GEOMETRIC TRANSFORMATION TECHNIQUES FOR DIGITAL IMAGES: A SURVEY

George Wolberg

Department of Computer Science
Columbia University
New York, NY 10027
wolberg@cs.columbia.edu

December 1988
Technical Report CUCS-390-88

ABSTRACT

This survey presents a wide collection of algorithms for the geometric transformation of digital images. Efficient image transformation algorithms are critically important to the remote sensing, medical imaging, computer vision, and computer graphics communities. We review the growth of this field and compare all the described algorithms. Since this subject is interdisciplinary, emphasis is placed on the unification of the terminology, motivation, and contributions of each technique to yield a single coherent framework.

This paper attempts to serve a dual role as a survey and a tutorial. It is comprehensive in scope and detailed in style. The primary focus centers on the three components that comprise all geometric transformations: spatial transformations, resampling, and antialiasing. In addition, considerable attention is directed to the dramatic progress made in the development of separable algorithms. The text is supplemented with numerous examples and an extensive bibliography.

This work was supported in part by NSF grant CDR-84-21402.

TABLE OF CONTENTS

1. INTRODUCTION

A geometric transformation is an image processing operation that redefines the spatial relationship between points in an image. This facilitates the manipulation of an image's spatial layout, i.e., its size and shape. This area has received considerable attention due to its practical importance in remote sensing, medical imaging, computer vision, and computer graphics. Typical applications include distortion compensation of imaging sensors, decalibration for image registration, geometrical normalization for image analysis and display, map projection, and texture mapping for image synthesis.

Historically, geometric transformations were first performed on continuous (analog) images using optical systems. Early work in this area is described in [Cutrona 60], a landmark paper on the use of optics to perform transformations. Since then, numerous advances have been made in this field [Horner 87]. Although optical systems offer the distinct advantage of operating at the speed of light, they are limited in control and flexibilty. Digital computer systems, on the other hand, resolve these problems and potentially offer more accuracy. Consequently, the algorithms presented in this survey deal exclusively with digital (discrete) images, the primary target of geometric transformations.

1.1. MOTIVATION

The earliest work in geometric transformations for digital images stems from the remote sensing field. This area gained attention in the early 1960s, when the U.S. National Aeronautics and Space Administration (NASA) embarked upon aggressive earth observation programs. Its objective was the acquisition of data for environmental research applicable to earth resource inventory and management. As a result of this initiative, programs such as Landsat and Skylab emerged. In addition, other government agencies were supporting work requiring aerial photos for terrain mapping and surveillance.

These projects all involved acquiring multi-image sets, i.e., multiple images of the same area taken either at different times or with different sensors. Immediately, the task arises to align each image with every other image in the set so that all corresponding points match. Misalignment can occur due to any of the following reasons. First, images may be taken at the same time but acquired from several sensors, each having different distortion properties. Second, images may be taken from one sensor at different times and at various viewing geometries. Furthermore, sensor motion will give rise to distortion as well.

Geometric transformations were originally introduced to invert (correct) these distortions and allow the accurate determination of spatial relationships and scale. This requires us to first estimate the distortion model, usually by means of reference points which may be accurately marked or readily identified (e.g., road intersections, land-water interface). In the vast majority of applications, the coordinate transformation representing the distortion is modeled as a bivariate polynomial whose coefficients are obtained by minimizing an error function over the reference points. Usually, a second-order polynomial suffices, accounting for translation, scale,

rotation, skew, and pincushion effects. For more local control, affine transformations and piecewise polynomial mapping functions are widely used, with transformation parameters varying from one region to another. A historical review of early remote sensing techniques can be found in [Haralick 76].

The methods derived from remote sensing have direct application in other related fields, including medical imaging and computer vision. In medical imaging, for instance, geometric transformations play an important role in image registration and rotation for digital radiology. In this field, images obtained after injection of contrast dye are enhanced by subtracting a mask image taken before the injection. This technique, known as digital subtraction angiography, is subject to distortions due to patient motion. Since motion causes misalignment of the image and its subtraction mask, the resulting produced images are degraded. The quality of these images is improved with superior transformation algorithms that increase the accuracy of the registration.

