Digital Image Warping:

Theory and Real Time Hardware Implementation Issues

by

Mark Sebastian Lohmeyer

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 17, 1996

Author ....
Department of Electrical Engineering and Computer Science
May 17, 1996

Certified by ....
George Verghese
Thesis Supervisor

Accepted by ....
[illegible]. Morgenthaler
[illegible]raduate Theses
[illegible]

Digital Image Warping: Theory and Real Time Hardware Implementation Issues

by

Mark Sebastian Lohmeyer

Submitted to the
Department of Electrical Engineering and Computer Science

May 17, 1996

In Partial Fulfillment of the Requirements for the Degree of Bachelor of Science in Electrical Science and Engineering and Master of Engineering in Electrical Engineering and Computer Science

Abstract

This thesis begins with a broad introduction to digital image warping. It includes background on spatial transformations and basic signal processing theory as they are used in the warping application. Next, an analysis of various interpolation kernels in the time domain is performed. The results from this study lead directly to a new approach to performing a generalized high-quality image warp that maps well to an efficient hardware implementation. Finally, the design of a hardware system capable of performing image warps in real-time is described.

Thesis Supervisor: George Verghese
Title: Professor of Electrical Engineering, Massachusetts Institute of Technology

Acknowledgments

First I would like to thank my 6-A internship company, the David Sarnoff Research Center for sponsoring my work. In particular, I would like to recognize Gooitzen van der Wal and Peter Burt for finding me such an interesting project, supervising my work, and always having the time to listen and help.

Thanks to Professor George Verghese for being my MIT thesis supervisor and providing great feedback throughout the process.

Thanks to my family for always being so supportive (and keeping me well supplied with candy and cookies.)

Thanks to my brother Dan for always using my car so I had nothing else to do but work on my thesis.

Thanks to all my friends at MIT for providing me with plenty of opportunities to relax and take my mind off of thesis.

Finally, thanks to Benjie Chen for providing me with disk space to store all those warped images.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Image Warping

Digital image warping is an important and expanding branch of the field of digital image processing. In general, a digital image warp is a geometric transformation of a digitized image. Simple examples of warps include translation, rotation, and scale change. More complex warps can apply an arbitrary geometric transformation to the image.

Image warping was first used extensively for geometric correction in remote sensing applications. In the past twenty years, the field has grown to encompass a broad range of applications. These include medical imaging, image stabilization, machine vision, image synthesis and special effects generation. Every application has different speed, accuracy, and flexibility requirements. For example, the warping for movie special effects has to result in high-quality video, but the frames can be processed off-line and then pasted together into a sequence afterwards. On the other hand, in certain machine vision applications, the warping must occur in real-time, but the quality of the resultant warped images is not as critical.

In recent years, with the advances in digital logic and memory, it has become possible to perform many image processing tasks in real-time on specialized hardware. Image warping is no exception. There is a wealth of applications where real-time image warping is required. Moving target detection from a moving platform is one example. In this case, it is not adequate to simply detect motion by frame differencing because the motion due to the target will be lost in the motion from the moving camera. Instead, if the frames are first warped to remove the motion due to the camera, and then differenced, the only motion left will be due to the target. This motion can then be detected more readily. The desire to perform image warping in real-time has lead to an increased interest in algorithms that map well to an efficient hardware implementation.

There are two basic components to an image warp: spatial transformation and resampling through interpolation. A spatial transformation defines a geometric relationship between each point in the input image and a corresponding point in the output image. There are many possible models for this geometric relationship. Typically, the spatial transformation will require the value of points in the source image that do not correspond to the integral sample positions. This is where the second component of image warping, namely age resampling through interpolation, becomes important. Some sort of interpolation is required to estimate the value of the underlying continuous source image at locations that don't correspond to the sample points. Together, these two components determine the type and the quality of the warp.

1.2 Thesis Overview

This thesis begins with a broad introduction to image warping. Background on spatial transformations and basic signal processing theory are introduced to the extent required in the warping application. Next, an analysis of various interpolation kernels in the time domain is performed. The results from this study lead directly to an interesting approach to performing a generalized high-quality image warp that maps well to an efficient hardware implementation. Finally, the thesis describes the design of a hardware system capable of performing image warps in real-time.

The thesis is divided into chapters as follows:

Chapter 1 is an introduction to image warping.

Chapter 2 gives background on spatial transformations. There are many ways to specify the relationship between points in the source and target images. Different representations allow for varying degrees of freedom. The price for greater freedom is often higher computational complexity. These trade-offs are analyzed, particularly in terms of their mapping to an efficient hardware implementation. Three transformations are described in some detail because they are each important for certain real-time machine vision tasks. These are: the polynomial transformation, the perspective transformation, and the flow-field transformation.

Chapter 3 gives the theoretical framework for image resampling. First it discusses sampling of continuous-time signals. Next it examines how a continuous signal can be reconstructed from its samples with a low-pass filter and then resampled to form a new discrete sequence. Finally, this chapter reviews aliasing within the image warping framework and discusses how it may be avoided through preprocessing.

Chapter 4 is an analysis of methods of resampling through interpolation. First, it describes some candidate interpolation kernels for resampling. Then it analyzes their performance in the time domain. For each interpolation method, and a specified sampling rate, the expected error between an original and reconstructed sinusoid of a certain frequency is calculated. The error is then computed and graphed for a wide range of frequencies to determine how the interpolation method performs as a function of the frequency of the original signal. Next, the sampling rate is allowed to vary, while the frequency of the analog sinusoid is fixed. In this way, it is possible to determine how much the sampling rate must increase for an otherwise poor interpolation method to perform adequately. The results of Chapter 4 lead directly to an interesting method of performing an image warp, which may have numerous advantages over some previous approaches. This method is analyzed in Chapter 5.

Chapter 5 describes a method for performing high quality, computationally efficient digital image warping through upsampling followed by warping and downsampling using a smaller kernel for interpolation. This approach makes it particularly suitable for a low-complexity hardware implementation. Next, the performance of this algorithm is tested under a variety of conditions on actual images, and compared to standard techniques. Finally, based on the test results and knowledge of the filter used in the warping stage, an attempt is made to select an optimal filter for the upsampling stage.

Chapter 6 describes the hardware design of a specific warper implementation for use in a real-time image processing system. This includes a block diagram, summary of components, and board level design description. The chapter also analyzes the engineering trade-offs encountered in the design.

Chapter 7 is a summary of the work completed, and a discussion of topics for further research.

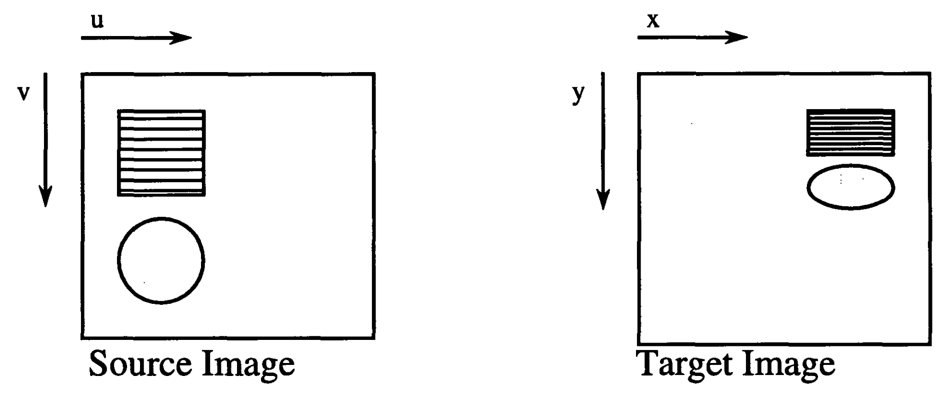
Chapter 2

Spatial Transformations

2.1 Definitions

In a digital image warp, the spatial transformation defines a geometric relationship between each point in the input or source image and its corresponding point in the output or target image [13]. This relationship between the points can be specified as either a forward or inverse mapping. If $[u,v]$ refers to the input image coordinates and $[x,y]$ refers to the output image coordinates, then a forward mapping gives the locations $[x,y]$ as some function of $[u,v]$. Likewise, an inverse mapping gives $[u,v]$ as a function of $[x,y]$. This is depicted in Figure 2.1, where U and V are the inverse mapping functions. For most hardware implementations, the inverse mapping is a more appropriate representation.

The inverse mapping specifies reference locations in the source image as a function of the current location in the target image.



$$[u,v] = [U(x,y), V(x,y)]$$

Figure 2.1: General inverse mapping.

Commonly, the locations in the target image are sequentially stepped through, and at each integer position the reference location in the source image is calculated. Because the inverse mapping function can be arbitrary, the referenced source location may not have integer coordinates. Therefore, some sort of interpolation is required to give the input value at nonintegral input locations. This will be discussed later. The second difficulty with the inverse mapping is that pixels in the source image may be left unaccounted for in the target image. If the input is not appropriately bandlimited prior to this sort of a mapping, aliasing can occur. Methods of avoiding this aliasing will also be discussed later.

In this chapter, three general categories of inverse mappings will be presented. These are: polynomial, perspective, and flow-field transformations. The selection of the most appropriate model depends upon both the application and the implementation requirements. The models will be discussed in these terms.

2.2 Polynomial Transformations

The polynomial inverse transformation is usually given in the form

$$u = \sum_{i=0}^{N} \sum_{j=0}^{N} a(i,j)\, x^i y^j$$

$$v = \sum_{i=0}^{N} \sum_{j=0}^{N} b(i,j)\, x^i y^j$$

Polynomial transformation (2.1)

where *a(i,j)* and *b(i,j)* are constant coefficients. This transformation originated in the remote sensing field, where it was used to correct both internal sensor distortions and external image distortions. It is currently being used in a wide range of applications. Mosaic construction is one example [5]. Typical hardware implementations use the forward differencing method to compute $[u,v]$. This is discussed in Appendix A.

One specific polynomial transformation that is commonly used is the affine transformation, specified as

$$u = a + bx + cy$$
$$v = d + ex + fy$$

Affine transformation (2.2)

This transformation allows for image translation, rotation, scale, and shear. It has six degrees of freedom, given by the six arbitrary constants. Therefore, only three corresponding points in a source and target image are needed to infer the mapping. The affine transformation can map a triangle to any other triangle. It can also map a rectangle to a parallelogram. It cannot, however, map one arbitrary quadrilateral into another.

The affine transformation is particularly well suited for implementation in hardware using the forward differencing method. Using the forward differencing method to calculate the source image addresses requires six registers to hold the values of the constant coefficients, four accumulators to store and update the current reference address, and four two-to-one multiplexers to pipe the correct increment values to the accumulators.

2.3 Perspective Transformation

The inverse mapping function of a perspective transformation is given as

$$u = \frac{a + bx + cy}{e + fx + gy}$$
$$v = \frac{h + ix + jy}{e + fx + gy}$$

Perspective transformation (2.3)

If f and g are taken to be zero, then the perspective transformation reduces to the affine transformation. If *f* and *g* are not zero, then it is a perspective or projective mapping. The perspective mapping preserves lines in all orientations. The perspective mapping has eight degrees of freedom, which permits arbitrary quadrilateral-to-quadrilateral mappings.

Because the perspective transformation requires a division, it is not as easy to implement in hardware as the affine transformation. The numerator and denominator for each [u,v] value can be computed using the forward differencing method, just as in the affine case. Then a division is needed. This could be performed in a specialized division chip, which may introduce a fairly long pipeline delay. Or, it could be done in a large look-up table. Depending on the required accuracy of the result, both of these methods may be prohibitively expensive in terms of cost or space. The perspective transformation can also be approximated by a polynomial transformation of high enough order. In many applications, a second or third degree polynomial will be adequate.

2.4 Flow-Field Transformation

The flow-field transformation is the most general transformation. It can be used to realize any other spatial mapping. It can be viewed simply as a table that gives an input location to reference for each output location. If these flow-field values are given to the warper module from other modules, the flow-field adds very little complexity to the warper module. The trade-off, of course, is that significant resources may be required elsewhere to compute these flow-field values.

2.5 Summary Remarks

This is only a brief description of some common spatial transformations, focusing on those transformations that are being considered for real-time hardware implementation at the David Sarnoff Research Center. A more detailed description of these and other geometric transformations can be found in [13]. A related problem is the estimation of the constants of the warp equation, given a set of constraints. For example, in performing image stabilization, interframe motion must be accurately estimated. It has been shown that a multiresolution, iterative process operating in a coarse-to-fine fashion is an efficient way to perform this estimation. During each iteration, an optical flow-field is estimated between the images through local cross-correlation analysis, then a motion model is fit to the flow-field using weighted least-squares regression. The estimated transform is then used to warp the previous image to the current image and the process is repeated between them, [5]. This and other methods of estimating the warping parameters will not be discussed in further detail here.

Chapter 3

Resampling Theory for Image Warping

3.1 Sampling of Continuous-Time Signals

Digital images are created by a variety of means. These fall into two general categories: computer-generated images and sampled images. Computer-generated graphics assign pixel values based on a mathematical formula. This can range from something as simple as a two-dimensional graph to a complex animation program derived from the laws of physics. If such a mathematical model is available, it is appropriate to warp the image simply by evaluating the formula at different points. In the case of sampled images, however, no such formula is available. Usually, all that is available is the actual samples. To understand what can be done under these conditions, one must understand the digital image as a representation of a continuous image in both the spatial and frequency domains. First, the one-dimensional case is examined.

The discrete representation of a continuous signal is usually obtained by periodically sampling the continuous signal. If the sampling rate is *T*, this relationship is expressed as:

$$f[n] = f_C(nT) \tag{3.1}$$

To derive the relationship between the Fourier transforms of *f[n]* and $f_c(x)$, it is helpful to analyze this conversion in two steps. In the first step, the continuous signal is sampled by multiplication with an impulse train of period *T*. The second step is to then convert this impulse train into a discrete sequence. This process is shown in the spatial domain in Figure 3.1.

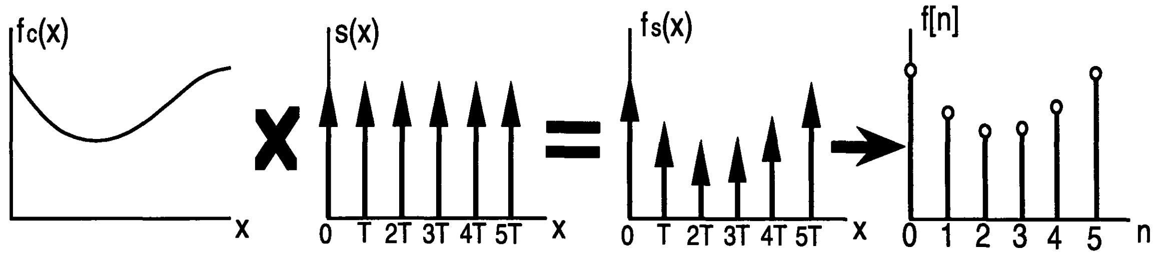


Figure 3.1: Continuous signal to discrete signal.

The impulse train of period *T* can be written mathematically as:

$$s(x) = \sum_{n=-\infty}^{\infty} \delta(x - nT) \tag{3.2}$$

Then

$$f_S(x) = f_C(x)\, s(x) \tag{3.3}$$

$$f_S(x) = \sum_{n=-\infty}^{\infty} f_C(nT)\, \delta(x - nT) \tag{3.4}$$

Since $f_s(x)$ is the product of $f_c(x)$ and $s(x)$, the Fourier transform of $f_s(x)$ is the convolution of the Fourier transforms of $f_c(x)$ and $s(x)$:

$$F_S(j\Omega) = \frac{1}{2\pi} F_C(j\Omega) * S(j\Omega) \tag{3.5}$$

The Fourier transform of $s(x)$ is:

$$S(j\Omega) = \frac{2\pi}{T} \sum_{k=-\infty}^{\infty} \delta(\Omega - k\Omega_S) \tag{3.6}$$

where $\Omega_S = 2\pi/T$, which is the sampling frequency. Doing the convolution then gives:

$$F_s(j\Omega) = \frac{1}{T}\sum_{k=-\infty}^{\infty} F_c(j\Omega - kj\Omega_s) \tag{3.7}$$

Equation 3.7 gives the frequency-domain relationship between the original analog signal and the impulse-sampled analog signal. The Fourier transform of the impulse sampled signal comprises copies of the Fourier transform of the original signal superimposed at integer multiples of the sampling frequency. The higher the sampling rate, the further these copies are spread apart. From this analysis, one can see the origins of the Nyquist Sampling Theorem. If the original analog signal is bandlimited, it is possible to sample at a high enough rate that the copies of the Fourier transform of the original signal do not overlap in the Fourier transform of the sampled signal. If there is no overlap, then the original signal can be exactly reconstructed with an ideal lowpass filter with the correct gain. An ideal lowpass filter will reject all copies except the one centered at the origin, giving the original signal back. For there to be no overlap, the sampling rate must be at least twice the maximum frequency component of the original signal:

$$\Omega_s > 2\Omega_n \tag{3.8}$$

If there is overlap in the frequency domain, the signal can no longer be exactly reconstructed from its samples. The error that occurs due to this overlap is referred to as aliasing.

The second step in the continuous to discrete process is the conversion from the impulse-sampled signal in the continuous domain to a discrete sequence. The values of the discrete time sequence at $n = 0, 1, 2, \ldots$ are the areas of the impulses at $0, \Omega_s, 2\Omega_s, \ldots$ But what does this do in the frequency domain?

First the results will be derived mathematically. Taking the Fourier transform of equation 3.4 gives:

$$F_s(j\Omega) = \sum_{n=-\infty}^{\infty} f_c(nT)\, e^{-j\Omega T n} \tag{3.9}$$

Since

$$f[n] = f_c(nT) \tag{3.10}$$

and

$$F(e^{j\omega}) = \sum_{n=-\infty}^{\infty} f[n]\, e^{-j\omega n} \tag{3.11}$$

looking at Equations 3.9, 3.10, and 3.11, it can be seen that

$$F_s(j\Omega) = F(e^{j\omega})\big|_{\omega = \Omega T} = F(e^{j\Omega T}) \tag{3.12}$$

Now, from Equations 3.7 and 3.13.

$$F(e^{j\Omega T}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} F_c(j\Omega - jk\Omega_s) \tag{3.13}$$

which can also be written

$$F(e^{j\omega}) = \frac{1}{T}\sum_{k=-\infty}^{\infty} F_c\left(j\frac{\omega}{T} - j\frac{2\pi k}{T}\right) \tag{3.14}$$

From these equations it can be seen that the conversion from the continuous domain train of impulses to a discrete sequence corresponds to a rescaling of the frequency axis in the frequency domain. The frequency scaling is given by $\omega=\Omega T$. This scaling normalizes the continuous frequency $\Omega=\Omega_s$ to $\omega=2\pi$ for the Fourier transform. This makes some intuitive sense as well. In the spatial domain, the impulses are spaced T apart, while the discrete sequence values are spaced apart by one. Thus, the spatial axis is normalized by a factor of T. In the frequency domain then, the axis is normalized by a factor of $1/T$. [10]

A graphical representation is quite useful in understanding the relationship between all these signals. This is shown in Figure 3.2.

