An Investigation of Methods for Digital Television Format Conversions

by

Chuohao Yeo

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degrees of

Bachelor of Science in Electrical Science and Engineering

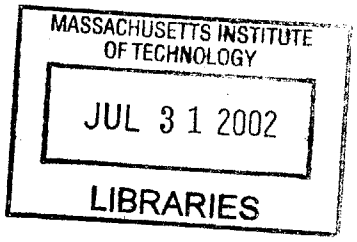
and

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

May 2002



Author .....
Department of Electrical Engineering and Computer Science
May 10, 2002

Certified by .....
Jae S. Lim
Professor of Electrical Engineering
Thesis Supervisor

Accepted by .....
Arthur C. Smith
Chairman, Department Committee on Graduate Theses

An Investigation of Methods for Digital Television Format Conversions

by

Chuohao Yeo

Submitted to the Department of Electrical Engineering and Computer Science
on May 10, 2002, in partial fulfillment of the
requirements for the degrees of
Bachelor of Science in Electrical Science and Engineering
and
Master of Engineering in Electrical Engineering and Computer Science

Abstract

For the last few years, broadcasters have been in transition from the old NTSC standard to the new digital television standard. A key feature of the new standard is the plethora of supported video formats. While this gives broadcasters greater flexibility in their operations, it has also necessitated format conversion at the transmitter-level and the receiver-level. It is possible for current production equipment to produce video at a resolution of 720x1280 progressively scanned (720p) at 60 frames per second, at a resolution of 1080x1920 interlaced (1080i) at 60 fields per second or at a resolution of 1080x1920 progressively scanned (1080p) at 30 frames per second. However, broadcasters can choose to transmit in many other video formats. In addition, a typical consumer HDTV set has only one display format, therefore other video formats that are received must be first converted to its native display format. The digital television standard also allows for multi-casting, or the broadcasting of multiple programs on the same channel. Therefore, broadcasters may be interested in broadcasting programs in a video format that has lower resolution than production formats. While computation time may be an issue in commercial systems, the only criterion used here to evaluate different methods is the output quality. A number of conversion techniques were investigated to determine the best method for a small subset of possible format conversions.

Thesis Supervisor: Jae S. Lim
Title: Professor of Electrical Engineering

Dedication

To my parents,

For bringing me up.

To my sister,
For bringing me joy.

Acknowledgments

This thesis would not have been possible without the help and support of the various people in my life.

First, I would like to thank Professor Jae Lim for giving me the chance to work in the Advanced Telecommunications and Signal Processing (ATSP) group. I am deeply indebted to him for his guidance and support.

I am very grateful to my colleagues at ATSP, Brian Heng and Wade Wan, for giving me the jump start on my research, and for giving me advice while I was writing this thesis. They were always helpful despite their own busy schedule.

I would also like to thank the secretary of the group, Cindy LeBlanc, for making my life here so much easier.

An education at MIT would have been difficult without the financial support of the Public Service Commission (PSC) of Singapore. I am grateful to them for this wonderful opportunity.

Finally, I owe my success to my parents, who have made countless sacrifices and have shown me endless love in bringing me up. Without them, I will not be where I am today.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In December 1996, the Federal Communications Commission (FCC) adopted the digital television standard to replace the old National Television System Committee (NTSC) analog standard used for terrestrial broadcast of television programs [1]. A key feature of the new standard is the large number of video formats that are allowed. Each resolution format is defined by the number of scan lines per video frame, the number of picture elements (pixels) per scan line, the frame rate and structure (interlaced or progressive) and the aspect (width to height) ratio. Table 1.1 shows the four basic resolution formats and their 18 variants supported by the digital television standard.

The large number of video formats available introduced the problem of format conversion: how to convert a digital video from one video format to another, with possible constraints on video quality and computation time. This thesis investigates two specific format conversions, with a single constraint of minimal loss of video quality.

Picture Size (lines x pixels)	Frame Rate/Structure				Aspect Ratio	
1080 x 1920	60i	-	30p	24p	16:9	-
720 x 1280	-	60p	30p	24p	16:9	-
480 x 704	60i	60p	30p	24p	16:9	4:3
480 x 640	60i	60p	30p	24p	-	4:3
i=interlaced, p=progressive						

Table 1.1: Video formats supported by the digital television standard (reproduced from [1])

1.1 Video Format Properties

As observed in table 1.1, there are three main properties that distinguish one video format from another. These properties will be discussed below.

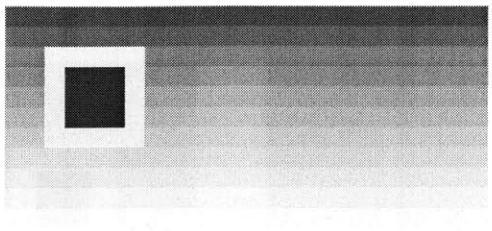
1.1.1 Picture Size

The picture size of the digital video format, otherwise known as its resolution, determines the sampling lattice that is used to sample the original analog video data. Therefore, the quality of a video depends heavily on its resolution. Figure 1-1 shows the effects of sampling without pre-filtering at two different resolutions.

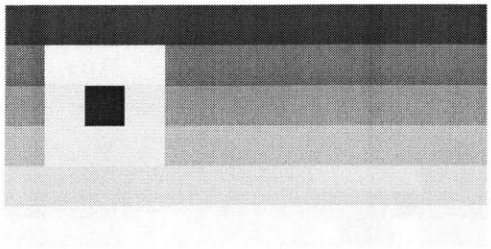
1.1.2 Frame Rate/Structure

The frame rate of a video format determines how many frames are shown to the viewer per second. A higher frame rate improves temporal resolution and provides smoother motion rendition but also requires a higher bit rate for transmission [2]. Closely tied to the frame rate is the split between progressive and interlaced scanning.

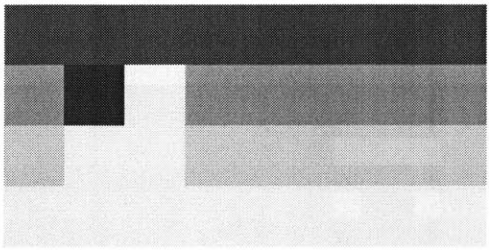
A progressively scanned video consists of frames in which every scan line in the picture is captured. Interlacing, an artifact of the NTSC standard, is a technique that



(a) Original picture at 12 lines by 24 pixels



(b) Sampled picture at 6 lines by 12 pixels



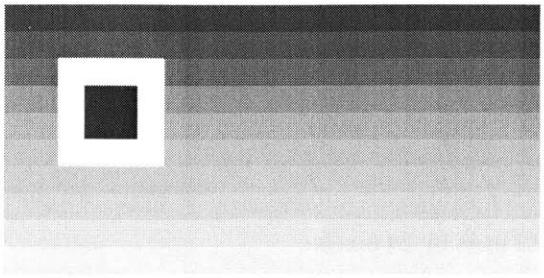
(c) Sampled picture at 4 lines by 8 pixels

Figure 1-1: Sampling at two different resolutions

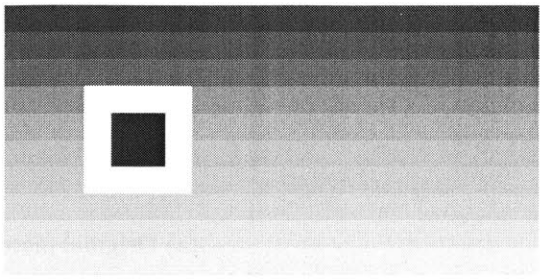
allows one to reduce the signal bandwidth of a video while keeping the same refresh rate and resolution. Instead of consisting of frames, an interlaced video consists of fields where each field alternates between capturing the even and odd scan lines in the picture. For example, all odd fields would consist of odd scan lines, and all even fields would consist of even scan lines. The difference between progressive and interlaced scanning is illustrated in figure 1-2.

1.1.3 Aspect Ratio

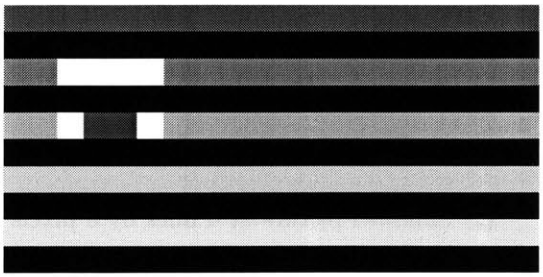
The aspect ratio of a video is the ratio of its frame width to its frame height. NTSC pictures have an aspect ratio of 4:3, while most digital television video formats have an aspect ratio of 16:9, which is closer to that of current theater motion pictures. Figure 1-3 shows the difference between pictures in these two aspect ratios.



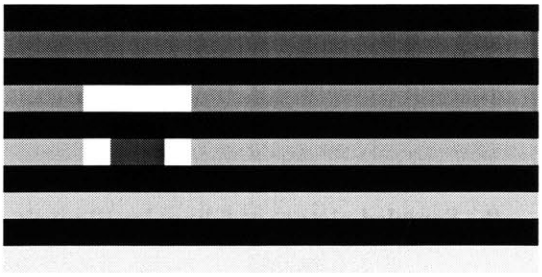
(a) Progressive scanning, frame i



(b) Progressive scanning, frame i+1

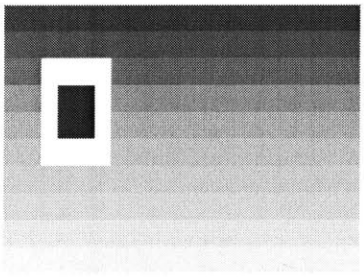


(c) Interlaced scanning, field i

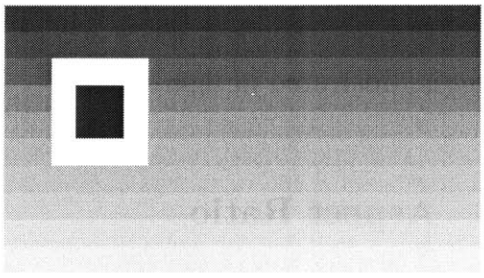


(d) Interlaced scanning, field i+1

Figure 1-2: Progressive scanning and interlaced scanning



(a) Aspect ratio of 4:3



(b) Aspect ratio of 16:9

Figure 1-3: Aspect ratio

1.2 Motivation

While the large number of video formats supported by the digital television standard gives broadcasters great flexibility in their operations, it has also made format conversion systems necessary at both the transmitter level and the receiver level.

Currently, most HDTV production equipment offer a resolution of either 1080x1920 interlaced at 60 fields per second (1080i) or 720x1280 progressively scanned at 60 frames per second (720p). However, a broadcaster can choose to transmit its programs in any of the supported digital television video formats. While broadcasting at a higher resolution would give consumers video of higher quality, there are other factors that broadcasters have to take into account when deciding which video format to use for transmission.

First, a quick survey of current HDTV display sets reveals that most of them have a single native display resolution, for example, 1080i or 720p. Hence, programs in other video formats would have to be converted to the display's native format before being shown. Depending on which display format gains greater acceptance in the marketplace, a broadcaster can then choose a transmission video format to ensure that programs are seen the way the broadcaster would like them to be seen by the majority of its viewers.

Second, the digital television standard allows for multi-casting, or the sending of multiple programs over the same channel. For example, instead of showing just one program at 720p, the broadcaster can choose to send two programs at 480p instead, and let the consumer decide which program to view. This allows the broadcaster to effectively double its revenue, since it can now theoretically broadcast twice the amount of advertisements. Of course, the broadcaster may still choose to broadcast selected programs at a higher resolution video format.

Given the large number of video formats possible, there exist an even greater

number of possible format conversions. Only a small subset of these is investigated to limit the scope of this study. However, it can still provide some insight into the general problem of format conversion. In particular, the following down-conversions of high-definition video are studied:

- 720p to 480p
- 1080i to 480p

We choose a destination video format of 480p, because a 480p digital television display will be cheaper to produce, and so market demand for it is likely to be higher initially. Therefore, while programs might be broadcasted at higher resolutions, the digital television set needs to be able to convert those programs to 480p in order to display it.

1.3 Outline

Chapter 2 discusses the background relevant to this thesis. There are two sub-problems which need to be solved to perform video format conversion: de-interlacing and frame resizing. Formal definitions for each of these problems would be provided. The chapter will then describe the various methods that can be used to solve the two sub-problems.

Chapter 3 describes the overall system of each type of format converter. Details of more sophisticated techniques will be presented and clarified in this chapter.

Chapter 4 describes the tests that were performed. The results and analysis of these experiments will also be provided.

Chapter 5 concludes the thesis. A summary of the study and future research directions will be given.

Chapter 2

Background

In the field of format conversion, most of the attention has been on de-interlacing and frame rate conversion, while video resizing has been given much less attention. For the format conversion problem introduced earlier, de-interlacing and frame resizing are the two techniques that are of relevance. While aspect ratio control and frame rate conversion might be important in other format conversions, they are not needed for the format conversions studied in this thesis.