Computer graphics offers another repertoire of methods and goals for geometric transformations. In this field, the goal is not geometric correction, but rather inducing geometric distortion. This inverse formulation is used to map 2-D images onto 3-D surfaces. This technique, known as texture mapping, has been used with much success in achieving visually rich and complicated imagery. Furthermore, additional sophisticated filtering techniques have been promoted to combat artifacts arising from the severe spatial distortions possible in this application. The thrust of this effort has been directed to the study and design of efficient space-variant low-pass filters. Since the remote sensing and medical imaging fields have generally attempted to correct only mild distortions, they have neglected this important area. The design of efficient algorithms for filtering fairly general areas remains a great challenge.

The continuing development of efficient algorithms for the geometric transformation of digital images has gained impetus from the growing availability of fast and cost-effective digital hardware. The ability to process high resolution imagery has become more feasible with the advent of fast computational elements, high-capacity digital data storage devices, and improved display technology. Consequently, the trend in algorithm design has been towards a more effective match with the implementation technology.

1.2. OVERVIEW

In this section, we briefly review the various stages in a geometric transformation. Each stage of the geometric transformation process has received much attention from a wide community of people in many diverse fields. As a result, the literature is replete with varied terminologies, motivations, and assumptions. A review of geometric transformation techniques, particularly in the context of their numerous applications, is useful for highlighting the common thread that underlies their many forms.

The purpose of this paper is to describe the algorithms developed in this field within a consistent and coherent framework. It centers on the three components that comprise all geometric transformations: spatial transformations, resampling, and antialiasing. Due to the central importance of sampling theory, a review is provided as a preface to the resampling and antialiasing

sections. In addition, a discussion of efficient separable implementations is given as well. We now briefly outline the contents of these sections.

1.2.1. Spatial Transformations

The basis of geometric transformations is the mapping of one coordinate system onto another. This is defined by means of a *spatial transformation* — a mapping function that establishes a spatial correspondence between all points in the input and output images. Given a spatial transformation, each point in the output assumes the value of its corresponding point in the input image. The correspondence is found by using the spatial transformation mapping function to project the output point onto the input image.

Depending on the application, spatial transformation mapping functions may take on many different forms. Simple transformations may be specified by analytic expressions including affine, projective, and polynomial transformations. More sophisticated mapping functions that are not conveniently expressed in analytic terms can be determined from a sparse lattice of control points for which spatial correspondence is known. This yields a spatial representation in which undefined points are evaluated through interpolation. Indeed, taking this approach to the limit yields a dense grid of control points resembling a 2-D spatial lookup table that may define any arbitrary mapping function.

In computer graphics, the spatial transformation is completely specified by the parameterization of the 3-D object and its position from the 2-D projecting plane (i.e., the viewing screen). The objects are usually defined as planar polygons or bicubic patches. Consequently, three coordinate systems are used: 2-D texture space, 3-D object space, and 2-D screen space. The various formulations for spatial transformations are discussed in section 3.

1.2.2. Sampling Theory

Sampling theory is central to the study of sampled-data systems, e.g., digital image transformations. It lays a firm mathematical foundation for the analysis of sampled signals, offering invaluable insight into the problems and solutions of sampling. It does so by providing an elegant mathematical formulation describing the relationship between a continuous signal and its samples. We will use it to resolve the problems of image reconstruction and aliasing that follow. Note that reconstruction is an interpolation procedure applied to the sampled data, and aliasing simply refers to the presence of unreproducibly high frequencies and the resulting artifacts.

Together with defining theoretical limits on the continuous reconstruction of discrete input, sampling theory yields the guidelines for numerically measuring the quality of various proposed filtering techniques. This proves most useful in formally describing reconstruction, aliasing, and the filtering necessary to combat the artifacts that may appear at the output. The fundamentals of sampling theory are reviewed in section 4.

1.2.3. Resampling

In the continuous domain, a geometric transformation is fully specified by the spatial transformation. This is due to the fact that an analytic mapping is bijective — one-to-one and onto. However, in our domain of interest, complications are introduced due to the discrete nature of digital images.

In digital images, the discrete picture elements, or *pixels*, are restricted to lie on a sampling grid, taken to be the integer lattice. The output pixels, now defined to lie on the output sampling grid, are passed through the mapping function generating a new grid used to resample the input. This new resampling grid, unlike the input sampling grid, does not generally coincide with the integer lattice. Rather, the positions of the grid points may take on any of the continuous values assigned by the mapping function.