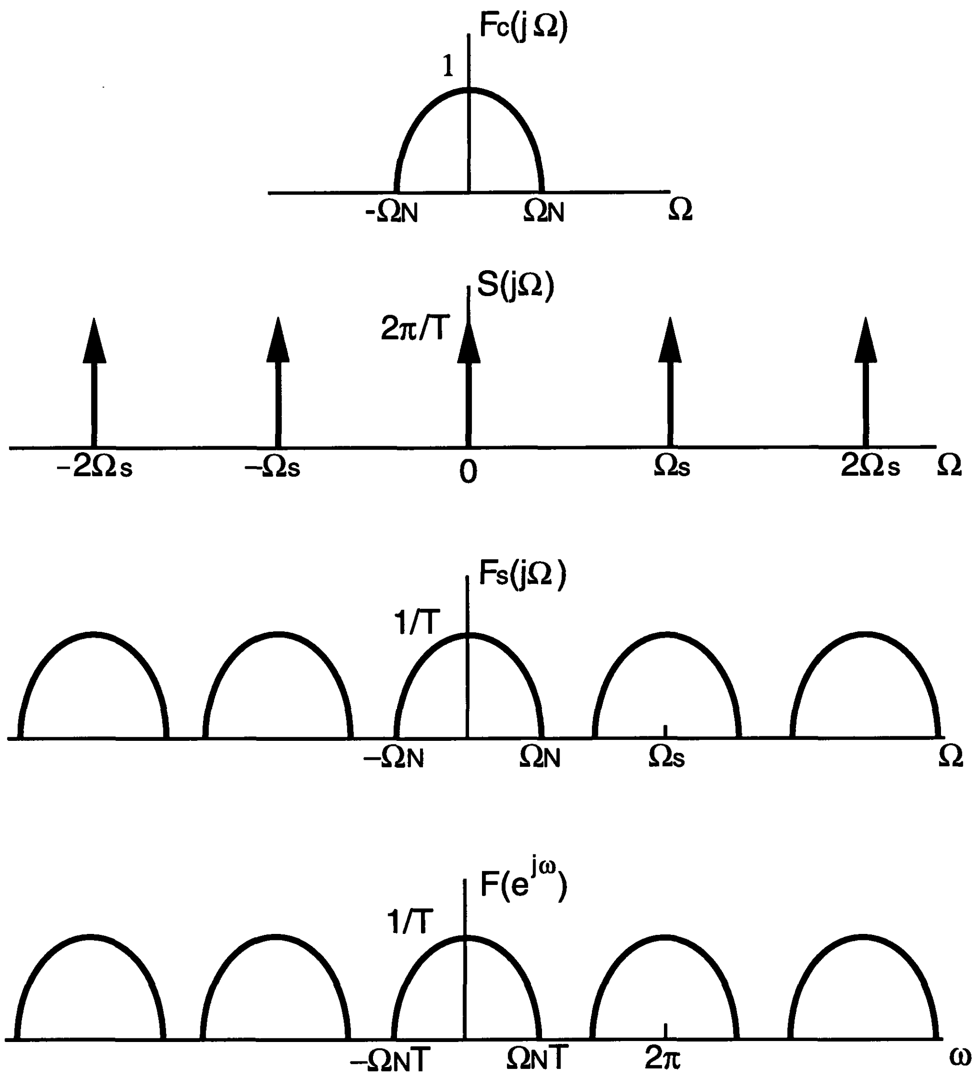


Figure 3.2: Continuous signal to discrete signal in the frequency domain.

The whole continuous-to-discrete sampling process can be summarized quite simply in the spatial and frequency domain. In the spatial domain, the values of the discrete sequence at n = 0, 1, 2, etc. are the values of the continuous-time signal at the sample points, 0, T, $2T$, etc. In the frequency domain, the Fourier transform of the discrete sequence can be produced by normalizing the frequency axis of the continuous signal by a factor of $1/T$, and then replicating this at all integer multiples of 2π. The two-dimensional case can be analyzed using the same methods as in the one-dimensional case here. The only difference is now one has to worry about frequencies in two dimensions.

For the purposes of this thesis, it will be assumed that no appreciable aliasing occurs in the continuous to discrete sampling of an image. This is indeed the case with most of the images encountered in typical image warping applications. However, understanding the relationships between the underlying analog image and its samples is still of extreme importance for warping. In this situation, a discrete image is all that is given. The values of the underlying continuous image are only known at integer locations in this discrete image. If this image is then to be warped based on a geometric transformation, it is very probable that the values of the continuous image at other positions than the original sample points will be needed. This is the fundamental resampling problem encountered in digital image warping.

3.2 Resampling Through Interpolation in Digital Domain

Resampling is the process of transforming a digital image from one coordinate system to another. In the case of image warping, these two coordinate systems are related to each other by the spatial transformation that defines the warp. Conceptually, resampling can be divided into two steps. First, interpolate the discrete image into a continuous one. Second, sample this continuous image at the desired locations to form the resampled image. In practice, these two steps are often consolidated so the interpolated values are only calculated for those locations that will be sampled. This makes it possible to implement the resampling procedure entirely in the digital domain. This section discusses the spatial and frequency-domain interpretations of the resampling process. It also examines how this resampling can be accomplished in digital hardware for an arbitrary geometric transformation function.

Conceptually, the first step in the resampling process is reconstruction of the bandlimited analog signal from its samples. In Section 3.1, it was shown that if the original signal after modulation by an impulse train was appropriately low-pass filtered, it would return the original analog signal. This impulse train, $f_s(t)$, can be constructed from the discrete samples as follows:

$$f_S(x) = \sum_{n=-\infty}^{\infty} f[n]\,\delta(x-nT) \tag{3.15}$$

Then if $H_r(j\Omega)$ is the frequency response of the low-pass filter, and $h_r(x)$ is its impulse response, the reconstructed signal at the output of the filter is:

$$f_r(x) = \sum_{n=-\infty}^{\infty} f[n]\,h_r(x-nT) \tag{3.16}$$

The perfect $H_r(j\Omega)$ is an ideal low-pass filter with cut-off frequency of π/T and a gain of T. In the spatial domain, this is a sinc function:

$$h_r(x) = \frac{\sin\left(\frac{\pi x}{T}\right)}{\frac{\pi x}{T}} \tag{3.17}$$

So, perfect reconstruction comes from convolving the samples with a sinc function. One important property of this sinc function is that it has a value of unity at $x=0$, and a value of zero at all other integer multiples of T. This means that the reconstructed signal has the same values at the sample points as the original continuous signal.

After the continuous signal is reconstructed, it can be resampled at different positions to form the resampled sequence. Of course, in real implementations, this is not how the resampling is performed. One approach to performing this resampling entirely in the digital domain is discussed below.

In the typical resampling problem, the samples at integer locations are given. From these, we wish to determine the value of the continuous sequence at other locations. This is shown in Figure 3.3.

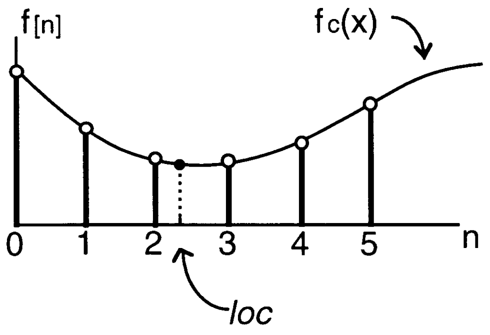


Figure 3.3: Resampling at an arbitrary location.

It is obvious at this point that the sinc function can not possibly be used for reconstruction, as it is infinitely long. Instead, a finite length interpolation function must be used so that the convolution can be performed. Of course, this means that it will not correspond to a perfect low-pass filter, which will result in some error. The trade-offs involved in picking an appropriate interpolation method are discussed in greater detail in Chapter 4. For practical implementations, a interpolation kernel of finite length is used. To demonstrate the approach, a kernel with a width of four will be used. This interpolation kernel will be referred to as $h_{r4}(x)$.

Call the spatial location we are interested in *loc*. This can be written as the sum of a fractional and an integer part:

$$loc = intloc + fracloc \tag{3.18}$$

So if *loc* is 2.3, *intloc* is 2 and *fracloc* is 0.3. Now, it can be seen that if the convolution was carried out using $h_{r4}(x)$ as the kernel, the value of the function at *loc* would be:

$$f_r(loc) = \sum_{k=-1}^{2} f[k + intloc] \times h_{r4}(fracloc - k) \tag{3.19}$$

This approach lends itself well to a hardware implementation. A look-up table can be used to store the values of the interpolation kernel. The integer portion of the referenced source location is used to determine which sample points are accessed, and the fractional part of the source location determines the weights of the coefficients as generated by the look-up table. The final value at the referenced location is given by a sum of products of the sample points and the weighting coefficients. This approach can easily be used in two dimensions for image warping resampling. For example, if the one dimensional interpolation kernel is of width four, then this method will require a pixel area of 4x4, with sixteen weighting coefficients.

Next, it will be helpful to examine the effects that upsampling and downsampling have in the frequency domain. In general, the geometric transform of an image warp may require upsampling in some areas of the image and downsampling in others. Or, it may simply require a shift, which is neither upsampling or downsampling. The frequency-domain interpretation of such arbitrary resampling is difficult to represent in closed mathematical form. Therefore, it will be useful to analyze the simpler cases of regular upsampling and downsampling to gain intuition into the frequency-domain effects of the resampling required by a geometric transformation. In this analysis, it will be assumed that the interpolation uses an ideal low-pass filter. If this is the case, then resampling in the digital domain corresponds exactly to sampling the original signal at different locations.

Upsampling refers to an increase in the sampling rate. For example, upsampling by two corresponds to doubling the sampling rate of the continuous signal. From the analysis in Section 3.1, it can easily be seen that upsampling corresponds to a compression of the Fourier transform. If the signal in upsampled by a factor of U, then the Fourier transform is compressed in frequency by that same factor. This is shown in Figure 3.4.

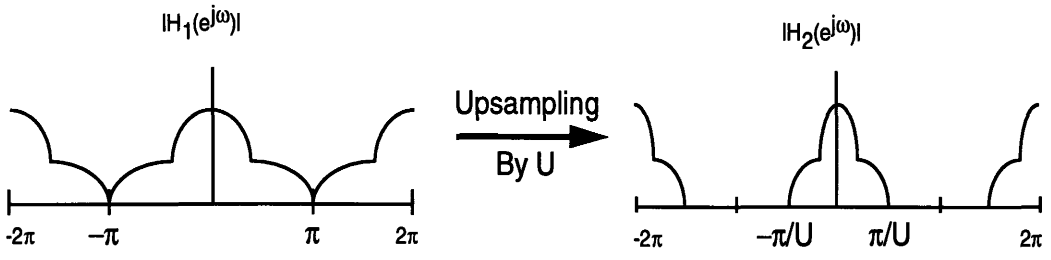


Figure 3.4: Upsampling by U in frequency domain.

Of course, the relationship above holds only if the low-pass filter used for interpolation is ideal; if it is not ideal, then some distortion will occur.

Downsampling refers to a decrease in the sampling rate. If the signal is not appropriately low-pass filtered before the downsampling occurs, aliasing can result. If the signal is downsampled by a factor of D, then the Fourier transform of the signal is expanded in frequency by that same factor. This can lead to an overlap in the frequency domain, which is the cause of the aliasing. The problem is identical to that of sampling the original continuous signal below the Nyquist rate. This aliasing problem can be avoiding by pre-filtering the signal with a discrete-time low-pass filter to remove the frequencies that would overlap, as shown in Figure 3.5.

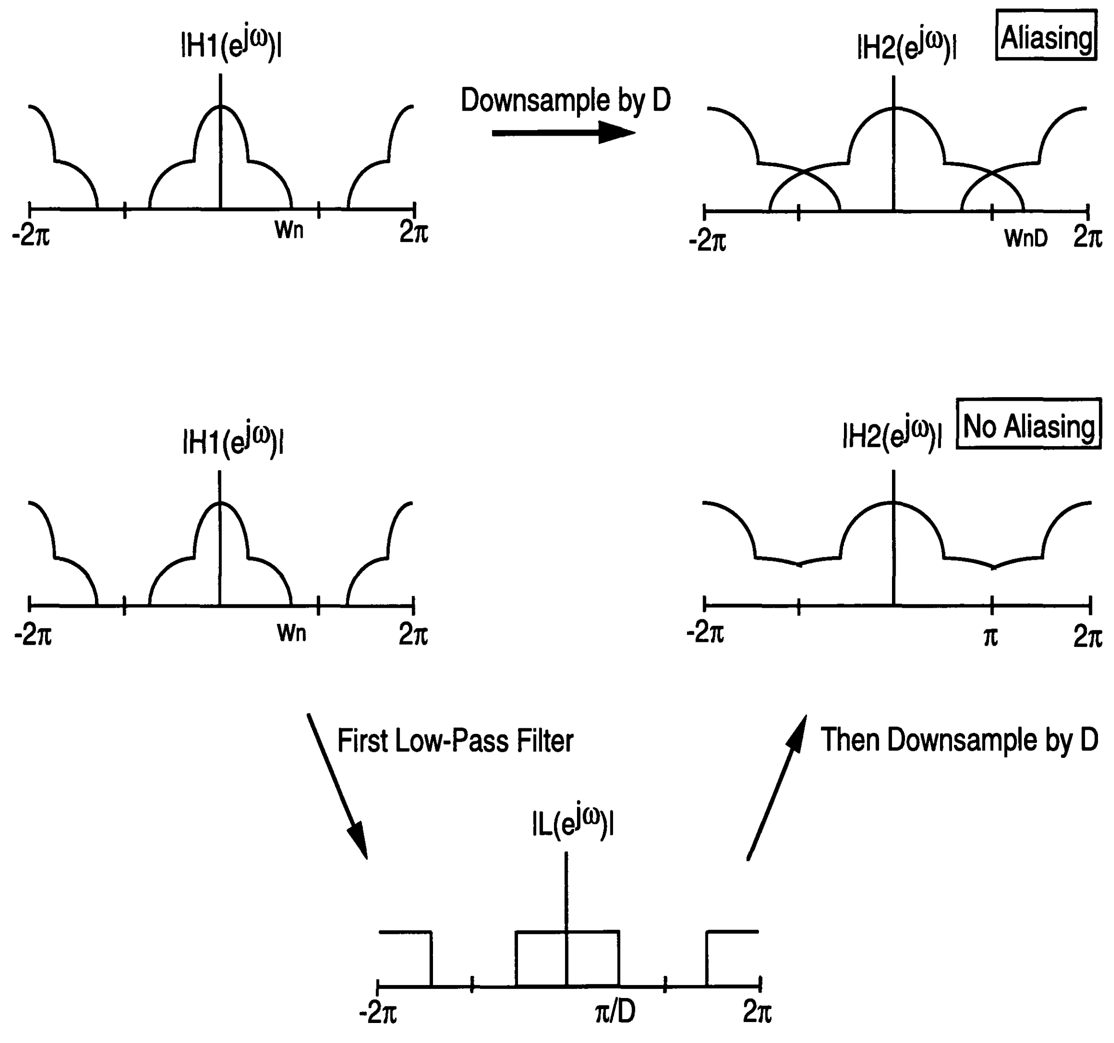


Figure 3.5: Downsampling, with and without aliasing.

3.3 Summary Remarks

In practice, for machine vision applications, aliasing usually is not a severe problem. The geometric transformations rarely require shrinking the image to a size where aliasing might result. Also, the frequency composition of typical images rarely contains much energy in the higher frequencies that would overlap. For example, if the image contains no energy in radian frequencies higher than π/2, then the image can be downsampled by a factor of two in both directions, and no aliasing will occur. Also, if the geometric transformation is known beforehand, then the image can be preprocessed with a low-pass filter to remove the frequencies that would cause problems. Methods of performing this pre-filtering will not be discussed in further detail here.

The usual culprit in poor quality image warps comes not from aliasing, but from the method used for interpolation prior to resampling. Chapter 4 will analyze the performance of some interpolation kernels, and discuss the implications of the results.

Chapter 4

Analysis of Interpolation Methods for Image Resampling

4.1 Description

The real problem to be analyzed here is how well a continuous image can be reconstructed from its samples using practical interpolators. If the original continuous image is bandlimited and sampled above the Nyquist frequency, sampling theory shows it can be exactly recovered with a perfect low-pass filter. In the spatial domain, this corresponds to convolving with a two dimensional sinc function. This is not a practical option, as the sinc function is infinitely long and falls off only as the reciprocal of distance. Instead, practical implementations use an interpolation kernel of finite support. In general, the larger the support of the interpolation kernel is allowed to be, the more accurately its frequency response can approximate the ideal. However, longer interpolation kernels require greater computational resources to implement. This is the fundamental trade-off: efficient computation versus accuracy of reconstruction. Much work has been done to identify methods of interpolation that strike an appropriate balance between these two competing requirements.

There are numerous papers that compare various interpolation methods for their accuracy of reconstruction [8]. Two general approaches to evaluation are usually taken. The first is simply to resample test images using different interpolation methods and visually assess the quality of the resultant image. The second, more analytical method is to base the evaluation on the frequency characteristics of the interpolation kernel.

The first method is very straightforward, and appropriate for applications where display for human viewing is the ultimate goal. It has the natural advantage that it includes the effects of human perception. However, it is somewhat ad hoc; the error is not quantified. For comparison of interpolation methods, all that can really be said is that one "looks" better that the other. What may be reasonable for viewing may not be adequate if further calculations are to be performed.

The second method, on the other hand, is quantitative. Analysis in the frequency domain is quite powerful. Given the frequency spectrum of an image, the sampling rate, and the interpolation method, one can easily determine how well the method approximates the ideal. This is the approach usually taken, and this sort of analysis can be found in most signal processing textbooks [10].

4.2 Expected Error Calculation

Rather than duplicate these results, a different approach is taken here. At a very basic level, it will attempt to quantify the error incurred by different interpolation methods in the spatial domain in a way that is appropriate to the image warping application. To facilitate analysis, this will be done in one dimension. First, take a continuous sinusoid of frequency f Hz, $actual(t) = \sin(2\pi f t)$. Next, sample this wave at $1/T$ Hz to produce, $samples[n] = \sin\left(2\pi f\frac{n}{T}\right)$. Finally, using these samples and the selected interpolation method, reconstruct the continuous sinewave. The reconstruction error is then calculated as follows:

$$ReconstructionError = \int_a^b \frac{|actual(t) - interpolated(t)|dt}{(b-a)}$$

Reconstruction Error Calculation (4.1)

This can be interpreted graphically as the area between the two functions divided by the length over which the area is being computed. Figure 4.1 shows this graphical interpretation.

