveillance systems, investigators are forced to view video obtained using line repetition. The remainder of this thesis investigates more advanced methods of deinterlacing the video, so that it may be analyzed with higher intelligibility on progressively scanned displays.

Chapter 3
Deinterlacing

A progressive scan sequence consists of frames, in which every line of the video is scanned. An interlaced sequence consists of fields, which alternate between scanning the even and odd lines of the corresponding progressive frames. A video sequence is a three-dimensional array of data in the vertical, horizontal, and temporal dimensions. Let $F[x,y,n]$ denote the pixel in this three-dimensional sequence at horizontal position x, vertical position y, and time n. Assuming that the video source is progressively scanned, $F[x,y,n]$ is defined for all integers x, y, and n within the valid height, width, and time duration of the sequence. If $F_i[x,y,n]$ is an interlaced source generated from the progressive source $F[x,y,n]$, then $F_i[x,y,n]$ is defined as follows

$$F_i[x,y,n] = \begin{cases} F[x,y,n] & \text{mod}(y,2) = \text{mod}(n,2) \\ \varnothing & \text{otherwise} \end{cases} \qquad (1)$$

where the null symbol, $\varnothing$, represents pixels that do not exist since they are not scanned, and $\text{mod}(y,2)$ is the modulus operator defined as

$$\text{mod}(z,2) = \begin{cases} 0 & \text{z even} \\ 1 & \text{z odd} \end{cases} \qquad (2)$$

Thus, the interlacing operation alternates between selecting the even field or the odd field from the corresponding progressively scanned frame.

Deinterlacing a video sequence involves converting the interlaced fields into progressively scanned frames. Ideal deinterlacing would double the vertical-temporal sampling rate and remove aliasing [5]. However, this process is not a simple rate conversion problem, since the Nyquist criterion requirement is generally not met. Video cameras are not typically equipped with the optical pre-filters necessary to prevent aliasing effects in the deinterlaced output, and if they were, severe blurring would result [5].

For a given interlaced input $F_i[x,y,n]$, the output of deinterlacing, $F_o[x,y,n]$, can then be defined as

$$F_o[x,y,n] = \begin{cases} F_i[x,y,n], & \mathrm{mod}(y,2) = \mathrm{mod}(n,2) \\ \hat{F}[x,y,n], & \text{otherwise} \end{cases} \tag{3}$$

where the pixels $\hat{F}[x,y,n]$ are the estimation of the missing lines generated by the deinterlacing algorithm. The existing, even or odd, lines in the original fields are directly transferred to the output frame. These definitions will be used throughout this thesis: $F[x,y,n]$ is the original frame, $F_i[x,y,n]$ is the interlaced field, and $F_o[x,y,n]$ is the result of deinterlacing.

The main difficulty in deinterlacing can be seen in the following figure. Figure 3-1 (a) shows the frequency spectrum of an interlaced signal along the vertical-temporal plane. The interlaced sampling lattice results in the quincunx pattern of spectral repetitions [5]. The darker regions in Figure 3-1 (a) represent aliasing due to the interlaced sampling. The ideal deinterlacing algorithm would remove the central repeated spectra to give the spectrum depicted in Figure 3-1 (b). The areas of overlap in (a) cannot be perfectly recovered by any deinterlacing algorithm, so the result in (b) is not possible in general.

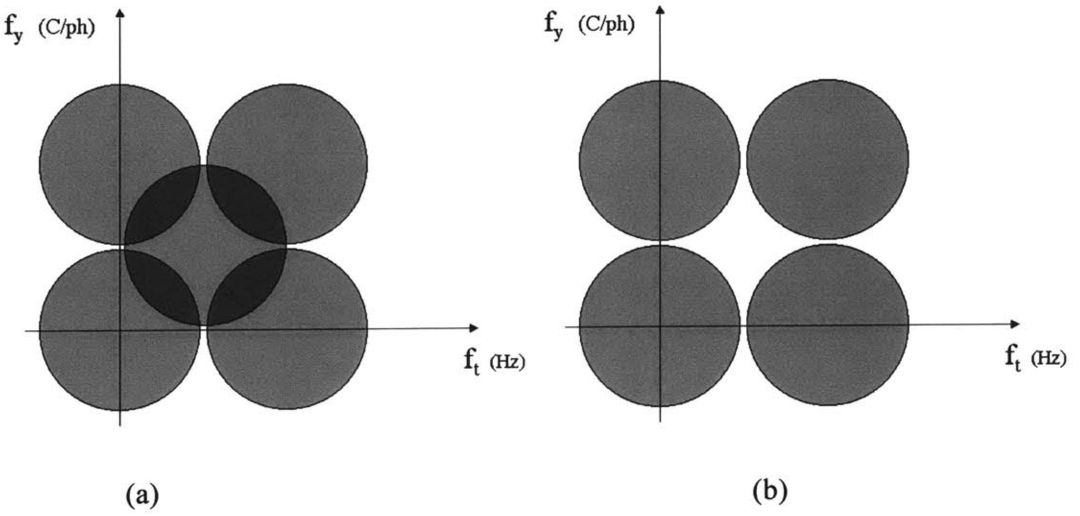


Figure 3-1: Vertical-temporal spectrum of interlaced video. (a) General vertical-temporal spectrum of interlaced video sequence. Interlaced sampling causes a quincunx pattern of spectral repetitions. Dark regions represent areas that are unrecoverable due to aliasing. (b) Ideal output of deinterlacing algorithm. Central spectral repetition is removed.

Chapter 4
Deinterlacing Algorithms

Deinterlacing methods have been proposed ever since the introduction of interlacing. With increases in technology, progressive displays have become feasible and the interest in deinterlacing has increased. The methods that have been suggested vary greatly in complexity and performance, but they can be segmented into two categories: *intraframe* and *interframe*. Intraframe methods only use the current field for reconstruction, while interframe methods make use of the previous and or subsequent frames as well. Note that, all the methods attempt to reconstruct the missing scan lines without the knowledge of the original progressive source.

The purpose of this thesis is to investigate a number of deinterlacing methods and analyze their performance when applied to surveillance camera footage. To this end, fourteen different algorithms have been compared. Given the extent to which this problem has been studied, it was impossible to implement every algorithm that has ever been suggested. However, these fourteen methods are believed to be a fair representation of the different types of deinterlacing algorithms that have been suggested to date.

The following sections specify the details of the algorithms that were explored as well as their individual advantages and disadvantages.

4.1 Intraframe Methods

Intraframe or spatial methods only use pixels in the current field to reconstruct missing scan lines. Therefore, they do not require any additional frame storage. Initially, the memory required to store one video frame was expensive, so intraframe methods were attractive. Since spatial methods consider only one frame at a time, their performance is independent of the amount of motion present in the sequence or the frame-recording rate. Robustness to motion is one characteristic that is very useful for deinterlacing low frame rate surveillance video. However, considering only the information in the present field greatly limits these algorithms due to the large temporal correlation that typically exists between successive fields.

4.1.1 Line Repetition

Line repetition is one of the simplest deinterlacing algorithms, and thus, was one of the first to be considered. In this method, the missing lines are generated by repeating the line directly above or below the missing line. The top field is copied down to fill in the missing lines, and the bottom field is copied up as illustrated in Figure 4-1.

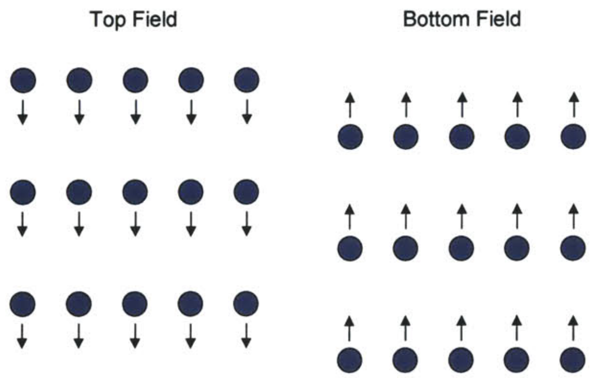


Figure 4-1: Illustration of line repetition. The top field is copied down to fill in the missing bottom field. The bottom field is copied up to fill in the missing top field.

The mathematical definition of line repetition is as follows.

For top fields:

$$F_o[x,y,n] = \begin{cases} F_i[x,y,n], & \text{y even} \\ F_i[x,y-1,n], & \text{otherwise} \end{cases}$$

(4)

For bottom fields:

$$F_o[x,y,n] = \begin{cases} F_i[x,y,n], & \text{y odd} \\ F_i[x,y+1,n], & \text{otherwise} \end{cases}$$

An example of the result of line repetition is shown in Figure 4-2.

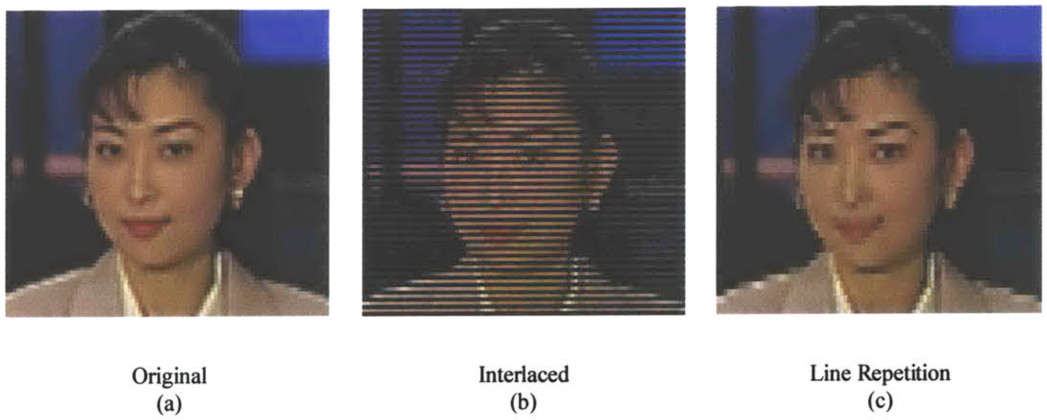


Figure 4-2: Example of line repetition. (a) Original progressive frame. (b) Interlaced field. (c) Output of line repetition. The lines in the interlaced field are repeated to generate the output frame.

As is shown by the figure, this type of zero-order-hold interpolation often has poor performance. In addition to introducing jagged edge artifacts caused by aliasing, line repetition can also cause severe "jitter" in deinterlaced video due to improper reconstruction of vertical details. A demonstration of the cause of this artifact is shown in Figure 4-3.

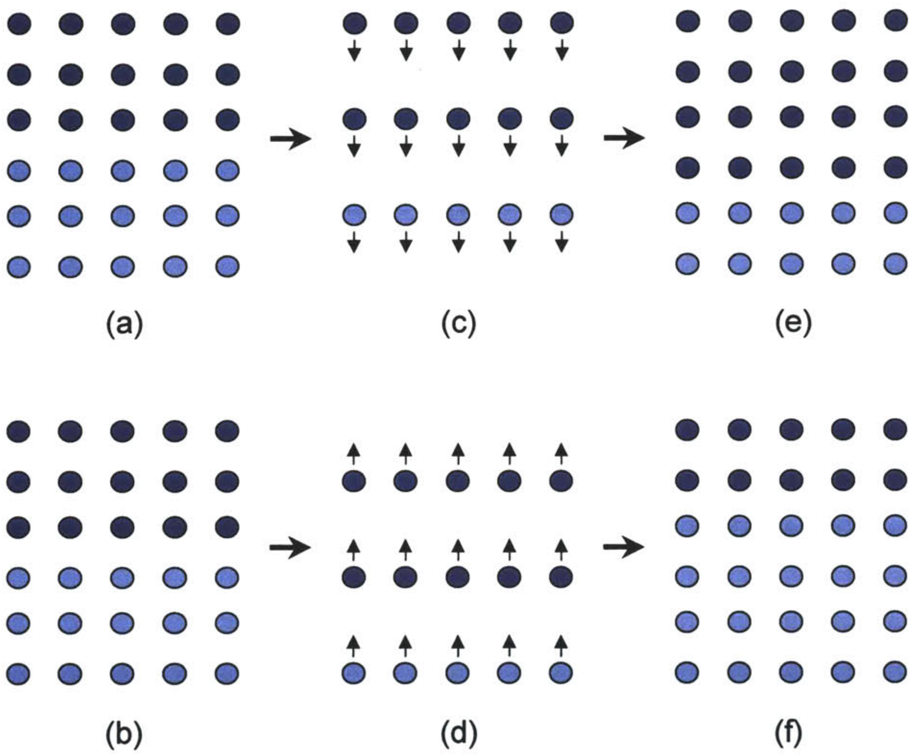


Figure 4-3: Jitter Caused by Line Repetition. (a) (b) Consecutive progressive frames of a stationary image with a horizontal edge. (c) (d) Corresponding interlaced fields. (e) (f) Result of reconstruction using line repetition. Comparing the deinterlaced frames, the incorrect reconstruction of the horizontal edge causes a 2 pixel shift in successive frames. This shift will make the output sequence appear to shake.

As seen in this example, vertical details such as the edge in Figure 4-3 will be incorrectly reconstructed. These errors will make video sequences appear to shake slightly in the vertical direction. The rate of this vibration corresponds directly to the frame rate of the video sequence. At high frame rates, this shaking may be so fast that it becomes unnoticeable to the human eye. However, at lower frame rates this effect becomes very noticeable and extremely annoying.

4.1.2 Linear Interpolation

Linear interpolation is another simple deinterlacing algorithm that is only slightly more advanced than line repetition. In linear interpolation the missing lines are reconstructed by averaging the lines directly above and below. The formal definition of linear interpolation is

$$F_o[x,y,n] = \begin{cases} F_i[x,y,n] \ , & \mathrm{mod}(y,2) = \mathrm{mod}(n,2) \\ \dfrac{F_i[x,y+1,n] + F_i[x,y-1,n]}{2} \ , & \text{otherwise} \end{cases} \qquad (5)$$

This type of algorithm results in a slightly smoother picture than line repetition, but still has fairly poor performance. The same problems caused by line repetition, such as aliasing artifacts and jitter, exist with linear interpolation as well.

4.1.3 Parametric Image Modeling

A more advanced form of spatial deinterlacing utilizes image modeling. Parametric image modeling algorithms attempt to model a small region of an image through a set of parameters and basis equations. Missing lines are reconstructed using linear interpolation. The model attempts to determine the contours of a shape in an image and interpolates along this direction to reduce interpolation errors. Martinez and Lim, and Ayazifar and Lim have suggested models, which can be used to spatially interpolate, interlaced video fields. Martinez suggests a Line Shift Model in which small segments of adjacent scan lines are assumed to be related to each other through a horizontal shift [18]. Ayazifar similarly suggests a generalization of the Line Shift Model through a Concentric Circular Shift Model in which small segments of concentric arcs of an image are related to adjacent arcs by an angular shift [1].

One of the benefits of these methods is that they tend to be less susceptible to noise. When attempting to fit an image to a model, pixels corrupted by noise often do not correctly fit the within the model, and thus the model tends to disregard them. However, the system must also be over-determined in order for the model to closely match the actual image, which requires considering a large area of the image as well as more computations. Depending on the order of model selected, the complexity of this algorithm can be greater than other methods.

The Line Shift Model method suggested by Martinez and Lim has been included in this study [18]. Figure 4-4 illustrates the general idea of this algorithm. The algorithm begins by selecting a certain group of pixels surrounding the missing pixel that is to be reconstructed. The algorithm implemented for this study uses a group of 10 pixels, 5 pixels on two lines, as suggested by Martinez in [18]. The parameters for the image model are then estimated for the selected group of pixels, and these parameters are used to estimate the appropriate line shift vector. Once the shift vector is determined, interpolation is done along the direction of the estimated line shift as is shown in Figure 4-5.

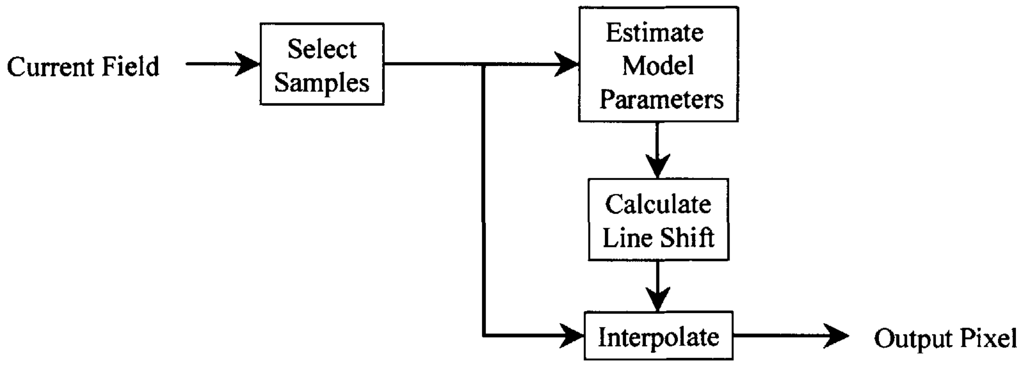


Figure 4-4: Implementation of Martinez-Lim spatial deinterlacing algorithm

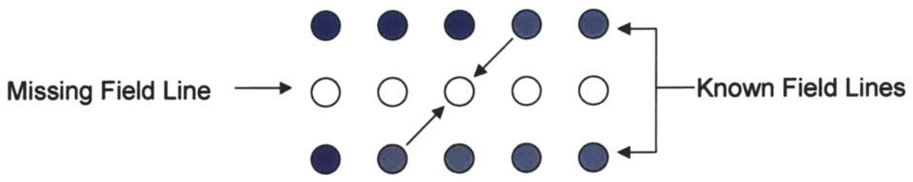


Figure 4-5: Interpolation by Martinez-Lim algorithm. Appropriate line shift is calculated and interpolation is done along this direction. In this example, the missing pixel at the center of the figure is reconstructed by averaging the two pixels along the edge direction.

A comparison of the spatial deinterlacing algorithms that have been mentioned is shown in Figure 4-6. Of these three, the Martinez-Lim algorithm tends to generate the smoothest, most realistic images. While it also suffers from some of the jitter effects found in line repetition, these effects are significantly reduced when compared to the other two methods.