Since the discrete input is defined only at integer positions, an interpolation stage is introduced to fit a continuous surface through the data samples. The continuous surface may then be sampled at arbitrary positions. This interpolation stage is known as *image reconstruction*†. Collectively, image reconstruction followed by sampling is known as *image resampling*.

Image resampling consists of passing the regularly spaced output grid through the spatial transformation, yielding a resampling grid that maps into the input image. Since the input is discrete, image reconstruction is performed to interpolate the continuous input signal from its samples. Sampling the reconstructed signal gives us the intensity values that are assigned to the output pixels.

The accuracy of interpolation has significant impact on the quality of the output image. Therefore, many interpolation functions have been studied from the viewpoints of both computational efficiency and approximation quality. Popular interpolation functions include cubic convolution, bilinear, and nearest neighbor. They can exactly reconstruct second-, first-, and zero-degree polynomials, respectively. More expensive and accurate methods include cubic spline interpolation and convolution with a sinc function. Using sampling theory, this last choice can be shown to be the ideal filter. However, it cannot be realized using a finite number of neighboring elements. Consequently, the alternate proposals have been given to offer reasonable approximations. Image resampling and reconstruction are described in section 5.

1.2.4. Aliasing

Through image reconstruction, we have solved the first problem that arises due to operating in the discrete domain — sampling a discrete input. Another problem now arises in evaluating the discrete output. The problem, related to the resampling stage, is described below.

The output image, as described above, has been generated by *point sampling* the reconstructed input. Point (or zero-spread) sampling refers to an ideal sampling process in which the value of each sampled point is taken independently of its neighbors. That is, each input point

† In the literature, the terms reconstruction and interpolation are used interchangeably.

influences one and only one output point.

With point sampling, entire intervals between samples are discarded and their information content is lost. If the input signal is smoothly varying, the lost data is recoverable through interpolation, i.e., reconstruction. This statement is true only when the input is a member of a class of signals for which the interpolation algorithm is designed. However, if the skipped intervals are sufficiently complex, interpolation may be inadequate and the lost data is unrecoverable. The input signal is then said to be *undersampled*, and any attempt at reconstruction gives rise to a condition known as *aliasing*. Aliasing distortions, due to the presence of unreproducibly high spatial frequencies, may surface in the form of jagged edges and moire patterns.

Aliasing artifacts are most evident when the spatial mapping induces large scale changes. As an example, consider the problem of image magnification and minification. When magnifying an image, each input pixel contributes to many output pixels. This one-to-many mapping requires the reconstructed signal to be densely sampled. Clearly, the resulting image quality is closely tied to the accuracy of the interpolation function used in reconstruction. For instance, high-degree interpolation functions can exactly reconstruct a larger class of signals than low-degree functions. Therefore, if the input is poorly reconstructed, artifacts such as jagged edges become noticeable at the output grid. Note that the computer graphics community often considers jagged edges to be synonymous with aliasing. As we shall see in section 4, this is sometimes a misconception. In this case, for instance, jagged edges are due to inadequate reconstruction, *not* aliasing.

Under magnification, the output contains at least as much information as the input, with the output assigned the values of the densely sampled reconstructed signal. When minifying an image, the opposite is true. The reconstructed signal is sparsely sampled in order to realize the scale reduction. This represents a clear loss of data, where many input samples are actually skipped over in the point sampling. It is here where aliasing is apparent in the form of moire patterns and fictitious low-frequency components. It is related to the problem of mapping many input samples onto a single output pixel. This requires appropriate filtering to properly integrate all the information mapping to that pixel.

The filtering used to counter aliasing is known as *antialiasing*. Its derivation is grounded in the well-established principles of sampling theory. Antialiasing typically requires the input to be blurred *before* resampling. This serves to have the sampled points influenced by their discarded neighbors. In this manner, the extent of the artifacts is diminished, but not eliminated.

Completely undistorted sampled output can only be achieved by sampling at a sufficiently high frequency, as dictated by sampling theory. Although adapting the sampling rate is more desirable, physical limitations on the resolution of the output device often prohibit this alternative. Thus, the most common solution to aliasing is smoothing the input prior to sampling.