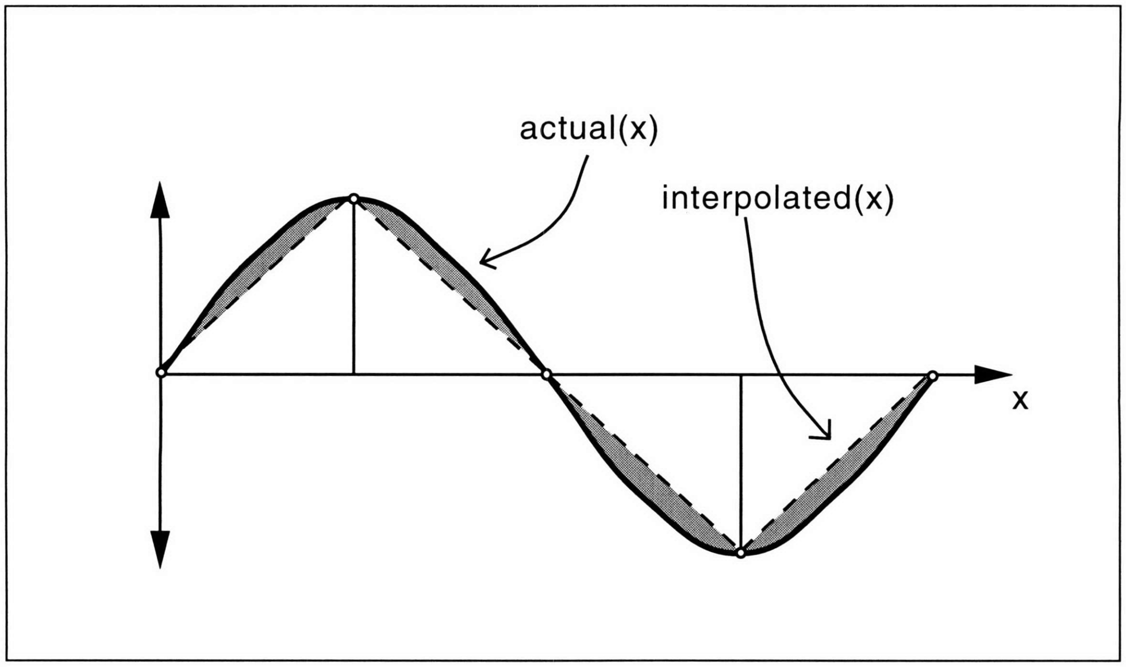


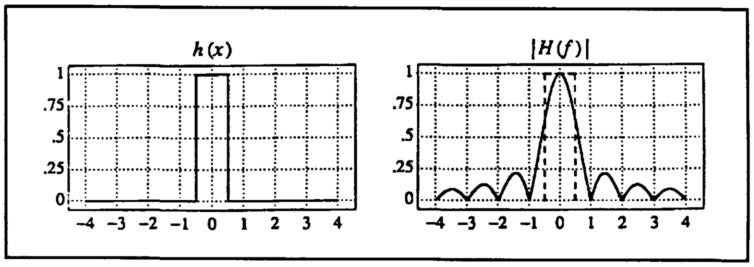
Figure 4.1: Reconstruction error illustration.

The length of integration is taken to be one complete period of the sinewave. The reconstruction error is calculated and averaged for all possible phase shifts of the sample points relative to the sinewave to give the expected error. This gives the average vertical separation between the two functions, and because it is computed for all phase shifts, expected error is an appropriate name.

A few different interpolation kernels will be examined in this manner. Figure 4.2 shows these kernels and their frequency responses.

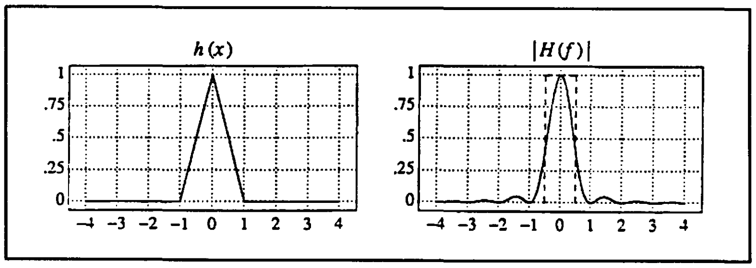
Nearest Neighbor:

$$h(x) = \begin{cases} 1 & 0 < |x| < .5 \\ 0 & .5 < |x| \end{cases}$$



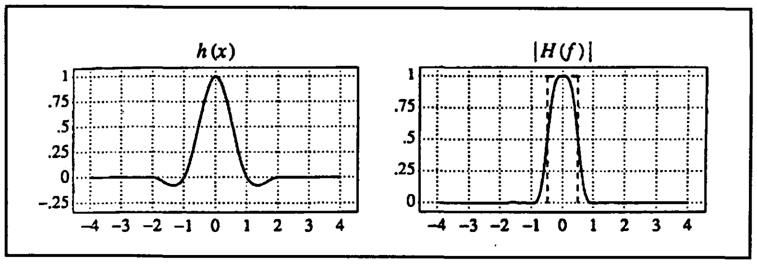
Linear:

$$h(x) = \begin{cases} 1-|x| & 0 < |x| < 1 \\ 0 & 1 < |x| \end{cases}$$



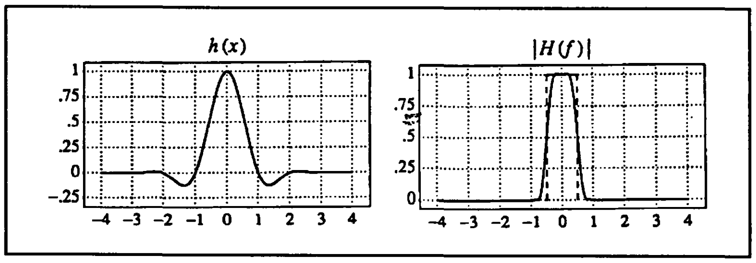
3rd Order Key's Cubic Conv:

$$h(x) = \begin{cases} 1.5|x|^3 - 2.5|x|^2 + 1 & 0<|x|<1 \\ -.5|x|^3 + 2.5|x|^2 - 4|x| + 2 & 1<|x|<2 \\ 0 & 2<|x| \end{cases}$$



Hamming Windowed Sinc:

$$h(x) = \begin{cases} \mathrm{sinc}(\pi x)*(.54 + .46\cos(\pi x/2)) & 0<|x|<2 \\ 0 & 2<|x| \end{cases}$$



Lanczos Windowed Sinc:

$$h(x) = \begin{cases} \mathrm{sinc}(\pi x)*2\sin(\pi x/w)/\pi x & 0<|x|<2 \\ 0 & 2<|x| \end{cases}$$

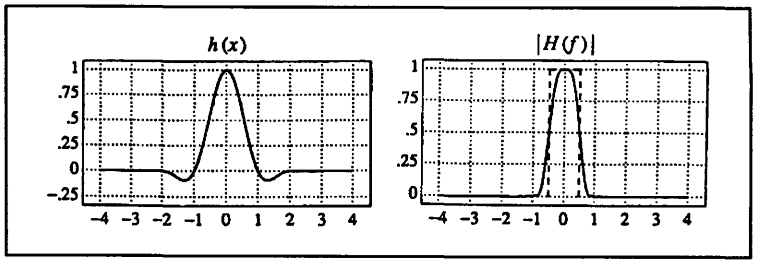


Figure 4.2: Interpolation kernels and their frequency responses.

For each interpolation method, given a fixed sampling frequency of $1/T$, a plot of expected error as a function of frequency of the underlying sinusoid is generated. This data is presented graphically in Figure 4.3 and Figure 4.4.

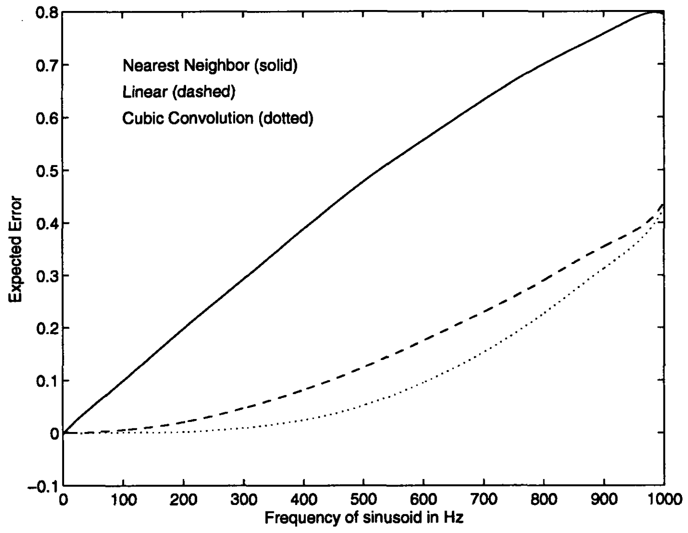


Figure 4.3: Expected error vs. frequency for sampling rate of 2000Hz.

Figure 4.3 shows the expected error for three basic interpolation methods. These are nearest neighbor interpolation, linear interpolation, and third order cubic convolution. The sampling rate is fixed at 2000Hz in this experiment. The horizontal axis is the frequency of the underlying sinusoid. It ranges from 0 to 1000Hz. If the frequency is allowed to go beyond 1000Hz, then the signal is no longer Nyquist sampled and the expected error value will include the affects of aliasing, which is not meant to be measured. The vertical axis is the expected error, as defined in Equation 4.1.

As expected, nearest neighbor interpolation is the poorest performer. Linear interpolation is the next best, followed by cubic convolution interpolation. There is a marked difference between all three methods. This is directly linked to the fact that nearest neighbor uses one sample point for interpolation, and linear and cubic use two and four sample points, respectively. This basic result is also confirmed by frequency domain interpretation.

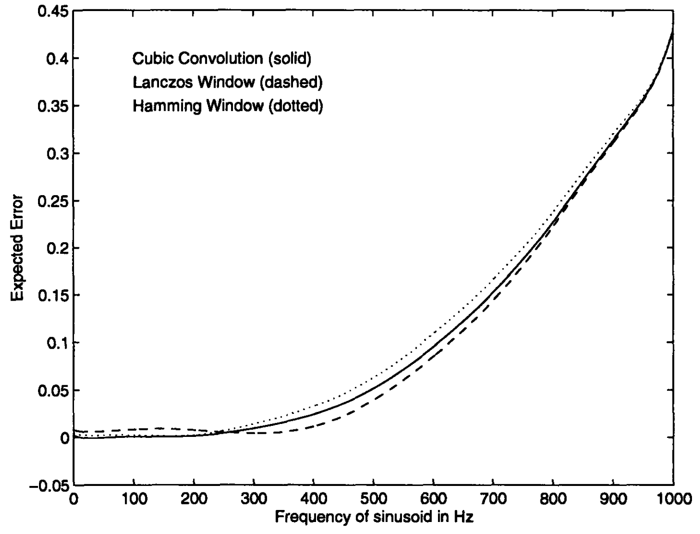


Figure 4.4: Expected error vs. frequency for sampling rate of 2000Hz.

Figure 4.4 shows expected error for two other interpolation methods. Cubic convolution is included as a reference to Figure 4.3. The Lanczos windowed sinc kernel is the best performer for the higher frequencies, but cubic convolution is better at lower frequencies. Overall though, their performance is very similar because they all use four sample points. In hardware, if the interpolation kernel values are precalculated, then all methods that use a four-by-four interpolation kernel should produce nearly identical results, given a fixed sampling rate.

In the preceding calculations of expected error, the sampling rate was fixed, and the expected error was found as a function of frequency of the underlying sinusoid. An equivalent analysis could be made by fixing the frequency of the sinusoid and varying the sampling rate, since the expected error will be the same for two sets of sampling rates and frequencies, *(sample_rate_1, frequency_1)* and *(sample_rate_2, frequency_2)* if *freq_1/*

sample_1 = *freq_2/sample_2*. For example, using the same interpolation method, the expected error would be identical for a 200Hz sinusoid sampled at 1000Hz as for a 400Hz sinusoid sampled at 2000Hz.

4.3 Sampling Rate Issues

An interesting question then develops. Could an otherwise poor interpolation method give superior results if we oversample the analog signal. For example, could linear interpolation with an oversampled signal give less error than cubic convolution interpolation with a Nyquist sampled signal? The answer is clearly, yes. But, how much oversampling is needed? The following analysis will attempt to quantify these things through analysis in the time domain.

Figure 4.5 shows the "sampling rate multiplier" for linear interpolation versus cubic convolution interpolation as a function of the expected error. This multiplier is defined as follows. If we sample a sinusoid of a given frequency, the expected error under a cubic convolution interpolation scheme is the same as the expected error using a linear interpolation scheme when the sampling rate for the linear scheme exceeds that of the cubic scheme by the sampling rate multiplier for that particular expected error. If a large expected error is tolerable, then the sampling rate multiplier is quite small. However, as the desired expected error approaches zero, the multiplier grows rapidly.

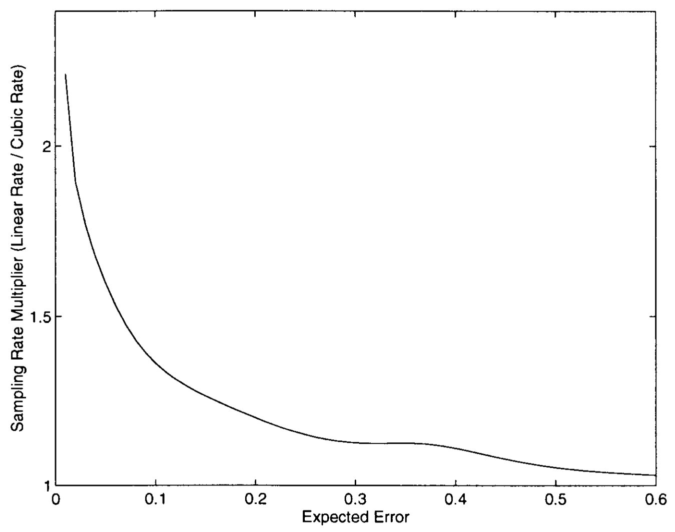


Figure 4.5: Sampling rate multiplier (linear vs. cubic convolution).

Figure 4.6 shows the same sort of information, except here the vertical axis is the sampling rate multiplier for nearest neighbor interpolation versus linear interpolation.

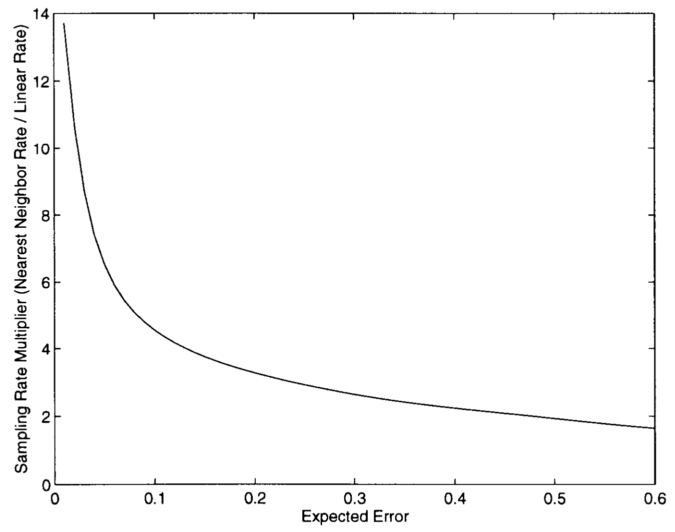


Figure 4.6: Sampling rate multiplier (nearest neighbor vs. linear).

It has the same basic shape as Figure 4.5 except that the multiplier is considerably larger for any given expected error. This is because nearest neighbor interpolation is such a poor overall performer. Because the multiplier is so large and results at least as good as linear interpolation on a Nyquist sampled image are desired, this case will not be analyzed further.

There are many interesting implications that develop from Figure 4.5. Specifically, note that for a tolerable error of .02, doubling the sampling rate and using a linear interpolation scheme is comparable to cubic convolution. This leads to many interesting possibilities for an efficient hardware implementation. Instead of performing a two-dimensional third-order cubic convolution interpolation for resampling on a Nyquist sampled image, it might be possible to perform two-dimensional linear interpolation (commonly referred to as bilinear interpolation) for resampling on a double density sampled image. The interpolation kernel size would be decreased from 4x4 to 2x2. This leads to a reduction in the cost and number of components needed. The penalty in this approach is incurred in increasing the sampling rate by a factor of two. There are two approaches to this. First, the analog image could be sampled at twice the previous rate. The prohibitive disadvantage of this approach is that the warper can no longer be a modular part of a complete vision system, since it requires a different digitizer and image size for input. The second approach, which is the one that will be examined, is to include a digital upsampler in the warper. This takes the Nyquist sampled digital image and upsamples it by a factor of two in both directions. Then this image is warped, using bilinear interpolation for resampling. This is a specific example of a more general approach to performing high quality image warping efficiently in hardware. The general approach is to upsample using a good filter, next warp using a lower quality filter, finally downsample to give the final image. This approach will be examined in detail.

Chapter 5

High Quality, Efficient Image Warping

5.1 General Description of Method.

As discussed in previous chapters, there are two components to an image warp: spatial transformation and resampling through interpolation. In the interpolation step, an area of pixels around the referenced input location is used to compute the output pixel value. The larger the number of pixels used, the more accurate the resampling can be. However, as the number of pixels used increases, so does the cost and complexity of the hardware implementation. The method described in this chapter deals primarily with a way to improve the accuracy of the resampling without a drastic increase in complexity.

One standard approach to performing the interpolation required for the resampling uses a 2x2 neighborhood of pixels around the referenced address in the source image to calculate each output pixel value. This is commonly called bilinear interpolation, which simply refers to linear interpolation in two dimensions. Chapter 6 describes such a hardware design in detail. For a real-time implementation with reasonable clock rates, this means that every clock cycle, four pixels values must be accessed simultaneously. These pixel values are then multiplied by the appropriate weights and summed to give the output pixel value. In hardware, this corresponds to four separate memory banks that can be accessed in parallel. Also, the four weighting coefficients must be generated, based on the sub-pixel location of the reference point. Finally, a four-term sum of products must be performed. The difficulty with this model comes when bilinear interpolation is no longer adequate. In applications that require repeated warping of the same image, or even just high quality sub-pixel translations, bilinear interpolation gives poor results. The next highest quality interpolator uses a 3x3 pixel area in the source image to compute each output pixel value. If the same paradigm is used, it leads to an unwieldy implementation. Now, nine separate memory banks, nine coefficients, and a nine-term sum of products are required. In applications where size, power and cost are at a premium, this is an unacceptable solution. If an even better interpolation is required, for example 4x4 or 6x6, the problem is even worse. There are some optimizations that can be made under this approach, but the fundamental problem of quadratic growth will always exist. One common approach to avoiding this problem is the use of a two-pass method. This approach works well in many specialized cases. It cannot, however, accommodate a general flow-field warp without costly computation overhead.

The following provides a high level description of a method for performing an arbitrary image warp that can achieve high-quality interpolation while bypassing some of the limitations described above. The basic system is shown in Figure 5.1.