2.1 Definitions

A progressively-scanned digital video sequence is a three-dimensional array of data in a horizontal-vertical-temporal space. Let $F_p[x, y, n]$ be the data that represents a pixel, where x is the horizontal position of the pixel within a scan line, y is the vertical position of the scan line, and n denotes the frame number. If the video has N frames, Y rows and X columns, then $F_p[x, y, n]$ is defined for all integer values of $0 \leq x < X$, $0 \leq y < Y$, and $0 \leq n < N$.

These definitions will be used for the rest of the thesis.

2.2 De-interlacing

If an interlaced video was produced from a progressive source, then it is possible to express the interlaced video data in terms of the progressive video data. Let $F_i[x, y, n]$ be the data that represents a pixel in an interlaced video, where x is the horizontal position of the pixel within a scan line, y is the vertical position of the scan line, and n denotes the field number. Then,

$$F_i[x, y, n] = \begin{cases} F_p[x, y, n] & \text{if } y \equiv n \pmod 2, \\ \text{undefined} & \text{otherwise.} \end{cases} \tag{2.1}$$

Equation (2.1) implies that only even lines are scanned in even fields, and only odd lines are scanned in odd fields.

The task of de-interlacing is to reverse the interlacing that took place when converting a progressively scanned video to an interlaced video. In other words, we wish to find an estimate, $\hat{F}_p[x, y, n]$, of $F_p[x, y, n]$. The standard approach is to define this estimate as:

$$\hat{F}_p[x, y, n] = \begin{cases} F_i[x, y, n] & \text{if } y \equiv n \pmod 2, \\ \hat{F}[x, y, n] & \text{otherwise.} \end{cases} \tag{2.2}$$

where $\hat{F}[x, y, n]$ is the estimate of the missing lines in the interlaced video. Ideally, we would like $\hat{F}_p[x, y, n] = F_p[x, y, n]$.

De-interlacing cannot be treated as a simple sampling rate up-conversion problem, because the sampling involved in interlacing does not in general fulfill the Nyquist criterion [3]. In addition, pre-filtering prior to interlacing is not an option because it results in objectionable blurring of the picture [3].

Many de-interlacing methods exist for obtaining $\hat{F}[x, y, n]$. As outlined by Lee et al. [4], de-interlacing algorithms can be classified roughly as intraframe techniques and interframe techniques. The class of interframe techniques can be further sub-divided into non-motion-compensated and motion-compensated methods.

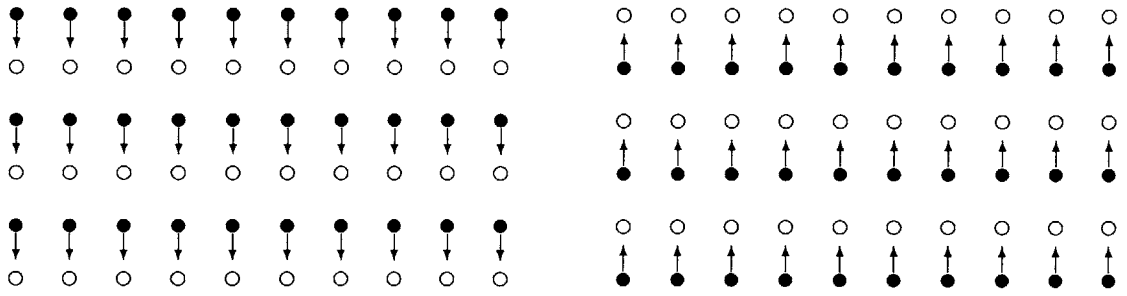
2.2.1 Intraframe Techniques

Spatial Filtering

The simplest form of de-interlacing techniques involves spatial filtering of each frame to exploit spatial correlation. In particular, line repetition can be used to simply fill in every other line. Mathematically,

$$\hat{F}[x,y,n] = \begin{cases} F_i[x,y-1,n] & \text{if } n \equiv 0 \pmod 2, \\ F_i[x,y+1,n] & \text{otherwise.} \end{cases} \tag{2.3}$$

Line repetition is illustrated in figure 2-1 below. While line repetition is simple, it causes both aliasing and jittering [3] in the de-interlaced output.



(a) Field n (b) Field n+1

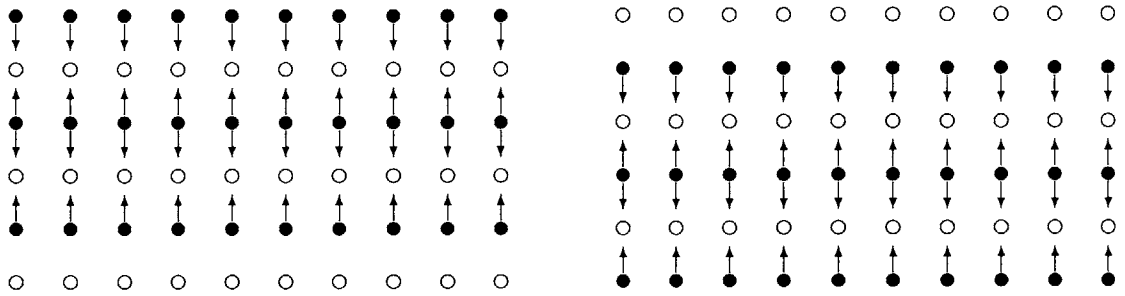
Figure 2-1: Line-repetition de-interlacing. Filled-in circles represent the pixels that are available and the empty circles represent the pixels to be estimated.

Better results can be achieved by using line averaging, which generates a missing line by taking the mean of the scan lines immediately above and below it. Hence,

$$\hat{F}[x,y,n] = \frac{F_i[x,y-1,n] + F_i[x,y+1,n]}{2} \tag{2.4}$$

Figure 2-2 illustrates the pixels that are used for line averaging. Known as "Bob" this technique is one of the most popular because it is simple to implement and

produces a much better picture at the cost of slightly more computations than line repetition. While it results in a smoother picture than line repetition, there is still notable aliasing and jittering.



(a) Field n (b) Field n+1

Figure 2-2: Line-averaging ("Bob") de-interlacing

Edge Detection

These techniques rely on edge detection algorithms to interpolate along the detected edges with the aim of preserving it. In previous work, Doyle et al. [5] looks at pixels in the neighborhood around the pixel to be interpolated, determines the least harmful orientation, and then interpolates along that direction. In his work,

$$\hat{F}[x,y,n] = \begin{cases} \frac{1}{2}(a+f) & \text{if } (|a-f| < |c-d|) \text{ and } (|a-f| < |b-e|), \\ \frac{1}{2}(c+d) & \text{if } (|c-d| < |a-f|) \text{ and } (|c-d| < |b-e|), \\ \frac{1}{2}(b+e) & \text{otherwise.} \end{cases} \tag{2.5}$$

where

$$\begin{aligned} a &= F_i[x-1, y-1, n] & d &= F_i[x-1, y+1, n] \\ b &= F_i[x, y-1, n] & e &= F_i[x, y+1, n] \\ c &= F_i[x+1, y-1, n] & f &= F_i[x+1, y+1, n] \end{aligned} \tag{2.6}$$

Figure 2-3 shows the points that are used for de-interlacing as described above.

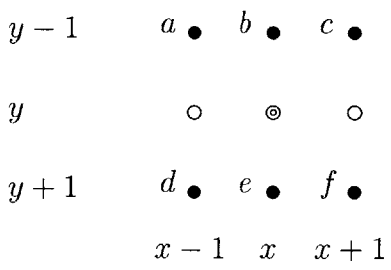


Figure 2-3: Aperture of edge-preserving de-interlacing. The concentric circle in the middle represents the pixel to be estimated.

Salonen et al. [6] attempts to enforce consistency of edge directions by checking the edge orientations of neighboring pixels using *directional edge-detection operators.*

Parametric Image Modeling

By assuming a model for the local region around the pixel to be estimated, it is possible to determine the spatial path of a pixel, and then interpolate along this direction. Martinez and Lim [7] suggested a *line shift model* in which vertically adjacent scan lines are assumed to be related by horizontal shifts. In other words,

$$F_p[x, y, n] = F_p[x - v(y - y_0), y_0, n] \tag{2.7}$$

where v is the local horizontal shift per scan line. Using the parametric model described in [7], it is possible to estimate v from a selection of pixels around the desired

pixel. We then perform the interpolation:

$$\hat{F}[x, y, n] = \frac{F_i[x - v, y - 1, n] + F_i[x + v, y + 1, n]}{2} \tag{2.8}$$

This is illustrated in figure 2-4.

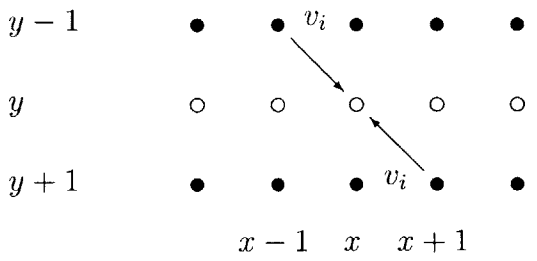


Figure 2-4: Martinez-Lim de-interlacing

However, we recognize that v in general will not be an integer value. Hence, the values $F_i[x - v, y - 1, n]$ and $F_i[x + v, y + 1, n]$ would have to be obtained through some kind of interpolation, such as bilinear interpolation.

Ayazifar and Lim [8] built upon the above work and proposed a *concentric circular shift model* where small segments of arcs are assumed to be related to adjacent arcs through an angular shift.

2.2.2 Non-motion-compensated Interframe Techniques

Temporal Filtering

Analogous to spatial filtering, temporal filtering techniques exploit correlation in the time domain. Instead of line repetition, field repetition can be used to fill in the missing lines by making use of pixels in the previous field. Mathematically,

$$\hat{F}[x, y, n] = F_i[x, y, n - 1] \tag{2.9}$$

Figure 2-5 illustrates how this method works. Also known as "Weave", it performs well for stationary objects, but causes serration of moving edges [3].

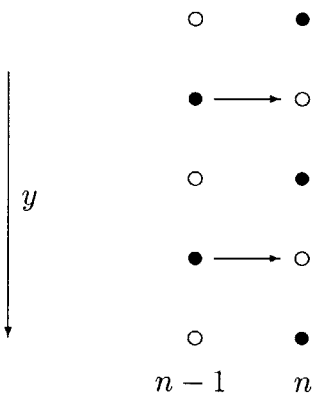


Figure 2-5: Field repetition("Weave") de-interlacing. Each filled-in circle represents a scan line that is available and each empty circle represents a scan line to be estimated.

Another simple temporal filter would be bilinear field interpolation, where

$$\hat{F}[x, y, n] = \frac{F_i[x, y, n-1] + F_i[x, y, n+1]}{2} \tag{2.10}$$

Figure 2-6 illustrates the scan lines that are used for this operation.

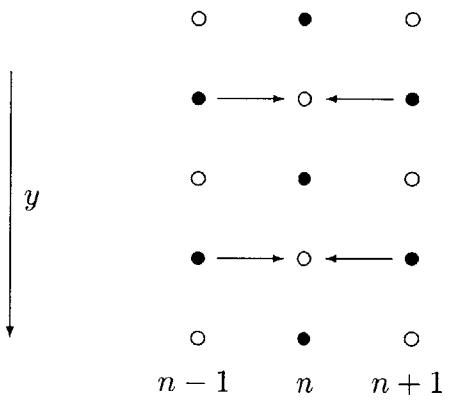


Figure 2-6: Bilinear field de-interlacing

If the input had been band-limited prior to interlacing, then it can be fully recovered by a vertical-temporal (VT) filter [3]. Unfortunately, this is seldom the case, so it remains a textbook solution. However, with a well-implemented VT filter, it is still able to do reasonably well by allowing a small amount of aliasing and blurring.

Motion Adaptive

Motion adaptive algorithms use a mix of interpolation techniques for different parts of the image, depending on whether a local region is classified as being stationary or as being in motion. Temporal filtering performs well for stationary images, while spatial filtering does a reasonable job otherwise. A variety of motion adaptive techniques has been proposed [9, 10], and they differ in how motion detection is accomplished and how fading between different interpolation methods is performed.

Median Filtering

Median filtering is an implicit adaptive method, in the sense that its frequency response changes depending on the region it is applied on. As its name suggests, each missing pixel is taking to be the median of a set of spatially and temporally neighboring points. The simplest form of VT median filtering is a three-tap median filter, where the immediate vertical neighbors and the temporal neighbor in the previous field are considered [11]. Hence,

$$\hat{F}[x, y, n] = \text{median}(F_i[x, y-1, n], F_i[x, y+1, n], F_i[x, y, n-1]) \tag{2.11}$$

This is shown graphically in Figure 2-7.