The well understood principles of sampling theory offer theoretical insight into the problem of aliasing and its solution. However, due to practical limitations in implementing the ideal filters suggested by the theory, a large number of algorithms have been proposed to yield approximate solutions. Section 6 details the antialiasing algorithms.

1.2.5. Separable Geometric Transformation Algorithms

A large body of work in the geometric transformation of digital images has been directed towards optimizing special cases to obtain major performance gains. In particular, the use of separable techniques has reduced complexity and processing time. Separable geometric algorithms reduce 2-D problems into a sequence of 1-D (scanline) resampling problems. This makes them amenable to streamline processing and allows them to be implemented with conventional hardware. Separable techniques have been shown to be useful for affine and perspective transformations, as well as mapping onto bilinear, biquadratic, bicubic, and superquadric patches. Contributions in this area are discussed in section 7.

2. DIGITAL IMAGE ACQUISITION

Consider the imaging system shown in Fig. 2.1. The entire imaging process can be viewed as a cascade of filters applied to the input image. The scene $f(x,y)$ is a continuous two-dimensional image. It passes through an imaging subsystem which acts as the first stage of data acquisition. Due to the *point spread* function (PSF) of the image sensor, the output $g(x,y)$ is a degraded version of $f(x,y)$.

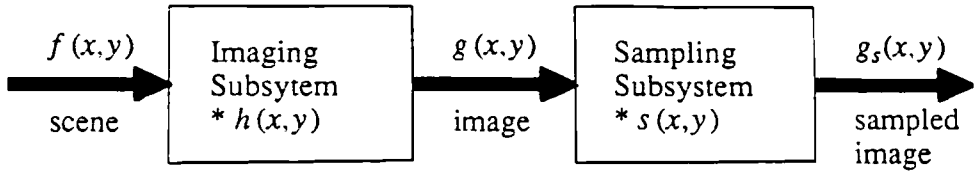


Figure 2.1: Imaging system.

As its name suggests, the PSF is taken to be a bandlimiting filter, $h(x,y)$, having blurring characteristics. It reflects the physical limitations of an optical lens to accurately resolve each input point without the influence of neighboring points. Consequently, the PSF is typically modeled as a low-pass filter given by a bell-shaped weighting function over a finite aperture area. A PSF profile is depicted in Fig. 2.2. By definition,

$$g(x,y) = f(x,y) * h(x,y) \tag{2.1}$$

where * denotes convolution. The problems addressed in this paper also assume that the imaging device induces geometric distortion in addition to blurring. Examples are given in section 3.

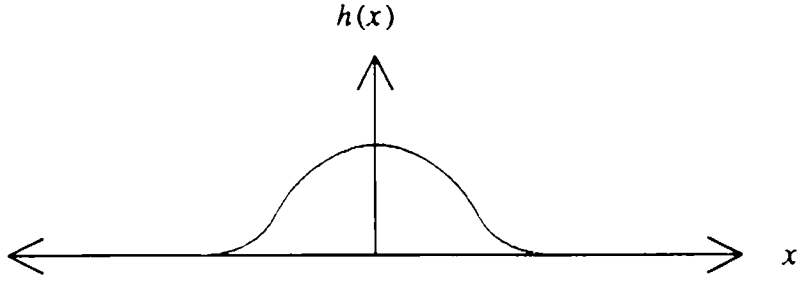


Figure 2.2: PSF profile.

The continuous image $g(x,y)$ then enters a sampling subsystem which digitizes the analog input and completes the data acquisition stage. The sampled image $g_s(x,y)$ is given by

$$g_s(x,y) = g(x,y) * s(x,y) \tag{2.2}$$

where

$$s(x,y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(x-m, y-n) \tag{2.3}$$

is the two-dimensional *comb function*, depicted in Fig. 2.3, and

$$\delta(x, y) = \begin{cases} 1 & \text{if } (x,y) = (0,0) \\ 0 & \text{otherwise} \end{cases} \tag{2.4}$$

is an *impulse function*, known also as the *Kronecker* or *Dirac delta function.*