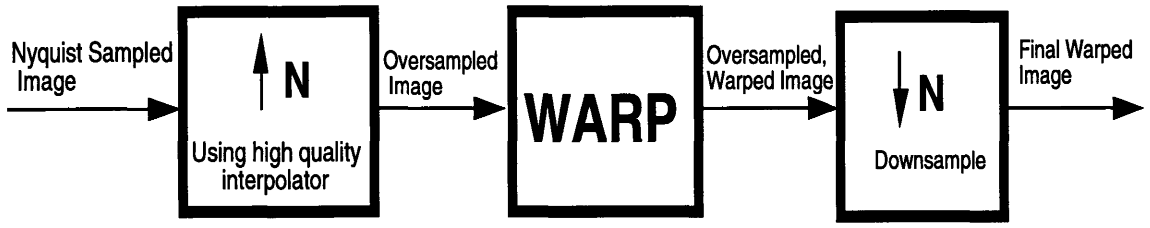


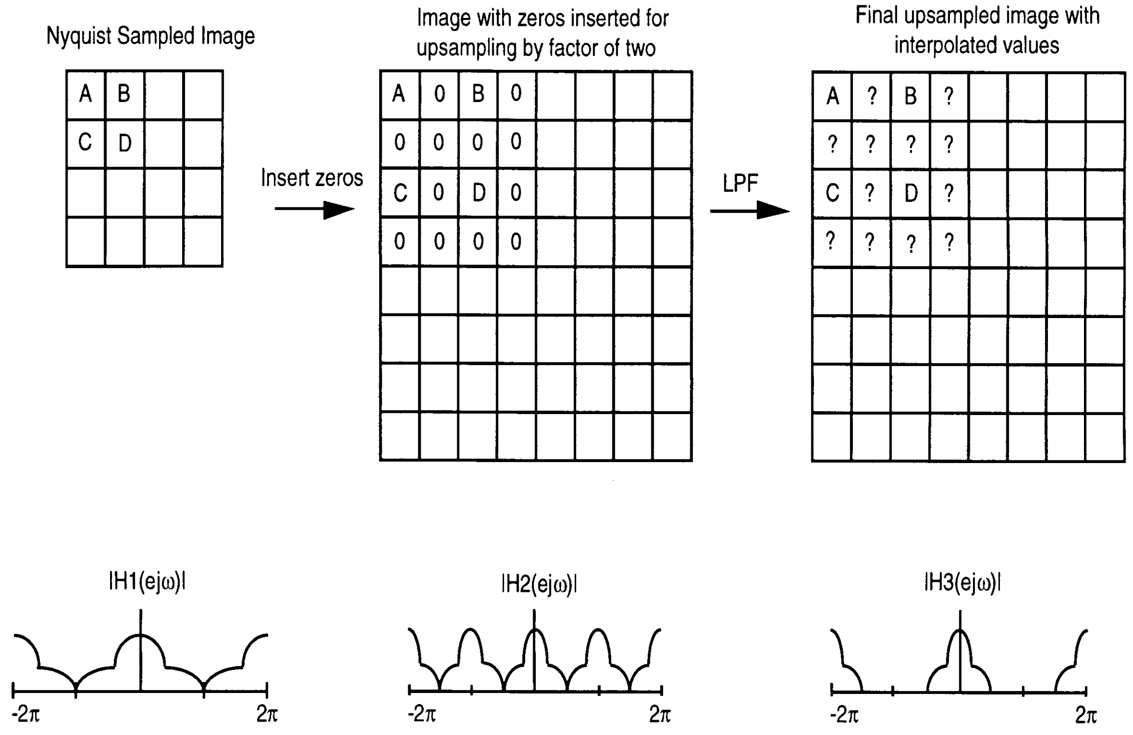
Figure 5.1: High quality warp system.

The three basic steps are to first increase the sampling rate, then warp the image using a lower quality interpolator for resampling, and finally downsample the image to its original size. These steps and their implementation are described below.

Typical digital images to be warped are sampled at the Nyquist rate. The first step in the new warping method is to increase the sampling rate above this point. This can be done by either sampling the analog image at a higher rate or by digitally processing an already Nyquist-sampled image to increase its sampling rate. The latter of these two approaches will be taken to maintain the modularity of the warper within a larger image processing system.

Because the input image to the system is sampled at or above the Nyquist rate, it is a complete representation of the underlying analog image. Therefore, theoretically, it is possible to obtain the exact value of that analog image at any point, even though only the sampled points are available. To achieve an efficient implementation, we limit the operation to an upsampling by a factor of 2^N, where N is a positive integer. This upsampling is done in both the vertical and horizontal directions. For example, if the input image is 512x512 pixels, upsampling by a factor of 2 would give an image of size 1024x1024. Upsampling by a factor of 4 would give a resulting image size of 2048x2048. It is important to use a very high quality interpolation method to obtain the values of the upsampled image.

Conceptually, the upsampling occurs in two steps. The first step is to insert the appropriate number of zeros into the image. For example, if the image is being upsampled by 2, then every other value in the upsampled image is zero. The values at the other locations are the original samples from the source image. Then this intermediate image is lowpass filtered to give the final upsampled image. The original image is critically sampled. Inserting zeros compresses the Fourier transform by a factor of two. The final lowpass filter leaves only a single copy of the compressed transform centered around every integer multiple of 2π. The spatial and corresponding frequency domain interpretation of this process are shown in Figure 5.2.



Nyquist Sampled Image
Image with zeros inserted for upsampling by factor of two
Final upsampled image with interpolated values
A
B
C
D
Insert zeros
LPF
0
?
|H1(ejω)|
|H2(ejω)|
|H3(ejω)|
-2π
2π

Figure 5.2: Upsampling stage.

At this point, it is clear why a high quality lowpass filter is needed for the interpolation. This lowpass filter must cleanly eliminate the undesired frequency components while leaving the desired frequency components relatively undisturbed.

This sort of constrained upsampling can be performed quite efficiently in hardware. The Pyramid Chip, developed at the David Sarnoff Research Center, is an excellent example of how this operation may be performed [12]. The upsampling can be implemented as a separable operation, which maps well to hardware. For the warping application, it is expected that a higher quality filter than the Pyramid Chip can provide will be needed. Still, a 4-tap or 6-tap fixed coefficient filter to perform the upsampling can be implemented in an field programmable gate array fairly easily.

In summary, there are two variables to consider in the upsampling procedure. The first is the amount by which the image will be upsampled. We constrain these values to be 2, 4, 8, etc. The second is the quality of the lowpass filter employed to perform the interpolation. It is expected that a 4-tap or 6-tap filter will be adequate. However, implementing a filter with more taps does not lead to a huge increase of complexity. The desired quality of the resultant warped image will determine appropriate values for these two variables.

Step two in the procedure is to warp this oversampled image using a lower quality interpolation filter for resampling. Because the image has been upsampled, its frequency content has been compressed. A lower quality filter used in the resampling step for the warp can give good results, as long as it has good characteristics over this smaller region of the frequency spectrum. This is shown in Figure 5.3.

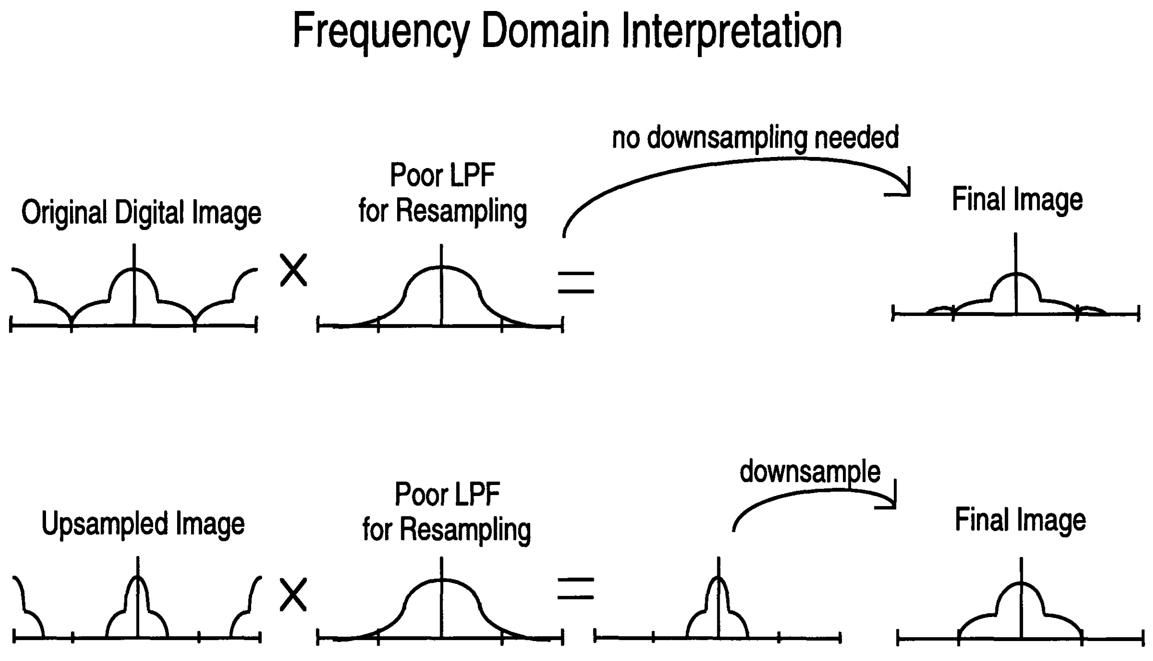


Figure 5.3: Frequency domain comparison.

Overall, the warp using the upsampled image does a better job of eliminating the unwanted duplicate high frequencies while leaving the desired frequencies undisturbed. Using a lower quality filter at this warping stage can drastically reduce the complexity, as it will use a smaller neighborhood of pixels.

Finally, the warped image is downsampled to the size it was before the upsampling occurred. In practice, this can be performed quite efficiently by modifying the geometric transformation function. The downsampling is then inherent in the warp equation. This approach combines steps two and three into one process.

Figure 5.4 shows how the downsampling is performed as part of the image warp.

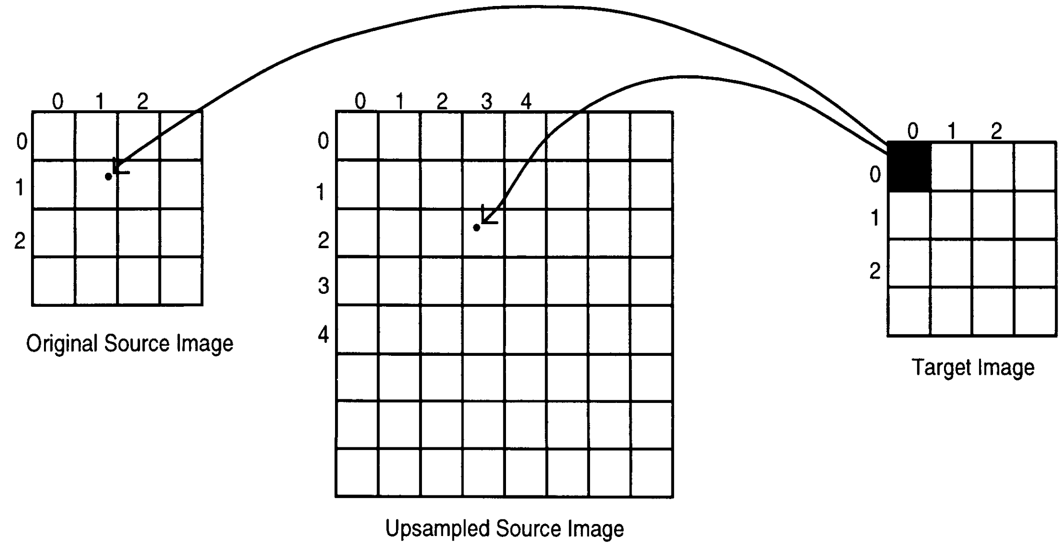


Figure 5.4: Downsampling as part of image warp.

In this example, the standard geometric transformation specifies that the value in the target image at [0,0] should come from the source image location (x1 = 1.7, y1 = 1.2). This value at a nonintegral location would be determined by looking at a neighborhood of pixels around this location. To access the same point in the upsampled image, simply multiply the referenced address by 2. In this case, the address to reference in the upsampled source image would be (x2 = 3.4, y2 = 2.4). If the image was upsampled by a factor of 4, the referenced address would be multiplied by 4. Then a neighborhood of pixels around this address is used to determine the output value. Besides this modification, the warp is performed identically to the standard method described earlier. In effect, the neighborhood of pixels that is used in the upsampled image corresponds to sample points of the original analog image that are closer to the desired location than an equal-sized neighborhood of pixels in the original Nyquist sampled image.

At first glance, one drawback to this approach for a hardware implementation seems to be that the warp will require more time to complete. The upsampling stage increases the size of the image. This data then needs to be passed to the warper memory, where it can be accessed at random. If the image data comes to the upsampling stage at a rate of one pixel per clock cycle and leaves the upsampling stage to be stored in the memory for the warp at the same rate, then there is a backlog delay incurred in this process, because more pixels are leaving the upsampling stage than are coming in.

However, this problem can be avoided. Basically, all that is required is to increase the bandwidth leaving the upsampling stage relative to the bandwidth entering the upsampling stage by the same factor as the increase in image size. There are many possible ways to do this. For example, assume that the warping stage uses bilinear interpolation. Then, there are four separate memory banks holding the image to be warped. If the upsampling stage upsamples by a factor of two in both directions, then the required bandwidth leaving the upsampling stage is four times that of the bandwidth entering it. So, if the image is entering the upsampling stage at one pixel per clock cycle, it should leave at four pixels per clock cycle. Because there are four independent memory banks holding the image, it is possible to write four pixels to this memory each clock cycle, thus meeting the bandwidth requirements. Then the only delay incurred will be a minor pipeline delay.

In conclusion, this approach to image warping gives a number of advantages (particularly for a hardware implementation) over the standard one-pass approach. It provides:

1. A method of image warping that doesn't grow quadratically in complexity with the resampling accuracy. The complete warping process is implemented in two steps. The first is upsampling. The second is warping with downsampling. These two steps can be tuned together to produce the desired quality warp. The complexity of the second step can be fixed at a reasonable setting. Then the upsampling step can be changed to give the desired quality.

2. A modular, independent, warping function that can process arbitrary spatial transformations. There are many specialized warping algorithms that achieve excellent results for particular applications; for example, rotation [11]. The warping procedure described in this thesis takes a Nyquist sampled image as input and produces a warped image as output. The warp function is not constrained to a particular type; it can be an arbitrary flow field.

3. A method of upsampling that is well suited for hardware implementation. The upsampling is constrained to a factor of 2^N, and is implemented using a separable filter. Both these factors lead to an efficient hardware implementation.

4. A method of downsampling that is simple to implement within the warping structure. Because the image was upsampled by a factor of 2^N, the downsampling can be performed as part of the warp by multiplying the vertical and horizontal components of the referenced source address by this factor. This multiplication is simply a bit shift. This approach also avoids unwanted aliasing in the downsampling step.

5.2 Comparison with Standard Techniques

Section 5.1 described the proposed approach to warping in a very general sense. Now, we examine the performance of the method for some very specific cases. There are basically three degrees of freedom in the general method. These are: the amount by which the original image is upsampled, the interpolation method used for upsampling, and finally, the interpolation method used for warping. In determining these factors, it must be kept in mind that the final goal is an efficient method for real-time image warping in hardware. Therefore, in the following analysis two of these factors will be fixed. First, the original image will always be upsampled by a factor of two in the vertical and horizontal directions. Second, the interpolation method in the warping stage will be fixed at bilinear. There are a number of advantages to doing this. Upsampling by a non-integer factor adds unnecessary complexity to the hardware. And, upsampling by a factor of two (as opposed to larger factors) can be implemented in hardware very efficiently. If the warping stage uses bilinear interpolation, a 2x2 pixel neighborhood is used. As discussed earlier, for a real-time hardware implementation, this leads to four separate memory banks, four weighting coefficients, and a four-term sum of products. Better interpolation methods for the warping stage would required an increase in all these quantities. Finally, if the image is upsampled by a factor of two, and if the warping stage has four separate memory banks that can be accessed in parallel, then it is possible to pipe the image data through the upsampling stage and into the warp memory without a backlog of the data in the upsampling stage. Under these constraints, the only variable left to manipulate is the upsampling method. The proposed method of image warping under different upsampling methods will then be compared with other common approaches.

The familiar Lena image will be used for all of the tests. Two measures will be used to assess the quality of the warp. The first will be a measure of Root Mean Square (RMS) error between the final warped image and the original image. The second will be a visual comparison. A total of five methods will be examined here. These are listed in Table 5.1.

Name	Description
Bilinear	No upsampling. Bilinear interpolation used in warp. (2x2 pixel area.)
Keys-3	No upsampling. Third order cubic convolution (a=-.5) used in warp. (4x4 pixel area).
Keys-4	No upsampling. Fourth Order Cubic Convolution used in warp. (6x6 Pixel Area).
Up2(Keys-3)	Upsample by two using Keys-3. (4 tap separable) Bilinear interpolation used in warp. (2x2 pixel area).
Up2(Keys-4)	Upsample by two using Keys-4. (6 tap separable) Bilinear interpolation used in warp.

Table 5.1: Warping Method Descriptions

The first test will be image rotation. The Lena image is rotated by 22.5 degrees a total of 16 times to bring it back to its original orientation. Then, the RMS error between this image and the original is taken. The results are summarized in Table 5.2.

Warping Method	RMS Error
Bilinear	11.384572
Keys-3	5.638434
Keys-4	4.430744
Up2(Keys-3)	7.712898
Up2(Keys-4)	6.915473

Table 5.2: RMS Error for Rotation

As expected, bilinear gives the largest error, and the pure fourth order cubic convolution gives the smallest error. It is worth noting that both upsampling methods do perform better than just bilinear, but not as well as the pure cubic convolution methods. To get a better feeling for what these numbers mean in terms of actual image quality, the images will now be examined.

Figure 5.5 shows the original lena image and the resultant image after being rotated 16 times using bilinear interpolation. The bilinear image is noticeable blurred. The is because bilinear interpolation attenuates the high-frequency components in the image.

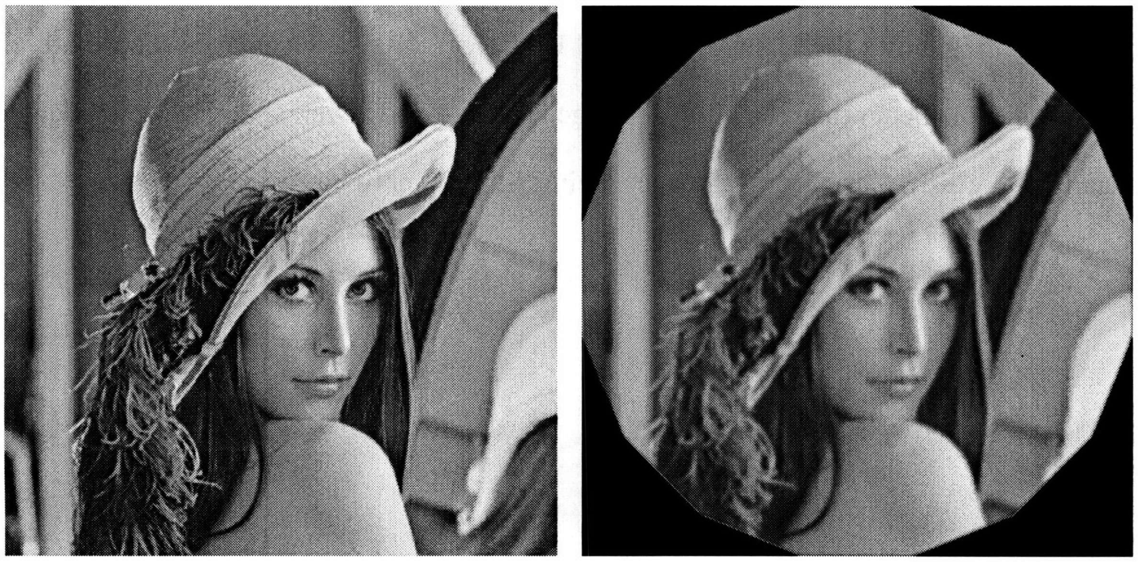


Figure 5.5: Original and bilinear rotation images.