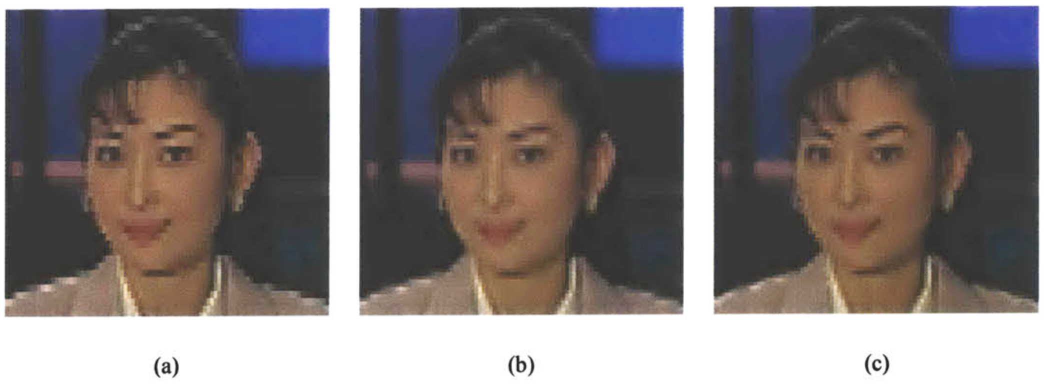


Figure 4-6: Comparison of intraframe interpolation methods. (a) Line repetition. (b) Linear interpolation. (c) Martinez-Lim algorithm.

4.2 Interframe Methods

While the performance of spatial methods is independent of the amount of motion present in an image, they essentially ignore a significant amount of information in adjacent frames that may be useful. For instance, if a video sequence contains one perfectly stationary image, this sequence could be perfectly deinterlaced by simply combining the even and odd fields. Interframe or temporal methods consider previous and/or subsequent frames to exploit temporal correlation. These methods require storage for one or more frames in their implementation, which may have been a serious difficulty in the past. However, the cost of frame storage is not as serious an issue with the reduced cost of memory.

4.2.1 Field Repetition

Field repetition refers to the generation of missing scan lines by copying lines from the previous frame at the same vertical position. Specifically, field repetition is defined as:

$$F_o[x,y,n] = \begin{cases} F_i[x,y,n], & \mathrm{mod}(y,2) = \mathrm{mod}(n,2) \\ F_i[x,y,n-1], & \text{otherwise} \end{cases} \qquad (6)$$

This type of deinterlacing would provide essentially perfect reconstruction of stationary video. However, field repetition can cause severe blurring if the video contains motion. Consider the example presented in Figure 4-7.

At high frame rates, the temporal correlation between two adjacent fields may be very high. If this is the case, field repetition can perform well with minor noticeable blurring. Yet, at lower temporal rates, as is used in surveillance video, this blurring effect becomes more pronounced. Figure 4-8 shows an example of the effects of field repetition being used in a moving region of a video sequence.

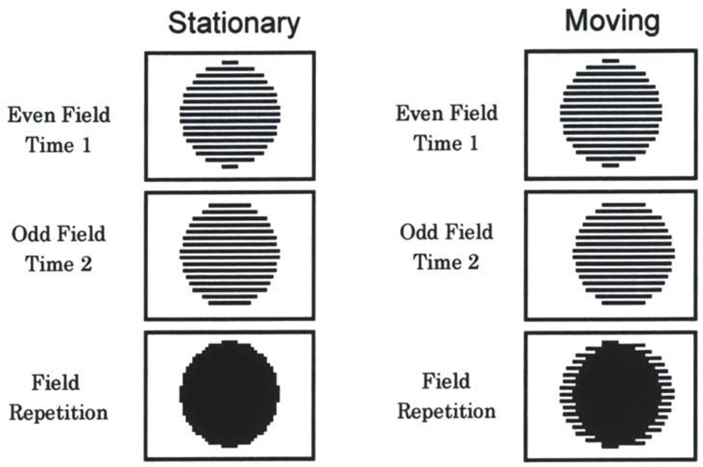


Figure 4-7: Example of deficiency in field repetition algorithm. The stationary circle is perfectly reconstructed by combining the even and odd fields. However, in the second example, the image is blurred since the even and odd fields are no longer properly aligned due to the motion of the circle between frames.

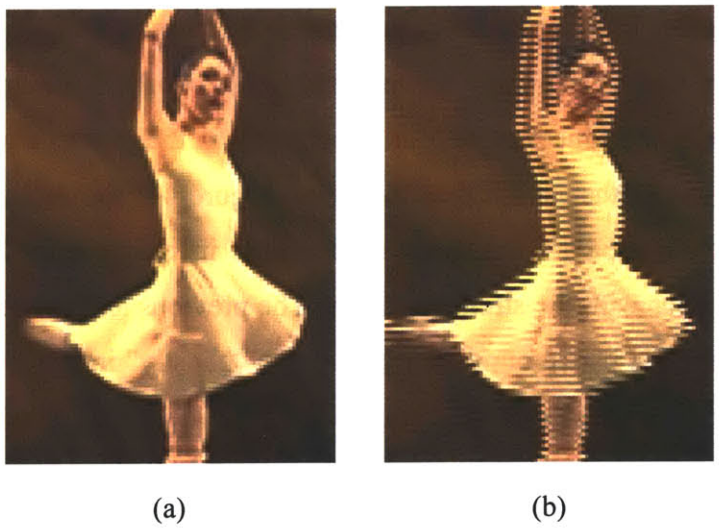


Figure 4-8: Blurred frame caused by field repetition. (a) Original progressive frame. (b) Frame produced using field repetition. Motion of the ballerina causes edges to be distorted resulting in a blurred image.

Since surveillance video systems use stationary video cameras, most regions of the video contain no motion. For instance, the background regions, where no motion is taking place, make up the majority of these video sequences. In these regions, field repetition will have excellent performance. Yet, given the extremely low frame rates used in surveillance video, field repetition can cause severe artifacts wherever motion is present. Thus, it is not suitable for this application, since the areas of motion are likely to be the main areas of interest. However, it was included in this study for comparison purposes.

4.2.2 Bilinear Field Interpolation

Bilinear field repetition uses the average of the previous and future lines to deinterlace the current missing line. The formal definition is

$$F_o[x,y,n]=\begin{cases} F_i[x,y,n] \ , & \operatorname{mod}(y,2)=\operatorname{mod}(n,2) \\ \dfrac{F_i[x,y,n+1]+F_i[x,y,n-1]}{2} \ , & \text{otherwise} \end{cases} \tag{7}$$

Field interpolation has essentially the same benefits and issues as field repetition. This method works very well in stationary regions, and works poorly in moving regions. Field interpolation actually introduces more blurring artifacts in the presence of motion since it blends three frames together rather than just two. Figure 4-9 shows an example of this distortion. The three separate fields are each visible in image (b). The current field shows up in the center with the strongest color. The previous and future fields show up, slightly blended into the background. The result is a "ghostly" image with badly blurred edges. In this sense, when compared to field repetition, field interpolation is even more unacceptable.

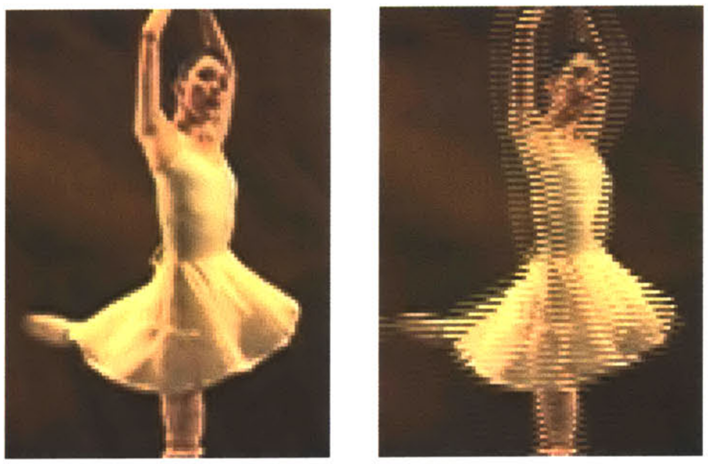


Figure 4-9: Blurred frame caused by field interpolation. (a) Original progressive frame. (b) Frame produced using bilinear field interpolation. Three fields of the video sequence are blended together leaving a severely distorted result in the presence of motion.

4.2.3 Vertical-Temporal Median Filtering

As with all interframe methods Vertical-Temporal (VT) filters use pixels in adjacent frames, however VT filters also utilize neighboring pixels in the current frame. In VT median filtering, as its name implies, a median operation is used rather than a linear combination of the surrounding pixels. VT median filtering has become very popular due to its performance and ease of implementation. The simplest example of a VT median filter is the 3-tap algorithm as suggested in [5]. To calculate the pixel *X* as seen in Figure 4-10, one simply finds the median of the two vertically adjacent points, *A* and *B*, and the temporally adjacent point in the previous field, *C*, as described in (8).

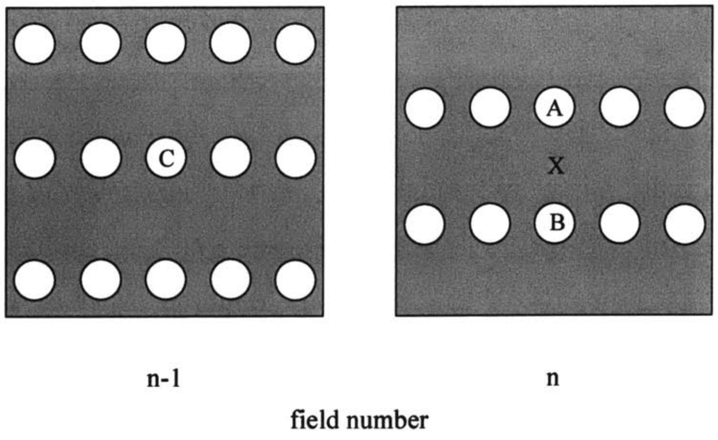


Figure 4-10: Pixel definitions for 3-tap VT Median Filtering

$$X = med(A, B, C) \tag{8}$$

This is formally described as

$$F_o[x,y,n] = \begin{cases} F_i[x,y,n] \;, & \operatorname{mod}(y,2) = \operatorname{mod}(n,2) \\ \text{median}\left(F_i[x,y+1,n], F_i[x,y-1,n], F_i[x,y,n-1]\right) \;, & \text{otherwise} \end{cases} \tag{9}$$

Median filtering adapts itself to moving and stationary regions on a pixel-by-pixel basis. If the image is stationary, pixel C is likely to take on a value somewhere between A and B. Thus, pixel C will be chosen by the median operation resulting in temporal filtering. If the region is moving, pixel C is likely to be very different than A or B, and thus, A and B will typically have a higher correlation with each other than with pixel C. The median filter will then select either pixel A or pixel B resulting in spatial filtering. A number of variations on this median filter have been suggested. For instance, filters with more taps have been used such as the seven-tap filters suggested by Salo *et al* [21] or Lee *et al* [15].

For this research two median filters have been implemented; the three-tap median filter as presented in (9), and a modification of the seven-tap weighted median filter presented by Lee *et* al [15]. In their weighted median filter, the following median operation is implemented.

Let

$$\begin{aligned} A &= F_i[x,y-1,n] & E &= \frac{A+B}{2} \\ B &= F_i[x,y+1,n] & F &= \frac{C+D}{2} \\ C &= F_i[x,y,n-1] & & \\ D &= F_i[x,y,n+1] & & \end{aligned}$$

$$F_o[x,y,n] = \begin{cases} F_i[x,y,n] \ , & \operatorname{mod}(y,2) = \operatorname{mod}(n,2) \\ \operatorname{median}(A,B,C,D,E,E,F) \ , & \text{otherwise} \end{cases} \qquad (10)$$

Again, this type of median filter will be robust to motion and will be likely to select a spatial deinterlacing method in the presence of motion. However, in the presence of motion, linear interpolation (pixel E) becomes somewhat more likely than line repetition (pixels A and B) since pixel E is given twice as much weight in the median filter.

4.2.4 Motion Compensated Deinterlacing

The most advanced techniques for deinterlacing generally make use of motion estimation and compensation. There are many motion estimation algorithms that can be used to estimate the motion of individual pixels between one field and then next. If accurate motion vectors are determined, motion compensation (MC) can basically remove the motion from a video sequence. If motion were perfectly removed, temporal interpolation could theoretically restore the original frames. However, motion vectors are often incorrect and any motion compensated deinterlacing methods have to be robust enough to account for this. In general any non-MC algorithm can be converted to a MC algorithm. However, only those techniques that perform better on stationary images will most likely be improved. Some straightforward MC methods include field repetition, field interpolation, and VT median filtering.

The most common method for determining motion vectors is through block matching algorithms. The image is segmented into blocks, for instance, 8x8 blocks of pixels are fairly common, and then each of those blocks is compared to the previous frame in a given search area, such as –15/+16 pixels in both the horizontal and vertical directions. The block within the search area, which most closely "matches" the original block, is selected to compute the motion vector. The "matching" criterion that is typically used is either mean square error or mean absolute error. There are a number of variations on this algorithm that involve different search regions or different criteria for selecting the most closely matching block. Other algorithms have been used which use phase correlations of adjacent frames to determine shifts of objects within the frames. The two-dimensional Fourier transforms for the two fields are calculated and a phase correlation surface is determined. From this surface it is possible to locate peaks corresponding to shifts of objects from one frame to the next, and map these shift back onto the original field.

A number of problems can arise when using this type of motion estimation scheme. For instance when objects pass in front of one another, there are appearing and disappearing regions of the fields, which cannot be accounted for with motion vectors. Also, if an object moves beyond the motion vector search area, the motion vector will not be accurately determined. Another fundamental problem with motion compensation is that the pixels may have sub-pixel motion. In this case, the motion vector actually should point between pixels or at missing lines in the previous field. In the horizontal direction, the video is assumed to be sampled at a high enough rate that the general sampling theorem can be used to correctly interpolate the needed value. However, in the vertical direction, the data is sub-sampled and band limited interpolation will not provide an appropriate result.

In the general deinterlacing problem, motion compensation can be used to compensate for translational motion of static objects, meaning objects that shift from one position to another without deforming or changing shape. Thus, motion compensation can be very effective when deinterlacing a video sequence acquired from a panning camera. However, this type of motion seldom occurs in surveillance situations that use a stationary camera. The objects that are static, such as walls and doors, are not moving because the camera is not moving. The objects that do move are not static; they are typically individuals who change shape quite often. This type of motion is not translational, and is thus beyond the scope of the two-dimensional translational model. In this type of situation, accurate motion vectors cannot be found, since they do not exist. This problem is compounded by the low temporal rates used in these sequences, which decreases temporal correlation between adjacent frames.

This thesis looks at three different methods that use motion compensation: field repetition, field interpolation, and a three-tap VT median filter. Half-pixel motion estimation algorithms were used with 8x8 blocks and a search range of $-31/+32$ pixels. The large search range was selected because of the large amounts of motion that are possible between successive frames due to the low temporal recording rates.

4.2.5 Motion Adaptive Deinterlacing

Interlaced sequences can be essentially perfectly reconstructed by interframe methods in the absence of motion, while intraframe methods perform well in the presence of motion [25]. Motion adaptive methods use a motion detector to take advantage of this fact by separating each field into moving and stationary regions. Based on the output of the motion detector, the deinterlacing algorithm then fades between a temporal method that works well for stationary images and a spatial method for moving images. The output of this type of motion adaptive algorithm can be represented as in (11) where α represents a motion detection parameter that indicates the likelihood of motion for a given pixel

$$F_o[x,y,n] = \begin{cases} F_i[x,y,n], & \text{mod}(y,2) = \text{mod}(n,2) \\ \alpha \cdot F_{mot}[x,y,n] + (1-\alpha) \cdot F_{stat}[x,y,n], & \text{otherwise} \end{cases} \tag{11}$$

The pixels $F_{stat}(x,y,t)$ are calculated for stationary regions using a temporal method, and the pixels $F_{mot}(x,y,t)$ are calculated for moving regions using a spatial method.

Preliminary results indicated that motion adaptive methods had the most potential for use with low frame rate surveillance video. Motion adaptive methods correspond well with surveillance video. For example, the majority of these sequences consist of stationary background type regions due to the stationary camera system. These background areas can be deinterlaced using field repetition or interpolation with very high accuracy. Another benefit of motion adaptive methods is that their performance is not significantly degraded when used on sequences with low frame rates. For these reasons, motion adaptive methods were given the most attention during this study. The details of motion detection and motion adaptive methods are examined in detail in the following chapter.

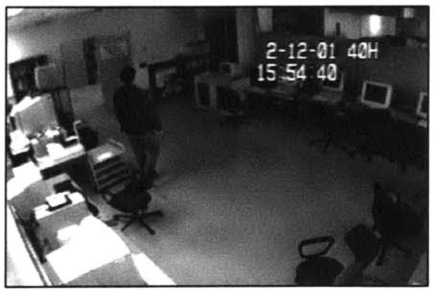
Chapter 5
Motion Adaptive Deinterlacing

Motion adaptive deinterlacing methods are designed to take advantage of the varying information that is present in different regions of a video sequence. In stationary images, there exists significant correlation between the present field and the adjacent fields. Temporal methods such as field repetition are designed to benefit from this fact. In moving regions, this temporal correlation does not exist, and more information can be found in adjacent pixels on the current field. Spatial methods take advantage of this fact. Motion adaptive deinterlacing attempts to combine the benefits of both temporal and spatial methods by segmenting each image into moving and stationary regions. Stationary regions are reconstructed with a temporal method while moving regions are reconstructed with a spatial method.

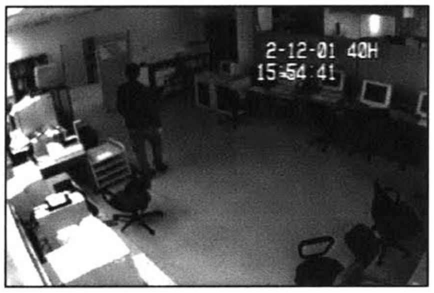
In order to segment a video sequence into moving and stationary regions, motion adaptive methods must use some form of motion detection. The next section covers some of the issues with detecting motion in interlaced sequences and describes the different algorithms that were implemented in this study.