If a region is stationary, $F_i[x, y, n-1]$ is likely to have a value between $F_i[x, y-1, n]$ and $F_i[x, y+1, n]$. Hence, it will be chosen, resulting in field repetition de-interlacing which works well in stationary regions. On the other hand, in a moving region, the

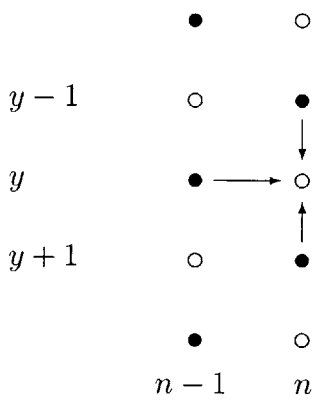


Figure 2-7: 3-tap median filtering

two vertical neighbors are likely to be closer in values. Therefore, one of them will be chosen depending on which of them is closer to $F_i[x, y, n-1]$ in value.

By increasing the number of taps, a VT median filter can be made to be edge adaptive as well [12]. Using the neighborhood of pixels as defined in equation (2.6), Salo et al. [12] suggested the following median operation:

$$\hat{F}[x, y, n] = \text{median}(a, b, c, d, e, f, F_i[x, y, n-1]) \tag{2.12}$$

2.2.3 Motion-compensated Interframe Techniques

Motion-compensated (MC) techniques are the most advanced class of techniques available for de-interlacing [3]. The main characteristic of MC de-interlacing is that it attempts to remove motion from each image by making use of motion vectors, effectively making the video sequence stationary. De-interlacing can then be done by using one of the interframe methods, which performs better for stationary images.

Motion vectors will of course have to be estimated from the video sequence before MC de-interlacing can be carried out. While dense pixel-based vector fields will yield better de-interlacing performance, they are computationally more expensive to

estimate than block-based motion vectors used for MPEG-2 encoding. Furthermore, sub-pixel accuracy in the motion vector itself is required [3].

Unfortunately, accurate motion estimation is hard to implement, and it is likely that some estimates of motion vectors will be incorrect. This is especially true in the case of interlaced video. Hence, MC de-interlacers have to be robust against such errors. Furthermore, in the presence of complex motion such as rotation or zooming, MC de-interlacing is unlikely to have any significant improvement over spatial interpolation.

2.3 Frame Resizing

The problem of frame resizing can be viewed as a sampling rate change problem. Unlike de-interlacing, most frame resizing methods proposed thus far are intra-frame in nature. In other words, they treat the problem of video resizing as a series of static image resizing problems. It is this image resizing problem that has been addressed in the literature.

If the original progressively-scanned video sequence (with Y rows, X columns and N frames) is represented by $F_p[x, y, n]$, we wish to find a new sequence, represented by $F'_p[x', y', n]$, with a frame size of Y' rows and X' columns.

2.3.1 Time-domain Techniques

The most straight forward way of performing the resizing is to fit the image data with a continuous model, and then resample this function on an appropriate sampling lattice. We could approach this problem by transforming the image such that it exist in a square of height and width 1.0. In other words, the original image will now be

represented by $f_p(x_t, y_t, n)$, such that:

$$x_t = \frac{x}{X} \qquad y_t = \frac{y}{Y} \tag{2.13}$$

$$F_p[x, y, n] = f_p\left(\frac{x}{X}, \frac{y}{Y}, n\right) \tag{2.14}$$

Hence, to find the new image sequence, we could use the following:

$$F'_p[x', y', n] = f_p\left(\frac{x'}{X'}, \frac{y'}{Y'}, n\right) \tag{2.15}$$

Of course, $f_p(x_t, y_t, n)$ does not exist for all $0 \leq x_t, y_y < 1$ because of the way it is defined in equation (2.14). To use equation (2.15), we would need to interpolate the values of pixels that are not originally defined.

Sinc Interpolation

Ideally, if the original sampling was performed on a band-limited signal, then according to sampling theory, perfect interpolation can be achieved with a weighted sum of shifted 2-D sinc functions. Mathematically,

$$f_p(x_t, y_t, n) = \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} F_p[x, y, n] \frac{sin(\pi X(x_t - \frac{x}{X}))}{\pi X(x_t - \frac{x}{X})} \frac{sin(\pi Y(y_t - \frac{y}{Y}))}{\pi Y(y_t - \frac{y}{Y})} \tag{2.16}$$

However, this is almost never done in practice, because of the slow decay of sinc and the resulting Gibbs oscillations [13].

Nearest Neighbor Interpolation

Nearest neighbor interpolation guesses each pixel as having the same visual quality as its closest neighbor. In other words,

$$f_p(x_t, y_t, n) = F_p[x_{nn}, y_{nn}, n] \tag{2.17}$$

where

$$x_{nn} = \text{round}(x_t X) \qquad y_{nn} = \text{round}(y_t Y) \tag{2.18}$$

and round(x) returns the nearest integer to x. It is the simplest method to implement, but introduces very noticeable block-like artifacts [13].

Bilinear Interpolation

Bilinear interpolation is slightly more sophisticated, and calculates each new pixel as a linear weighted sum of the 4 closest neighboring pixels. If

$$\begin{aligned} x_f &= \text{floor}(x_t X) & y_f &= \text{floor}(y_t Y) \\ \Delta x_f &= x_t X - x_f & \Delta y_f &= y_t Y - y_f \end{aligned} \tag{2.19}$$

where floor(x) returns the largest integer less than or equal to x, then

$$\begin{aligned} f_p(x_t, y_t, n) = \; & (1-\Delta x_f)(1-\Delta y_f) F_p[x_f, y_f, n] \\ & + (1-\Delta x_f)\Delta y_f F_p[x_f, y_f+1, n] \\ & + \Delta x_f (1-\Delta y_f) F_p[x_f+1, y_f, n] \\ & + \Delta x_f \Delta y_f F_p[x_f+1, y_f+1, n] \end{aligned} \tag{2.20}$$

It performs better than nearest neighbor interpolation, but tends to blur images [13]. Figure 2-8 shows the points used for bilinear interpolation.

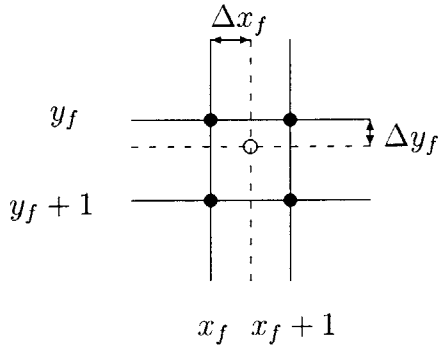


Figure 2-8: Bilinear interpolation. The 4 filled in circles are used to interpolate the empty circle.

Bicubic Interpolation

More powerful models are constructed by polynomials of higher powers [14] and B-splines [15]. An simple example of higher-order interpolation is bicubic interpolation. Using the definitions given in equation (2.19),

$$f_p(x_t, y_t, n) = \sum_{a=-1}^{2} \sum_{b=-1}^{2} F_p[x_f + a, y_f + b, n] R(\Delta x_f - a) R(\Delta y_f - b) \tag{2.21}$$

where the cubic weighting function $R(x)$ is defined as:

$$R(x) = \frac{1}{6}\left[P(x+2)^3 - 4P(x+1)^3 + 6P(x)^3 - 4P(x-1)^3\right] \tag{2.22}$$

$$P(x) = \begin{cases} x & x > 0 \\ 0 & x \leq 0 \end{cases} \tag{2.23}$$

While polynomials of higher degrees are expected to give better performance, they require more computations, and might exhibit over-fitting, leading to strange artifacts. In addition, in the application of image reduction, there might be potential aliasing problems [13].

2.3.2 Frequency-domain Techniques

Another class of methods attempts to perform the resizing in frequency domain. The main reason for doing so is that many image-compression and video-compression methods are based on discrete cosine transform (DCT) coding, so manipulating images in the compressed domain can save computations by avoiding converting to and from the spatial domain [16, 17]. Another reason is that such methods may not be as susceptible to image noise as time-domain methods.

The simplest method of resizing in the DCT domain is to keep all the low-frequency DCT coefficients of the original image [17, 18]. Typical images have very little energy

in the high frequency components in the DCT domain, hence keeping only the lower frequency components would not result in too large a loss in energy.

In a typical compression application, 8x8 or 16x16 block DCTs are used, and so the arbitrary resizing needed for format conversion is in general hard to implement purely in the DCT domain. However, in this thesis, both the input and output are video in the spatial domain, and so an arbitrary (but appropriate) DCT block size can be used. The specific implementation used in this thesis will be described in the next chapter.

Chapter 3

Format Conversion System

To perform format conversion, all that is required is to use the techniques described in Chapter 2 to carry out de-interlacing or frame resizing where appropriate.

3.1 Format Conversion From 720p To 480p

As illustrated in figure 3-1, the conversion from 720p to 480p involves a single step of frame resizing. It consists of down-sampling by a factor of 3:2 (from 720 lines to 480 lines) in the vertical direction, and down-sampling by a factor of 20:11 (from 1280 pixels to 704 pixels) in the horizontal direction.

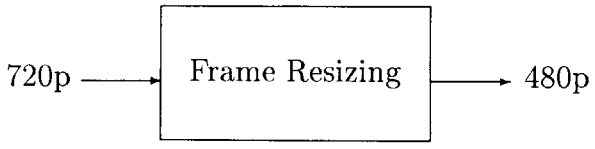


Figure 3-1: Block diagram for 720p to 480p format conversion

In this thesis, nearest neighbor interpolation, bilinear interpolation and bicubic interpolation as discussed in Chapter 2 are used for frame resizing. In addition, the line shift model proposed by Martinez and Lim [7] for de-interlacing is also extended to the problem of frame resizing. This is described below in 3.1.1. Resizing in the DCT domain is also used, and is described in 3.1.2. Hence, a total of five methods are used for the 720p to 480p format conversion.

3.1.1 Resizing Using Martinez-Lim De-interlacing

Frame resizing can be seen as being closely related to de-interlacing. When performing de-interlacing, we wish to interpolate the missing lines. However, the previous frame does not really help in estimating the needed pixels when performing resizing because unlike de-interlacing, the available pixel positions do not alternate from frame to frame. Hence, it would suffice to apply the best spatial de-interlacing method to frame resizing. A good candidate would be Martinez-Lim's line shift model.

The idea is as follows: we would first interpolate in the vertical direction, filling in the scan lines which are missing as per de-interlacing. However, because the missing lines may not necessarily lie in the middle of two existing scan lines, we would need to modify the scheme as shown in equation (2.8). Instead, we would have to rescale the velocity accordingly, and then weigh the contributions according to which existing line the missing line is closer to. One way to do so is:

$$f_p(x_t, y_t, n) = \frac{(1-\Delta y_f)F_p[x-\Delta y_f v, y_f, n] + \Delta y_f F_p[x+(1-\Delta y_f)v, y_f+1, n]}{2} \tag{3.1}$$

where y_f and Δy_f are as defined in equation (2.19) and illustrated in figure 2-8.

We can then interpolate in the horizontal direction in a similar fashion. This would be analogous to de-interlacing of the image columns (as opposed to rows).

3.1.2 Resizing in the DCT Domain

Suppose we want to resize an image by a factor of N_y:M_y in the vertical direction, and N_x:M_x in the horizontal direction. For the 720p to 480p format conversion, $N_y = 3$, $N_x = 20$, $M_y = 2$, $M_x = 11$. Because this resizing method can be used for both down-conversion and up-conversion, there is no constraint on the relationship between N_y and M_y, and between N_x and M_x.

The idea is to divide up the input image into non-overlapping blocks of block size $N_y \times N_x$. Let each of these blocks be denoted by $F_{ij}[x, y, n]$, where $(i \cdot N_x, j \cdot N_y)$ is the position of the block within $F_p[x, y, n]$. We then take the $N_y \times N_x$ DCT of each block, producing $C_{ij}[k_x, k_y, n]$, which is defined as:

$$C_{ij}[k_x, k_y, n] = \sum_{x=0}^{N_x-1} \sum_{y=0}^{N_y-1} 4F_{ij}[x, y, n] \cos\left(\frac{\pi}{2N_x} k_x(2x+1)\right) \cos\left(\frac{\pi}{2N_y} k_y(2y+1)\right) \tag{3.2}$$

for $0 \leq k_x \leq N_x, 0 \leq k_y \leq N_y$.

Next, we keep the lowest $M_y \times M_x$ terms of the DCT of each block. For a down-conversion, $M_y < N_y$ and $M_x < N_x$, and so we just need to discard the higher frequency terms. For an up-conversion however, $M_y > N_y$ and $M_x > N_x$, and hence we have more terms in the new DCT block than originally. One way to handle this is just to set all the extra terms to 0.