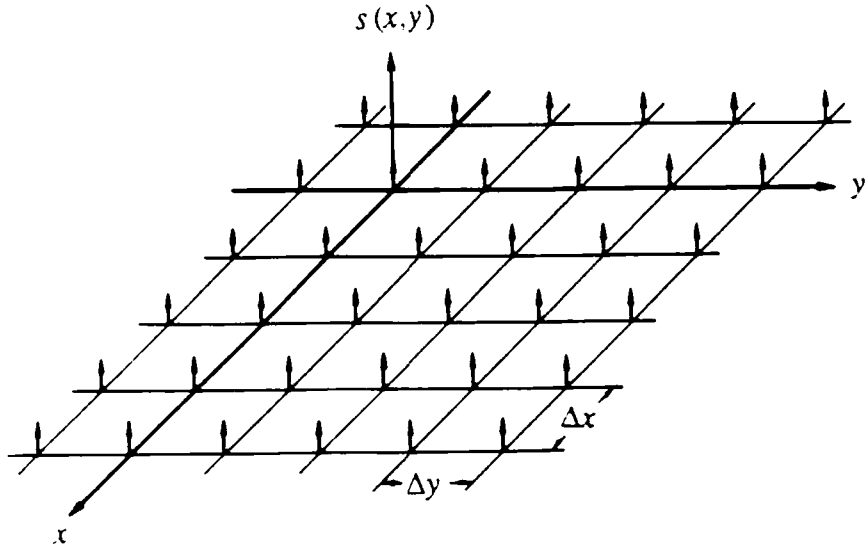


Figure 2.3: Comb function.

The comb function comprises our *sampling grid* which is conveniently nonzero only at integral (x,y) coordinates. Therefore, $g_s(x,y)$ is now a digital (discrete) image with intensity values defined only over integral indices of x and y. Each sample represents a picture element, or *pixel*. Collectively, they comprise the 2-D array of pixels that serve as input to the subsequent processing.

The process of mapping real numbers onto a range of integers is called *quantization*. Digital images are the product of both spatial and intensity quantization. Spatial quantization is achieved through the use of a sampling grid. Intensity quantization is the result of representing pixel values with a finite number of bits. A tradeoff exists between sampling rate and quantization levels. An interesting review of early work in this area is found in [Knowlton 72]. Related work in image coding is described in [Netravali 80]. Finally, a recent analysis on the tradeoff between sampling and quantization can be found in [Lee 87].

3. SPATIAL TRANSFORMATION

This section describes the various mapping formulations derived for geometric transformations. We begin with a brief review of affine and perspective transformations. This provides the basis for the more sophisticated mappings described in the remainder of this section. They include the most common spatial transformations used in remote sensing, medical imaging, computer vision, and computer graphics.

3.1. DEFINITIONS

A *spatial transformation* defines a geometric relationship between each point in the input and output images. An input image consists entirely of reference points whose coordinate values are known precisely. The output image is comprised of the observed (warped) data. The general mapping function can be given in two forms: either relating the output coordinate system to that of the input, or vice versa. Respectively, they can be expressed as

$$[x, y] = [X(u,v), Y(u,v)] \tag{3.1}$$

or

$$[u, v] = [U(x,y), V(x,y)] \tag{3.2}$$

where $[u,v]$ refers to the input image coordinates corresponding to output pixel $[x,y]$, and X, Y, U and V are arbitrary mapping functions that uniquely specify the spatial transformation. Since X and Y map the input onto the output, they are referred to as the forward mapping. Similarly, the U and V functions are known as the inverse mapping since they map the output onto the input.

3.1.1. Forward Mapping

The *forward mapping* consists of copying each input pixel onto the output image at positions determined by the X and Y mapping functions. Figure 3.1 illustrates the forward mapping for the 1-D case. The discrete input and output are each depicted as a string of pixels lying on an integer grid (dots). Each input pixel is passed through the spatial transformation where it is assigned new output coordinate values. Notice that the input pixels are mapped from the set of integers to the set of real numbers. In the figure, this corresponds to the regularly spaced input samples and the irregular output distribution.