5.1 Motion Detection in Interlaced Sequences

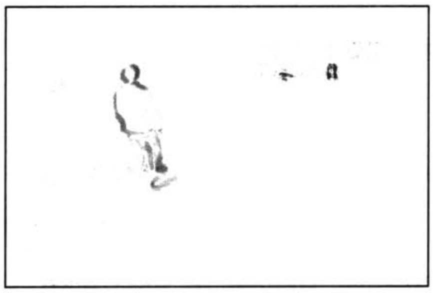
The goal of motion detection is to detect changes in a video sequence from one field to the next. If a significant change is detected, that region declared to be moving. If no significant changes are detected the region is assumed to be stationary. This task is straightforward for a progressive sequence. A simple frame difference can be used to detect changes and a threshold can be applied to indicate the presence or absence of motion. An example of this idea is shown in Figure 5-1.



(a)



(b)



(c)

Figure 5-1: Example of motion detection using progressive frames. (a) (b) Two successive frames from a video sequence. (c) Result of absolute frame difference. Result of frame differencing can be used as an indication of the presence or absence of motion.

Frame differencing cannot be used for interlaced sequences. In fact, it is impossible since consecutive fields do not have the same pixel locations, i.e. an even field cannot be subtracted from an odd field since the corresponding pixels do not exist. The following sections discuss some of the proposed solutions for motion detection in interlaced sequences.

5.1.1 Two-Field Motion Detection

The simplest approach to interlaced motion detection is to convert the fields to frames, using any deinterlacing method. Once the fields have been deinterlaced, motion detection is straightforward, and can be performed as in the progressive case discussed above [17]. Any spatial deinterlacing method can be used for this purpose, such as line repetition, linear interpolation, or the Martinez-Lim algorithm. Figure 5-2 illustrates this idea.

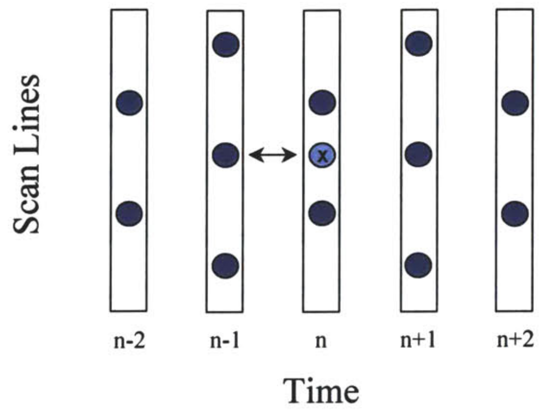


Figure 5-2: Two-field motion detection scheme for determining the presence or absence of motion at pixel location 'X'. The dark circles represent pixels in a video sequence on different scan lines at the same horizontal position. The pixel at location 'X' is interpolated, represented in the picture by the light circle, and then an absolute difference is computed between the interpolated pixel and the pixel from the previous field.

This figure depicts five separate fields of a video sequence. The dark circles symbolize pixels on varying scan lines at the same horizontal position. The alternating odd and even fields can be seen in this figure. Suppose the location 'X' is the pixel for which motion is being estimated. In two-field motion detection the pixel at location 'X' is interpolated spatially, which is represented here by the light-colored circle. Then an absolute difference is computed between the interpolated pixel and the pixel from the previous field. This pixel difference is represented in this figure by the double arrow symbol.

A more theoretical approach to this problem is presented by Li, Zeng, and Liou [17]. Since the difference between an even sampled field and an odd sampled field is a phase shift, they suggest applying a phase correction filter to one field so that it may be more appropriately compared with the opposite polarity field for motion detection. They use the following 6-tap filter for phase correction of the current frame:

$$F_o[x,y,n] = \begin{cases} F_i[x,y,n], & \text{mod}(y,2) = \text{mod}(n,2) \\ 3 \cdot F_i[x,y,n \pm 5] - 21 \cdot F_i[x,y,n \pm 3] + 147 \cdot F_i[x,y,n \pm 1], & \text{otherwise} \end{cases} \tag{12}$$

Obviously, their method does not result in perfect reconstruction, since it is unable to remove the aliasing terms. However, this method does provide an interesting theoretical solution to the problem through an application of sampling theory. Their method also showed the best performance over the other spatial deinterlacing methods for use in this type of two-field motion detection and was used for two-field motion detection in this study.

The problem that arises with two-field motion detection is that, without perfect reconstruction, there will be some errors made in the interpolation step. These errors can make vertical details in the picture appear as motion. For instance, the pixels above and below pixel 'X' in Figure 5-2 do not necessarily have any correlation to the actual value of pixel 'X'. Thus, the interpolation may be done incorrectly and the absolute difference will not be an appropriate measure of motion. Thus, motion may be detected when none was present. This error leads to a significant number of false-detections. As a result, more regions of the video sequence will be deinterlaced spatially than necessary, resulting in lower quality.

5.1.2 Three-Field Motion Detection

The only way to avoid the problems introduced by two-field motion detection is to only compare pixels on identical lines. For instance, no interpolation should be done in order to compare two pixels that are not on corresponding lines. With this in mind, the simplest method of motion detection is a three-field motion detection scheme suggested in [15] and presented in Figure 5-3.

This figure is similar to Figure 5-2, except that no interpolation is needed, and so the light colored pixel is omitted. Assume that motion is being detected for the location marked by the symbol 'X'. In a three-field motion detection scheme, the pixel in the previous field is compared to the same pixel in the next field. Since the previous and the next field have the same spatial location, the absolute difference can be computed without any interpolation. An absolute difference of these two pixels is computed as an indication of the likelihood of motion.

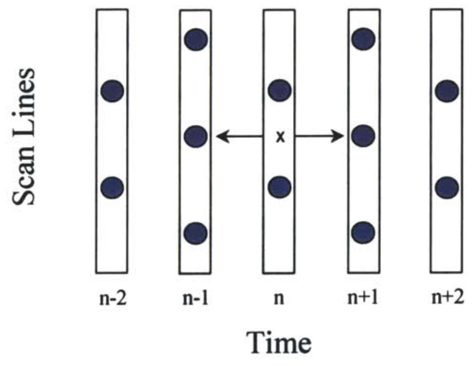


Figure 5-3: Three-field motion detection scheme for determining the presence or absence of motion at pixel location 'X'. An absolute difference is computed between the pixel in the previous frame and the pixel in the next frame. Only pixels on corresponding scan lines are compared, so no interpolation is needed.

While two-field motion detection results in many false alarms (detecting motion when none is present), three-field motion detection results in many missed detections (not detecting motion when motion actually is present) because it is unable to detect fast moving regions. The following example illustrates an example of a missed detection when utilizing three-field motion detection.

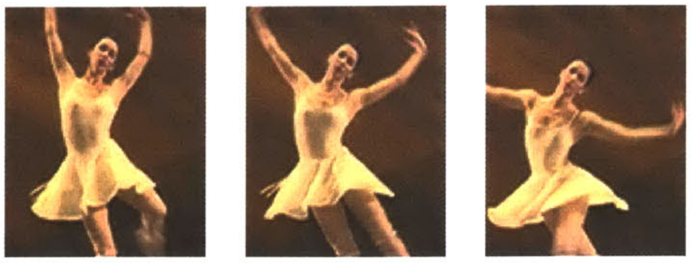


Figure 5-4: Sequence of frames which will cause three-field motion detection to fail.

Figure 5-4 presents three sequential frames of a video sequence. In Figure 5-4 (b), consider the ballerina's left arm. In this example, the three-field motion detection scheme would incorrectly label this area as stationary. In this region, images (a) and (c) look exactly alike. This motion detection algorithm will compare (a) with (c), assume that nothing has changed in that particular region, and label it as stationary. The result of deinterlacing using these erroneous motion detection parameters can be found in Figure 5-5.

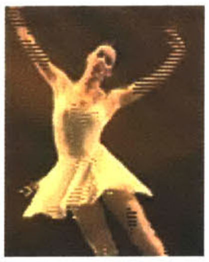


Figure 5-5: Results of motion adaptive deinterlacing when errors are made in motion detection

The artifacts caused by missed detection are similar to those caused by field repetition. The dancer's left arm is blurred out, leaving only a ghostly outline. The same artifact occurs in other regions of this image for identical reasons.

5.1.3 Four-Field Motion Detection

In an attempt to improve upon three-field motion detection by reducing the number of missed detections, one can use a four-field motion detection algorithm [15]. The idea behind four-field motion detection is presented in Figure 5-6.

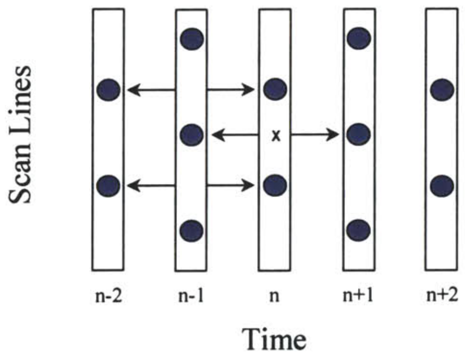


Figure 5-6: Four-field motion detection scheme for determining the presence or absence of motion at pixel location 'X'. Three absolute differences are computed. This results in a better detection of fast motion. Only pixels on corresponding scan lines are compared, so no interpolation is needed.

In this scheme, three pixel comparisons are used rather than the one set of pixels that are compared in the previous two algorithms. The additional two pixel differences help to protect against the error that was demonstrated in Figures 5-4 and 5-5. Figure 5-7 shows the final result after this error has been corrected.

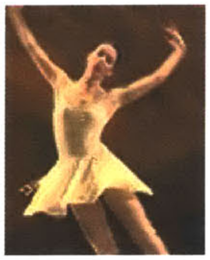


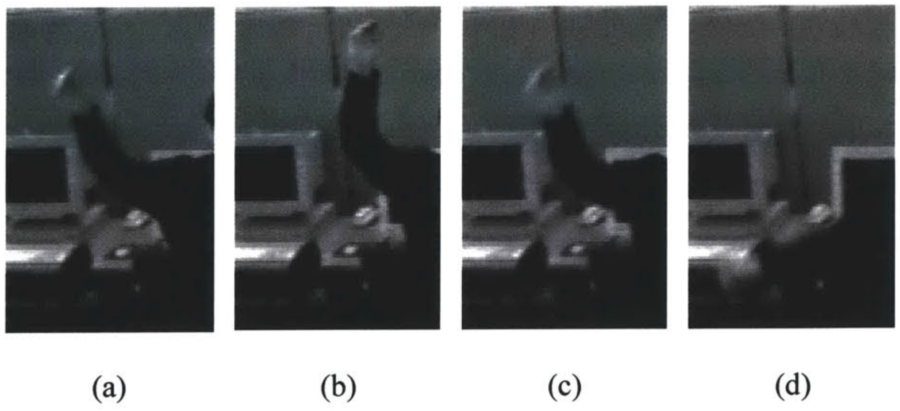
Figure 5-7: Results of motion adaptive deinterlacing using four-field motion detection. Missed detections that were made by the three-field motion detection have been removed.

The output of this motion detector is the maximum of the three pixel differences.

$$A = \left|F_i[x,y,n-1] - F_i[x,y,n+1]\right|$$
$$B = \left|F_i[x,y-1,n] - F_i[x,y-1,n-2]\right|$$
$$C = \left|F_i[x,y+1,n] - F_i[x,y+1,n-2]\right|$$

$$\text{motion\_detected} = \max(A,B,C) \tag{13}$$

While this method has better performance than three-field motion detection, it can still miss detections. The occurrences are far less likely, but they can occur. The following sequence of sequential frames demonstrates an example where four-field motion detection claims a region is stationary when motion is actually present.



(a) (b) (c) (d)

Figure 5-8: Sequence of frames which will cause four-field motion detection to fail.

In this sequence, consider the position of the waving hand in frame (c). In frame (a) the hand is also approximately in the same position. The two frames are not identical, but at least some regions of the hand and arm are similar. The four field motion detection will compare frames (a) and (c) and will assume that nothing has changed for the majority of the pixels. Similarly, the algorithm will perform a comparison of frames (b) and (d). While these frames (b) and (d) are clearly very different, they are nearly identical in the area in question, so the algorithm will assume nothing has changed in that region either. The end result will be that no motion will be detected in this area, while frame (c) and frame (b) are quite different. The effects of this type error are similar to that shown in Figure 5-5.

5.1.4 Five-Field Motion Detection

The final motion detection algorithm that was investigated is a five-field based method that was suggested by the Grand Alliance, developers of the United States HDTV standard, for HDTV format conversion [2]. This method improves upon four-field motion detection by reducing the number of missed motion detections. The method they propose is illustrated in Figure 5-9.

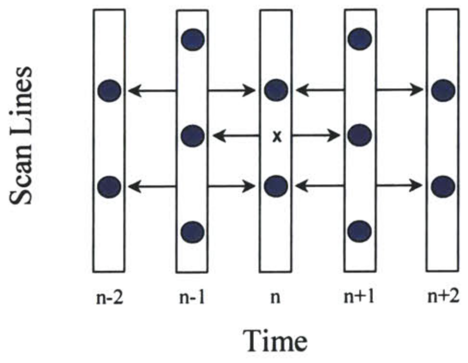


Figure 5-9: Five-field motion detection scheme for determining the presence or absence of motion at pixel location 'X'. A weighted combination of five absolute differences is used.

Let

$$\begin{aligned}
A&=\left|F_i[x,y,n-1]-F_i[x,y,n+1]\right|\\
B&=\left|F_i[x,y-1,n]-F_i[x,y-1,n-2]\right|\\
C&=\left|F_i[x,y+1,n]-F_i[x,y+1,n-2]\right|\\
D&=\left|F_i[x,y-1,n]-F_i[x,y-1,n+2]\right|\\
E&=\left|F_i[x,y+1,n]-F_i[x,y+1,n+2]\right|
\end{aligned}$$

The output of this motion detector is then the following combination of these five pixel differences.

$$\text{motion\_detected} = \max\left(A, \frac{B+C}{2}, \frac{D+E}{2}\right) \tag{14}$$

In this maximum operation, the two pixel differences between the fields at time *n* and *n-2* are averaged, and the two pixel differences between the fields at *n* and *n+2* are averaged.

This method also allows for one additional improvement. The previous methods, with the exception of the three-field motion detector, look for motion in one direction in time. In particular, they are all orientated to find differences between the current frame and the previous frames. Thus these methods will all use the previous field for temporal deinterlacing. However, the five-field method, in some sense, looks for motion in both the forward and backward directions. Thus it is not immediately clear whether the motion adaptive deinterlacing algorithm should use the previous field or the next field for temporal interpolation. In this case, a type of median filter is used for temporal deinterlacing, as suggested by the Grand Alliance, the following median operation is used for temporal deinterlacing [2].

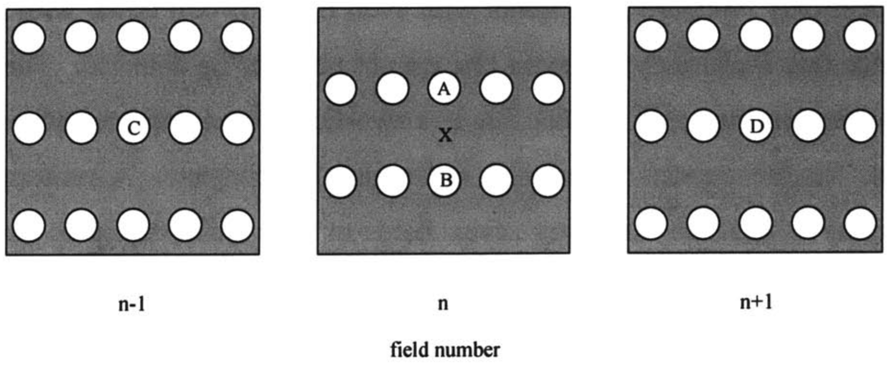


Figure 5-10: Pixel definitions for modified VT-median filter used in five-field motion detection algorithm

$$X = \operatorname{med}\left(\frac{A+B}{2}, C, D\right) \tag{15}$$

$$F_o[x,y,n] = \begin{cases} F_i[x,y,n] \ , & \operatorname{mod}(y,2) = \operatorname{mod}(n,2) \\ \operatorname{median}\left(\dfrac{F_i[x,y+1,n]+F_i[x,y-1,n]}{2}, F_i[x,y,n+1], F_i[x,y,n-1]\right) \ , & \text{else} \end{cases} \tag{16}$$

This operation calculates the median of the pixel in the previous frame, the pixel in the next frame, and the linear interpolation of the adjacent pixels. In some sense, it repeats either the pixel in the previous frame or the pixel in the future frame depending on which pixel is closest to the value generated by spatial linear interpolation.

While this method does perform the best of all the methods studied it is still possible for a combination of frames to cause the motion detection algorithm to miss areas of motion.

This algorithm will not detect a sequence of fields with some region that flickers between two images once every field over a five-field duration. These types of motion detection algorithms can only compare even fields with even fields or odd fields with odd fields. Any sequence that flickers at the frame rate cannot possibly be detected. The only type of motion detection that could detect this is a two-field based scheme, which has poor performance for the reasons mentioned earlier in this chapter. A motion detection scheme similar to this one but using seven fields or nine fields could be used to help reduce this problem, but eventually it becomes unreasonable to look so far into the future or past. Using five fields was determined to be sufficient in reducing missed detections. Some errors occasionally occur, but these are rare and require very fast, periodic motion. At the rates typically used in surveillance video this type of error rarely occurred and it is more unlikely at higher frame rates.

5.2 Motion Adaptive Deinterlacing

Using a motion detection algorithm, the benefits of both spatial and temporal methods can be combined. The idea is to switch or fade between a spatial method and a temporal method based on the presence or absence of motion as indicated by the motion detector [5]. Equation (17) and Figure 5-11 depict this central idea.

$$F_o[x,y,n]=\begin{cases} F_i[x,y,n], & \mathrm{mod}(y,2)=\mathrm{mod}(n,2) \\ \alpha\cdot F_{mot}[x,y,n]+(1-\alpha)\cdot F_{stat}[x,y,n], & \text{otherwise} \end{cases} \tag{17}$$

The pixels $F_{stat}(x,y,t)$ are calculated for stationary regions using a temporal method, and the pixels $F_{mot}(x,y,t)$ are calculated for moving regions using a spatial filter. The motion parameter α must be a real number between 0 and 1. However, the motion detector outputs a pixel difference with values between 0 and 255. This conversion, symbolized by the "switch" in Figure 5-11 was implemented with the transfer function depicted in Figure 5-12.