Before taking the inverse DCT of the new block, it is necessary to rescale the DCT terms. This is done to normalize the total energy, such that the average energy of the whole picture remains the same, which results in the resized output appearing similar to the input. One way to do so is to require that the average energy per pixel in each block remains approximately the same, i.e.

$$\frac{1}{N_x \cdot N_y} \sum_{x=0}^{N_x-1} \sum_{y=0}^{N_y-1} |F_{ij}[x, y, n]|^2 \approx \frac{1}{M_x \cdot M_y} \sum_{x'=0}^{M_x-1} \sum_{y'=0}^{M_y-1} |F'_{ij}[x', y', n]|^2 \tag{3.3}$$

for all i, j and n. $F'_{ij}[x', y', n]$ represents the resized block at $(i \cdot M_x, j \cdot M_y)$ of the resized

image, $F'_p[x', y', n]$. Applying the energy relationship of the DCT to equation (3.3), we have:

$$\frac{1}{4(N_x \cdot N_y)^2} \sum_{k_x=0}^{N_x-1} \sum_{k_y=0}^{N_y-1} w_x(k_x) w_y(k_y) |C_{ij}[k_x, k_y, n]|^2 \approx \frac{1}{4(M_x \cdot M_y)^2} \sum_{k'_x=0}^{M_x-1} \sum_{k'_y=0}^{M_y-1} w'_x(k'_x) w'_y(k'_y) |C'_{ij}[k'_x, k'_y, n]|^2 \tag{3.4}$$

where $w_x(k_x)$, $w_y(k_y)$, $w'_x(k'_x)$ and $w'_y(k'_y)$ are weighting functions defined as:

$$w_x(k_x) = \begin{cases} \frac{1}{2}, & k_x = 0 \\ 1, & 1 \le k_x \le N_x - 1 \end{cases} \qquad w_y(k_y) = \begin{cases} \frac{1}{2}, & k_y = 0 \\ 1, & 1 \le k_y \le N_y - 1 \end{cases}$$

$$w'_x(k'_x) = \begin{cases} \frac{1}{2}, & k'_x = 0 \\ 1, & 1 \le k'_x \le M_x - 1 \end{cases} \qquad w'_y(k'_y) = \begin{cases} \frac{1}{2}, & k'_y = 0 \\ 1, & 1 \le k'_y \le M_y - 1 \end{cases} \tag{3.5}$$

$C'_{ij}[k'_x, k'_y, n]$ is the $M_y \times M_x$ DCT of each of the resized blocks, $F'_{ij}[x', y', n]$. Equation (3.4) tells us that each of the DCT term should be scaled by:

$$\text{Scale factor} = \frac{M_x \cdot M_y}{N_x \cdot N_y} \tag{3.6}$$

Now, we can obtain an expression for $C'_{ij}[k'_x, k'_y, n]$, which is:

$$C'_{ij}[k'_x, k'_y, n] = \frac{M_x \cdot M_y}{N_x \cdot N_y} C_{ij}[k'_x, k'_y, n] \tag{3.7}$$

for a down-conversion, and

$$C'_{ij}[k'_x, k'_y, n] = \begin{cases} \frac{M_x \cdot M_y}{N_x \cdot N_y} C_{ij}[k'_x, k'_y, n], & \text{for } k'_x < N_x, k'_y < N_y \\ 0, & \text{otherwise} \end{cases} \tag{3.8}$$

for a up-conversion.

The last step is to take the inverse DCT of each of the new DCT blocks, and just place those resized blocks in the output image. The inverse DCT is given by:

$$F'_{ij}[x', y', n] = \frac{1}{M_x \cdot M_y} \sum_{k'_x=0}^{M_x-1} \sum_{k'_y=0}^{M_y-1} w'_x(k'_x) w'_y(k'_y) C'_{ij}[k'_x, k'_y, n] \cos\left(\frac{\pi}{2M_x} k'_x(2x'+1)\right) \cos\left(\frac{\pi}{2M_y} k'_y(2y'+1)\right) \tag{3.9}$$

for $0 \leq x' \leq M_x, 0 \leq y' \leq M_y$.

3.2 Format Conversion From 1080i To 480p

Figure 3-2 shows the steps involved in the conversion from 1080i to 480p. De-interlacing is first carried out, followed by frame resizing. In this conversion, frame resizing consists of down-sampling by a factor of 9:4 (from 1080 lines to 480 lines) in the vertical direction, and down-sampling by a factor of 30:11 (from 1920 pixels to 704 pixels) in the horizontal direction.

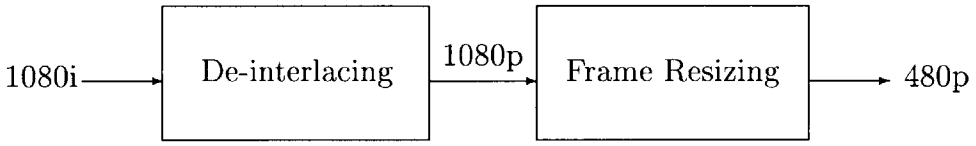


Figure 3-2: Block diagram for 1080i to 480p format conversion

It should be noted that it is possible to perform both de-interlacing and frame resizing in a single step. While doing so could be computationally more efficient, it requires that software be written for each combination of de-interlacing and resizing,

which results in a significantly longer development time. In addition, it is not clear if combining both operations would result in video output of higher quality. Therefore, de-interlacing and frame resizing are carried out as two separate steps in this thesis.

For de-interlacing, the following methods are used: line repetition, line averaging, Martinez-Lim de-interlacing, field repetition, bilinear field interpolation, motion adaptive de-interlacing, 3-tap vertical-temporal median filtering as described by equation (2.11) and 7-tap vertical-temporal median filtering as described by equation (2.12). In addition, a variation of the 3-tap median filtering was used. This is described below in 3.2.2. Hence, there is a total of nine methods considered for de-interlacing.

The five methods used for frame resizing in 3.1 are used here as well.

3.2.1 Motion Adaptive De-interlacing

While motion adaptive de-interlacing was described in Chapter 2, there are many variations available. In particular, there are different ways of performing motion detection and fading between spatial and temporal de-interlacing methods.

In this thesis, motion detection is accomplished by comparing the neighborhood of a pixel in the field before and after the current field. Figure 3-3 shows the comparisons that are made. Note that fields $n-1$ and $n+1$ need to be de-interlaced prior to motion detection, and this can be achieved by using any intraframe de-interlacing method, such as Martinez-Lim de-interlacing.

Mathematically, the amount of motion $m[x, y, n]$ is calculated as follows:

$$m[x, y, n] = \max(A, \frac{1}{4}(B + C + D + E)) \tag{3.10}$$

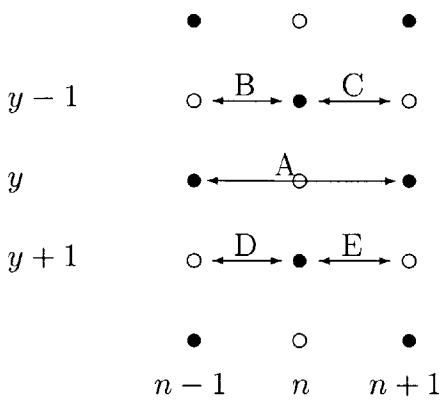


Figure 3-3: Motion detection

where

$$\begin{aligned} A &= |F_i[x, y, n+1] - F_i[x, y, n-1]| \\ B &= |\hat{F}_p[x, y-1, n-1] - F_i[x, y-1, n]| \\ C &= |\hat{F}_p[x, y-1, n+1] - F_i[x, y-1, n]| \\ D &= |\hat{F}_p[x, y+1, n-1] - F_i[x, y+1, n]| \\ E &= |\hat{F}_p[x, y+1, n+1] - F_i[x, y+1, n]| \end{aligned} \tag{3.11}$$

Here, spatial de-interlacing is performed using Martinez-Lim, while temporal de-interlacing is performed using a median filter described mathematically as:

$$\hat{F}[x, y, n] = \text{median}(F_i[x, y, n-1], F_i[x, y, n+1], \frac{F_i[x, y-1, n] + F_i[x, y+1, n]}{2}) \tag{3.12}$$

This median filter will perform better than field repetition, because it will choose a point from either the previous or the next frame depending on which point is closer to the estimate of the point in the current frame.

Fading is accomplished by using the following equation:

$$\hat{F}[x, y, n] = \alpha \hat{F}_{\text{spatial}}[x, y, n] + (1 - \alpha)\hat{F}_{\text{temporal}}[x, y, n] \tag{3.13}$$

where $\hat{F}_{\text{spatial}}[x, y, n]$ is the spatially de-interlaced image, $\hat{F}_{\text{temporal}}[x, y, n]$ is the temporally de-interlaced image, and α is a parameter that determines the fading between the spatially de-interlaced image and the temporally de-interlaced image. Here, α is a non-linear function of the amount of motion $m[x, y, n]$. This function is shown in the graph in figure 3-4.

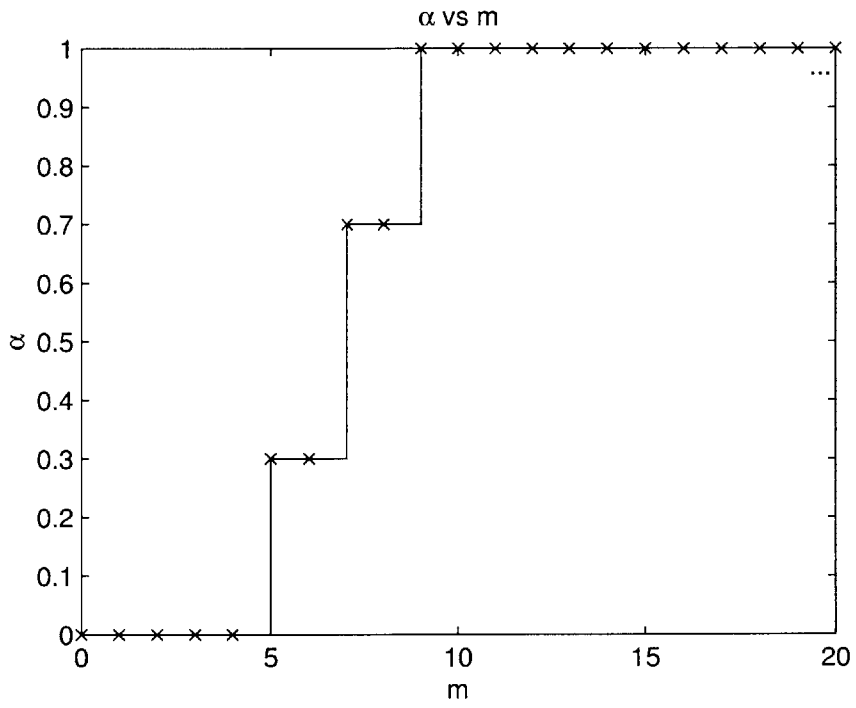


Figure 3-4: Graph of α vs m

3.2.2 De-interlacing Using Martinez-Lim/3-tap Median Filtering

As shown in figure 2-7, the 3-tap median filter uses the immediate vertical neighbors and the temporal neighbor in the previous field for de-interlacing. This can be improved by first determining the line-shifts using the parametric model presented by

Martinez, and then using that information to decide which points to consider in the median operation. Hence,

$$\hat{F}[x, y, n] = \text{median}(A, B, C) \tag{3.14}$$

where

$$\begin{aligned} A &= F_i[x - v, y - 1, n] \\ B &= F_i[x + v, y + 1, n] \\ C &= F_i[x, y, n - 1] \end{aligned} \tag{3.15}$$

Consider the definitions for the points A, B and C as given in equation 3.15. A corresponds to the point on the edge in the scan line above, B corresponds to the point to the edge in the scan line below, and C corresponds to the point that is the temporal neighbor in the previous field. If there is motion in the region, then A and B will be more closely related than C, and one of the two points will be chosen by the median filter, which results in the preservation of the edge. If the region is stationary, then C is likely to be between A and B, and will hence be chosen.

In the thesis, this method will be referred to as Martinez-Lim/3-tap VT median filtering.

Chapter 4

Experimental Results

In this thesis, a number of experiments are performed to determine the best procedure for carrying out each format conversion. For the 720p to 480p format conversion, five different frame resizing methods are tested. They are listed in table 4.1. For the 1080i to 480p format conversion, a total of nine de-interlacing methods, as listed in table 4.2, is used. These algorithms are used in conjunction with the five frame resizing methods, giving rise to 45 possible format converters which are tested.

Algorithm Code	**Algorithm Name**
NN	Nearest Neighbor Interpolation
BL	Bilinear Interpolation
BC	Bicubic Interpolation
ML	Martinez-Lim Interpolation
DCT	DCT Domain Resizing

Table 4.1: Frame resizing methods used for testing

The algorithm codes (for example, NN for nearest neighbor interpolation) listed in the tables 4.1 and 4.2 will be used in this chapter to refer to the corresponding algorithms.