Figure 5.6 shows the resultant image after rotation under the Keys-3 and Keys-4 interpolation methods. Both of these look much better than bilinear, with not nearly as much blurring. This is because these interpolation kernels have a much sharper cut-off in the frequency domain. The Keys-4 image is slightly better than Keys-3.

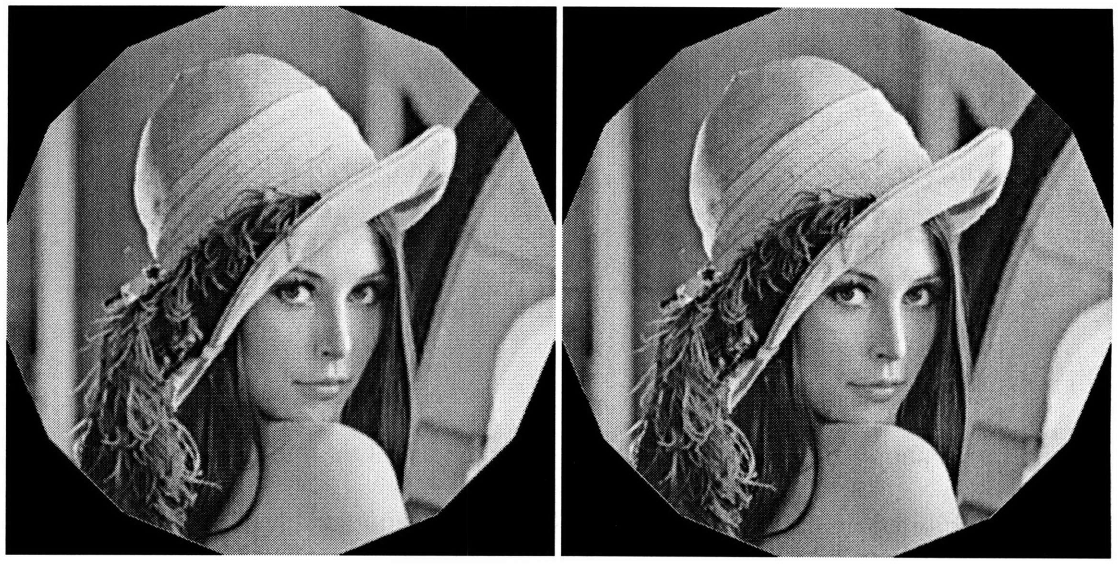


Figure 5.6: Keys-3 and Keys-4 rotation images.

Figure 5.7 shows the Up2(Keys-3) and Up2(Keys-4) rotation images. These are also both noticeable better than the bilinear image. They are, however, slightly worse in quality than the pure Key-3 and Keys-4 images, being slightly more blurred.

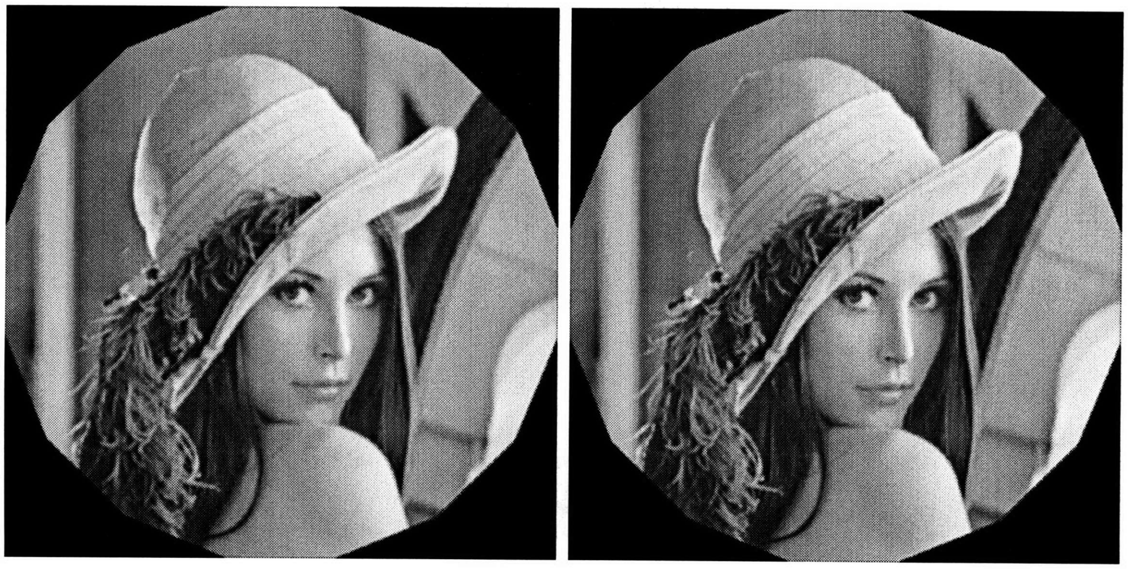


Figure 5.7: Up2(Keys-3) and Up2(Keys-4) rotation images.

All of these results, both visual and RMS error, are consistent with our expectations. Doing the upsampling followed by bilinear warping does improve the warped image quality over just bilinear warping, but, is still slightly inferior to just using better interpolation filters in the warping stage. However, the upsampling method does lead to a more efficient hardware implementation. The results from just this one test seem to suggest that using this upsampling method can greatly increase the warped image quality over pure bilinear, without a drastic increase in hardware complexity.

The second test is a translation test. The same methods for warping will be examined. The image is translated in a diamond pattern. The diamond cycle is composed of four steps. First translate the image up and right by 1/4 pixel, then down and right by 1/4 pixel, then down and left by 1/4 pixel and finally up and left by 1/4 pixel. This brings it back to the original location, where the RMS error relative to the original image can be measured. This process is then repeated on the same image, each time measuring the RMS error at the end of each diamond cycle. The purpose of this test is to determine how the different methods perform as a function of the number of times the image is warped. The results are shown in graph form in Figure 5.8 below.

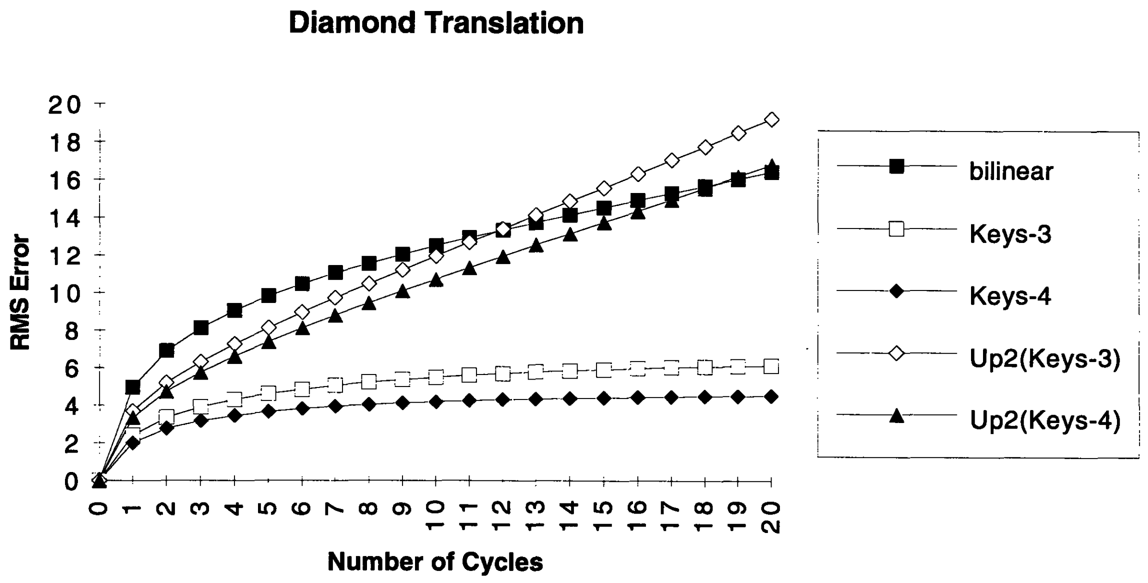


Figure 5.8: Diamond cycle RMS error.

Figure 5.9 provides a closer view of the graph for the first few values.

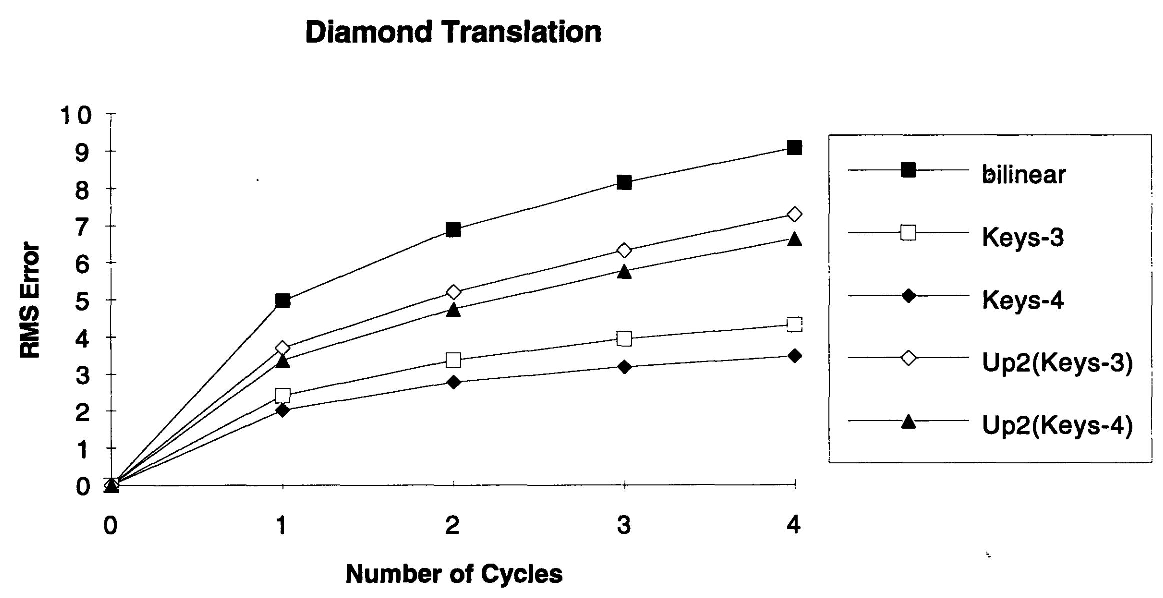


Figure 5.9: Diamond cycle RMS error.

There are a few interesting things to note from these figures. First, when the image has only undergone of few cycles, the RMS error for the images is similar to that of the rotation test. However, as the number of warps performed on the same image increases, the upsampling methods rapidly degrade the image quality, while the standard methods (Bilinear, Keys-3 and Keys-4) do not worsen as much. In fact, after 12 cycles, bilinear is actually better than Up2(Keys-3). This seems to suggest that the upsampling method is not appropriate for applications where the same image is to be warped many times in a repetitive fashion.

To get a feel for how the RMS error numbers correspond to the image quality, the images after 12 diamond cycles are shown below. Cycle number 12 was selected because this is where the RMS error for the bilinear and Up2(Keys-3) method are the same.

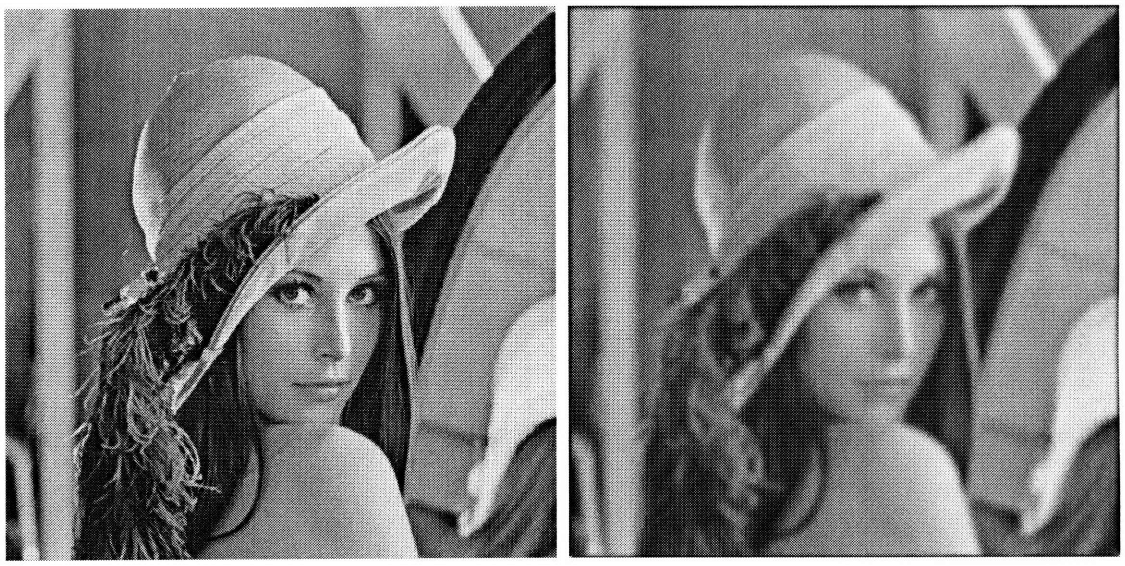


Figure 5.10: Original and bilinear after 12 diamond cycles.

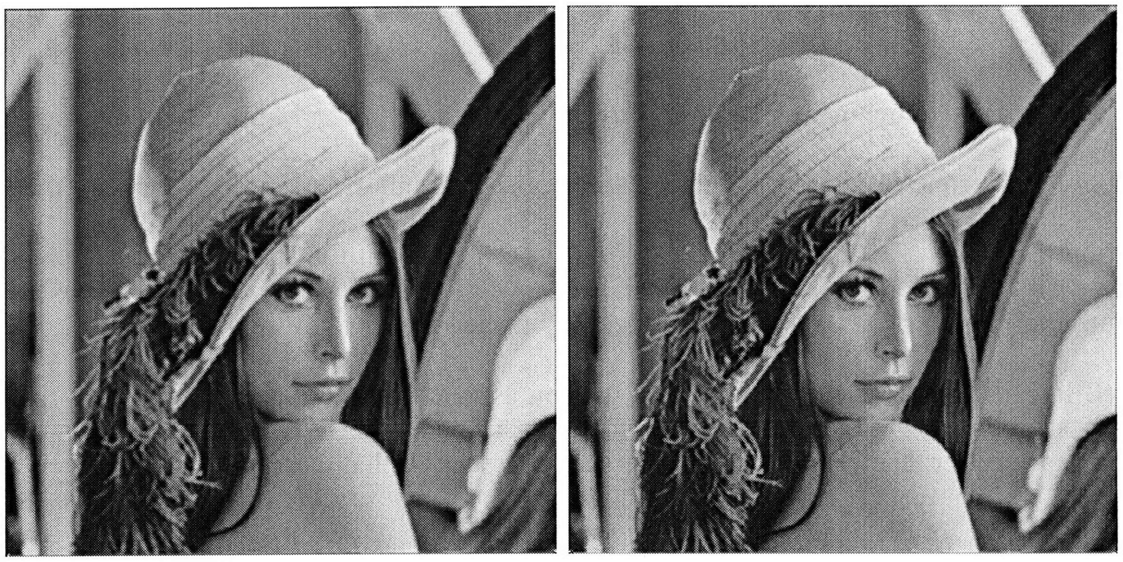


Figure 5.11: Keys-3 and Keys-4 after 12 diamond cycles.

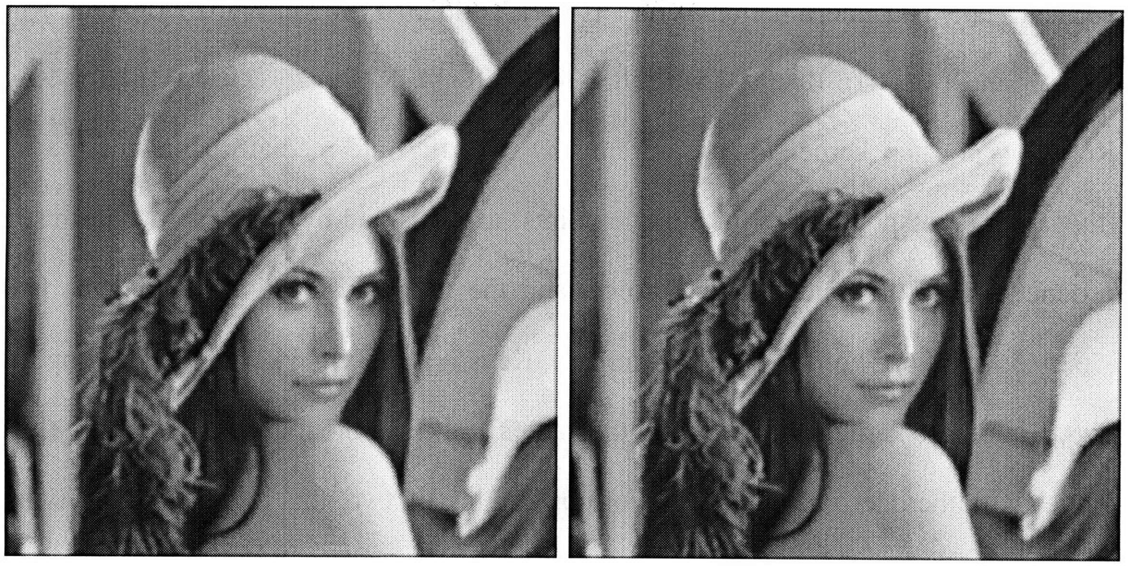


Figure 5.12: Up2(Keys-3) and Up2(Key-4) after 12 diamond cycles.

At this point, the relative image quality rankings are the same as for the rotation test. From best to worst: Keys-4, Keys-3, Up2(Keys-4), Up2(Keys-3), and Bilinear. It is interesting that even though Bilinear and Up2(Key-3) have the same RMS error at this point, the type of error is quite different. Both images are blurred; however, the blurring of the bilinear image is clearly more pronounced. The error in the Up2(Keys-3) image seems to also be coming from a reduction in the number of grey levels in the image. Over her face, blotches of the same intensity can be seen. This is very likely due to the repetitive warping pattern.

5.3 Optimal Upsampling Kernel Design

In the previous section, two methods were examined for the upsampling interpolation kernel, namely Keys-3 and Keys-4. The complete upsampling process occurs in two steps. In the first step, the image is expanded by a factor of two in both directions, inserting zeros in every other location. Then this image is convolved with the interpolation kernel. Since the kernel is separable, the convolution can be done in two passes, one horizontal and one vertical. The 1-D interpolation kernel is then [-.0625 0 .5625 1 .5625 0 -.0625] under Keys-3 and [.010416 0 -.09375 0 .583334 1 .58334 0 -.09375 0 .010416] for Keys-4. Keys-3 is a 4-tap kernel for upsampling by two; the intermediate values of the image (those not corresponding to the sample points) are obtained by taking a four-term sum of products of the four nearest data points and the four odd-location coefficients [-.0625 .5625 .5625 -.0625] of the kernel. Similarly, Keys-4 is a six-tap kernel for upsampling by two. Both these kernels will preserve the values at the sample locations and fill in the zero values with the interpolated values. For upsampling by two, this is achieved by fixing the center coefficient to 1, and fixing the coefficients at even locations to 0. Another important property that both kernels possess is unity D.C. gain. That is, when convolved with an image of constant value, the result is also that same constant. For upsampling by two, this can be achieved by requiring that the sum of the coefficients be 2.