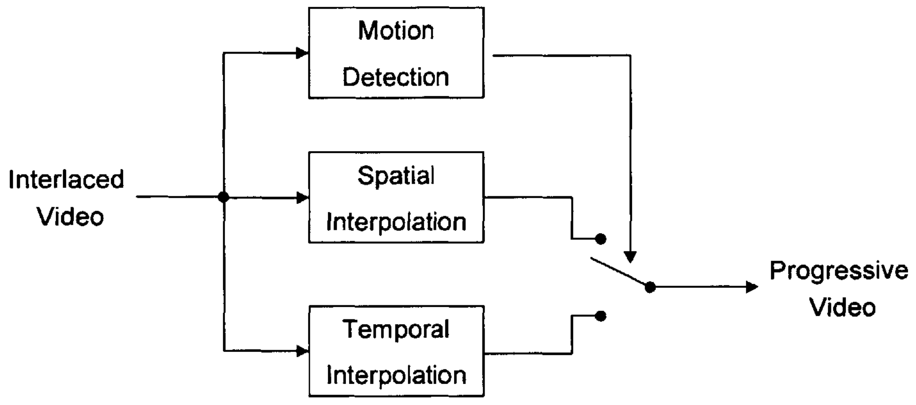


Figure 5-11: General concept behind motion adaptive deinterlacing. Motion detection is used to switch or fade between the use of spatial deinterlacing and temporal deinterlacing.

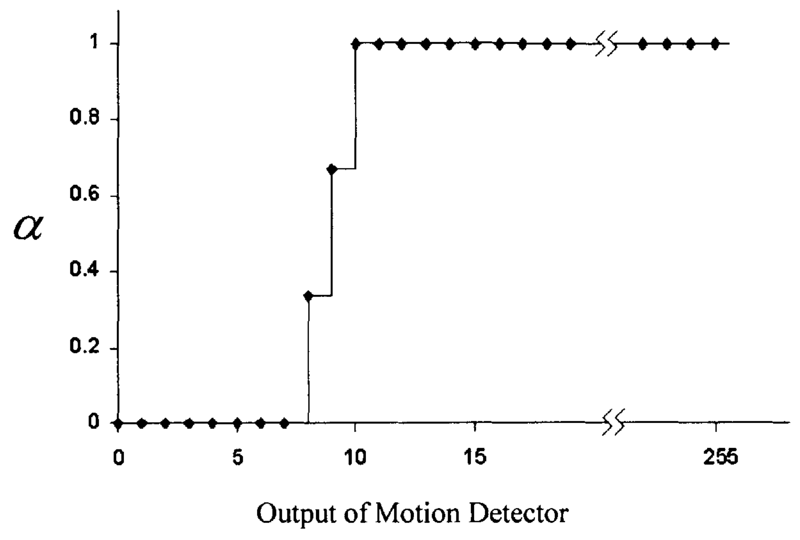


Figure 5-12: Transfer function between the output of motion detection and the motion parameter α

Some initial tests indicated that a pixel intensity difference greater than 10 resulted in noticeable artifacts in the deinterlaced output, so the switch between stationary and moving regions was chosen just under this point.

In this thesis, the Martinez-Lim algorithm [18] was used for the spatial deinterlacing and field repetition was used for temporal deinterlacing. The only exception is with the five-field motion detection algorithm as was mentioned in the previous section.

The security video that was looked at in this thesis contained a significant amount of noise due to VHS tape degradations. This type of noise can cause the motion detector to indicate motion in areas that should not be considered as such. To help reduce this effect, a low-pass filter was used before motion detection. This adjustment led to significant improvements in video quality. The low-passed video was not used for deinterlacing, only for motion detection. The final implementation of the motion adaptive deinterlacing algorithms used in this thesis is depicted in Figure 5-13.

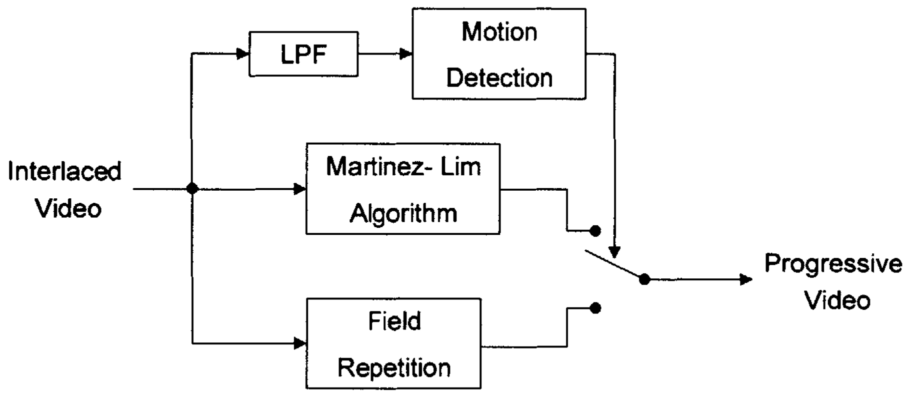


Figure 5-13: Motion adaptive deinterlacing algorithm

Chapter 6

Experiments and Results

6.1 Experimental Setup

The experiments conducted in this thesis compare the performance of fourteen different deinterlacing algorithms when applied to surveillance camera video sequences. The list of algorithms that were used is shown in Table 6-1.

This thesis uses Peak Signal-to-Noise Ratio (PSNR) of the luminance channel as a quantitative measure of the video quality. PSNR is defined as

$$PSNR = 10\log_{10}\left(\frac{255^2}{MSE}\right)$$

and the Mean Square Error (MSE) is defined to be the average squared difference between the original and the resulting video.

$$MSE = \tfrac{1}{M \cdot N}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left(F_o[x,y,n] - F[x,y,n]\right)^2$$

It should be noted that PSNR and perceived quality are not always directly correlated. Higher PSNR does not always indicate superior video, but the use of PSNR is a common practice and has been found to be a sufficient measure of video quality.

Table 6-1: Summary of deinterlacing algorithms used for testing

Algorithm	Algorithm Name
	Intraframe Methods
1	Line Repetition
2	Linear Interpolation
3	Martinez Lim Algorithm
	Interframe Methods
4	Field Repetition
5	Bilinear Field Interpolation
	Median Filtering
6	VT Median Filtering
7	Weighted VT Median Filtering
	Motion Compensated Methods
8	Motion Compensated Field Repetition
9	Motion Compensated Field Interpolation
10	Motion Compensated VT Median Filtering
	Motion Adaptive Methods
11	Two Field Motion Detection
12	Three Field Motion Detection
13	Four Field Motion Detection
14	Five Field Motion Detection

This thesis has studied three separate sets of video data. The first set was made up of two MPEG test sequences that had properties similar to those of surveillance video sequences. In particular, both sequences use a stationary camera system and contain moving individuals. This data was temporally downsampled to simulate the low frame rates found in surveillance systems. These two sequences were used primarily for preliminary testing of deinterlacing algorithms. Table 6-2 summarizes the properties of this set of data. These sequences were originally progressively scanned. Thus, they were interlaced so that the deinterlacing algorithms could be applied. The results of deinterlacing these sequences were then compared to the original progressive source.

Table 6-2: Summary of MPEG test sequences used for algorithm testing

	Sequence Name: MPEG Test Sequence News CIF **Scan Mode:** Progressive **Fields:** 50 **Rows:** 288 **Cols:** 352 **Frame Rate:** 10 frames / sec
	Sequence Name: MPEG Test Sequence Silent Voice CIF **Scan Mode:** Progressive **Fields:** 50 **Rows:** 288 **Cols:** 352 **Frame Rate:** 10 frames / sec

The second set of data was made up of five sequences taken from an actual surveillance system. The system used for this purpose contained a Panasonic WV-BP330 series black-and-white security camera and a Panasonic AG-RT600A time-lapse recorder. All video sequences were recorded at 12 fields/sec on a professional quality VHS tape made especially for time-lapse recording. This video was then played off the VHS tape and recorded digitally using a Pinnacle Systems DV500 digital video editing system. The edges of the video were cropped slightly to remove some edge artifacts caused by the time-lapse recorder and the analog to digital converter resulting in interlaced video of size 464 rows by 704 columns. The digital video obtained through this process is believed to imitate an actual security system as accurately as possible.

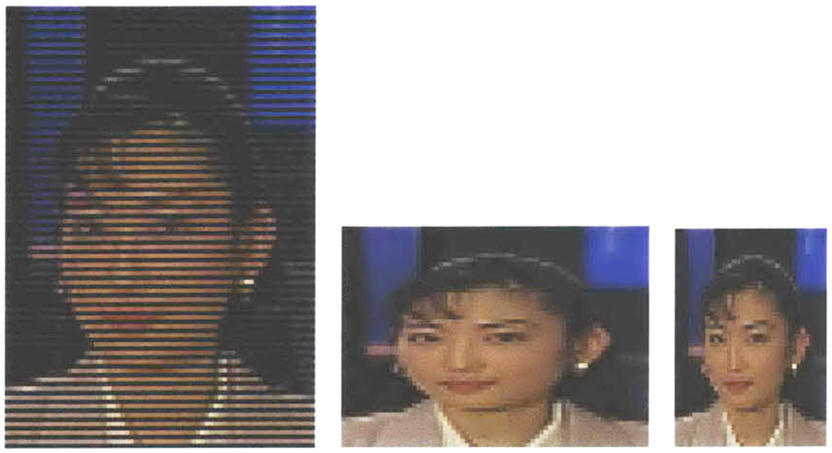


Figure 6-1: (a) Original interlaced field. (b) Discarding every other field downsamples the sequence by two in the vertical and temporal direction. (c) Downsampling in the horizontal direction generates a new progressive frame with the same proportions as the original field.

The full resolution video was used for some informal qualitative tests to verify the results suggested by quantitative experiments. However, no quantitative performance evaluations could be performed on the full resolution video since an original progressive source does not exist. To overcome this, the full resolution video was sub-sampled in all dimensions to generate smaller sequences that use progressive scanning. This sub-sampling process is demonstrated through the example in Figure 6-1.

This procedure yielded a set of progressive sequences, which could then be interlaced and used in quantitative experiments. Table 6-3 summaries the properties of the video sequences in this set of data.

Table 6-3: Summary of surveillance footage sequences used for algorithm comparison

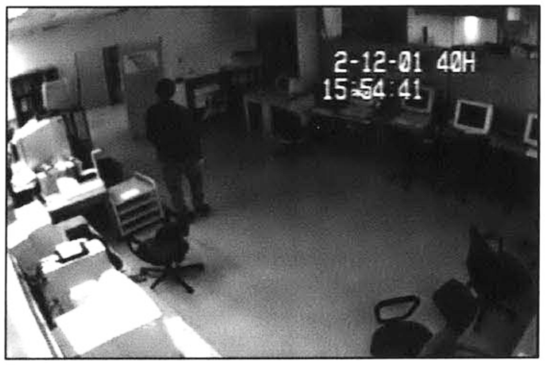
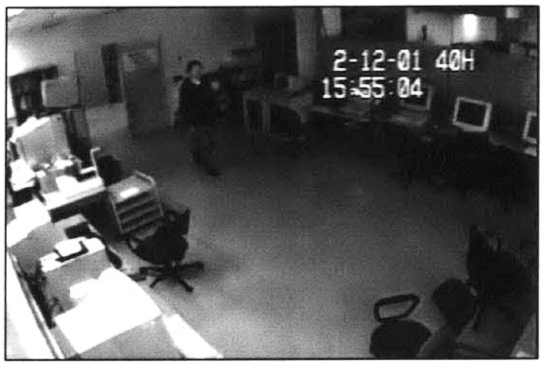
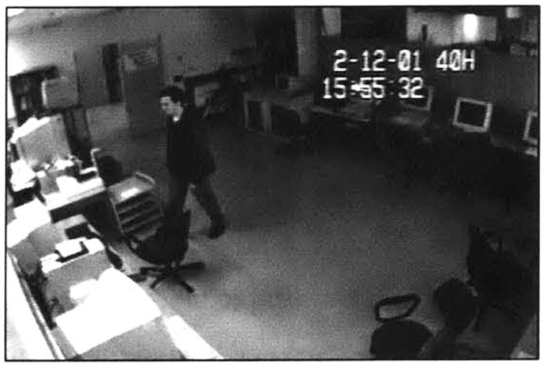
	**Sequence Name:** Surveillance Footage ATRP Lab Sequence A **Scan Mode:** Progressive **Fields:** 50 **Rows:** 232 **Cols:** 352 **Frame Rate:** 6 frames / sec
	**Sequence Name:** Surveillance Footage ATRP Lab Sequence B **Scan Mode:** Progressive **Fields:** 50 **Rows:** 232 **Cols:** 352 **Frame Rate:** 6 frames / sec
	**Sequence Name:** Surveillance Footage ATRP Lab Sequence C **Scan Mode:** Progressive **Fields:** 50 **Rows:** 232 **Cols:** 352 **Frame Rate:** 6 frames / sec

Table 6-3 (continued)

	Sequence Name:	Surveillance Footage ATRP Lab Sequence D
	Scan Mode:	Progressive
	Fields:	50
	Rows:	232
	Cols:	352
	Frame Rate:	6 frames / sec

	Sequence Name:	Surveillance Footage ATRP Lab Sequence E
	Scan Mode:	Progressive
	Fields:	50
	Rows:	232
	Cols:	352
	Frame Rate:	6 frames / sec

The final set of sequences was generated from the same surveillance system as above. However, the purpose of this set was to evaluate the performance of each deinterlacing method at various temporal rates. To accomplish this, a video was recorded at 60 fields per second and was then sub-sampled, just as above, to generate a progressive source. This 30 frame per second progressive source was temporally downsampled three times to generate an additional sequence at 10 frames per second, 5 frames per second, and 2.5 frames per second. Then performance comparisons could be performed with these four progressive sources. The details of this sequence are presented in Table 5-4.

Table 5-4: Summary of surveillance footage sequence used for frame rate comparison

	Sequence Name:	Surveillance Footage ATRP Lab Frame Rate Comparison
	Scan Mode:	Progressive
	Fields:	Various (120 / 40 / 20 / 10)
	Rows:	232
	Cols:	352
	Frame Rate:	Various (30 / 10 / 5 / 2.5)

6.2 Results

To compare the fourteen deinterlacing algorithms, each was used to deinterlace the sequences mentioned above. Both qualitative and quantitative comparisons of all the algorithms have been made.

Using the video sequences for which the original progressive source existed or was generated (see Figure 6-1), a number of quantitative comparisons have been made. The PSNR of each frame was computed and the results were averaged over the entire sequence. The results of these calculations can be seen in Figures 6-2 through 6-8.

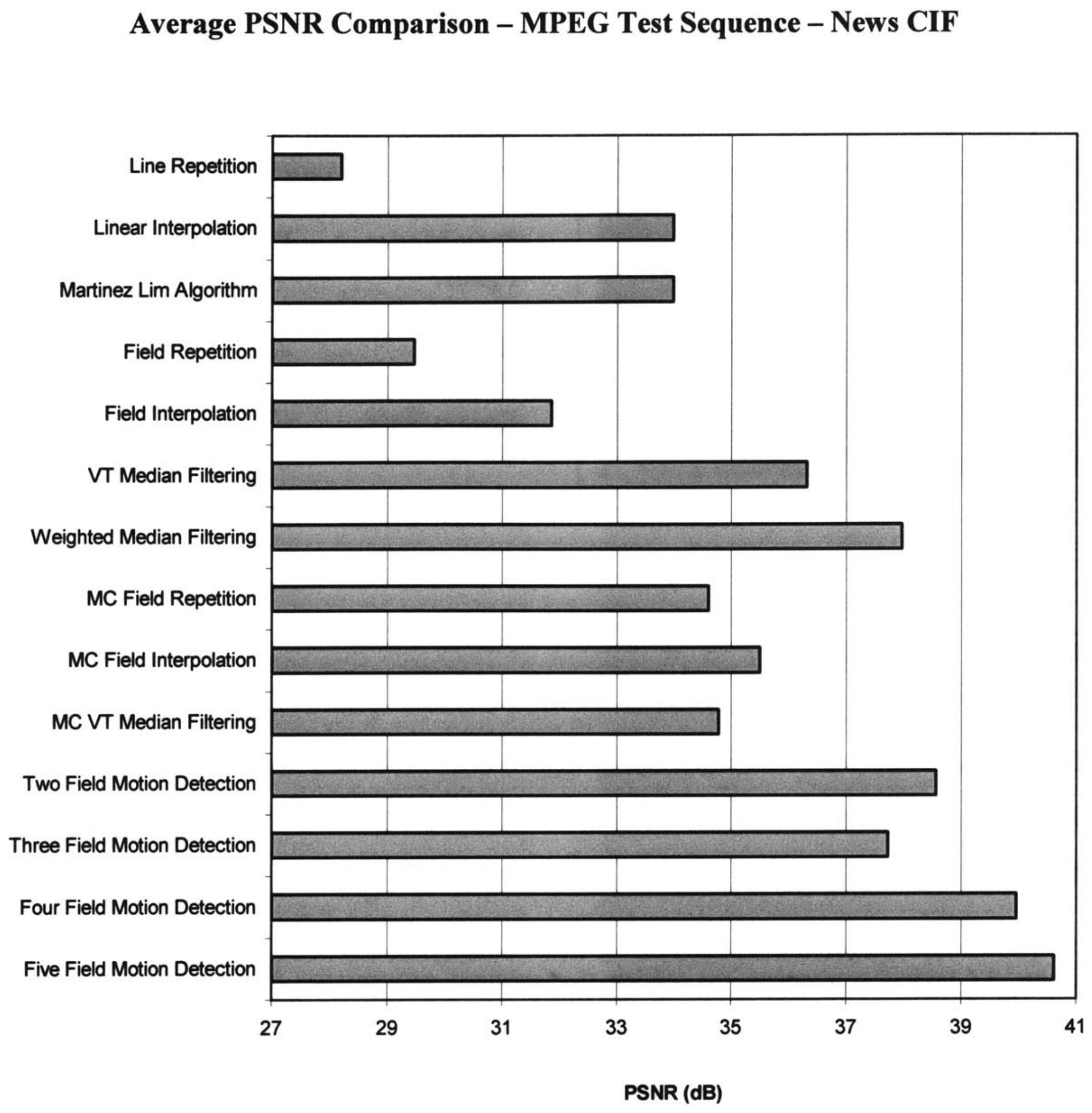


Figure 6-2: Average PSNR comparison.