Algorithm Code	Algorithm Name
LR	Line Repetition
BOB	Line Averaging
ML	Martinez-Lim De-interlacing
WEAVE	Field Repetition
BL	Bilinear Field Interpolation
VT3	3-tap Vertical-Temporal Median Filtering
VT7	7-tap Vertical-Temporal Median Filtering
MA	Motion Adaptive De-interlacing
MLVT3	Martinez-Lim/3-tap Vertical-Temporal Median Filtering

Table 4.2: De-interlacing methods used for testing

When possible, Peak Signal-to-Noise Ratio (PSNR) is used as a quantitative measure of video quality, and is defined as:

$$\text{PSNR} = 10 \log_{10} \left(\frac{255^2}{\text{MSE}} \right) \tag{4.1}$$

where Mean Square Error (MSE) is defined to be the average squared difference between all channels of the original and the resulting video. Mathematically,

$$\text{MSE} = \frac{1}{N \cdot X \cdot Y} \sum_{n=0}^{N-1} \sum_{x=0}^{X-1} \sum_{y=0}^{Y-1} \left| \hat{F}[x, y, n] - F[x, y, n] \right|^2 \tag{4.2}$$

where $\hat{F}[x, y, n]$ is the sequence to be compared and $F[x, y, n]$ is the original video sequence.

While PSNR and perceived video quality of a sequence are not always perfectly correlated, PSNR provides a quantity that can be used for comparison. Given the PSNR of an output sequence, it is also possible to have a good idea of its video quality.

4.1 Source Material

A total of four different video sequences were used as source material. Each is a different combination of static/moving background and static/moving foreground.

4.1.1 Computer Generated Static Video Sequence

The first sequence, referred to as 'Static' in the thesis, consists of 10 frames of the same computer generated image. The image, shown below in figure 4-1, has four quadrants. Each of the quadrants consists of either straight bars, horizontal bars, circular bars or diagonal bars.

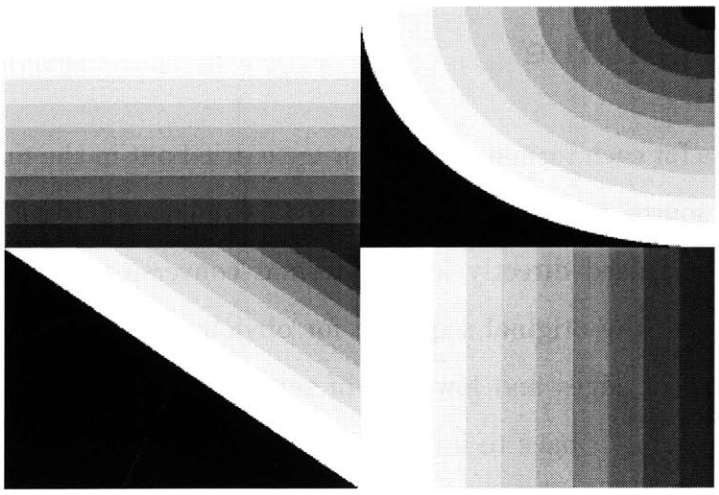


Figure 4-1: 'Static' sequence

The image is generated by using a set of mathematical definitions for each quadrant. Since it is computer generated, the same image can be rendered at different resolutions, which makes it useful for obtaining data for quantitative comparisons. To prevent aliasing, the image is first rendered at twice the required height and width. Then, it is low-pass filtered and sub-sampled to obtain the final image.

4.1.2 Captured Video Sequences

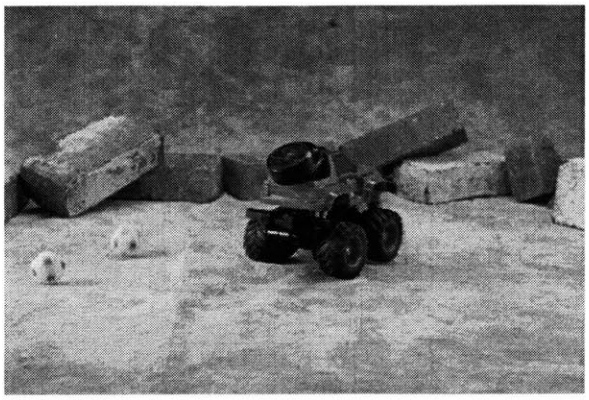
Three other video sequences were selected for testing. The first sequence is named 'Car', which is a video of a moving car in a largely static background. The second sequence, named 'Tulipz', consists of zooming and panning of a static background. The last sequence is named 'Train', and has a moving train in a panning background. Summaries of these sequences are shown in figure 4-2.

Since these sequences come in varying frame sizes, it may be necessary to crop them before carrying out tests. More details about the preparation of input data would be given for each specific format conversion.

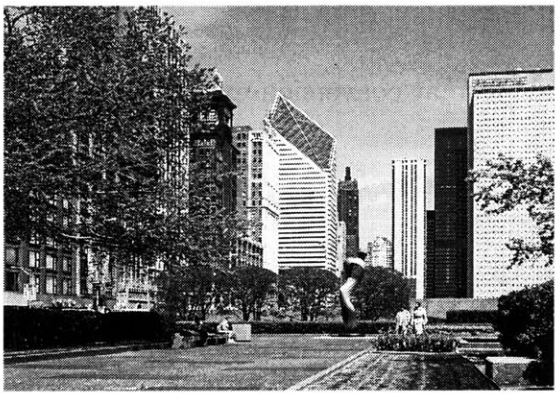
4.2 Test Outline

The experiments for each format conversion are carried out in the manner as shown in figure 4-3. The source materials described in 4.1 are treated as input data (in either 720p or 1080i) and used directly for the format conversion. The output is viewed and compared with the original sequence for obvious degradations such as aliasing, jittering, blurring of edges and lowering of text readability. It is also subjected to up-conversion to bring it back to its original format. That is then compared with the original source to obtain a PSNR for analysis.

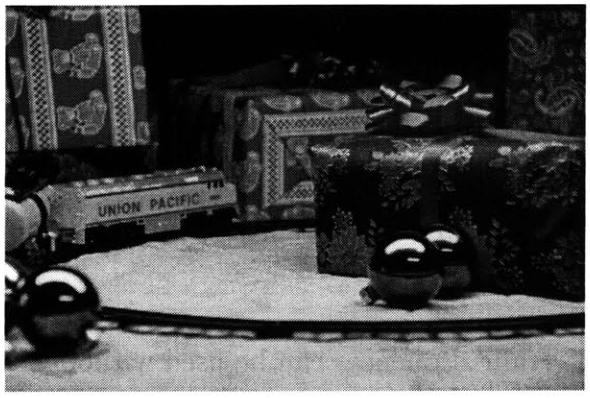
We recognize that the PSNR obtained may not be an accurate measure of the performance of the format converter. However, with the exception of the 'Static' sequence which can be generated for both input and output formats, the rest of the video sequences exist in only one format. This presents a problem in the collection of quantitative data. In the absence of better options, the scheme described in figure 4-3 provides a tolerable way of obtaining a quantitative measure of how well each format converter does.



Name: Car
Scan Mode: Progressive
Frames: 17
Rows: 480
Cols: 720
Frame Rate: 30 frames/sec



Name: Tulipz
Scan Mode: Progressive
Frames: 20
Rows: 720
Cols: 1024
Frame Rate: 30 frames/sec



Name: Train
Scan Mode: Progressive
Frames: 20
Rows: 480
Cols: 720
Frame Rate: 30 frames/sec

Figure 4-2: Captured video sequences

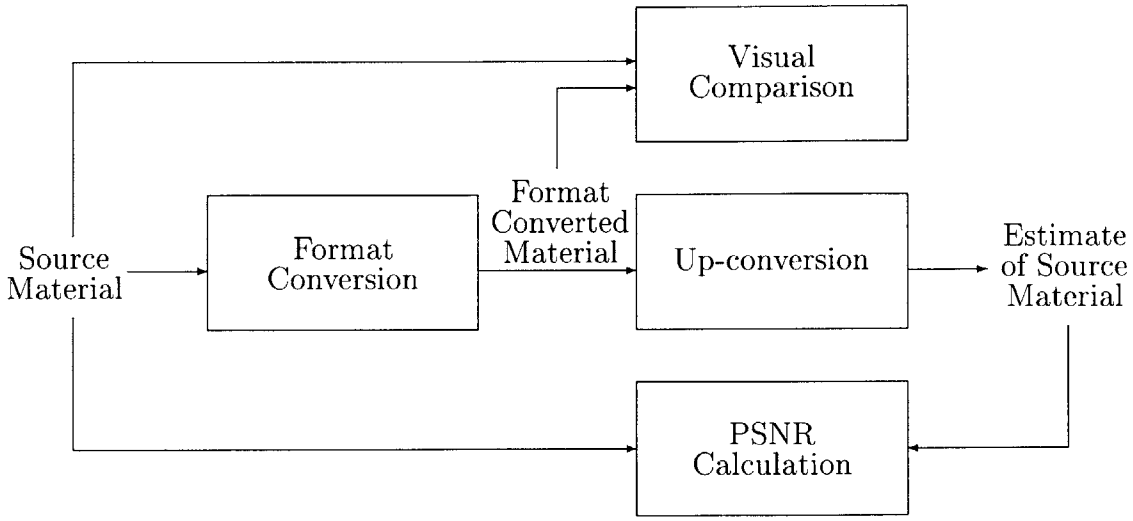


Figure 4-3: Block diagram for format conversion experiments

4.3 Format Conversion From 720p To 480p

The original video sequences are treated as 720p materials. All the sequences have a frame rate of 30 frames/sec, and their frame sizes are given in table 4.3.

Sequence	720p	480p
Static	360 x 640	240 x 352
Car	480 x 720	320 x 396
Tulipz	720 x 1020	480 x 561
Train	480 x 720	320 x 396

Table 4.3: Frame size of source materials used in experiments of 720p to 480p format conversion

The original video of the 'Car' and 'Train' sequences can be used without modification as input data. Input data for the 'Tulipz' sequence is obtained by cropping a 720x1020 window from the original video. The 'Static' sequence can be generated in both the 720p and 480p format by simply changing the parameters of the renderer.

The 720p video sequence is fed as input into each of the five possible format converters and the output is up-converted (using the same resizing algorithm as in the format conversion) to obtain an estimate of the original 720p input. This estimate is compared with the 720p input to obtain a PSNR measure. Note that two sets of PSNR measures can be generated for the 'Static' sequence: one by comparing the format converted 480p output with the generated 480p sequence, and the other by comparing the up-converted 720p output with the original 720p input. Figure 4-4 shows the results of this experiment.

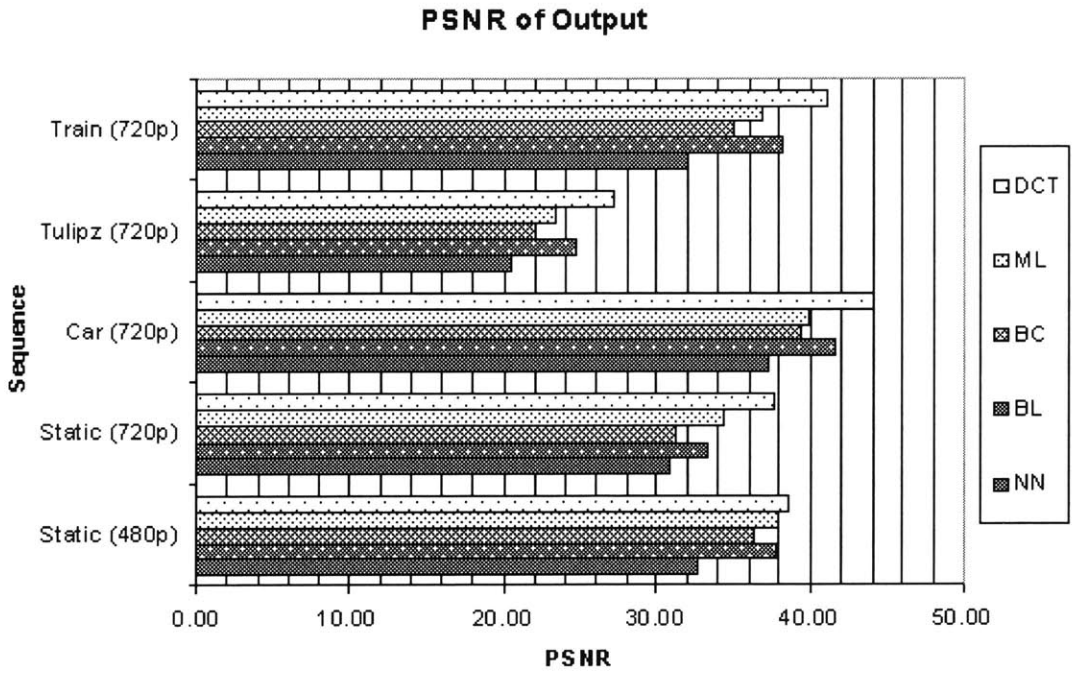


Figure 4-4: PSNR of output of 720p to 480p format conversion experiments for test set 1. Note that for the 'Static' sequence, two sets of PSNR data are available. One compares the format converted 480p output with the computer generated 480p sequence, and the other compares the 720p estimate with the original 720p sequence. The algorithm names for the codes listed in the legend are given in table 4.1

The results show that for all the sequences, resizing in the DCT domain has the

best performance. Martinez-Lim interpolation has the next best performance for the 'Static' sequence, while bilinear interpolation has the next best performance for the remaining three sequences.