The real-valued output positions assigned by X and Y present complications at the discrete output. In the continuous domain, where pixels may be viewed as points, the mapping is straightforward. However, in the discrete domain pixels are now taken to be finite elements defined to lie on a (discrete) integer lattice. It is therefore inappropriate to implement the spatial transformation as a point-to-point mapping. Doing so can give rise to two types of problems: holes and overlaps. Holes, or patches of undefined pixels, occur when mapping contiguous input samples to sparse positions on the output grid. In Fig. 3.1, F' is a hole since it is bypassed in the

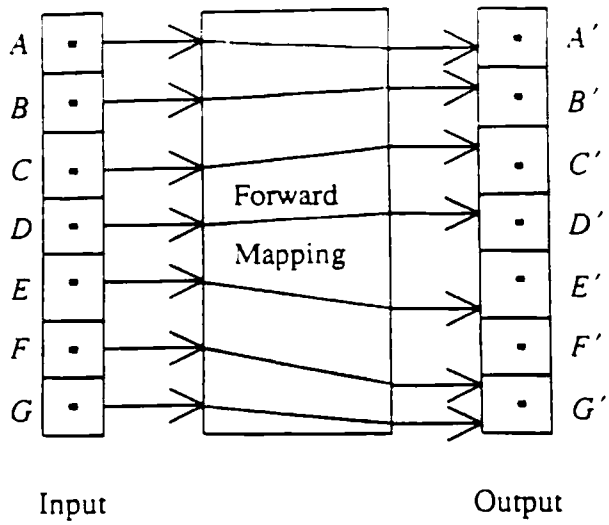


Figure 3.1: Forward mapping.

input-output mapping. In contrast, overlaps occur when consecutive input samples collapse into one output pixel, as depicted in Fig. 3.1 by output pixel G'.

The shortcomings of a point-to-point mapping are avoided using a four-corner mapping paradigm. This considers input pixels as square patches that may be transformed into arbitrary quadrilaterals in the output image. This has the effect of allowing the input to remain contiguous after the mapping.

Due to the fact that the projected input is free to lie anywhere in the output image, input pixels often straddle several output pixels, or lie embedded in one. These two instances are illustrated in Fig. 3.2. An *accumulator array* is required to properly integrate the input contributions at each output pixel. It does so by determining which fragments contribute to each output pixel and then integrating over all contributing fragments. The partial contributions are handled by scaling the input intensity in proportion to the fractional part of the pixel that it covers. Intersection tests must be performed to compute the coverage. Thus, each position in the accumulator array evaluates $\sum_{i=0}^{N} w_i f_i$, where f_i is the input value, w_i is the weight reflecting its coverage of the output pixel, and N is the total number of deposits into the cell. Note that N is free to vary among pixels and is determined only by the mapping function and the output discretization.

Formulating the transformation as a four-corner mapping problem allows us to avoid holes in the output image. Nevertheless, this paradigm introduces two problems in the forward mapping process. First, costly intersection tests are needed to derive the weights. Second, magnification will possibly cause the same input value to be applied onto many output pixels unless additional filtering is employed.

Both problems can be resolved by adaptively sampling the input based on the size of the projected quadrilateral. In other words, if the input pixel is mapped onto a large area in the output image, then it is best to repeatedly subdivide the input pixel until the projected area reaches some acceptably low limit, i.e., one pixel size. As the sampling rate rises, the weights converge

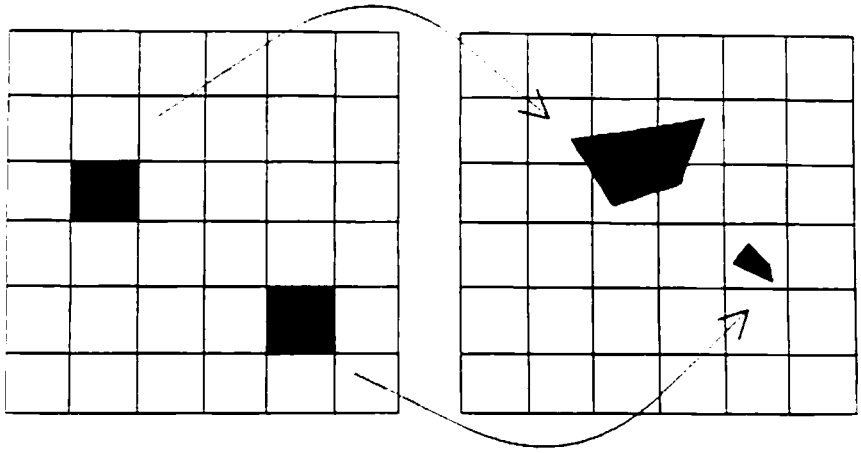


Figure 3.2: Accumulator array.

to a single value, the input is resampled more densely, and the resulting computation is performed at higher precision.