Average PSNR Comparison – MPEG Test Sequence – Silent Voice CIF

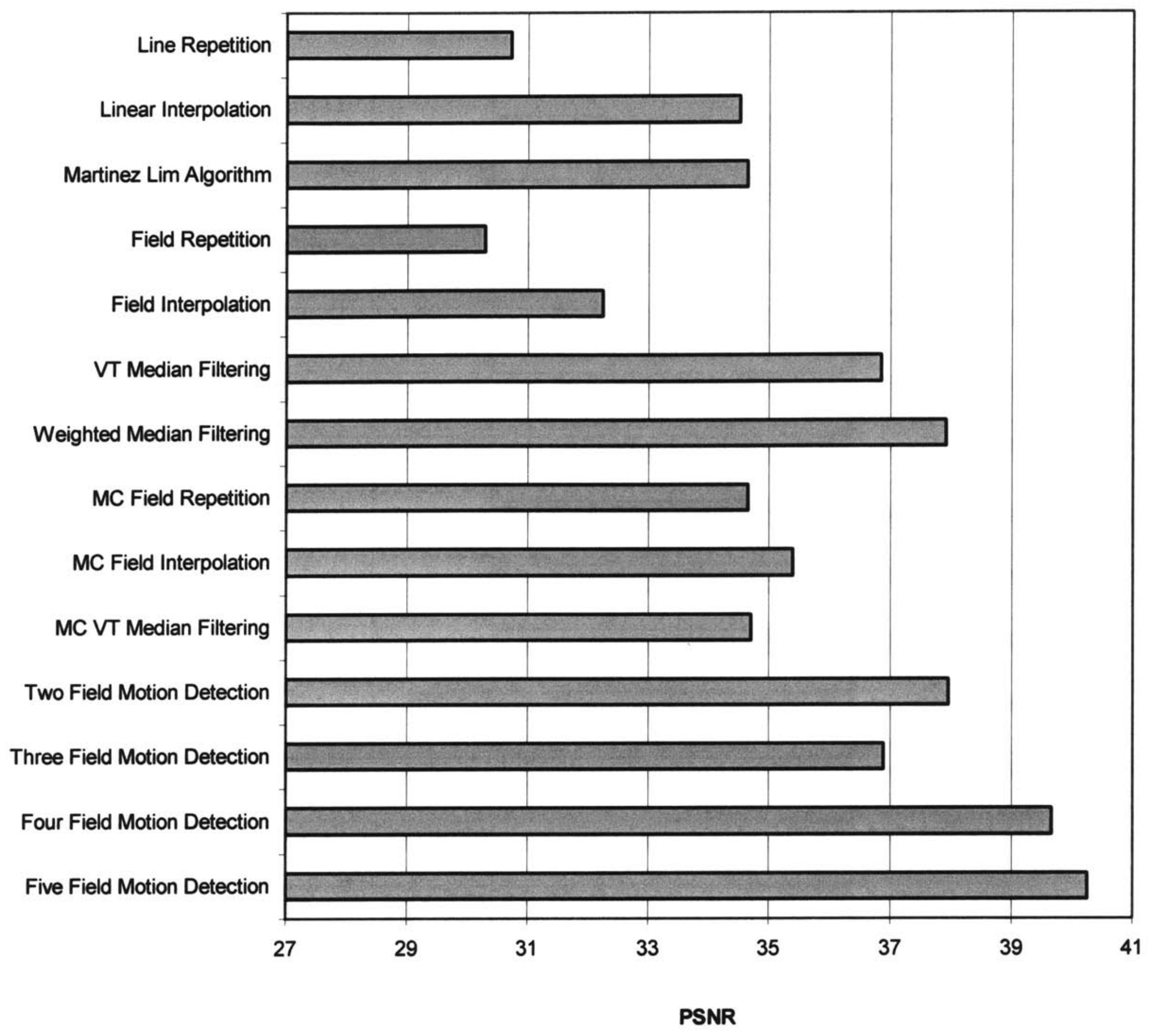


Figure 6-3: Average PSNR comparison.

Average PSNR Comparison – Surveillance Footage
ATRP Lab – Sequence A

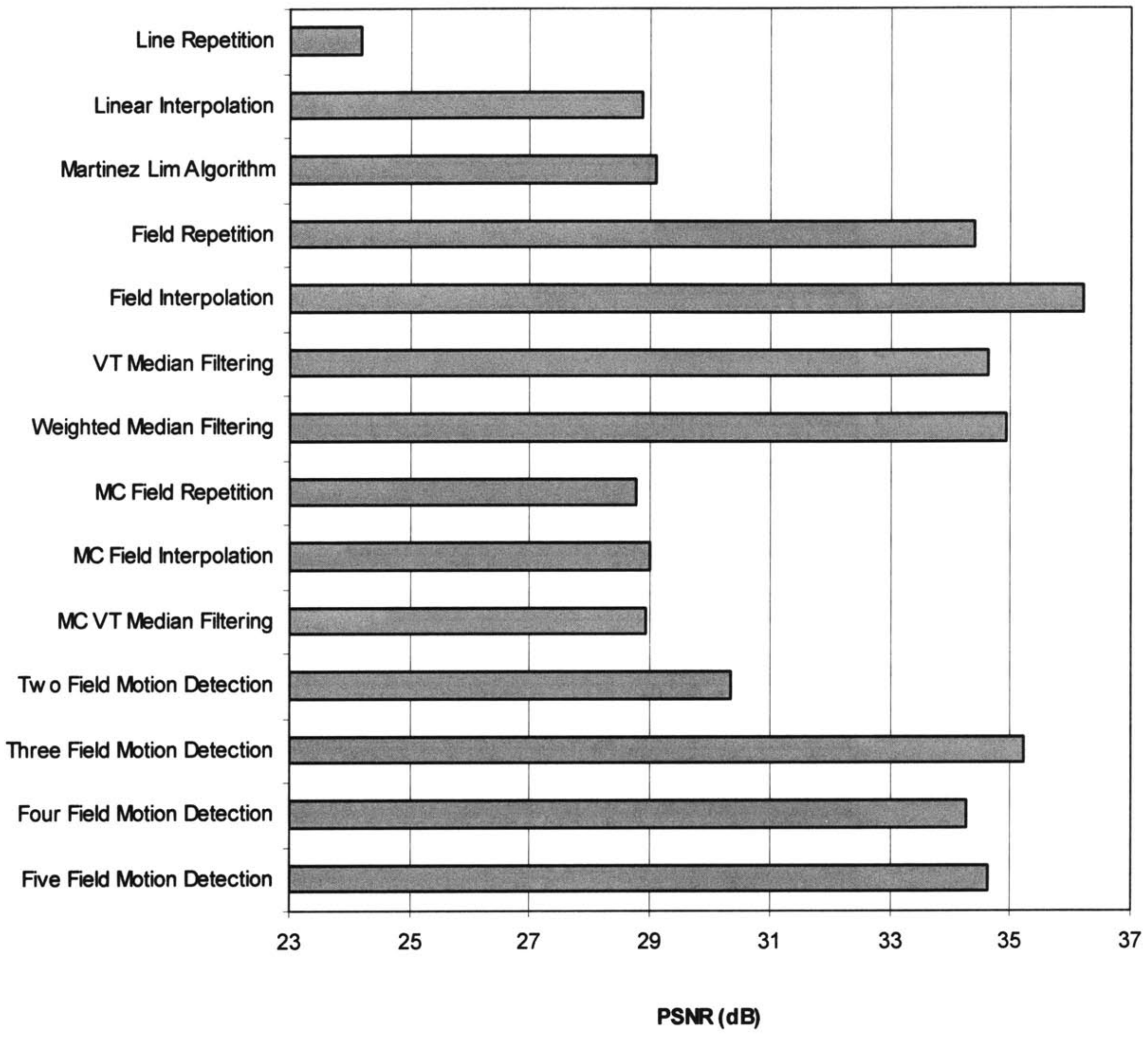


Figure 6-4: Average PSNR comparison.

Average PSNR Comparison – Surveillance Footage
ATRP Lab – Sequence B

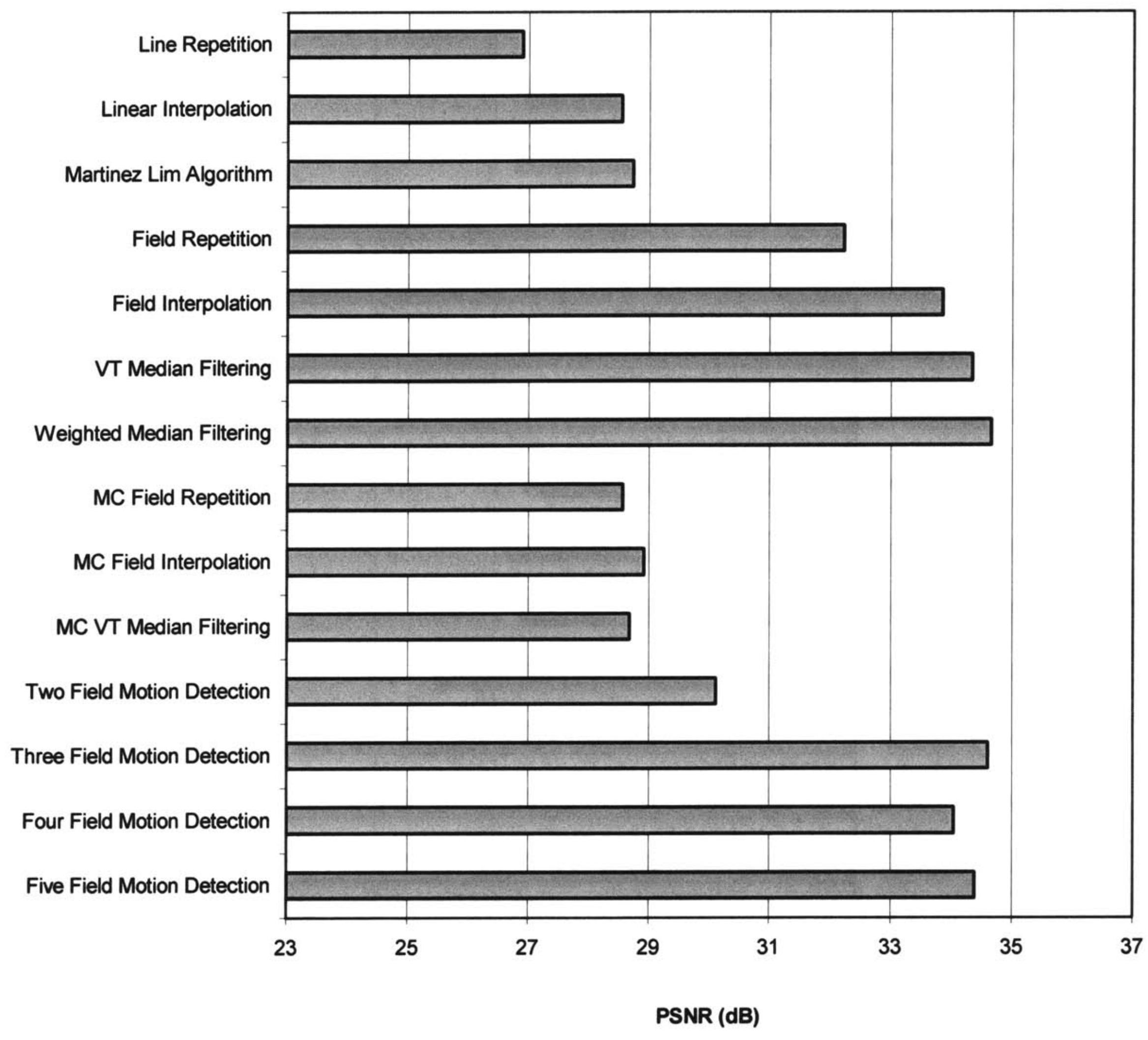


Figure 6-5: Average PSNR Comparison.