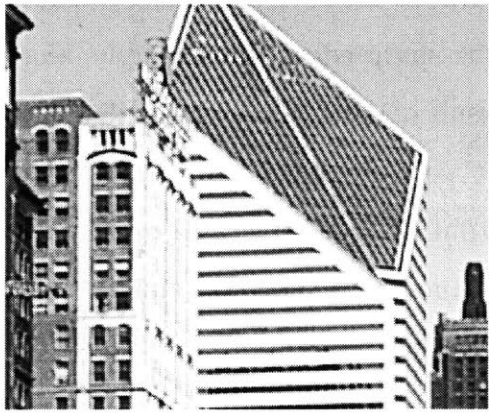
It is no surprise that resizing in the DCT domain gave the highest PSNR. The energy compaction property of the DCT ensures that most of the energy of each block is in the lower frequency components. Hence, the process of down-conversion and subsequent up-conversion amounts to nothing more than discarding all the higher frequency terms of the DCT. The MSE, which in turn affects the PSNR, can be rewritten in terms of the differences of the DCT terms, and that is just the sum of the energy of all the higher frequency components of the DCT. As long as the energy compaction property holds for the image, this method will always give the lowest MSE.

The PSNR results also seem to suggest that Martinez-Lim interpolation performs well for synthetic data where the edges are well-defined. In particular, the 'Static' sequence consists of vertical, horizontal, diagonal and circular edges which are well represented by the parametric model used in Martinez-Lim interpolation. Therefore, the resizing of these edges will be more accurately performed. However, for "real-world" sequences, the behavior of the edges are not so well predicted by the parametric model, and so bilinear interpolation performs better than Martinez-Lim interpolation.

The format converted 480p output is also compared with the original 720p input visually and degradations are observed. This provides another way of judging the performance of each format converted, albeit subjectively.

Figure 4-5 shows a frame from the 'Tulipz' video sequence. The diagonal edges on the top of the building in the center of the frame is a good area to focus on to distinguish the various resizing methods.

The results show that the output from the format converter that uses nearest neighbor interpolation produced the most degradation in video quality. The output



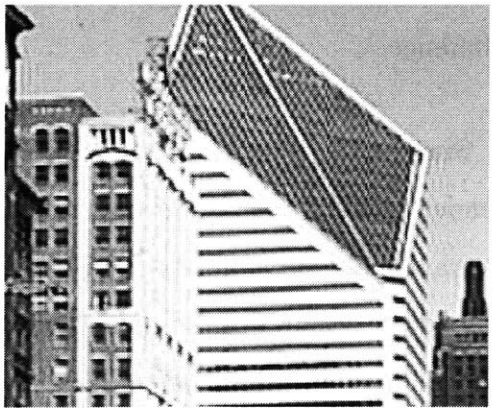
(a) Original frame

(b) Nearest neighbor interpolation

(c) Bilinear interpolation

(d) Bicubic interpolation

(e) Martinez-Lim interpolation



(f) DCT domain resizing

Figure 4-5: Visual results of 720p to 480p format conversion experiments. The 'Tulipz' sequence was used. Shown here is the reconstructed 720p input.

from DCT domain resizing is distinctly the sharpest. However, on closer examination, one can see ringing artifacts around where the sharp edges (for example, along the side of a building) are. This is most likely a result of a block having significant energy content in the higher energy components that were discarded. Bilinear interpolation and Martinez-Lim interpolation produces output of about the same quality. Output from bicubic interpolation has a considerable amount of blurring, mainly because it is using a higher order polynomial for interpolation.

When viewing the 'Tulipz' sequence as a video, the output from nearest neighbor interpolation shows considerable jittering along the edges. The output from DCT domain resizing and bicubic interpolations show minimal jittering, while that from bilinear interpolation and Martinez-Lim interpolation show moderate amounts of jittering.

Figure 4-6 is another example of output from the format converter, this time from the 'Car' sequence. Again, the nearest neighbor interpolation output shows some aliasing degradation, particularly along the edge of the "spare tire" of the toy vehicle. On the other hand, the output from the rest of the resizing methods shows no obvious difference.

Since all the resizing methods being tested are intraframe methods, one might wonder if experiments on single images would be sufficient. We argue that this is not the case, because artifacts from resizing that show up in the format converted video may not be apparent in single image comparisons, and vice versa. For example, for the 'Tulipz' sequence, it is possible to see the jittering of edges (especially around where the windows of the buildings are) when viewing the video. However, this cannot be seen at all in any single frame of the sequence if they were viewed separately. Hence, it is still necessary to do comparisons of whole videos rather than just single images.

(a) Original frame

(b) Nearest-neighbor interpolation

(c) Bilinear interpolation

(d) Bicubic interpolation

(e) Martinez-Lim interpolation

(f) DCT domain resizing

Figure 4-6: Visual results of 720p to 480p format conversion experiments. The 'Car' sequence was used.

4.4 Format Conversion From 1080i To 480p

For this format conversion experiment, the original video sequences are handled as 1080p materials. They are then interlaced to produce 1080i video for use as input in the experiments. All sequences have a frame rate of 30 frames/sec, and their frame sizes are given in table 4.4.

Sequence	1080i	480p
Static	540 x 960	240 x 352
Car	477 x 720	212 x 264
Tulipz	720 x 1020	320 x 374
Train	477 x 720	212 x 264

Table 4.4: Frame size of source materials used in experiments of 1080i to 480p format conversion

For the 'Car' and 'Train' sequence, a 477x720 window is cropped out of the original sequence to use as 1080p material. For the 'Tulipz' sequence, a 720x1020 window is cropped for the same purpose. As in the 720p to 480p format conversion, the 'Static' sequence can be generated in both the input and output format.

The tests follow the block diagram laid out in figure 3-2. For each input sequence, the 1080i version is first fed into one of the nine de-interlacers, and its output compared with the 1080p version to obtain the PSNR of the de-interlaced output. Then, the output from each de-interlacer is fed into each of the five frame-resizers, which gives rise to 45 possible format converted output. Each of those is up-converted to 1080p using the same resizing method as was used in the down-conversion. This produces an estimate of the original 1080p material, which is then compared with the original 1080p material to obtain a PSNR measure.

4.4.1 De-interlacing

Figure 4-7 shows the PSNR of de-interlaced video. As expected, WEAVE gives the best PSNR for the 'Static' sequence since all the frames are the same. Also, all the interframe methods outperform the intraframe methods because they make use of information from other frames.

For the other three sequences, because of changes between frames, the purely temporal methods (WEAVE and BL) are the worst performers in terms of PSNR. LR also does not do very well. The three methods that performs the best for these three sequences are BOB, ML and MA.

For the 'Car' sequence, MA has the highest PSNR, followed by BOB and ML. This is because there is a great deal of stationary regions. Hence, making judicious use of information from previous frames will give better de-interlacing results.

The 'Tulipz' sequence however has BOB as its best de-interlacer. This is due to the fact that there are many fine details which the parametric model in ML cannot pick up well, and so ML does not do as well as BOB. Also, the background is always moving, so it is almost impossible to get any useful information from the previous frame. This implies that MA cannot improve upon the performance of ML, and thus MA is also behind BOB in PSNR.

Finally, the 'Train' sequence has MA as its best performer. Again, this is because the sequence has a slowly moving background.

It should also be noted that the median de-interlacing methods (VT3, VT7 and MLVT3) are well-rounded de-interlacers in the sense that they perform above average for all sequences, regardless of whether they are static or not. For example, they perform better than the intra-frame methods for the 'Static' sequence, and perform better than LR, WEAVE and BL for the other three sequences. This shows the strength of the adaptive behavior implicit in these de-interlacers.

While the PSNR is no doubt useful for analysis, it is still necessary to look at

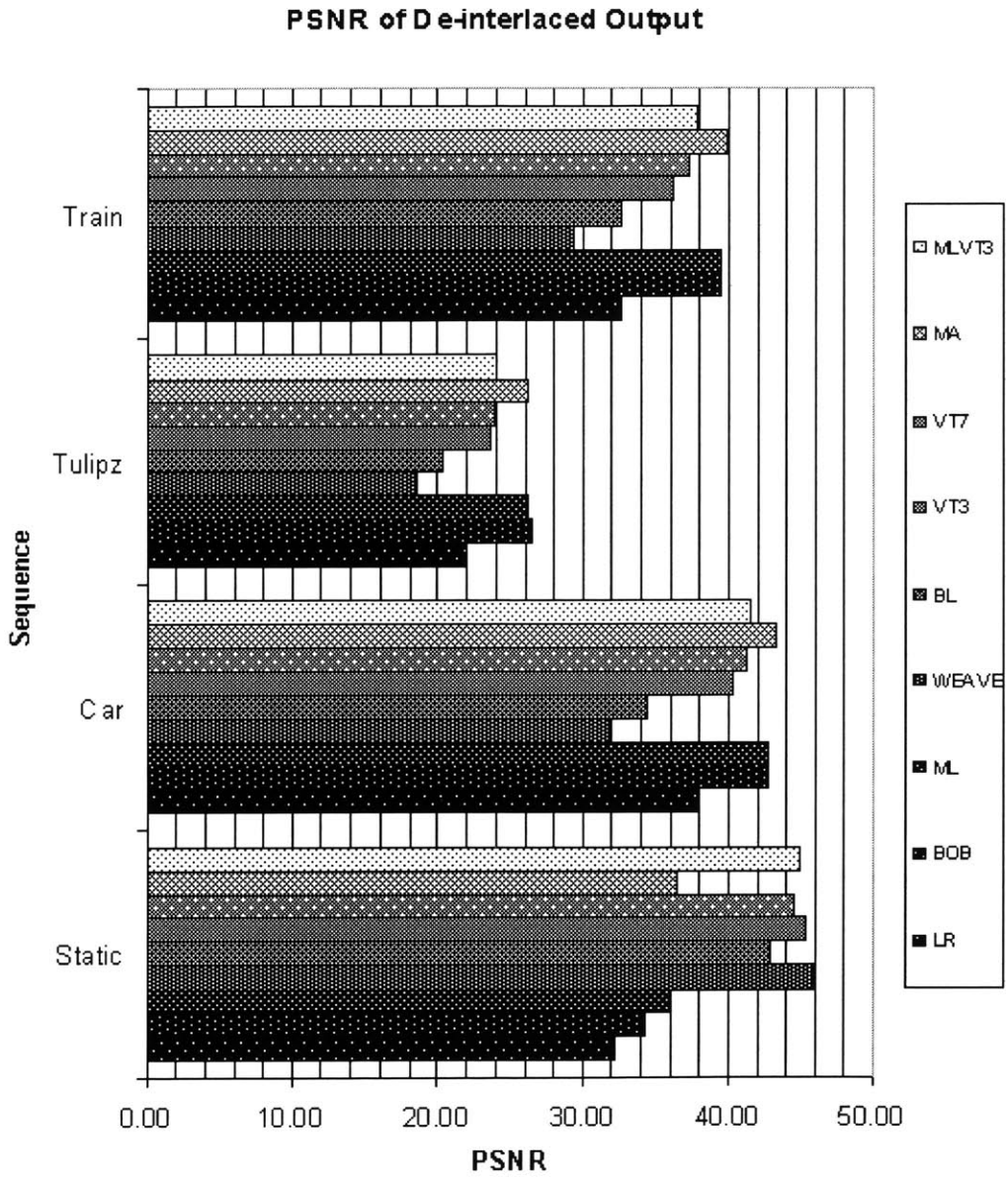


Figure 4-7: PSNR of de-interlaced video in 1080i to 480p format conversion. The algorithm names for the codes listed in the legend are given in table 4.2

each of the de-interlaced videos, because that is the product that viewers will see. As mentioned earlier, it is possible that a video with a lower PSNR could look better than a video with a higher PSNR.

The 'Car' sequence is a typical example of a clip one might see on television, because it has an almost stationary background with a number of moving objects. Hence, it is used here for visual comparison. Figure 4-8 is the original progressively scanned and the corresponding interlaced video frame.