It is important to note that uniformly sampling the input image does not guarantee uniform sampling in the output image unless X and Y are affine (linear) mappings. Thus, for nonaffine mappings, e.g., perspective, the input image must be adaptively sampled at rates that are spatially varying. For example, the oblique surface shown in Fig. 3.3 must be sampled more densely near the horizon to account for the foreshortening due to the perspective mapping. In general, forward mapping is useful when the texture image must be read sequentially or will not reside entirely in memory.

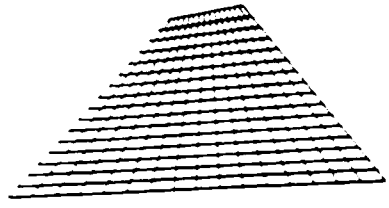


Figure 3.3: An oblique surface requiring adaptive sampling.

3.1.2. Inverse Mapping

The *inverse mapping* operates in screen order, projecting each output coordinate into the input image via U and V. The value of the data sample at that point is copied onto the output pixel. Again, filtering is necessary to combat the aliasing artifacts described in more detail later. This is the most common method since no accumulator array is necessary and since output pixels which lie outside a clipping window need not be evaluated. This method is useful when the screen is to be written sequentially, U and V are readily available, and the input image can be

stored entirely in memory. As a result of these advantages, the spatial transformations described in the remainder of the paper will be in the inverse mapping form.

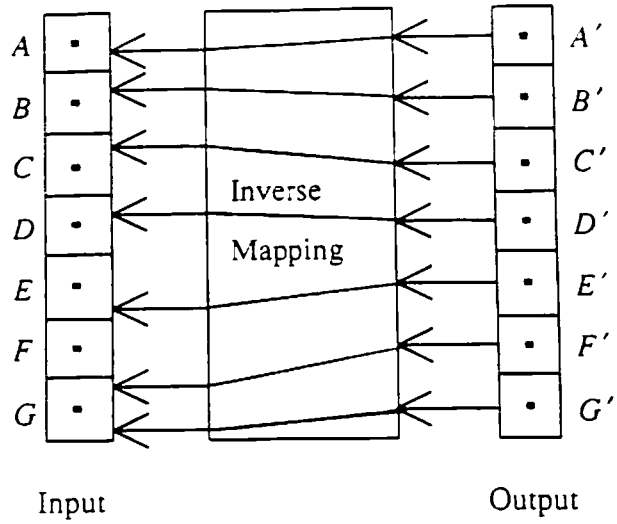


Figure 3.4: Inverse mapping.

Figure 3.4 depicts the inverse mapping, with each output pixel mapped back onto the input via the spatial transformation (inverse) mapping function. Notice that the output pixels are centered on integer coordinate values. They are projected onto the input at real-valued positions. As we will see later, an interpolation stage must be introduced in order to retrieve input values at undefined (nonintegral) input positions.

Unlike the point-to-point forward mapping scheme, the inverse mapping guarantees that all output pixels are computed. However, the analogous problem remains to determine whether large holes are left when sampling the input. If this is the case, large amounts of input data may have been discarded while evaluating the output, thereby giving rise to artifacts described in section 6. Thus, filtering is necessary to integrate the area projected onto the input. In general, though, this arrangement has the advantage of allowing interpolation to occur in the input space instead of the output space. This proves to be a much more convenient approach than forward mapping.

In their most unconstrained form, U and V can serve to scramble the image by defining a discontinuous function. The image remains coherent only if U and V are piecewise continuous. Several common forms of U and V have been isolated for geometric correction and geometric distortion.

We begin with a discussion of the general transformation matrix, the elementary form to specify simple mappings including affine and perspective transformations. This is followed by the methods advanced in remote sensing. The work in this area is motivated by the need to perform geometric correction. Many of the methods presented here apply equally to medical imaging and computer vision, two fields which share this related problem. Finally, additional mapping formulations used in computer graphics are presented.