Average PSNR Comparison – Surveillance Footage
ATRP Lab – Sequence C

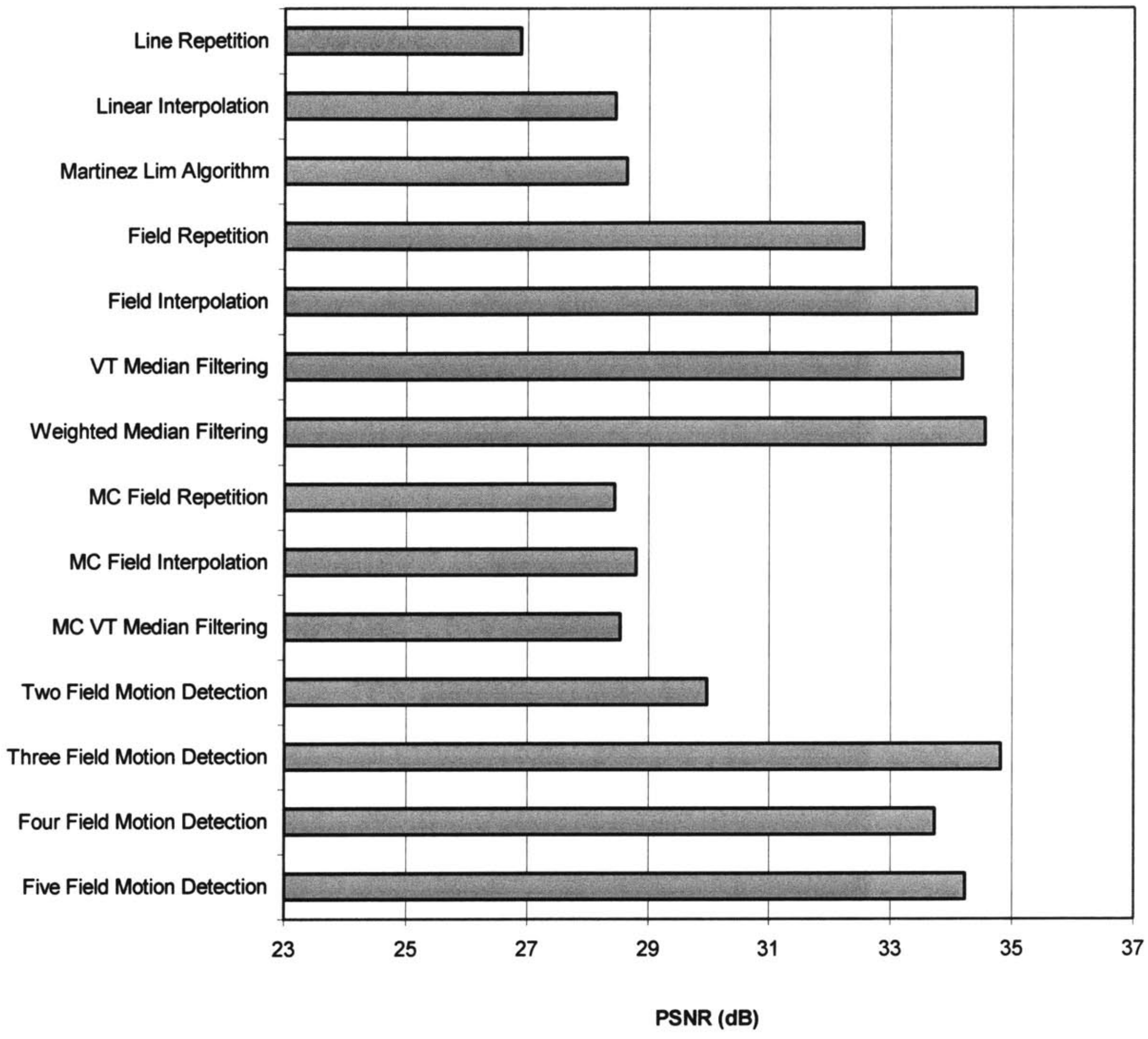


Figure 6-6: Average PSNR comparison

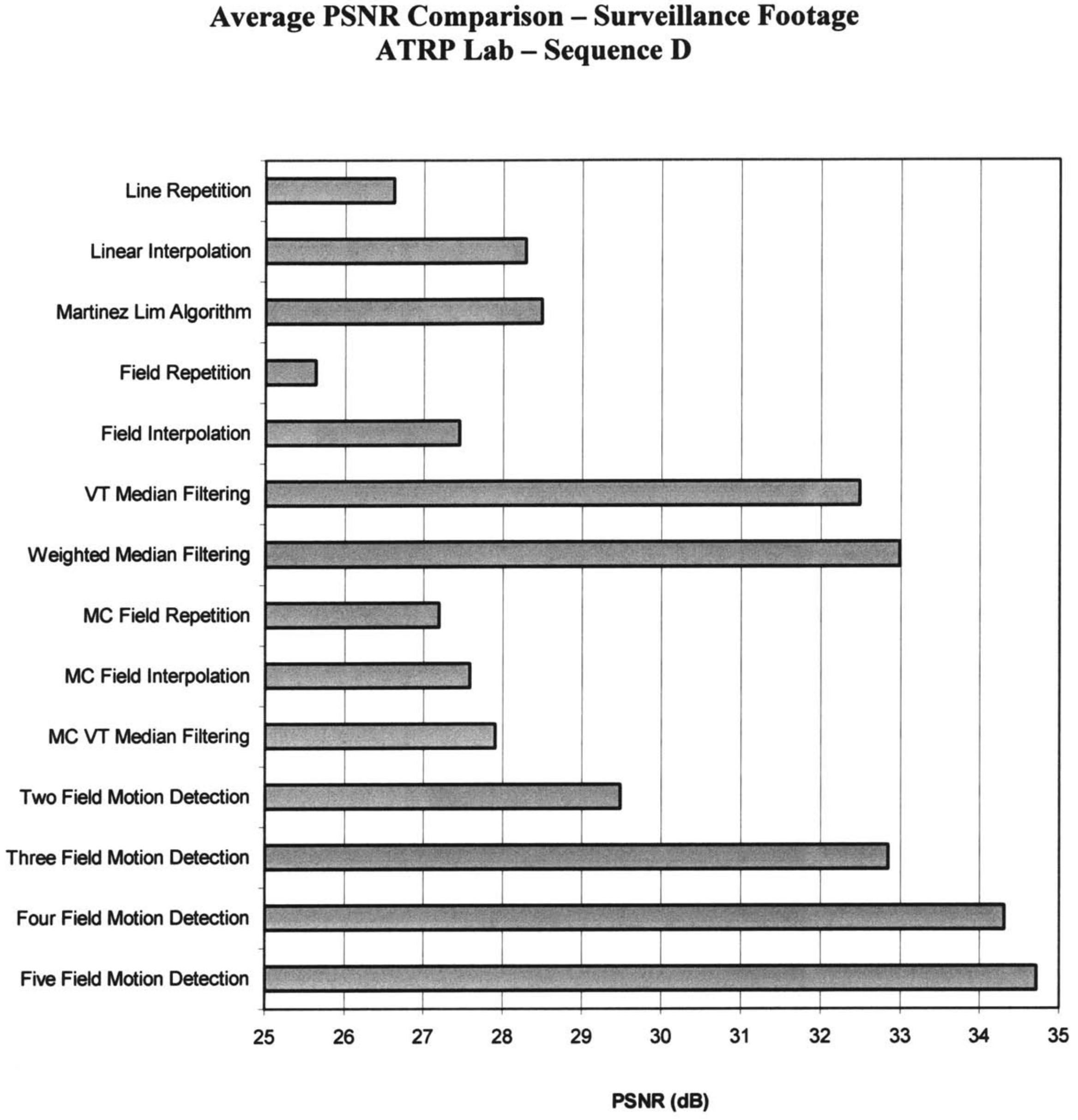


Figure 6-7: Average PSNR Comparison

Average PSNR Comparison – Surveillance Footage
ATRP Lab – Sequence E

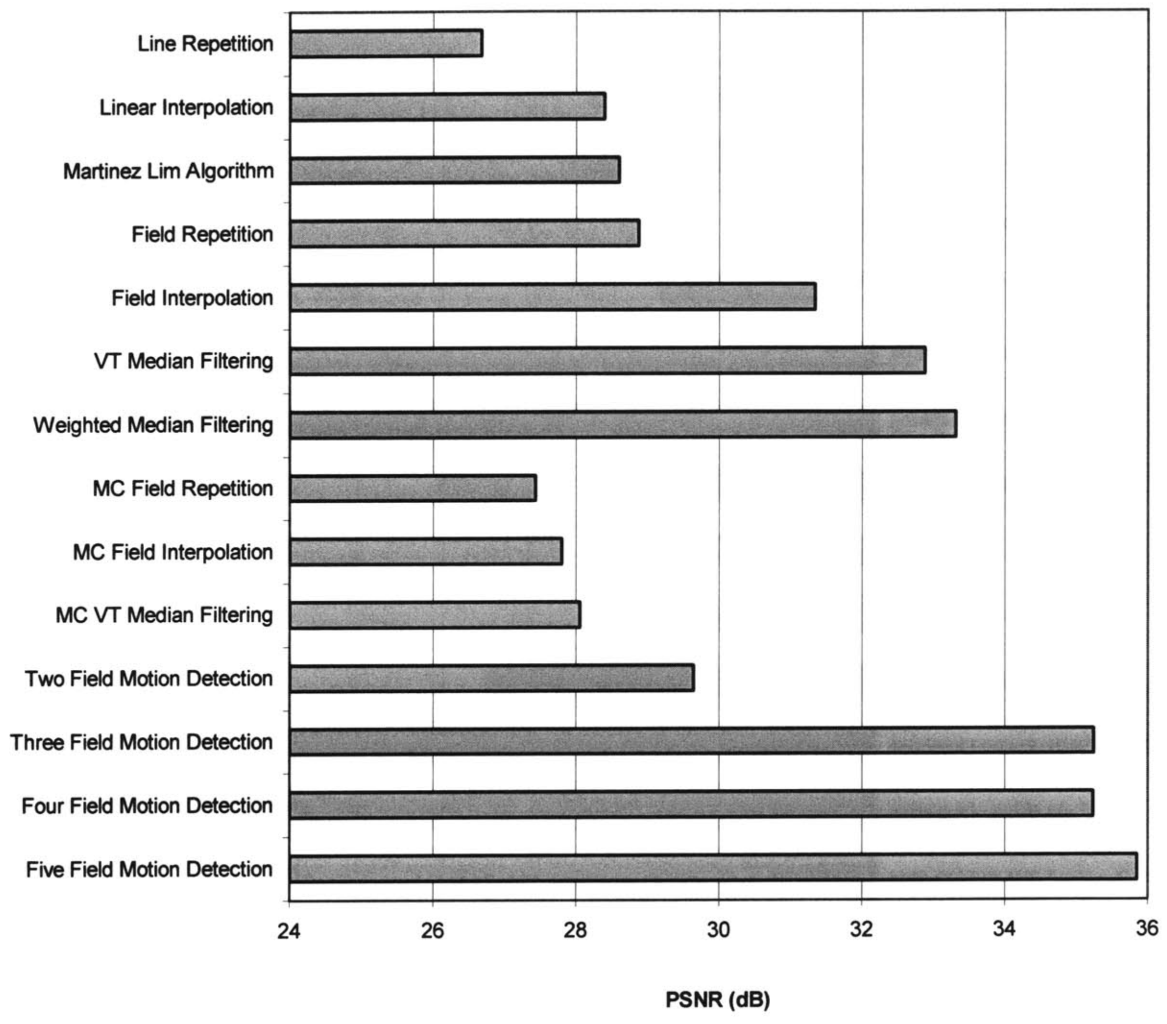
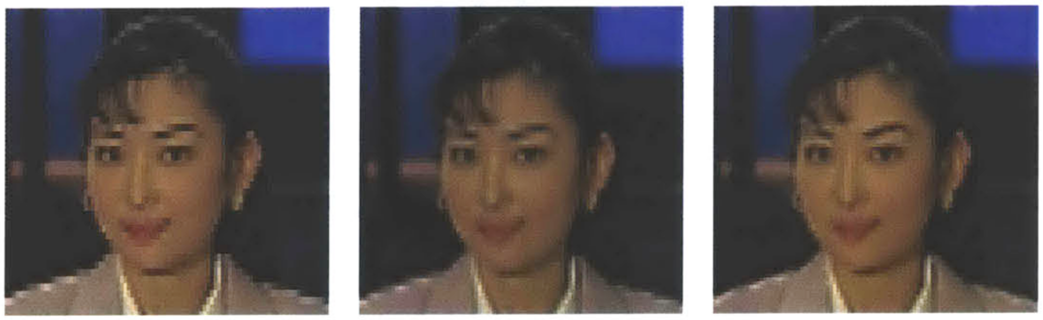


Figure 6-8: Average PSNR comparison.

From these results, a number of comments can be made. Of the three intraframe methods in this study, the Martinez-Lim algorithm consistently outperforms the others in PSNR for each of these seven video sequences. This is also confirmed through visual inspection since the Martinez-Lim algorithm tends to generate smoother more realistic images, as shown in Figure 6-9. This is the main reason why the Martinez-Lim algorithm was chosen as the spatial method in the motion adaptive algorithms.



(a) (b) (c)

Figure 6-9: (a) Line repetition. (b) Linear interpolation. (c) Martinez –Lim algorithm.

Another observation is that all of the motion compensated methods perform poorly compared to the other methods. If the video was panning or if there were a number of static objects, then motion compensation methods could potentially provide a significant amount of improvement. However, in this application, translational motion rarely occurs. The performance of some methods is actually decreased in some cases due to the errors made in motion estimation.

It should also be noted that a number of artifacts do not manifest themselves in this type of PSNR comparison. For instance, consider the surveillance sequences A, B and C. These sequences contain very little motion, only a small number of pixels in each image actually change. Thus, the temporal methods fair very well. In particular consider *Sequence A* shown in Figure 6-4. The PSNR comparison would indicate that field interpolation is clearly superior to the others. However, this is clearly not the case in a qualitative comparison, see Figure 6-10.

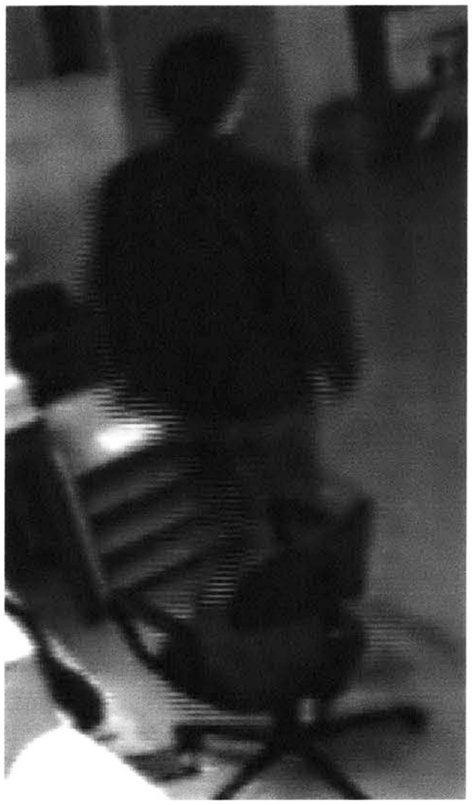


Figure 6-10: Bilinear field interpolation. Significant errors are introduced in regions of motion, however since the majority of pixels do not move, PSNR is still high.

Just as in Figure 4-10, field interpolation introduces significant artifacts in the presence of motion. However, since these artifacts are localized to a very small number of pixels, the PSNR does not drop significantly. Yet since the areas of motion are likely to be the areas of interest in surveillance sequences, this type of performance is unacceptable, regardless of the PSNR obtained. This is one example where PSNR and video quality do not directly coincide. As motion increases, as is the case in the other four sequences, the PSNR of these temporal methods begins to drop significantly coinciding with their poor performance. For this reason, even though field repetition, field interpolation, and the three field motion detection methods show good PSNR performance for these three particular sequences, they cannot be reasonably considered.

Of the remaining methods, the five-field motion detection algorithm demonstrates the best performance in the majority of these sequences and fairs very well in the others. The median filter methods also show good results, yet their performance tends to fall off more rapidly as the amount of motion increases. In visual comparisons, the median filters were seen to result in noisier images than the motion adaptive methods. This is perhaps due to the fact that, in the presence of motion, median filters default to line repetition, or at best linear interpolation, while the motion detection methods use the Martinez-Lim algorithm.

The PSNR results from all sequences have been combined in Figure 6-11 showing that, on average, the five-field motion detection algorithm outperforms other types of methods. Obviously this is dependant on the particular sequence, for instance field interpolation will perform best for a sequence with no motion. Yet, on average motion adaptive methods perform very well in all situations.

Average PSNR Comparison – Combined Results

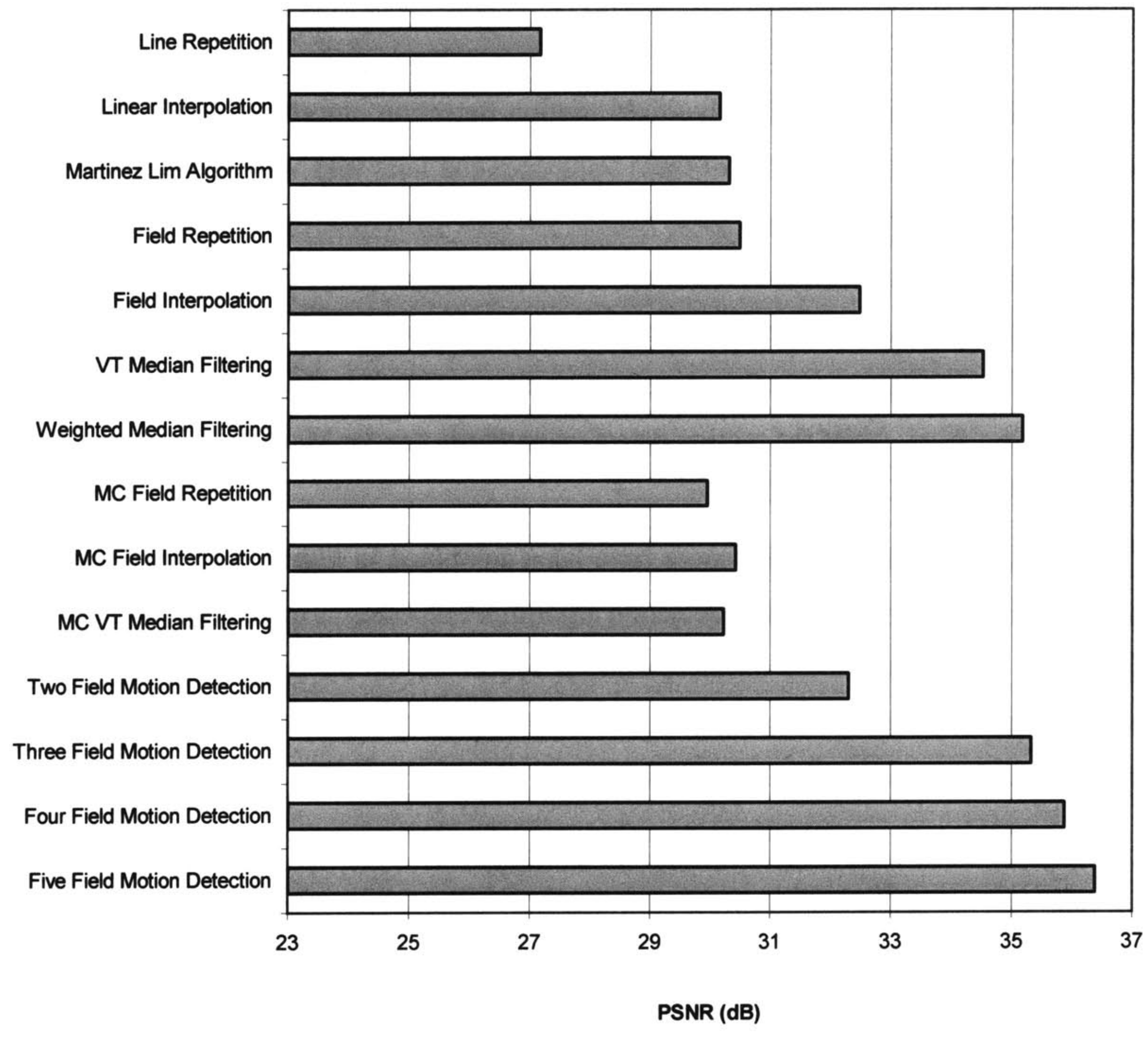


Figure 6-11: Average PSNR comparison. Results of all seven sequences have been averaged together.

The following figures present some visual results of this experiment. These figures all make comparisons between line repetition, weighted median filtering, and the five-field motion detection algorithm. Five-field motion detection is used since it was the method shown to have the highest performance while line repetition is included in these comparisons since it is the method most often used in current systems. Weighted median filtering is also compared here since it is the method that showed the second best performance behind the motion adaptive algorithms.

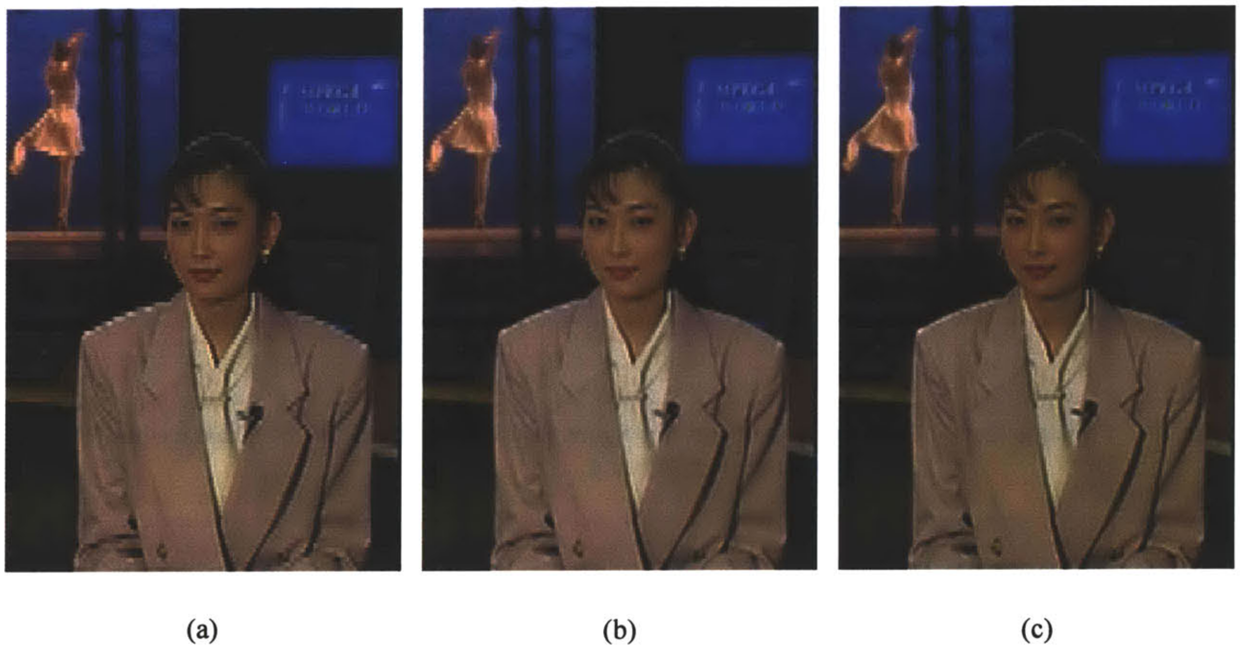


Figure 6-12: MPEG test sequence, News CIF. (a) Line repetition. (b) Weighted median filtering. (c) Motion adaptive processing