(a) Progressive video (b) Interlaced video

Figure 4-8: Original/Interlaced frame

Figure 4-9 shows the output of de-interlacing using each of the three intraframe de-interlacing methods. In figure 4-9(a), we can see the aliasing that occurs when LR is used. Take note of the edge along the "spare tire" of the toy vehicle. Figure 4-9(b) shows that while the de-interlaced picture using BOB is smoother than in figure 4-9(a), aliasing is still present, particularly along the edge of the "spare tire". The output in figure 4-9(c) uses ML, and shows very little aliasing in the de-interlaced frame. In particular, the edge of the "spare tire" looks very smooth. Clearly, this is

(a) Line repetition de-interlacing

(b) Line averaging de-interlacing

(c) Martinez-Lim de-interlacing

Figure 4-9: Visual results of intraframe de-interlacing

the superior intraframe de-interlacing method. Note that while it is visually superior, ML yielded a slightly lower PSNR than BOB.

The results of the two purely temporal de-interlacing methods are shown in figure 4-10. The serration that is characteristic of this class of methods is clearly visible in both output frames. The output from BL in figure 4-10(b) shows some smoothing of the aberrations, but the serrations are still fairly obvious.

(a) Field repetition de-interlacing (b) Bilinear field interpolation

Figure 4-10: Visual results of temporal de-interlacing

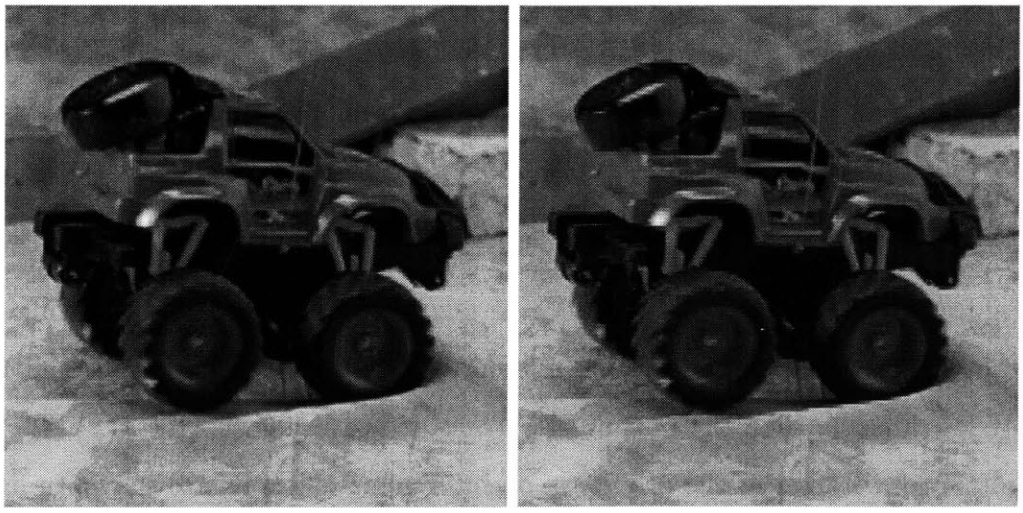
Figure 4-11 shows the performance of median filtering in de-interlacing. As a result of having a wider aperture, the output of VT7 as shown in figure 4-11(b) shows visibly less aliasing than the output of VT3 as shown in figure 4-11(a).

The output frames of the other two de-interlacing methods are shown in figure 4-12. Visually, the output of MA is the best. This comes as no surprise, since it combines the best of spatial de-interlacing and temporal de-interlacing and chooses which to use when appropriate. MLVT3 improves upon the performance of both VT3 and VT7, and reduces the amount of aliasing. However, when compared to MA,

(a) 3-Tap VT median filtering

(b) 7-Tap VT median filtering

Figure 4-11: Visual results of median filtering de-interlacing



(a) Motion adaptive de-interlacing

(b) Martinez-Lim/3-Tap VT median filtering

Figure 4-12: Visual results of alternative de-interlacing

MLVT3's output has visibly more aliasing.

For the 'Tulipz' and 'Train', it was also evident that MA gave the best visual performance. This is in agreement with the high PSNR that it had for its de-interlaced output.

Based on these results, it seems reasonable to assume that the best format conversion system would employ MA for de-interlacing. However, if one knows beforehand that a video program is just a series of stationary frames, then it would of course be better to use WEAVE for de-interlacing.

4.4.2 Resizing

From the 720p to 480p format conversion experiments, we have already know that the resizing algorithms with the best performance are bilinear interpolation, Martinez-Lim interpolation and DCT domain resizing. Hence, there is no need to include the results of nearest neighbor interpolation and bicubic interpolation for analysis purposes.

Similarly, from the de-interlacing experiments, we know that the best de-interlacers are either BOB, ML or MA. Therefore, these are the only three de-interlacers that will be considered for use in the format converter.

With three resizing algorithms and three de-interlacing algorithms, we can construct a total of nine possible format converters. For the purpose of analysis, this is definitely a more manageable number than 45. Nevertheless, tests were still conducted for all 45 format converters, and all the PSNR results can be found in Appendix A.

The nine format converters under consideration would be referred to by a combination of the de-interlacing algorithm code and the resizing algorithm code. For example, BOB/BL refers to the format converter that uses line averaging for de-interlacing, and bilinear interpolation for resizing, while MA/ML refers to the format converter that uses MA for de-interlacing, and Martinez-Lim interpolation for resiz-

ing.

Figures 4-13 shows the PSNR of output from the nine selected format converters. This figure shows a number of important points that are relevant for all the video sequences.

First, MA/DCT is almost always the best performing format converter. For the 'Tulipz' sequence, it is 0.01dB behind BOB/DCT, which for all practical purposes is almost negligible. From earlier results, this is the outcome to be expected. DCT domain resizing is the best frame resizer in the 720p to 480p format conversion experiments, and MA de-interlacing gives the best de-interlacing performance. Hence, we could have guessed that the combination of these two methods would give the best 1080i to 480p format converter, and this is now verified by quantitative data.

Second, given the same de-interlacing algorithm, DCT domain resizing always performs best. While we know from earlier results that DCT domain resizing is the best frame resizer, we did not know how robust it is. In particular, it is possible that given a sub-optimal de-interlacer, DCT domain resizing may not do as well as other resizing methods. The results show that this is not the case.

Third, given the same frame resizing algorithm, MA de-interlacing almost always performs best. Again, we have known MA to be the best de-interlacer from earlier results. Given that de-interlacing is the first step of the 1080i to 480p format conversion, we naturally expect that the final format converted output will depend a great deal on the performance of the de-interlacer. This verifies the hypothesis that was raised in 4.4.1, that the best format converter would employ MA for de-interlacing.

Figure 4-14 shows one frame from the output of the nine format converters under consideration. All the images look very much the same. What this suggests is that because of the large reduction in frame size, it may not matter which de-interlacing method is used in the format conversion. In a similar fashion, it may not matter which frame resizing method is used in the format conversion if the de-interlacer does

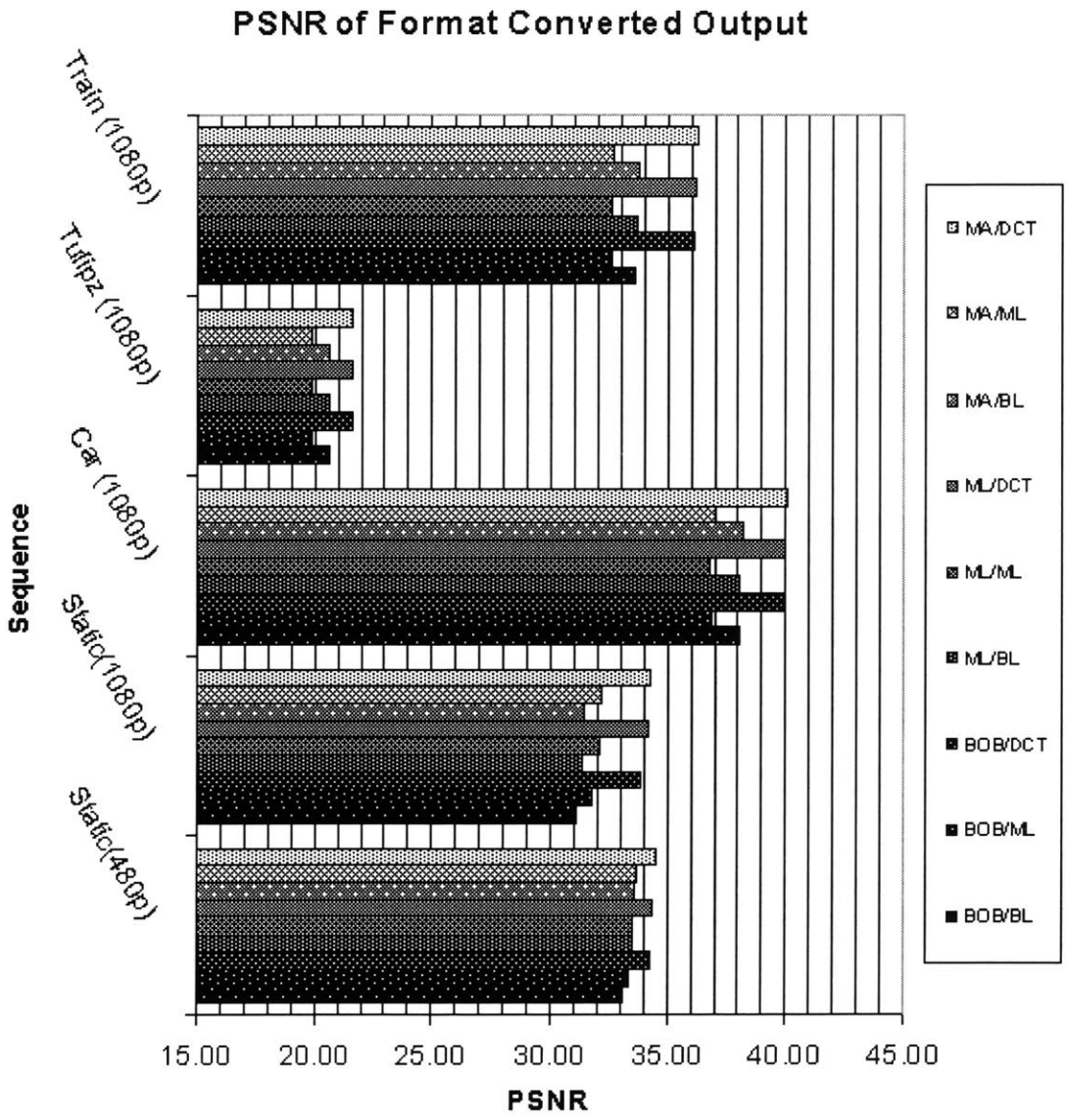
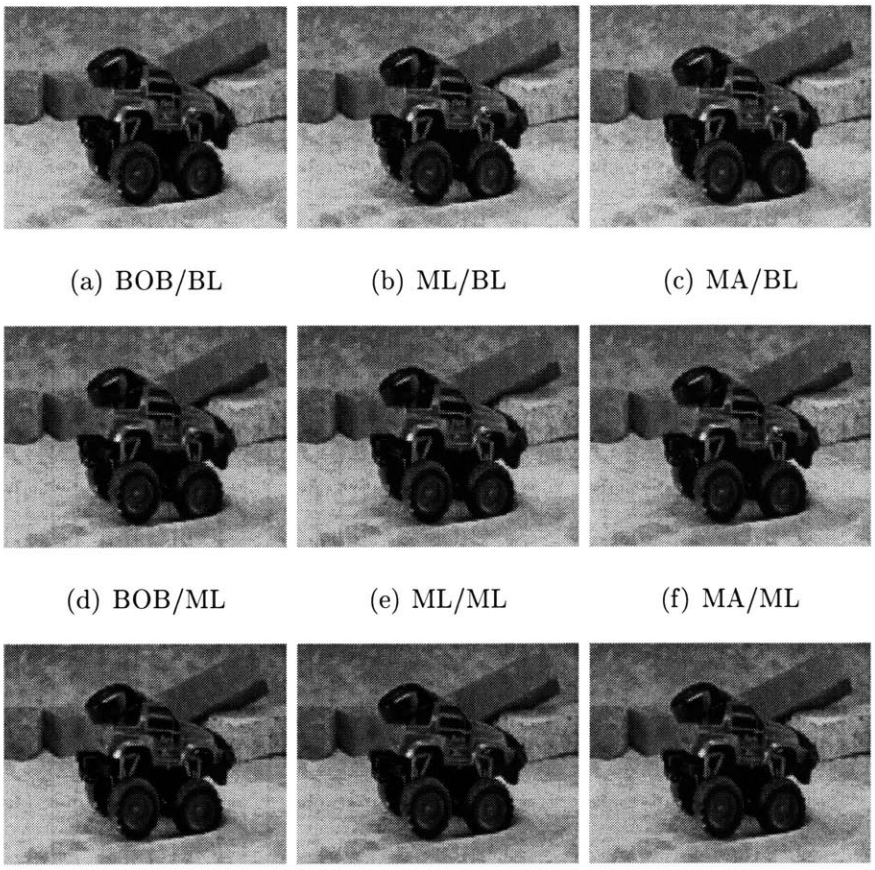


Figure 4-13: PSNR of output of 1080i to 480p format conversion experiments for the selected format converters. Note that for the 'Static' sequence, two sets of PSNR data are available. One compares the format converted 480p output with the computer generated 480p sequence, and the other compares the 1080p estimate with the original 1080p sequence.