Using the Keys-3 and Keys-4 kernels for the upsampling step works relatively well. However, improvements can be made. These kernels were meant to approximate a low-pass filter. In this respect they work well. However, for our application, it is known that after upsampling, the image will be warped using bilinear interpolation. The frequency response of the bilinear interpolation kernel is known. In particular, it attenuates higher spatial frequencies. Using this fact, and the other constraints placed on the upsampling kernel, a four-tap kernel (same length as the Keys-3 upsampling kernel) will be designed to give the best possible final image quality.

To satisfy the sample point preservation and unity D.C. gain properties, the general form of the symmetric 1-D four-tap kernel used for upsampling by 2 is:

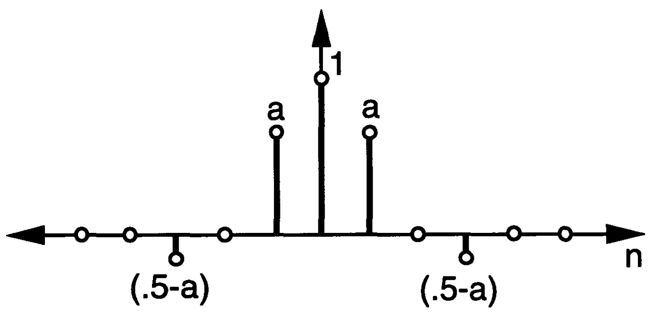


Figure 5.13: General 4 tap upsampling filter.

Note that after applying all these constraints, there is still one degree of freedom left, namely, the value of the constant *a*. To get the frequency response of this kernel, take the Fourier transform. After combining the terms, this reduces to:

$$H_1\left(e^{j\omega}\right) = 1 + 2a\cos(\omega) + 2\left(\frac{1}{2} - a\right)\cos(3w) \tag{5.1}$$

In general, increasing the value of *a* beyond 1/2 will make the cut-off from passband to stopband sharper, but will also add ripples. If *a* is exactly 1/2, the four-tap filter reduces to a two-tap filter corresponding to a linear interpolation kernel. The frequency responses for a few different values of *a* are shown in the graphs below.

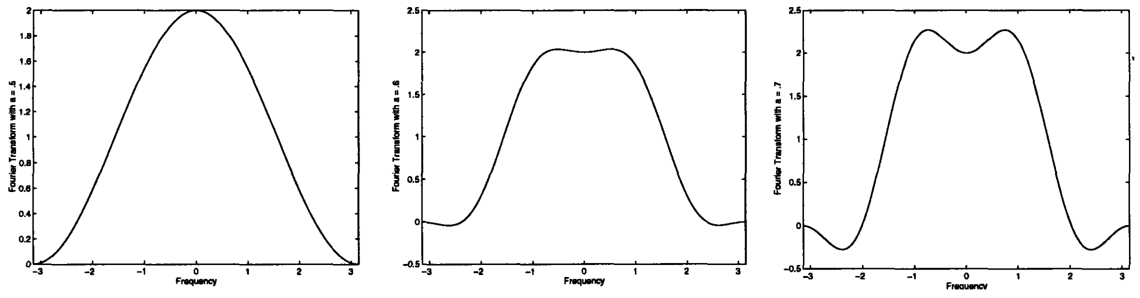
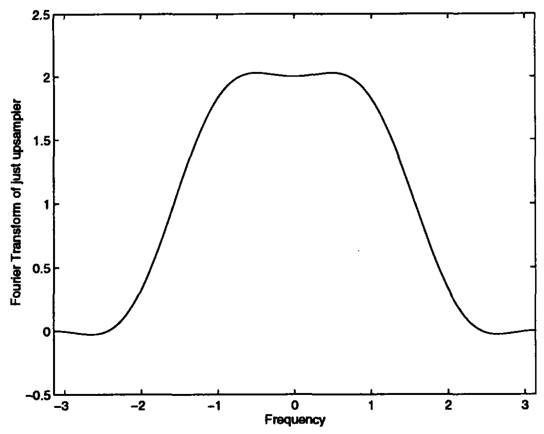


Figure 5.14: Frequency response of various 4-tap filters.

After upsampling using a filter like one of these, the image will be warped using a bilinear interpolation method. Since it is known the bilinear interpolation attenuates the high spatial frequency components, it makes sense to pick the upsampling filter in a way that increases these same high frequency components, so that the overall effect on them is as small as possible. Basically, the first filter must be picked so that the two filters together closely approximate an ideal low-pass filter. There is a complication to this selection: the bilinear interpolation method used for warping doesn't have the same spectral characteristics for every warp. For example, if the image is merely translated by an integer amount, there is no degradation of the high-frequency components. On the other hand, for a half-pixel translation, the high spatial frequency components will be maximally attenuated. Between these two extremes, there are many levels of attenuation. Therefore, the upsampling kernel must be selected based on the range of possibilities that can occur in the warping stage. Finally, for a hardware implementation, the kernel coefficients should all be sums of powers of 2. This reduces the multiplication of the data values by the coefficients to an arithmetic bit shift and sum operation. Taking these factors into account, through examination of a range of Fourier transforms, and also through tests on many different warp types, a value of a that gives good overall results was determined. This is a = .59375. With this value for a, the kernel becomes [-.09375 0.0 .59375 1.0 .59375 0.0 -.09375]. This kernel will be affectionately referred to as the Lohmeyer kernel. Figure 5.15 shows both the frequency response of this upsampling kernel followed by the bilinear kernel used for integer translation and the frequency response of this upsampling kernel followed by the bilinear kernel used for a half pixel translation on the upsampled image. These two graphs represent the extremes of possible total frequency response of the system before downsampling.



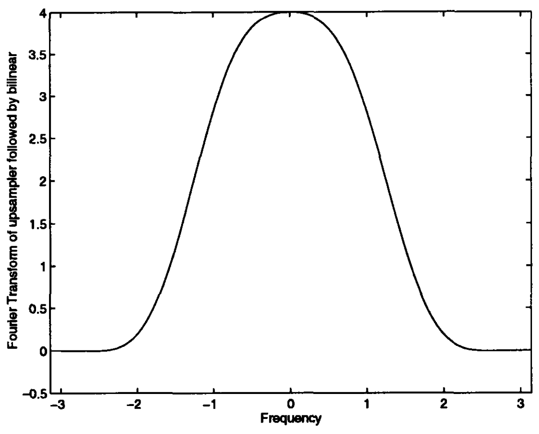
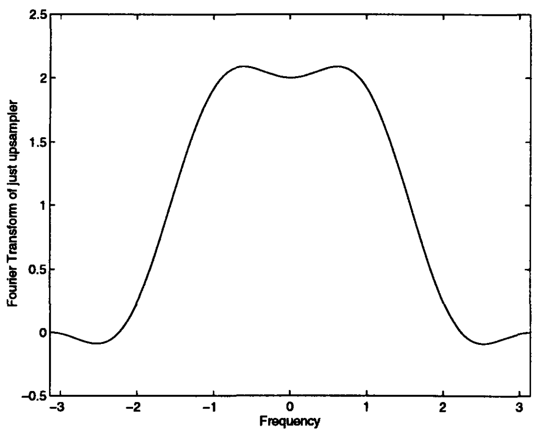


Figure 5.15: Frequency response of overall system for integer and half-pixel translation using the Lohmeyer kernel for upsampling.

To measure the performance of this upsampling method, the same tests as before will be used. At this point, we will also examine the performance of one other four-tap kernels defined by [-.125 0 .625 1 .625 0 -.125]. It is called the Pyramid kernel because this upsampling kernel can be performed by a currently available VLSI chip known as the Pyramid Chip, [12]. The Pyramid Chip was designed at the David Sarnoff Research Center. It can perform the standard filter and resampling operations required to generate a variety of image pyramids that are used in many machine vision tasks [2]. The Pyramid kernel is the best kernel available in the Pyramid Chip for use in the upsampling operation for the image warping application. Figure 5.16 shows the frequency response of this kernel, and of this kernel cascaded with the frequency response of the bilinear kernel used for 1/2 pixel translation on the upsampled image.



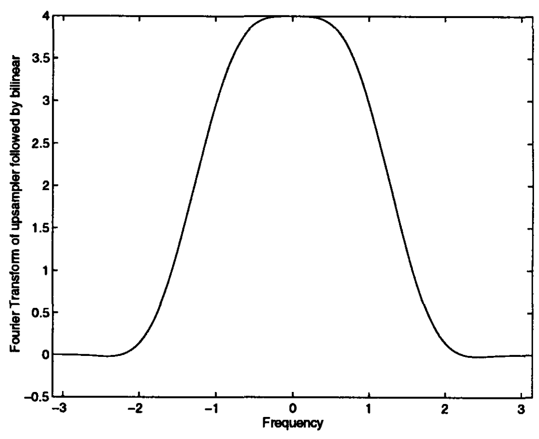


Figure 5.16: Frequency response of overall system for integer and half-pixel translations using Pyramid kernel for upsampling.

Note that the Pyramid upsampling kernel peaks the high frequency components more than the Lohmeyer kernel.

The first test is rotation. All the RMS error results are summarized in Table 5.3 below.

Warping Method	RMS Error
Bilinear	11.384572
Keys-3	5.638434
Keys-4	4.430744
Up2(Keys-3)	7.712898
Up2(Keys-4)	6.915473
Up2(Loh)	5.494398
Up2(Pyr)	7.153037

Table 5.3: RMS Error for Rotation

Up2(Loh) performs very well in this test. In fact, the RMS error is slightly lower than Keys-3. Up2(Pyr) is better than bilinear in this test, but is outperformed by nearly all the other methods.

Figure 5.17 shows the images after rotation under these two new methods.

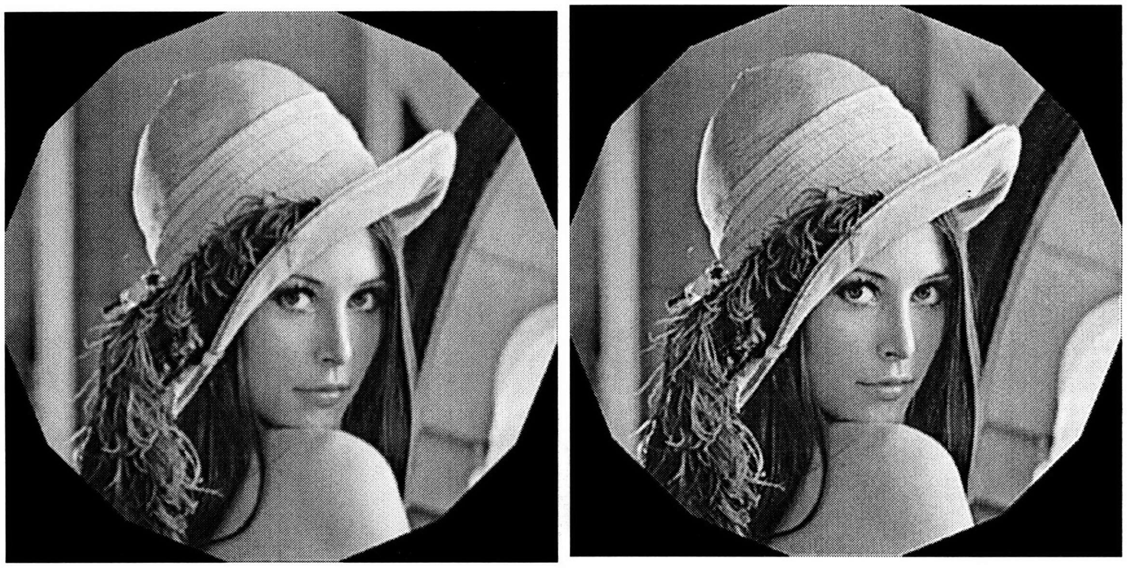


Figure 5.17: Up2(Loh) and Up2(Pyr) images after 16 rotations.

As expected, both look fairly reasonable. It is hard to perceive a difference in quality between the Keys-3 and Up2(Loh) image. The Up2(Pyr) image is slightly different from the other images because it has the high frequency components increased, rather than attenuated.

As before, the second test is translation using the diamond pattern. The same graphs as before are shown in Figures 5.18 and 5.19, except that now they include the two new methods.

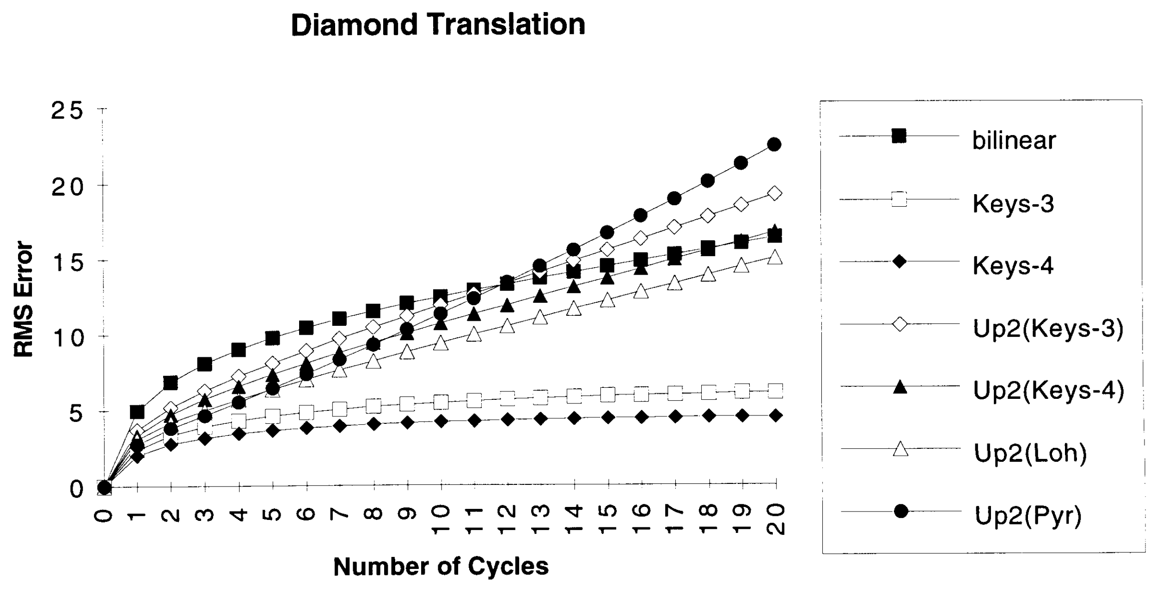


Figure 5.18: RMS error for diamond test.

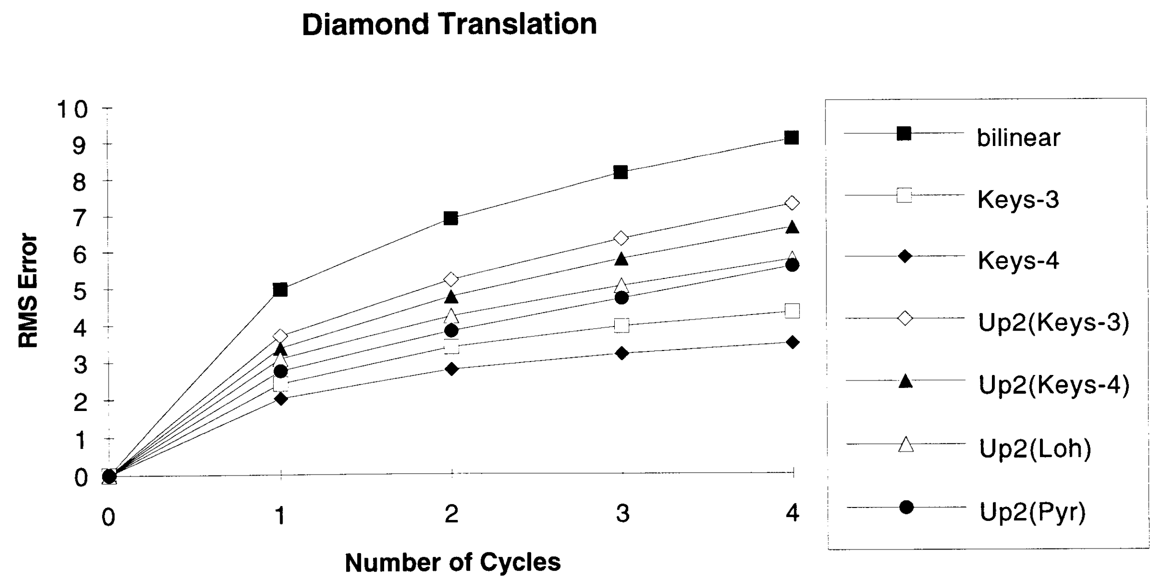


Figure 5.19: RMS error for diamond test.

Note that although both of the new methods perform well at the start, they degrade the image more rapidly than other methods as the number of warps increases. To get a feel for the type of error that is being accumulated, the images obtained after 12 cycles of the diamond pattern are shown in Figure 5.20.

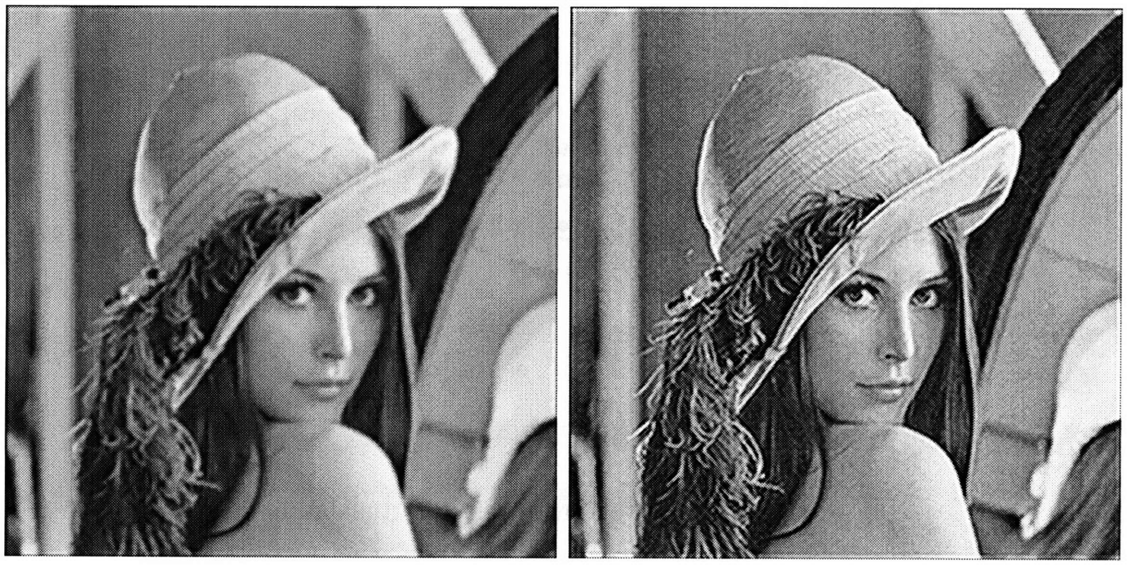


Figure 5.20: Up2(Loh) and Up2(Pyr) for diamond test.