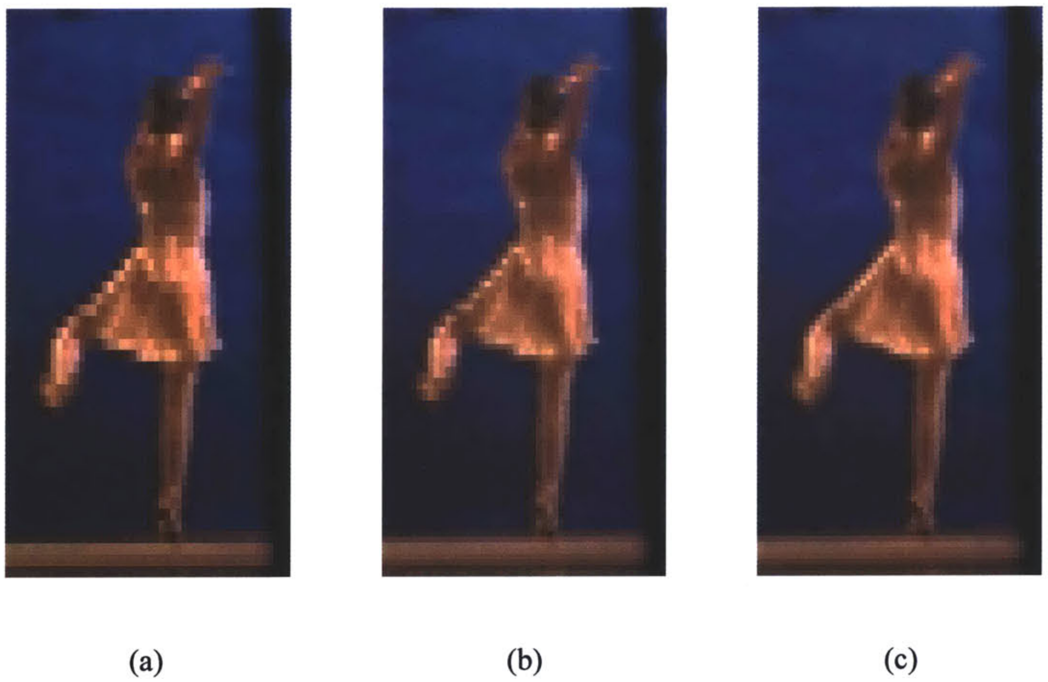
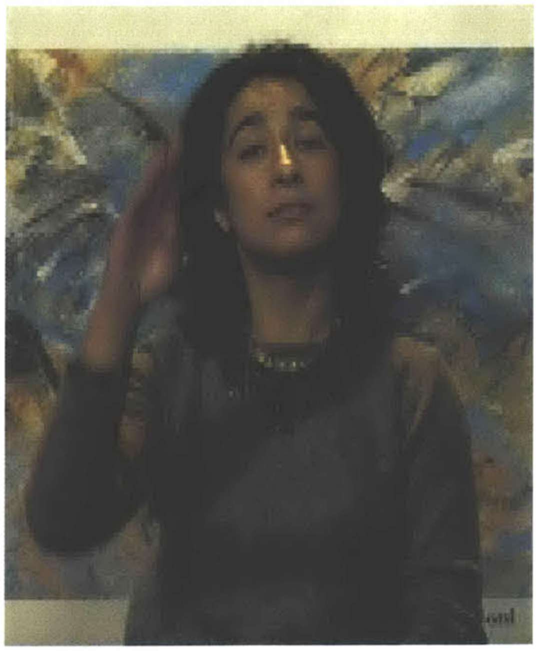
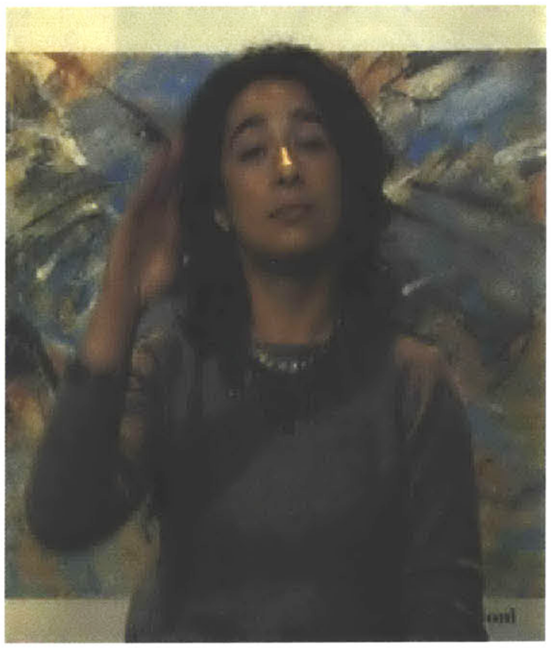


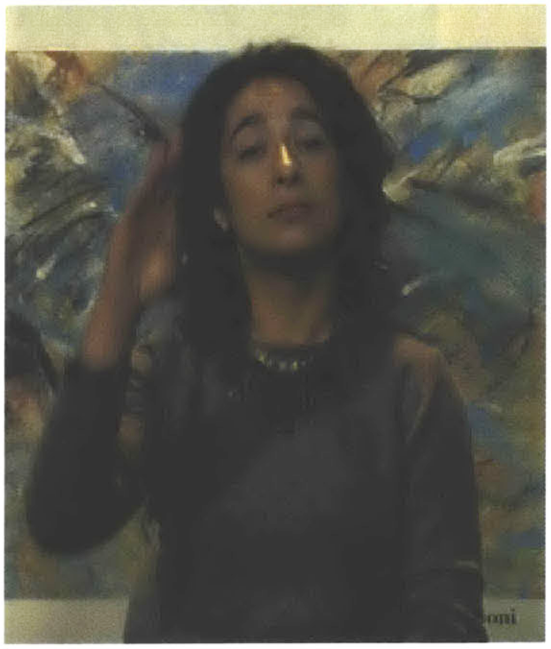
Figure 6-13: Enlarged view of previous figure. (a) Line repetition. (b) Weighted median filtering. (c) Motion adaptive processing



(a)

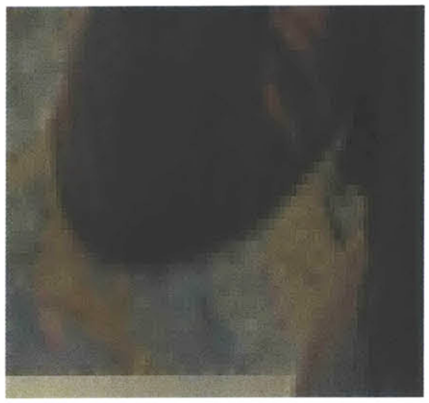


(b)

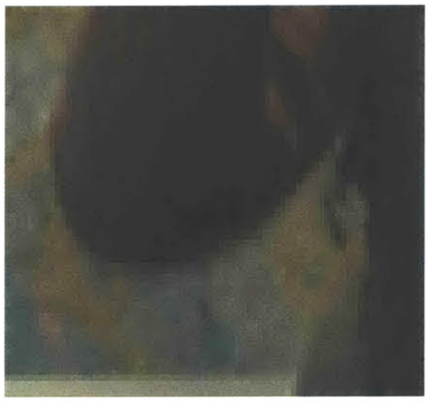


(c)

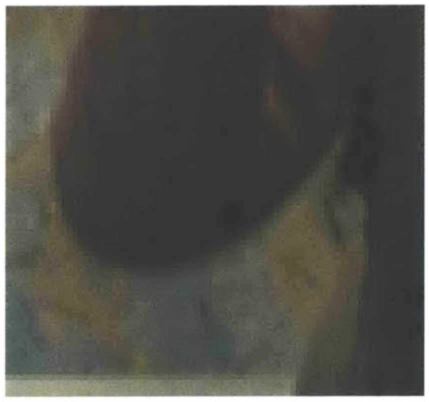
Figure 6-14: MPEG Test Sequence – Silent Voice CIF. (a) Line repetition. (b) Weighted median filtering. (c) Motion adaptive processing.



(a)

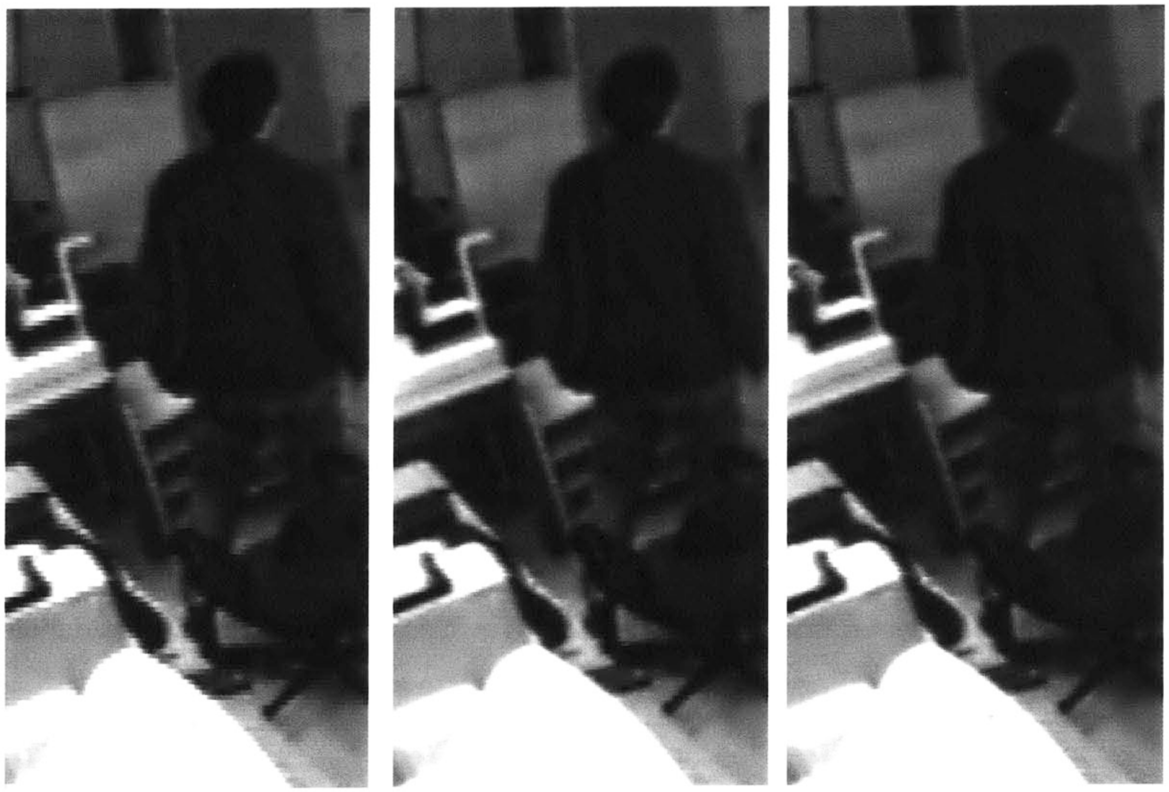


(b)



(c)

Figure 6-15: Enlarged view of previous figure. (a) Line repetition. (b) Weighted median filtering. (c) Motion adaptive processing.



(a) (b) (c)

Figure 6-16: Surveillance footage – sequence A. (a) Line repetition. (b) Weighted median filtering. (c) Motion adaptive processing.

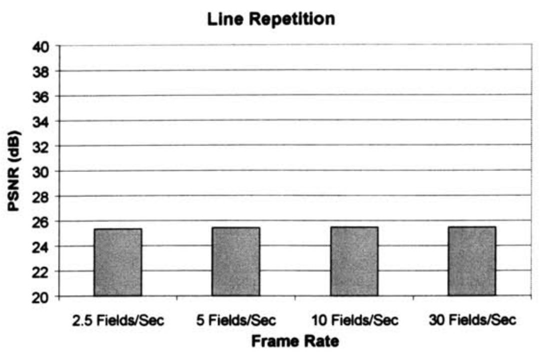
This qualitative analysis also leads to the same conclusion that the motion adaptive algorithm generally gives the best results. The resulting pictures are smoother and more realistic than the spatial methods or the median filters, and yet do not contain the serious artifacts introduced by some of the temporal methods such as field repetition.

The second experiment that was run was an analysis of the performance of each method versus frame rate. As mentioned in section 6.1, one particular surveillance camera sequence was used to generate four sequences at various frame rates. The results of deinterlacing these sequences were then used for PSNR comparison as shown in Figure 6-16.

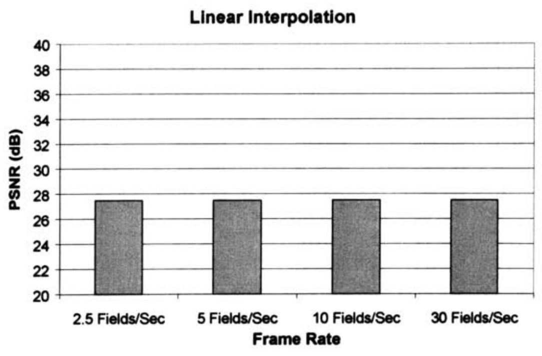
As would be expected the spatial methods show no variation in performance as frame rate is decreased. These methods only rely on the current frame, and thus have no dependence on the frame rate of the sequence. Similarly, the temporal methods, field repetition and field interpolation, perform very poorly as frame rates are decreased. Since these methods rely on the correlation between successive frames, the further apart in time each frame is, the worse these methods will do. The median filters show little variation to frame rate, demonstrating their robustness in the presence of motion.

The motion compensated methods surprisingly show little improvement at higher frame rates. It was thought that, at higher frame rates, the successive frames might have high enough correlation that these methods could be used to compensate for the motion. Yet, it appears that the same problems that the motion compensation methods suffer from at low frame rates still exist at higher frame rates. The types of motion, which can be appropriately compensated for, do not typically exist in these types of sequences.

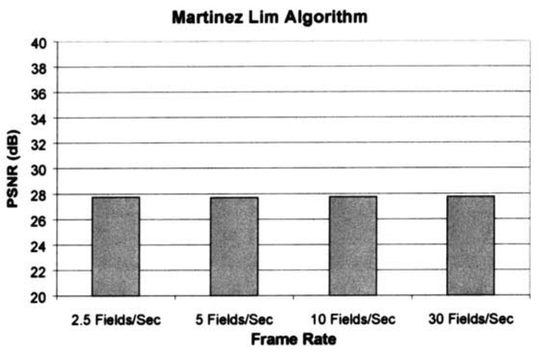
Finally, the five-field motion detection algorithm shows the best performance at each frame rate. The algorithm has even higher performance as the frame rate is increased.



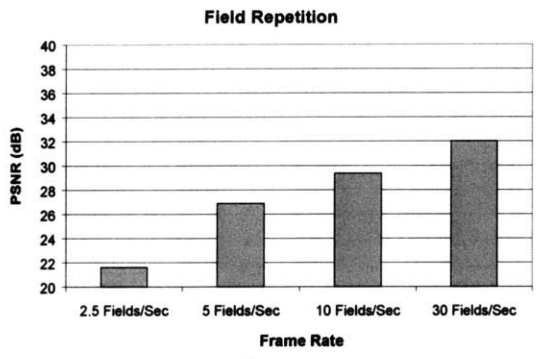
(a)



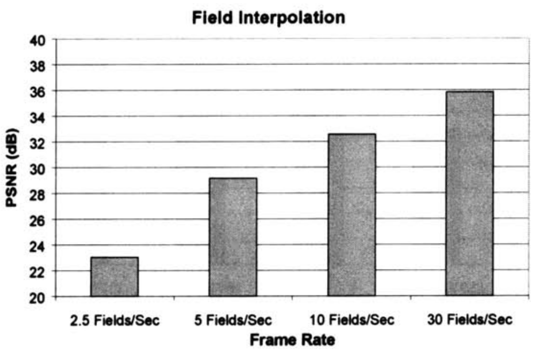
(b)



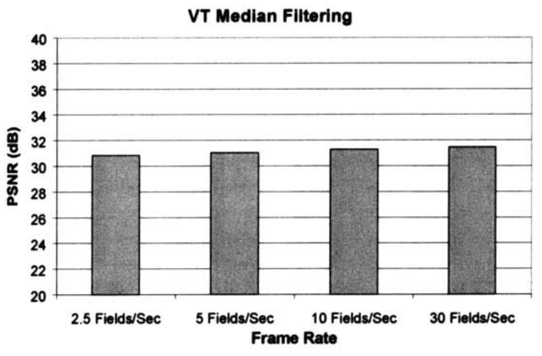
(c)



(d)

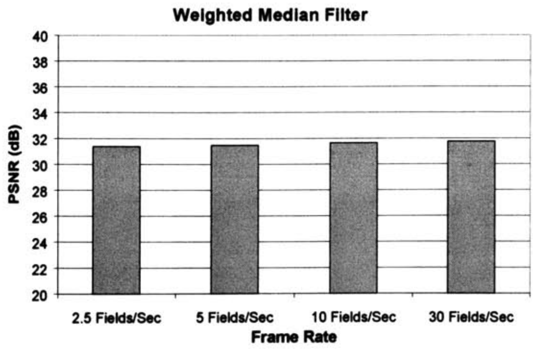


(e)

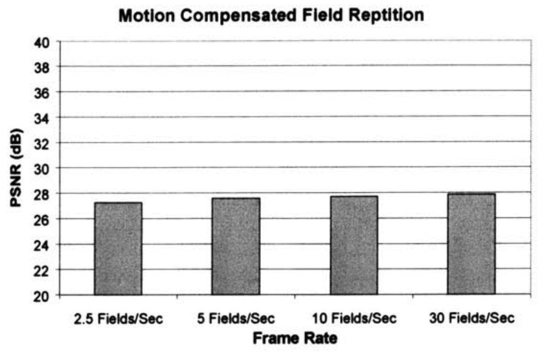


(f)

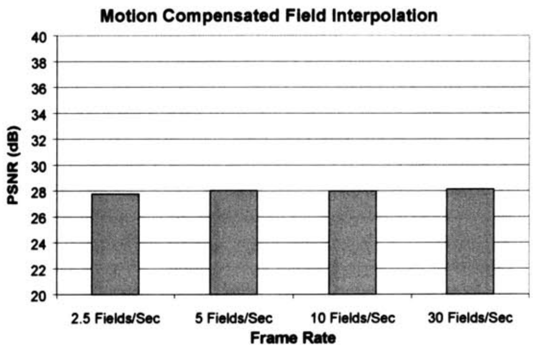
Figure 6-17: Comparison of average PSNR versus frame rate. Spatial methods (a), (b), and (c) show no dependence on frame rate, while the performance of the temporal methods (d) and (e) drops significantly at lower frame rates. Median filters (f) and (g) show good robustness to low frame rates with only a slight drop in performance as the frame rate is lowered.