(a) BOB/BL (b) ML/BL (c) MA/BL

(d) BOB/ML (e) ML/ML (f) MA/ML

(g) BOB/DCT (h) ML/DCT (i) MA/DCT

Figure 4-14: Visual results of 1080i to 480p format conversion. Results are only shown for the nine format converters under consideration.

a reasonable job.

Therefore, if all that is desired is the 480p output, then any of the nine format converters can be used. However, if the de-interlaced output is important, then MA de-interlacing should be used. In addition, if up-conversion of the format converted output is necessary at a later stage, then resizing in the DCT domain should be used. In any case, the format converter that uses MA de-interlacing and DCT domain resizing will give the best results, even if the difference is not immediately visible.

Chapter 5

Conclusion

5.1 Summary

This thesis investigates the problem of format conversion: how to convert a digital video from one video format to another. In particular, two down-conversions were studied:

- 720p to 480p
- 1080i to 480p

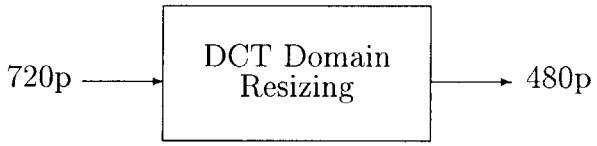
Sub-systems for de-interlacing and frame resizing were needed to perform the format conversions under study. The thesis studied a number of algorithms that can be used to implement each of the sub-system. Tests were performed to determine which algorithms would contribute to the format converter with the best performance.

For the 720p to 480p format conversion, it was found that DCT domain resizing gave output with the highest PSNR. Visually, its performance was better than the other resizing methods.

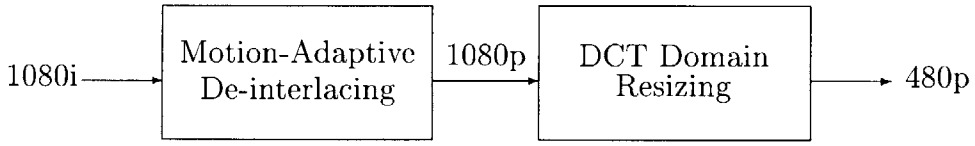
For the 1080i to 480p format conversion, the combination of motion adaptive de-interlacing and DCT domain resizing for frame resizing was found to give the best

visual results. The PSNR of its result was also among the highest.

Figure 5-1 summarizes the best systems found in this thesis.



(a) 720p to 480p



(b) 1080i to 480p

Figure 5-1: Block diagrams for format conversion

5.2 Future Research Directions

This research can be furthered in many ways. One is to expand the scope of this study by investigating other format conversions. That would involve investigating techniques for aspect ratio conversion and frame rate conversion. This thesis only studied image size reduction; it will be interesting to see how the techniques discussed in this thesis would perform when used for video up-conversion; for instance, to go from 480p to 720p.

While computation time is not an issue in this study, real-time format conversion is

important especially for HDTV displays which may have to perform format conversion on the fly. Further research may investigate ways to speed up format conversion.

There has been some recently published image resizing algorithms that make use of high order B-splines and polynomials [19, 20, 21]. These should be studied to see if they would be suitable for frame size conversion. Furthermore, image enhancement should be looked into as a means of improving visual quality. In particular, the work by Greenspan et al. [22] allows one to enhance the perceptual sharpness of a format converted video.

Finally, more quantitative results can be obtained by varying the collection of source material. Specifically, it may be helpful to make use of a computer 3-D renderer to provide video sequences of different resolutions. This will make it possible to obtain meaningful quantitative data for analysis.

Appendix A

Experimental Data

Format Conversion From 720p To 480p

Resizing Algorithm	Static (480p)	Static (720p)	Car	Tulipz	Train
NN	32.70	30.84	37.22	20.40	32.06
BL	37.75	33.32	41.59	24.65	38.24
BC	36.44	31.22	39.37	21.97	35.05
ML	37.96	34.30	39.91	23.25	36.88
DCT	38.55	37.58	44.10	27.13	41.17

Table A.1: PSNR of 720p-480p format conversion experiments

Format Conversion From 1080i To 480p

De-interlacing Algorithm	Static	Car	Tulipz	Train
LR	32.18	37.86	22.03	32.57
BOB	34.16	42.84	26.52	39.40
ML	35.88	42.80	26.12	39.52
WEAVE	45.88	31.80	18.52	29.30
BL	42.87	34.30	20.39	32.64
VT3	45.34	40.29	23.55	36.08
VT7	44.61	41.28	23.86	37.22
MA	36.44	43.29	26.15	39.95
MLVT3	44.99	41.51	24.00	37.70

Table A.2: PSNR of de-interlaced output

De-interlacing Algorithm	NN	BL	BC	ML	DCT
LR	30.69	32.55	33.04	32.66	33.65
BOB	31.86	33.12	33.22	33.31	34.25
ML	31.98	33.46	33.52	33.52	34.37
WEAVE	33.12	35.66	35.56	35.53	35.84
BL	32.84	35.13	35.06	35.03	35.64
VT3	33.11	35.63	35.52	35.50	35.84
VT7	33.18	35.78	35.57	35.64	35.89
MA	32.03	33.57	33.64	33.60	34.45
MLVT3	33.12	35.61	35.50	35.49	35.84

Table A.3: PSNR of format converted output for 'Static' sequence (480p)

De-interlacing Algorithm	NN	BL	BC	ML	DCT
LR	26.86	30.44	30.05	30.99	33.31
BOB	27.85	31.07	30.21	31.72	33.86
ML	27.99	31.34	30.39	32.07	34.15
WEAVE	27.63	31.66	30.72	32.57	35.77
BL	27.69	31.60	30.67	32.49	35.55
VT3	27.64	31.66	30.72	32.57	35.73
VT7	27.66	31.67	30.72	32.58	35.70
MA	27.94	31.37	30.42	32.13	34.26
MLVT3	27.65	31.66	30.72	32.58	35.72

Table A.4: PSNR of reconstructed output for 'Static' sequence (1080p)

De-interlacing Algorithm	NN	BL	BC	ML	DCT
LR	34.22	36.97	36.23	36.28	38.27
BOB	35.66	38.04	36.62	36.85	39.98
ML	35.63	38.03	36.60	36.79	39.98
WEAVE	30.61	32.75	33.09	32.44	33.90
BL	32.43	34.45	34.27	33.90	35.83
VT3	35.02	37.58	36.40	36.57	39.25
VT7	35.37	37.85	36.53	36.76	39.59
MA	35.59	38.21	36.74	37.00	40.08
MLVT3	35.33	37.82	36.51	36.67	39.65

Table A.5: PSNR of reconstructed output for 'Car' sequence

De-interlacing Algorithm	NN	BL	BC	ML	DCT
LR	17.75	19.98	19.23	19.45	20.85
BOB	18.93	20.54	19.38	19.87	21.67
ML	18.93	20.57	19.39	19.89	21.66
WEAVE	16.59	18.83	18.60	18.62	19.58
BL	17.61	19.43	18.84	19.11	20.38
VT3	18.43	20.15	19.21	19.58	21.23
VT7	18.73	20.27	19.24	19.67	21.36
MA	18.93	20.58	19.39	19.89	21.66
MLVT3	18.58	20.26	19.25	19.65	21.34

Table A.6: PSNR of reconstructed output for 'Tulipz' sequence

De-interlacing Algorithm	NN	BL	BC	ML	DCT
LR	28.65	31.95	30.85	31.40	33.46
BOB	30.33	33.55	31.47	32.58	36.11
ML	30.31	33.60	31.50	32.56	36.15
WEAVE	27.34	30.29	29.86	30.13	31.33
BL	29.05	31.91	30.72	31.44	33.71
VT3	29.77	33.07	31.32	32.28	35.20
VT7	30.04	33.30	31.40	32.41	35.59
MA	30.29	33.71	31.57	32.65	36.23
MLVT3	30.06	33.38	31.45	32.44	35.73

Table A.7: PSNR of reconstructed output for 'Train' sequence

Bibliography

[1] B. Bhatt, D. Birks, and D. Hermreck, "Digital Television: Making it work," *IEEE Spectrum*, vol. 34, pp. 19–28, Oct. 1997.

[2] J. Lim, "High-Definition Television," *Wiley Encylopedia of Electrical and Electronics Engineering*, vol. 8, pp. 725–739.

[3] G. Hann and E. Bellers, "Deinterlacing - an overview," *Proceedings of the IEEE*, Sept. 1998.

[4] D. Lee, J. Park, and Y. Kim, "Video Format Conversions Between HDTV Systems," *IEEE Transactions on Consumer Electronics*, vol. 39, pp. 219–224, Aug. 1993.

[5] T. Doyle and M. Looymans, "Progressive scan conversion using edge information," in *Signal Processing of HDTV II* (L. Chiariglione, ed.), pp. 711–721, Amsterdam, The Netherlands: Elsevier, 1990.

[6] J. Salonen and S. Kalli, "Edge adaptive interpolation for scanning rate conversion," in *Signal Processing of HDTV IV* (E. Dubois and L. Chiariglione, eds.), pp. 757–764, Amsterdam, The Netherlands: Elsevier, 1993.

[7] D. Martinez and J. Lim, "Spatial Interpolation of Interlaced Television Pictures," *International Conference on Acoustics, Speech, and Signal Processing*, 1989.

[8] B. Ayazifar and J. Lim, "Pel-adaptive model-based interpolation of spatially subsampled images," *International Conference on Acoustics, Speech, and Signal Processing*, 1992.

[9] M. Achiha, K. Ishikura, and T. Fukinuki, "A Motion-Adaptive High-Definition Converter for NTSC color TV signals," *SMPTE Journal*, vol. 93, pp. 470–476, May 1984.

[10] R. Prodan, "Multidimensional Digital Signal Processing for Television Scan Conversion," *Philip Journal of Research*, vol. 41, no. 6, pp. 576–603, 1986.

[11] M. Annegarn, T. Doyle, P. Frencken, and D. Hees, "Video signal processing circuit for processing an interlaced video signal," Apr. 1998. U.S. Patent 4740842.

[12] J. Salo, Y. Nuevo, and V. Hameenaho, "Improving TV picture quality with linear-median type operations," *IEEE Transactions on Consumer Electronics*, vol. 34, pp. 373–379, Aug. 1998.

[13] C. Lee, M. Eden, and M. Unser, "High-Quality Image Resizing Using Oblique Projection Operators," *IEEE Transactions on Image Processing*, vol. 7, pp. 679–692, May 1998.

[14] H. Hou and H. Andrews, "Cubic Splines for Image Interpolation and Digital Filtering," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 26, pp. 508–517, 1978.

[15] M. Unser, A. Aldroubi, and M. Eden, "Fast B-spline transforms for Continuous Image Representation and Interpolation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 277–285, Mar. 1991.

[16] C. Yim and M. Isnardi, "An Efficient Method for DCT-Domain Image Resizing with Mixed Field/Frame-Mode Macroblocks," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 9, pp. 696–700, Aug. 1999.

[17] S. Martucci, "Image Resizing in the Discrete Cosine Transform Domain," *International Conference on Image Processing*, vol. 2, pp. 244–247, 1995.

[18] R. Dugad and N. Ahuja, "A Fast Scheme for Image Size Change in the Compressed Domain," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 11, pp. 461–474, Apr. 2001.

[19] T. Blu, P. Thevenaz, and M. Unser, "MOMS: Maximal-Order Interpolation of Minimal Support," *IEEE Transactions on Image Processing*, vol. 10, pp. 1069–1080, July 2001.

[20] A. Munoz, T. Blu, and M. Unser, "Least-Squares Image Resizing Using Finite Differences," *IEEE Transactions on Image Processing*, vol. 10, pp. 1365–1378, Sept. 2001.

[21] A. Gotchev, K. Egiazarian, J. Vesma, and T. Saramaki, "Edge-Preserving Image Resizing Using Modified B-Splines," *International Conference on Acoustics, Speech, and Signal Processing*, vol. 3, pp. 1865–1868, 2001.

[22] H. Greenspan, C. Anderson, and S. Akber, "Image Enhancement By Nonlinear Extrapolation in Frequency Space," *IEEE Transactions on Image Processing*, vol. 9, pp. 1035–1048, June 2000.