The Up2(Pyr) image has been sharpened too much. The Up2(Loh) image has the same type of defect as the other upsampling methods have. Along with some blurring, patches are beginning to form. This is very likely due to the repetitive warping pattern.

To determine if the errors in the upsampling methods are really partially due to the repetitive pattern of the warp, a final test is performed using all the different methods. This is a random translation test. The image is translated horizontally and vertically by a random amount between zero and one, and then translated back again. The RMS error is taken at this point, as in the diamond pattern. This process is then repeated. The same sequence of random translations is used for each method. The RMS error results for all the methods are shown in Figure 5.21 and Figure 5.22.

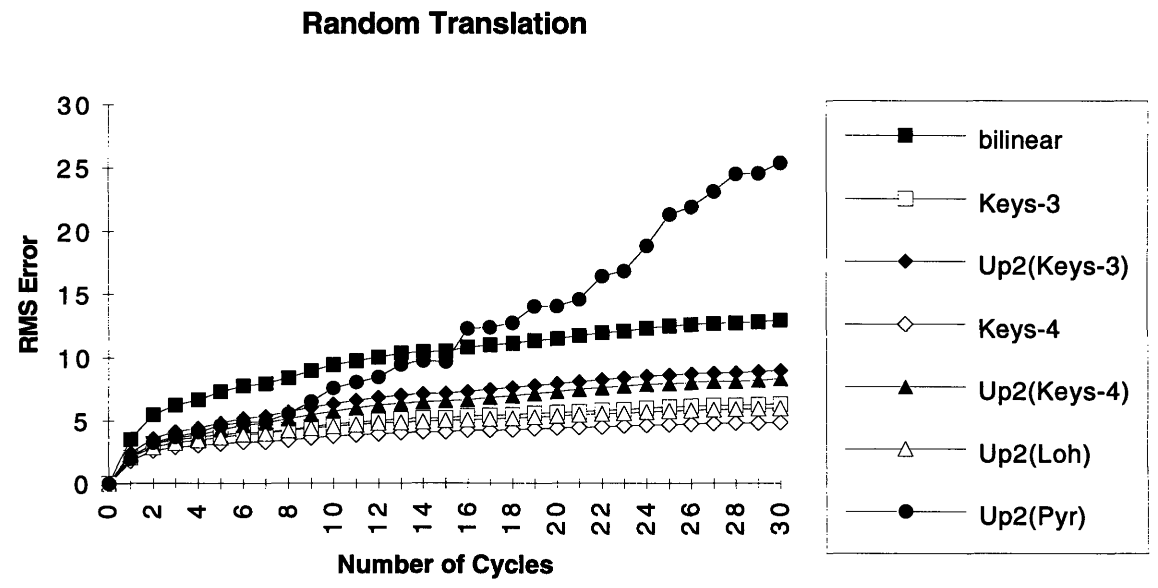


Figure 5.21: RMS error for random translation test.

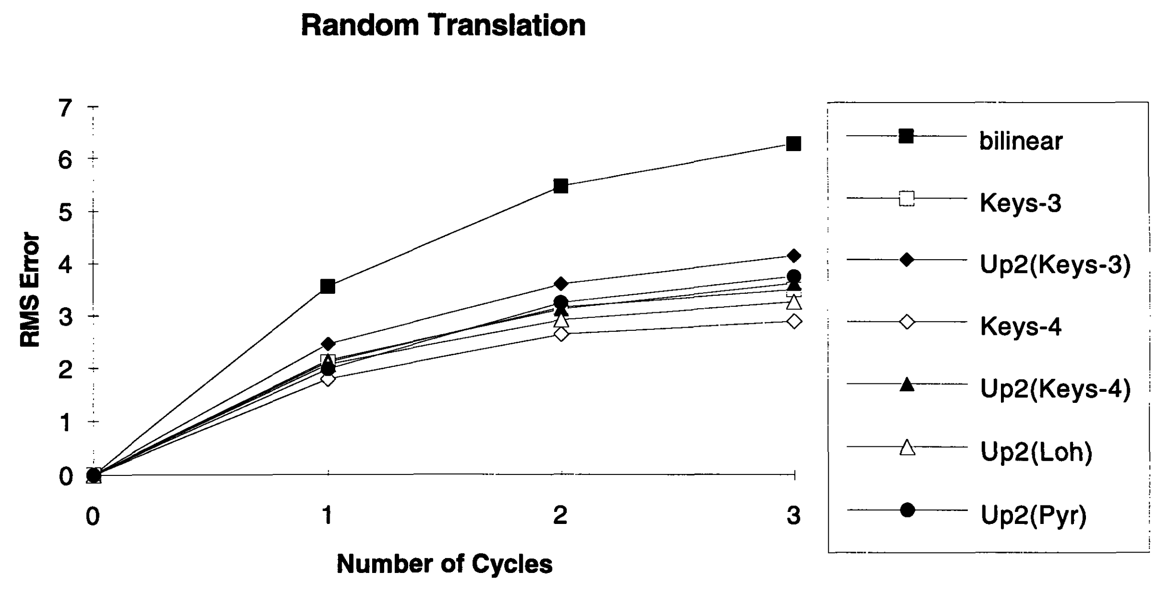


Figure 5.22: RMS error for random translation.

The Up2(Pyr) method still performs poorly as the number of warps increases. However, all the other upsampling methods do much better. In particular, the Up2(Loh) method performs well no matter how many times the image is translated. In fact, it is never worse than the Keys-3 method. This is due to the fact that the warps now have a random translation. Thus, the image experiences a wide range of the possible overall filter effects. Sometimes the high frequencies are attenuated, and sometimes they are peaked. If the translation is close to integer, they are peaked, since the bilinear interpolation for the warp did not attenuate them too much, and the upsampling increased them. If the translation is closer to a 1/2 pixel, then they are attenuated because now the attenuation in the warping stage overpowers the peaking in the upsampling stage. Overall, this has a balancing effect when the same image is warped many times. The images after 30 warp cycles are shown in the following figures.

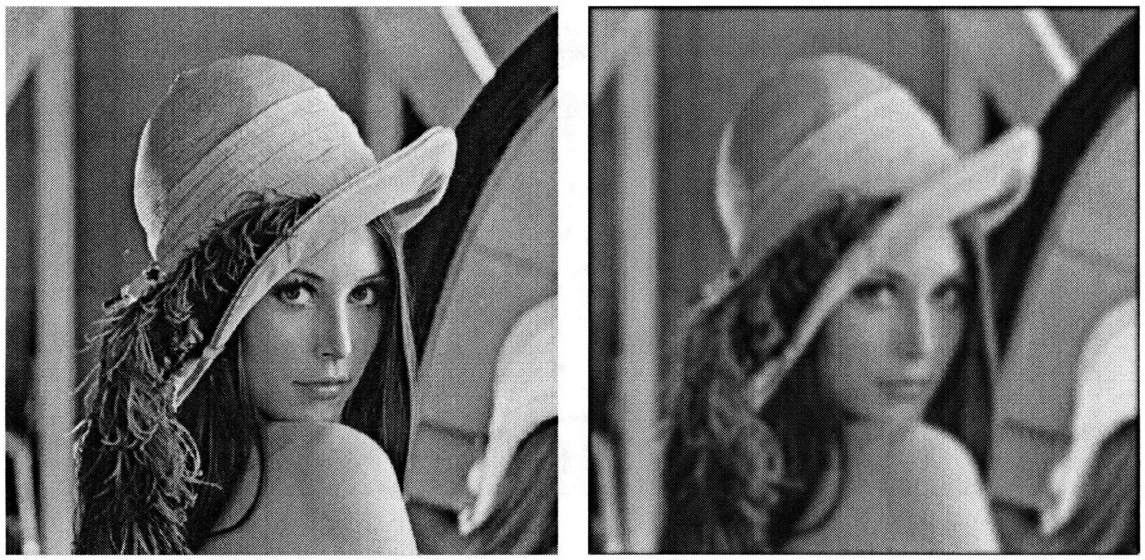


Figure 5.23: Original and bilinear image.

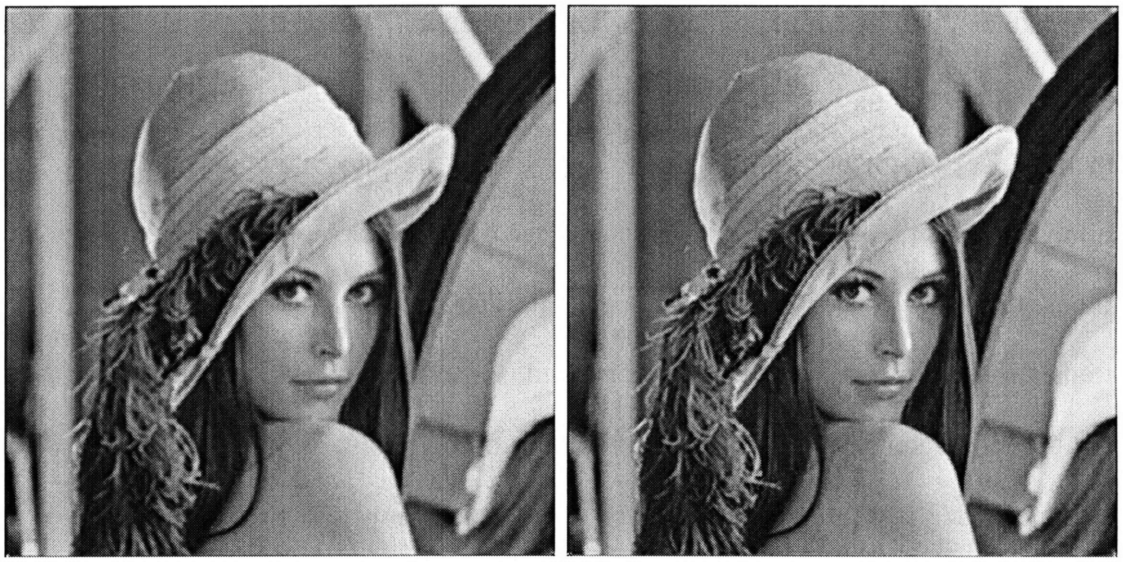


Figure 5.24: Keys-3 and Keys-4 images.

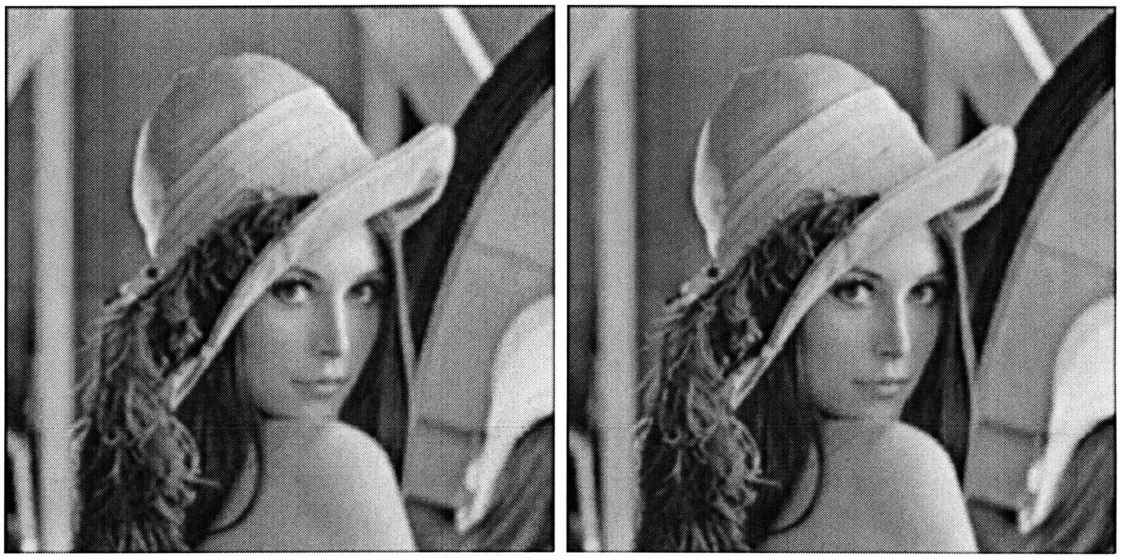


Figure 5.25: Up2(Keys-3) and Up2(Keys-4) images.

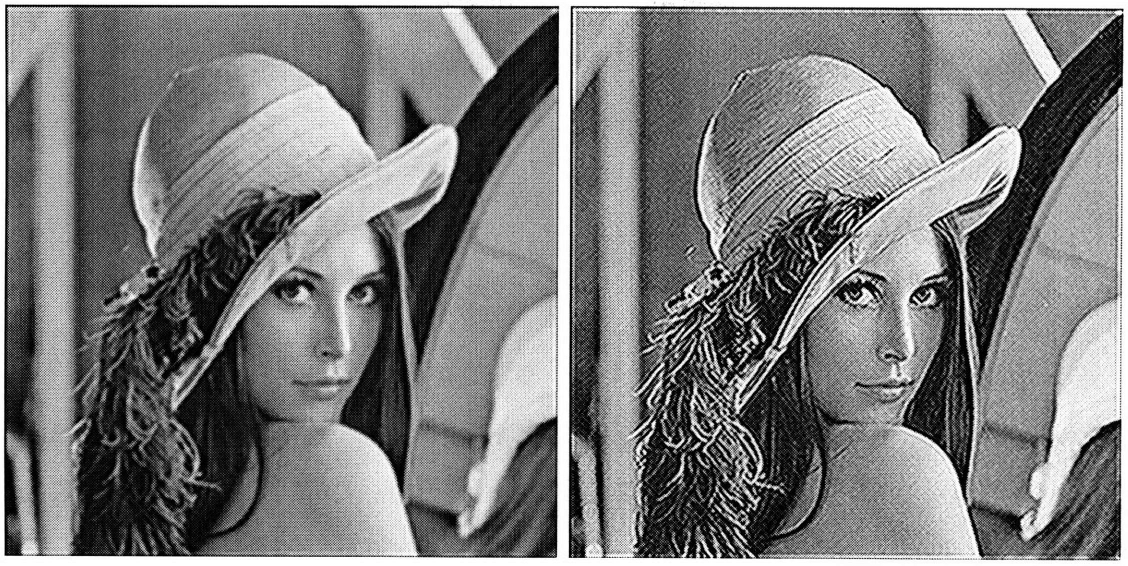


Figure 5.26: Up2(Loh) and Up2(Pyr) images.

The Up2(Pyr) image quality is very poor. This is because the high frequencies have been peaked too much. The Up2(Keys-3) and Up2(Keys-4) images are slightly blurred, and the bilinear image is very blurred, as before. The Keys-3, Keys-4 and Up2(Loh) images all look much better than the other images. There is very little noticeable blurring or other defects. This confirms the RMS error calculations.

5.4 Final Remarks

The proceeding tests have shown that this method of upsampling, warping, and downsampling can indeed give very good results. For multiple warps of the same image, as long as the warping pattern is not repetitive, the quality of a warp using a 2x2 pixel area for interpolation on an upsampled image can meet and even exceed that of a warp using a 4x4 pixel area for warp interpolation. These results were achieved for an upsampling kernel that had only four taps. Increasing the number of taps of the upsampling kernel may provide even better results. Also, this method of warping leads to a greatly simplified hardware implementation. The number of components required to do the warp can be drastically reduced, without a large decrease in image quality.

Chapter 6

Hardware Design

6.1 Background

This chapter describes the design of a hardware system that is capable of warping digital images in real-time. This means that it can warp images at a rate greater than thirty frames per second. It is one component in a very general machine vision hardware system that will be capable of performing a variety of tasks. These tasks include image stabilization, image registration, moving target detection, and obstacle avoidance for autonomous vehicle control. Image warping is a basic operation needed to perform most of these tasks.

6.2 Functionality Requirements

The requirements of this warping hardware are as follows.

- At a minimum, the geometric transform must be able to accommodate quadratic polynomial transformations.
- The interpolation method for resampling must be at least as good as bilinear.
- The system must run at 20Mhz, with relatively easy upgrade path to 30Mhz.
- The system must be able to accommodate an image size of 1024x512 pixels.
- For real-time operation, it must produce one output pixel each clock cycle (ignoring overhead).
- It must be able to read (warp) and write an image simultaneously.
- Board space and cost are at a premium.

6.3 Components

A block diagram of the overall system is shown in Figure 6.1 below.

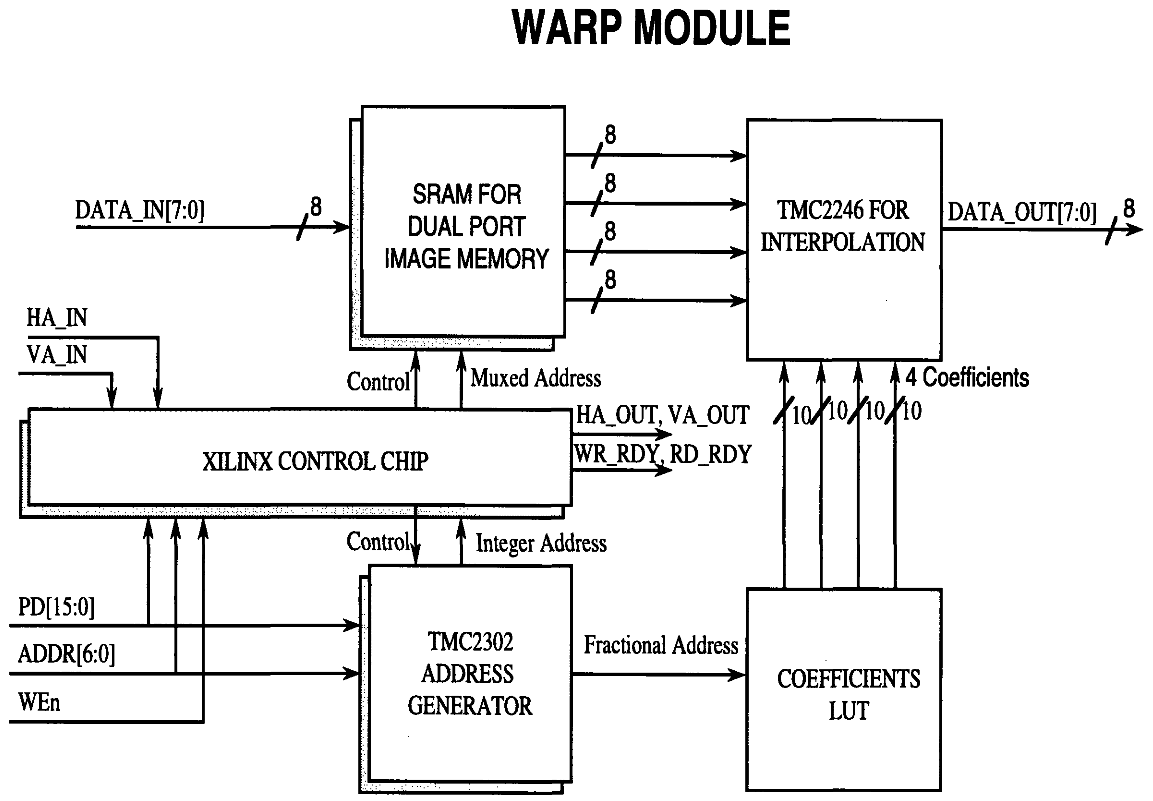


Figure 6.1: Warp module block diagram.