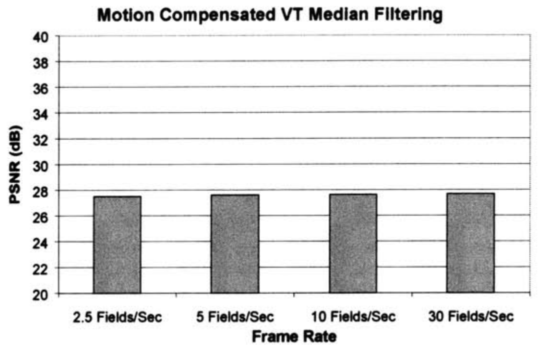
(g)



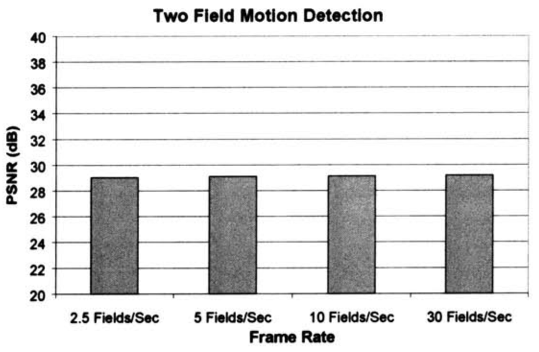
(h)



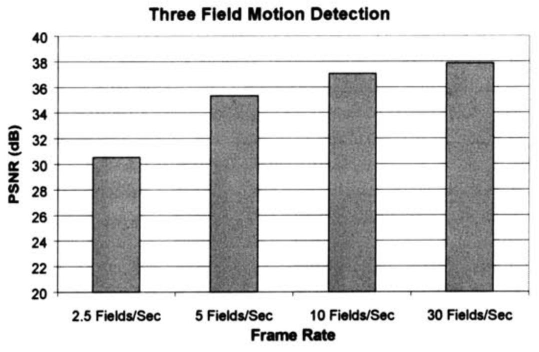
(i)



(j)

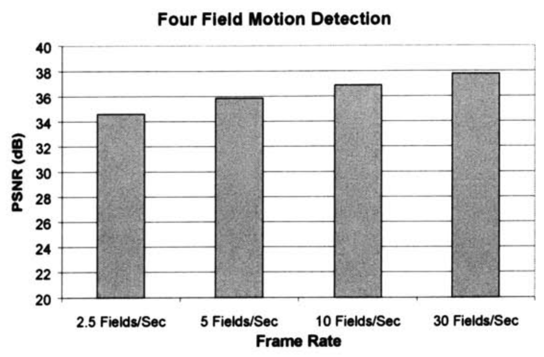


(k)

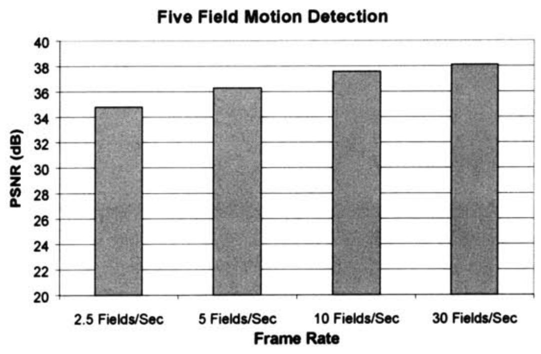


(l)

Figure 6-17 (continued): Surprisingly, the motion compensated methods, (h), (i), and (j), show little improvement even at high frame rates.



(m)



(n)

Figure 6-17 (continued): Five-field motion detection (n) shows excellent performance at all frame rates

Chapter 7
Conclusion

7.1 Summary

As shown in the previous chapter, motion adaptive deinterlacing can be used to effectively combine the benefits of both interframe and intraframe processing with good results. The resulting images are deinterlaced with the method that works best for each region depending on the presence or absence of motion.

When compared with interlaced displays, which has the same effect as using line repetition, motion adaptive deinterlacing has been shown to lead to a significant increase in video quality. Just the fact that the "jittering" effect of line repetition is removed significantly improves the video. Yet, there are many other benefits as well. Stationary regions of the video are deinterlaced using temporal methods, resulting in much higher resolution, while moving regions are deinterlaced with the high quality spatial method introduced by Martinez and Lim. For these reasons, it is clear that deinterlacing these surveillance sequences can help improve their quality significantly. These improvements may help improve the intelligibility of the tapes and help reduce some of the fatigue caused by analyzing these sequences in general.

When compared to other deinterlacing algorithms, motion adaptive methods have been found to have excellent performance in the presence or absence of motion and independent of the frame rate used.

7.2 Future Work

In the future, there are a few steps that could be taken to continue this research. The first is the addition of global motion compensation to the motion detection algorithm. When implemented as described in this thesis, the motion adaptive deinterlacing algorithm requires the use of a stationary camera to maximize performance. If the camera were to move, a larger portion of the video would be deinterlaced spatially, and performance would go down. The types of security cameras that pan back and forth could be accounted for with the use of global motion compensation. Global motion compensation would attempt to register the current field with the previous field in an attempt to compensate for this type of panning motion. Slight shaking due to camera vibrations could also be dealt with by using this type of idea. The addition of global motion compensation would most likely improve performance of the motion detection algorithm with a moving camera system, yet it would most likely decrease the performance for stationary cameras due to small motion compensation errors that might be made. Given that this research primarily looked at stationary camera systems, this has been left for future studies.

Another area of interest is a real-time implementation of this algorithm for use in a commercial system. This system would take a video signal from a time-lapse VCR, convert the analog video to digital video, deinterlace it with this type of algorithm, and then output the signal to a computer monitor or a high definition television display.

Appendix A
Experimental Data

This appendix provides the data and numerical results from which the charts in Chapter 6 were generated.

Sequence: MPEG Test Sequence - News CIF	
Deinterlacing Algorithm	**Average PSNR**
Line Repetition	28.19
Linear Interpolation	33.98
Martinez Lim Algorithm	33.98
Field Repetition	29.46
Field Interpolation	31.86
VT Median Filtering	36.31
Weighted Median Filtering	37.96
MC Field Repetition	34.60
MC Field Interpolation	35.50
MC VT Median Filtering	34.77
Two Field Motion Detection	38.56
Three Field Motion Detection	37.72
Four Field Motion Detection	39.97
Five Field Motion Detection	40.61

Table A-1: Average PSNR data used to generate Figure 6-2

Sequence: MPEG Test Sequence - Silent Voice CIF	
Deinterlacing Algorithm	**Average PSNR**
Line Repetition	30.72
Linear Interpolation	34.51
Martinez Lim Algorithm	34.64
Field Repetition	30.29
Field Interpolation	32.24
VT Median Filtering	36.84
Weighted Median Filtering	37.91
MC Field Repetition	34.65
MC Field Interpolation	35.39
MC VT Median Filtering	34.71
Two Field Motion Detection	37.95
Three Field Motion Detection	36.88
Four Field Motion Detection	39.66
Five Field Motion Detection	40.26

Table A-2: Average PSNR data used to generate Figure 6-3

Sequence: Surveillance Video - Febrary 12th, 2001 - Sequence A	
Deinterlacing Algorithm	**Average PSNR**
Line Repetition	24.20
Linear Interpolation	28.86
Martinez Lim Algorithm	29.09
Field Repetition	34.40
Field Interpolation	36.20
VT Median Filtering	34.62
Weighted Median Filtering	34.92
MC Field Repetition	28.77
MC Field Interpolation	29.01
MC VT Median Filtering	28.92
Two Field Motion Detection	30.35
Three Field Motion Detection	35.22
Four Field Motion Detection	34.27
Five Field Motion Detection	34.64

Table A-3: Average PSNR data used to generate Figure 6-4

Sequence: Surveillance Video - Febrary 12th, 2001 - Sequence C	
Deinterlacing Algorithm	**Average PSNR**
Line Repetition	26.90
Linear Interpolation	28.55
Martinez Lim Algorithm	28.74
Field Repetition	32.23
Field Interpolation	33.85
VT Median Filtering	34.34
Weighted Median Filtering	34.65
MC Field Repetition	28.57
MC Field Interpolation	28.92
MC VT Median Filtering	28.68
Two Field Motion Detection	30.11
Three Field Motion Detection	34.60
Four Field Motion Detection	34.04
Five Field Motion Detection	34.39

Table A-4: Average PSNR data used to generate Figure 6-5

Sequence: Surveillance Video - Febrary 12th, 2001 - Sequence E	
Deinterlacing Algorithm	**Average PSNR**
Line Repetition	26.88
Linear Interpolation	28.45
Martinez Lim Algorithm	28.64
Field Repetition	32.54
Field Interpolation	34.40
VT Median Filtering	34.18
Weighted Median Filtering	34.55
MC Field Repetition	28.44
MC Field Interpolation	28.80
MC VT Median Filtering	28.53
Two Field Motion Detection	29.97
Three Field Motion Detection	34.81
Four Field Motion Detection	33.72
Five Field Motion Detection	34.23

Table A-5: Average PSNR data used to generate Figure 6-6

Sequence: Surveillance Video - Febrary 20th, 2001 - Sequence B	
Deinterlacing Algorithm	**Average PSNR**
Line Repetition	26.61
Linear Interpolation	28.29
Martinez Lim Algorithm	28.48
Field Repetition	25.63
Field Interpolation	27.45
VT Median Filtering	32.48
Weighted Median Filtering	32.99
MC Field Repetition	27.19
MC Field Interpolation	27.58
MC VT Median Filtering	27.91
Two Field Motion Detection	29.48
Three Field Motion Detection	32.84
Four Field Motion Detection	34.31
Five Field Motion Detection	34.72

Table A-6: Average PSNR data used to generate Figure 6-7

Sequence: Surveillance Video - Febrary 20th, 2001 - Sequence C	
Deinterlacing Algorithm	**Average PSNR**
Line Repetition	26.67
Linear Interpolation	28.39
Martinez Lim Algorithm	28.59
Field Repetition	28.87
Field Interpolation	31.35
VT Median Filtering	32.88
Weighted Median Filtering	33.31
MC Field Repetition	27.43
MC Field Interpolation	27.80
MC VT Median Filtering	28.05
Two Field Motion Detection	29.65
Three Field Motion Detection	35.25
Four Field Motion Detection	35.24
Five Field Motion Detection	35.85

Table A-7: Average PSNR data used to generate Figure 6-8

Sequence: Surveillance Video - Febrary 21st, 2001 - 30 FPS	
Deinterlacing Algorithm	**Average PSNR**
Line Repetition	25.49
Linear Interpolation	27.54
Martinez Lim Algorithm	27.76
Field Repetition	32.05
Field Interpolation	35.83
VT Median Filtering	31.47
Weighted Median Filtering	31.75
MC Field Repetition	27.87
MC Field Interpolation	28.15
MC VT Median Filtering	27.68
Two Field Motion Detection	29.20
Three Field Motion Detection	37.88
Four Field Motion Detection	37.78
Five Field Motion Detection	38.13

Table A-8: Frame rate comparison data at 30 frames per second

Sequence: Surveillance Video - Febrary 21st, 2001 - 10 FPS	
Deinterlacing Algorithm	**Average PSNR**
Line Repetition	25.49
Linear Interpolation	27.52
Martinez Lim Algorithm	27.74
Field Repetition	29.37
Field Interpolation	32.56
VT Median Filtering	31.29
Weighted Median Filtering	31.66
MC Field Repetition	27.72
MC Field Interpolation	28.02
MC VT Median Filtering	27.65
Two Field Motion Detection	29.15
Three Field Motion Detection	37.06
Four Field Motion Detection	36.90
Five Field Motion Detection	37.58

Table A-9: Frame rate comparison data at 10 frames per second

Sequence: Surveillance Video - Febrary 21st, 2001 - 5 FPS	
Deinterlacing Algorithm	**Average PSNR**
Line Repetition	25.43
Linear Interpolation	27.49
Martinez Lim Algorithm	27.70
Field Repetition	26.89
Field Interpolation	29.17
VT Median Filtering	31.05
Weighted Median Filtering	31.48
MC Field Repetition	27.61
MC Field Interpolation	28.06
MC VT Median Filtering	27.62
Two Field Motion Detection	29.13
Three Field Motion Detection	35.35
Four Field Motion Detection	35.86
Five Field Motion Detection	36.30

Table A-10: Frame rate comparison data at 5 frames per second

Sequence: Surveillance Video - Febrary 21st, 2001 - 2.5 FPS	
Deinterlacing Algorithm	**Average PSNR**
Line Repetition	25.34
Linear Interpolation	27.50
Martinez Lim Algorithm	27.75
Field Repetition	21.63
Field Interpolation	23.06
VT Median Filtering	30.84
Weighted Median Filtering	31.39
MC Field Repetition	27.27
MC Field Interpolation	27.77
MC VT Median Filtering	27.53
Two Field Motion Detection	29.04
Three Field Motion Detection	30.54
Four Field Motion Detection	34.61
Five Field Motion Detection	34.82

Table A-11: Frame rate comparison data at 2.5 frames per second

References

[1] B. Ayazifar and J. S. Lim, "Pel-adaptive model-based interpolation of spatially subsampled images," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Volume: 3, 1992, Pages: 181-184.

[2] B. Bhatt, *et al.*, "Grand alliance HDTV multi-format scan converter," in *IEEE Transactions on Consumer Electronics*, Volume: 41 Issue: 4, Nov. 1995, Pages: 1020-1031.

[3] R. J. Clarke, "Basic principles of motion estimation and compensation," in *IEEE Colloquium on Applications of Motion Compensation*, 1990, Pages 1/1 - 1/7.

[4] G. de Haan and E. B. Bellers, "Advanced motion estimation and motion compensated deinterlacing," in *Society of Motion Picture and Television Engineers Journal*, Nov. 1997, Pages 777-786.

[5] G. de Haan and E. B. Bellers, "Deinterlacing – an overview," in *Proceedings of the IEEE*, Volume: 86 Issue: 9 , Sept. 1998, Pages: 1839-1857.

[6] G. de Haan and E. B. Bellers, "Deinterlacing video data," in *International Conference on Consumer Electronics*, Volume 33, Aug. 1997, Pages: 819-825.

[7] T. Doyle and M. Looymans, "Progressive scan conversion using edge information," in *Signal Processing of HDTV II*, L. Chiariglione, Ed. Amsterdam, The Netherlands: Elsevier, 1990, Pages: 711-721.

[8] D. Han, C. Shin, S Choi, and J. Park, "A motion adaptive 3-D de-interlacing algorithm based on the brightness profile pattern difference," in *IEEE Transactions on Consumer Electronics*, Volume: 45 Issue: 3, Aug. 1999, Pages: 690-697.

[9] D. Hargreaves and J. Vaisey, "Bayesian motion estimation and interpolation in interlaced video sequences," in *IEEE Transactions on Image Processing*, Volume: 6 Issue: 5, May 1997, Pages: 764-769.

[10] K. Jensen and D. Anastassiou, "Digitally assisted deinterlacing for EDTV," in *IEEE Transactions on Circuits and Systems for Video Technology*, Volume: 3 Issue: 2, April 1993, Pages: 99-106.

[11] O. Kalevo and P. Haavisto, "Deinterlacing of video signals using nonlinear interpolation with simple motion compensation," in *IEEE Winter Workshop on Nonlinear Digital Signal Processing, 1993*, Pages: P_4.1 - P_4.6.

[12] T. Koivunen, "Motion Detection of an Interlaced Video Signal," in *IEEE Transactions on Consumer Electronics*, Volume: 40 Issue: 3, Aug. 1994, Pages: 753-760.

[13] A. Kojima, N. Sakurai, and J. Kishigami, "Motion detection using 3D-FFT spectrum," in *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Volume: 5, 1993, Pages: 213-216.

[14] J. Kovačević, R. J. Safranek, and E. M. Yeh, "Deinterlacing by successive approximation," in *IEEE Transactions on Image Processing*, Volume: 6 Issue: 2, Feb. 1997, Pages: 339-344.

[15] C. Lee, S Chang, and C. Jen, "Motion detection and motion adaptive pro-scan conversion," in *IEEE International Symposium on Circuits and Systems*, 1991, Pages: 666-669.

[16] M. Lee, J. Kim, J. Lee, K. Ryu, and D. Song, "A new algorithm for interlaced to progressive scan conversion based on directional correlations and its IC design," in *IEEE Transactions on Consumer Electronics*, Volume: 40 Issue: 2, May 1994, Pages: 119-129.

[17] R. Li, B. Zheng, and M.L. Liou, "Reliable motion detection/compensation and its applications to deinterlacing," in *IEEE Transactions on Circuits and Systems for Video Technology*, Volume: 10 Issue: 1, Feb. 2000, Page(s): 23-29.

[18] D. M. Martinez and J. S. Lim, "Spatial interpolation of interlaced television pictures," in, *IEEE International Conference on Acoustics, Speech, and Signal Processing*, Volume:3, 1989, Pages: 1886-1889.

[19] C. Ryu and S. Kim, "Deinterlacing using motion compensated local spectra," in *IEEE Proceedings of Asilomar*, Volume: 2, 1996, Pages: 1394 -1397.

[20] Patti, M. Sezan, and A. Tekalp, "Robust methods for high quality stills from interlaced video in the presence of dominant motion," in *IEEE Transactions on Circuits and Systems for Video Technology*, Volume: 7 Issue: 2, April 1997, Pages: 328 -342.

[21] J. Salo, Y. Nuevo, and V. Hameenaho, "Improving TV picture quality with linear-median type operations," in *IEEE Transactions on Consumer Electronics*, Vol: 34 Issue: 2, Aug. 1988, Pages 373-379.

[22] J. Salonen and S. Kalli, "Edge adaptive interpolation for scanning rate conversion," in *Signal Processing of HDTV IV*, E. Dubois and L. Chiariglione, Eds. Amsterdam, The Netherlands: Elsevier, 1993, Pages 757-764.

[23] C. Stiller and J. Konrad, "Estimating motion in image sequences," in *IEEE Signal Processing Magazine*, Volume: 16 Issue: 4, July 1999, Pages: 70-91.

[24] The Grand Alliance. "The U.S. HDTV standard," in *IEEE Spectrum*, Volume: 32 Issue: 4, April 1995, Pages: 36-45.

[25] Y. Wang and S.K. Mitra, "Motion/pattern adaptive interpolation of interlaced video sequences," in *International Conference on Acoustics, Speech, and Signal Processing*, Volume: 4, 1991, Pages: 2829-2932.