The actual hardware components used to realized the blocks above are: two Xilinx chips (XC4005H) for control, two TMC2302 chips for warping address generation, one TMC2246 for the four-term sum-of-products needed for bilinear interpolation, eight 128K x 8 SRAM for image storage, and five 8Kx8 EEPROMS for the interpolation coefficient look-up table. The use of these components to perform the warp is described in the following sections.

6.4 High-Level Design

The following two tables list and describe the inputs and outputs to the warp board.

INPUT NAME	DESCRIPTION
DATA_IN[7:0]	Image Data Input
HA_IN	Horizontal Active In
VA_IN	Vertical Active In
WE\	Program Write Enable
ADDR[6:0]	Program Address Bus
PD[15:0]	Program Data Bus
WR_ENA	Write Enable Line
RD_ENA	Read Enable Line
RD_FIRE\	Read Synch Pulse
RESET	System Reset Line
CLK	System Clock

Table 6.1: Warp Board Inputs

OUTPUT NAME	DESCRIPTION
DATA_OUT[7:0]	Image Data Output
HA_OUT	Horizontal Active Out
VA_OUT	Vertical Active Out
WR_RDY	Write Ready Status Line
RD_RDY	Read Ready Status Line

Table 6.2: Warp Board Outputs

6.5 Warp Board Functional Description

The warp function has as input an image and a set of 32 motion (transform) parameters that describe a cubic polynomial geometric transform. The motion parameters are changed by the address generation block into motion vectors representing a pointer for each pixel in the warped image to the location of origin on the input image. These vectors point to sub-pixel locations, so in general some sort of interpolation is required. This interpolation is performed using the integer and fractional part of the reference source address. For bilinear interpolation, the integer part selects the upper left pixel of the 2x2 neighborhood of pixels to be used for the interpolation. The fractional part determines the weights to be applied to the sum of products of these four pixels to get the final interpolated result.

The hardware may also allow for a general flow-field input to handle arbitrary warps. This design does not include this option, but can be easily modified to do so. In this case, all that one needs are additional inputs to the control block for the values of the flow-field and the timing that accompanies them. Internal to the control block, these flow-field values will be used instead of the values that come from the address generation block.

6.6 Warp Control Block

The warp control block is implemented in two Xilinx chips. One chip handles all the vertical addresses and control, and the other handles the horizontal addresses and control. The internal logic of both chips is very similar. If only one chip was used to handle all the addresses, over 250 input and output pins would be required. The FPGA chip that has this many pins has much more logic resources than required. By spliting the control task into two chips, it is possible to use chips that are much smaller and contain logic resources that closer match the requirements. In this way, cost is also reduced.

The warp control block provides the control for all the other blocks. It takes inputs from external controllers and uses them to program and sequence all the operations that need to be done to perform the warp. The warp control module also takes the addresses produced by the address generation block and produces from these the correct addresses to access in the SRAM memory block to provide the four pixel values. Finally, this block provides the status and control outputs to the rest of the system, including the output warped image timing lines.

Before a warp can begin, all the registers of the TMC2302 and various other control registers need to be programmed. The program data and address buses are used for this. The complete machine vision hardware system consists of many modules. Each of the modules has one WE\ pulse associated with it, which is only activated when data is written to that module. The ADDR[6:0] and the PD[15:0] signals are shared by all modules. To load the program data into the warp module, set the data and address and hold down WE\ for two complete clock cycles. Listed below are the data and valid ranges for each program bus address.

WARP CONTROL REGISTER	DESCRIPTION
00/h - 2F/h	transform coefficients as defined for the TMC2302
30/h - 43/h	TMC2302 control regs
45/h RD_HSTART	Horizontal start delay of read operation
46/h RD_VSTART	Vertical start delay of read operation
47/h RD_HBLANK	Horizontal blanking time

Table 6.3: Control Registers

The image timing is defined by two signals that are associated with the digital video data paths: HA (Horizontal Active) and VA (Vertical Active). Each positive VA signal identifies the active (valid) time of the image. The internal images are always in progressive scan form (no interlace). Each positive HA signal within the positive VA signal identifies an active horizontal line of the image. The VA signal can change state any time during the horizontal blanking time (HA = low). The HA signal has to go high at the same clock as the first valid pixel data on a line, and has to turn to low on the clock after the last valid pixel data on that line. See Figure 6.2 for the relationship of the timing signals, data, and the system clock.

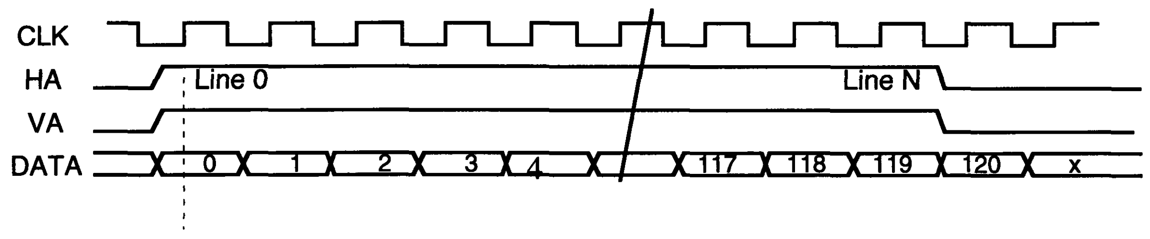


Figure 6.2: Image data timing.

When the warper is ready to be written to, WR_RDY will go high. After programming all the write parameter registers, the external controller will assert WR_ENA, WR_RDY will then go low. A write will then begin with HA and VA going high. VA may go high first, but the write does not begin until HA goes high. This process is shown in Table 6.4.

SYSTEM CONTROLLER	WARP MODULE
	WR_RDY -> LOW
	WR_RDY -> HIGH
PROGRAM REGISTERS	
WR_ENA -> HIGH	
	WR_RDY -> LOW
	Reset memory address counters

Table 6.4: Write Process

SYSTEM CONTROLLER	WARP MODULE
	Wait till HA and VA both go LOW
	Wait till HA and VA both go HIGH
	First Datum stored at 0,0
	Pixel Counter Incremented
	Continue Storing Data
HA -> LOW	
	Stop storing data
	Reset pixel counter
	Increment line counter
HA -> HIGH	
	Store next line
HA & VA -> LOW	
	Terminate write sequence
	set WR_RDY HIGH

Table 6.4: Write Process

When the warper is ready to be read from (and compute a warp), RD_RDY will go high. After programming all the read parameters (these go to the Xilinx as well as to the TMC2302's), the external controller will assert RD_ENA. RD_RDY will then go low. After RD_ENA, the read operation begins a deterministic number of clock cycles after the RD_FIRE/ pulse. This delay is determined by RD_HSTART and RD_VSTART. The entire read process is shown in Table 6.5.

SYSTEM CONTROLLER	WARPER MODULE
	RD_RDY -> LOW
	RD_RDY -> HIGH
PROGRAM REGISTERS	
RD_ENA -> HIGH	
	RD_RDY -> LOW
	wait
RD_FIRE\ -> PULSE	
	HA_OUT and VA_OUT -> LOW
	Initialize TMC2302
	HA_OUT and VA_OUT -> HIGH
	Output Data
	HA_OUT cycled
	VA_OUT -> LOW
	HA_OUT continues cycle
	RD_RDY -> HIGH

Table 6.5: Read Process

6.7 Address Generation Block

The address generation block comprises two TMC2302 chips. These chips are capable of calculating a third-order polynomial transformation. One chip will compute the vertical addresses, and the other will compute the horizontal. For a third-order polynomial in two variables, there are sixteen constants. The third-order polynomial transformation that will be used is:

$u[x,y] = a + bx + cx^2 + dx^3 + ey + fyx + gyx^2 + hyx^3 + iy^2 + jy^2x + ky^2x^2 + ly^2x^3 + my^3 + ny^3x + oy^3x^2 + py^3x^3$

and likewise for v[x,y].

The address generator computes a sub-pixel address (u,v) to reference in the source image for each pixel position [x, y] in the target image. Two TMC2302's are used to generate the vertical and horizontal addresses with 24 bits of precision each. Since the largest image size is 1K x 512 pixels, 10 bits are needed from each chip for the integer part of the address. That allows 14 bits for the fractional part. Only 5 bits of the fractional part will be used. These five fractional bits from both of the TMC2302 chips are fed to the interpolation coefficient look-up table. The fractional portion of the reference addresses controls the weight that each pixel is given in the sum of products computed for the interpolation.

6.8 Warp Memory Block

The warper memory component is configured as a ping-pong image memory. There are two complete SRAM memory buffers, each large enough to hold an entire image. While one is being written to, the other can be read from. In this way, it is possible for the warping system to simultaneously warp an image already stored in the warper while taking a new image as input from somewhere else.

Each of the two memory buffers is implemented with four 128K x 8 bit SRAM chips. Together four SRAM chips hold one entire image in a checkerboard interleaved storage pattern as shown in Figure 6.3.

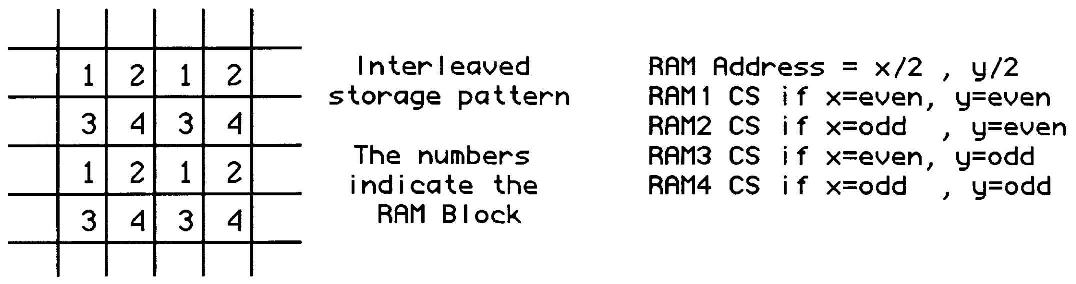

1 2 1 2
3 4 3 4
1 2 1 2
3 4 3 4
Interleaved storage pattern
The numbers indicate the RAM Block
RAM Address = x/2 , y/2
RAM1 CS if x=even, y=even
RAM2 CS if x=odd , y=even
RAM3 CS if x=even, y=odd
RAM4 CS if x=odd , y=odd

Figure 6.3: Image storage pattern.

Specifically, all pixel values that correspond to even horizontal and vertical addresses are stored in SRAM block number one; all pixel values that correspond to odd horizontal addresses and even vertical addresses are stored in block number two; etc. The entire image is stored in this manner so that the four pixel values required for bilinear interpolation can be accessed in parallel on a single clock cycle.

A write to the image memory is fairly simple. The data values are fed directly to the image memory module. The correct addresses and the appropriate memory block enables are provided by the control module. The image is stored sequentially, line by line, according to the HA and VA signals.

A random access read from the image memory is done as part of computing a warp. The control block takes the integer part of the reference source address. From this, it produces four addresses for the memory module, one for each block. In this way the four pixel values closest to the reference source address can be obtained simultaneously.

6.9 Coefficient Look-Up Table

The coefficient look-up table is implemented with five 8Kx8 EEPROMS. The address inputs to this LUT are simply the fractional part of the reference source address. The output is the set of weighting coefficients for the four pixel values that come from the SRAM module. The pixels are weighted in linear proportion to how close they are to the reference sub-pixel address. For example, if the fractional part of the horizontal address is 1/4 and the fractional part of the vertical address is 3/4, then the upper left pixel gets a weight of 3/16, the upper right gets a weight of 1/16, the lower left get a weight of 9/16, and the lower right gets a weight of 3/16.

6.10 Sum of Products Interpolator

The sum of products for bilinear interpolation is performed on a single TMC2246. This chip computes four 11-bit by 10-bit multiplications followed by a summation of the four resultant values. The multipliers perform signed-bit multiplications, so only 10-bit by 9-bit (unsigned) multiplications are available. Based on the 5-bit fractional source addresses, the coefficient look-up table should generate an 11-bin coefficient for each of the four data values. This 11-bit result consists of one integer bit and ten fractional bits. Since the integer bit can only be 1 when both fractional parts are 0, it is sufficient to represent only the fractional bits, and to set the result to $(1 - 2^{-10})$ when the output should be 1.0000. Therefore only a 10-bit result is generated from the LUT. The TMC2246 chip can compute the results either in fractional or integer representation. For proper results, the fractional representation should be used. In the fractional representation, the 10-bit input is represented as a sign with 9 fractional bits. The 11-bit input is represented as a sign bit, one integer bit, and 9 fractional bits. By setting the most significant bit and least significant bit to zero for the data and using the ten least significant bits of the 11-bit input (zero in the most significant bit) for the coefficient, the result will be eight fractional bits, six integer bits and a sign bit. The 8-bit result required comprises the seven most significant bits of the fraction and least significant bit of the integer of the result.

6.11 Conclusion

This hardware meets all the design specifications. In particular, it is capable of:

- performing a third-order polynomial transformation, with bilinear interpolation;
- having all components operate at 20Mhz;
- allowing most components to operate at 30Mhz (and those that currently do not will very likely be able to in future versions);
- warping a 1024x512 image in 26.2 msec, which corresponds to 38 frames/sec;
- simultaneously reading and writing an image.

Chapter 7

Conclusions

7.1 Summary of Work

This thesis first gives background on digital image warping theory and applications. Next, it analyzes the performance of various time-domain interpolation kernels that can be used for resampling. The results of this analysis lead to a new approach to performing general flow-field image warps in real-time using digital hardware. This approach is analyzed and compared to other methods. Finally, the thesis describes the design of a digital hardware system capable of performing real-time image warping.

7.2 Further Research

Image warping is a very large field, and research is being performed in a wide variety of areas within the field. Some possibilities for further work based on the contents of this thesis are listed below.

1. Include low-pass filtering in the upsampling stage with cut-off frequency based on the warp parameters, to guarantee that no aliasing will occur after the image is warped.

2. Apply the upsampling image warping method to other methods of image warping. For example, it may be useful in separable image warp algorithms.

3. Examine other upsampling and warping kernel combinations for even higher image quality.

Appendix A

Forward Difference Method

A.0.1 Algorithm

The forward difference method is used to simplify the computation of polynomials in specific applications. It can be used quite effectively for the address generation of a polynomial transformation. An example of a quadratic polynomial transformation will be used to demonstrate the method. The address generator needs to compute for each integer pixel position [x,y] in the resulting image, a pixel address [u,v] that references the input image. The equation for u is:

$$u[x, y] = a + bx + cy + dx^2 + ey^2 + fxy \tag{A.1}$$

The value of u could be found by calculating the whole sum of products each time x or y changes. This requires many multiplications and additions. The forward difference method takes advantage of the fact that x and y are integers and the target image locations are scanned left to right and top to bottom. If this is true, it can be shown that the current value of u can be found from the previous value of u and an increment. This is shown below.

Table 8.1 shows the value of u as y is increased. The increment is the difference between u[0,n] and u[0,n-1]:

Value	Increment
u[0,0] = a	-
u[0,1] = a+c+e	+c+e
u[0,2] = a+2c+4e	+c+3e
u[0,3] = a+3c+9e	+c+5e
u[0,4] = a+4c+16e	+c+7e

Table A.1: Increment Table

From this table it is apparent that each time y increases by one, the value for u increments in a consistent fashion. This is captured in the following equations.

$$u[0,y] = u[0,y-1] + acce[y]$$

$$acce[y] = acce[y-1] + 2e$$

and to be used later:

$$accf[y] = accf[y-1] + f$$

Row Equations (A.2)

Also, initial values need to be set:

$$u[0,0] = a$$

$$acce[0] = 0$$

$$acce[1] = e + c$$

$$accf[o] = 0$$

Initial Values (A.3)

Using these equations, the value of u to start every row with can be found. Once the beginning row value is computed a similar approach is used to update u as the horizontal position in the target image increments.

$$u[x,y] = u[(x-1),y] + accd[x]$$

$$accd[x] = accd[x-1] + 2d$$

$$accd[0] = 0$$

$$accd[1] = b + d + accf[y]$$

Column Equations (A.4)

Equivalent equations are used for the computation of v[x,y].

A.0.2 Implementation in Hardware

The mapping of these equations into a general hardware implementation is straightforward. Six registers are required to hold the constant coefficient values. Five accumulators are used to hold the current value u[0,y], acce[y], u[x,y], accf[y], and accd[x]. Three 2-1 multiplexors and one 3-1 multiplexor are used to pipe the appropriate initialization values to the accumulator.

Higher order polynomials can also easily be implemented using this method. As the order of the polynomial increases, more levels of accumulation are required, but the basic approach is the same.

References

[1] Andrews, H.C., C.L. Patternson, III, "Digital Interpolation of Discrete Images," *IEEE Transactions on Computers.*, vol c-25, no. 2, Feb. 1976.

[2] Burt, P.J., "The Pyramid as a Structure for Efficient Computation," Electrical Computer & Systems Engineering Department, Rensselaer Polytechnique Institute, Troy, NY 12181.

[3] Burt, P.J., P. Anandan, "Image Stabilization be Registration to a Reference Mosaic," *1994 Image Understanding Workshop*, Nov. 13-16, Monterey, CA, 1994.

[4] Burt, P.J. and R.J. Kolczynski, "Enhanced Image Capture Through Fusion," *Fourth International Conference on Computer Vision*, Berlin, Germany, 1993.

[5] Hansen, M., P. Anandan, K. Dana, G. van der Wal, P. Burt, "Real-time Scene Stabilization and Mosaic Construction," *David Sarnoff Research Center*, Princeton, NJ, 1994.

[6] Keys, R.G., "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Trans, Acoust., Speech, Signal Process.*, vol. ASSP-29, pp. 1153-1160, 1981.

[7] Norton, A., A.P. Rockwood, and P.T. Skolmoski, "Clamping: A Method of Antialiasing Textured Surfaces by Bandwidth Limiting in Object Space," *Computer Graphics*, (SIGGRAPH '82 Proceedings), vol. 16, no. 3, pp. 1-8, July 1982.

[8] Parker, A.J., R.V. Kenyon, and D.E. Troxel, "Comparison of Interpolation Methods for Image Resampling," *IEEE Trans. Medical Imaging*, vol. MI-2, no. 1, pp. 31-39, March 1983.

[9] Schafer, R.W. and L.R. Rabiner, "A Digital Signal Processing Approach to Interpolation," *Proc. IEEE*, vol. 61, pp. 692-702, June 1973.

[10] Oppenheim, A.V., R.W. Schafer, *Discrete-Time Signal Processing*, Prentice-Hall, NJ, 1989.

[11] Unser, M., P. Thevanaz, L. Yaroslavsky, "Convolution-Based Interpolation for Fast, High-Quality Rotation of Images," *IEEE Trans. on Image Processing*, vol. 4, No. 10, October 1995.

[12] van der Wal, G.S., P.J. Burt, "A VLSI Pyramid Chip for Multiresolution Image Analysis," *International Journal of Computer Vision*, 8:3, 177-189, 1992.

[13] Wolberg, G., *Digital Image Warping*, IEEE Computer Society Press, CA, 1990.