3.2. GENERAL TRANSFORMATION MATRIX

Many simple spatial transformations can be expressed in terms of the general 3×3 transformation matrix shown below. It handles local scaling, overall scaling, shearing, rotation, reflection, translation, and perspective in 2-space. Without loss of generality, we shall ignore the component in the third dimension since we are only interested in 2-D image projections. When multiplied with $[x, y, w]$, a 2-space vector represented in homogeneous coordinates, it yields the 2-space vector $[u, v, w']$.

$$[u, v, w'] = [x, y, w] \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{3.3}$$

The 3×3 transformation matrix can be best understood by partitioning it into four separate sections. The 2×2 submatrix

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$$

yields a linear transformation for local scaling, shearing, and rotation. The 1×2 matrix $[\, a_{31} \;\; a_{32} \,]$ produces translation. The 2×1 matrix $[\, a_{13} \;\; a_{23} \,]^T$ produces perspective transformation. The final element a_{33} is responsible for overall scaling.

3.2.1. Translation

All points are translated to new positions by adding offsets T_x and T_y to x and y, respectively. The translate transform is

$$[u, v, 1] = [x, y, 1] \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix} \tag{3.4}$$

3.2.2. Rotation

All points in the xy–plane are rotated about the origin through the clockwise angle θ.

$$[u, v, 1] = [x, y, 1] \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.5}$$

3.2.3. Scale

All points are scaled by applying the scale factors S_x and S_y to the x and y coordinates, respectively. Negative scale factors cause the image to be reflected, yielding a mirrored image. If the scale factors are not identical, then the image proportions are altered resulting in a differentially scaled image.

$$[u, v, 1] = [x, y, 1] \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.6}$$

3.2.4. Shear

The coordinate scaling described above involves only the diagonal terms a_{11} and a_{22}. We now consider the case where $a_{11} = a_{22} = 1$, and $a_{12} = 0$. By allowing a_{21} to be nonzero, u is made linearly dependent on both x and y, while v remains identical to y. A similar operation can be applied along the y-axis to compute new values for v while u remains unaffected. This effect, called *shear*, is therefore produced using the off-diagonal terms. The shear transform along the x-axis is

$$[u, v, 1] = [x, y, 1] \begin{bmatrix} 1 & 0 & 0 \\ H_x & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7a}$$

where H_x is used to make u linearly dependent on y as well as x. Similarly, the shear transform along the y-axis is

$$[u, v, 1] = [x, y, 1] \begin{bmatrix} 1 & H_y & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.7b}$$

3.2.5. Composite Transformations

Multiple transformations can be collapsed into a single composite transform. The transformations are combined by taking the product of the 3×3 matrices. For example, the composite transform representing a translation followed by a rotation and a scale change is given below.

$$[u, v, 1] = [x, y, 1]\, \mathbf{M}_{comp} \tag{3.8}$$

where

$$\mathbf{M}_{comp} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} S_x\cos\theta & S_y\sin\theta & 0 \\ -S_x\sin\theta & S_y\cos\theta & 0 \\ S_x(T_x\cos\theta - T_y\sin\theta) & S_y(T_x\sin\theta + T_y\cos\theta) & 1 \end{bmatrix}$$

3.2.6. Affine Transformations

All of the above examples are known as two-dimensional *affine transformations*. They are characterized by their last columns being equal to $[\,0\ 0\ 1\,]^T$. Since the product of affine transformations is also affine, they can be used to perform a general orientation of a set of points relative to an arbitrary coordinate system while still maintaining a unity value for the homogeneous coordinate. This is necessary for generating composite transforms. Furthermore, projections of 3-D affine transformations have the property of retaining parallelism among parallel lines. This allows us to avoid foreshortened axes when performing 2-D projections. The general representation of the affine transform is

$$[u, v, 1] = [x, y, 1] \begin{bmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{bmatrix} \tag{3.9}$$

3.2.7. Perspective Transformations

A *perspective transformation* is produced when $[\,a_{13}\ a_{23}\,]^T$ is nonzero. A perspective transformation is frequently used in conjunction with a projection onto a viewing plane. This combination is known as a *perspective projection*. The perspective projection of any set of parallel lines which are not parallel to the projection plane will converge to a vanishing point, the center of projection. This is useful for rendering realistic images. It has the property of foreshortening distant lines.

3.2.8. Homogeneous Coordinates

Projection into 2-space for viewing requires dividing the computed $[u',v']$ values by the homogeneous coordinate w'. This yields $[u, v] = [u'/w', v'/w']$, where $[u, v]$ is the projected vector which we sought from the transformation. Readers are referred to [Rogers 76] and [Foley 82] for a thorough treatment of 2-D and 3-D transformation matrix operations.

For affine transformations,

$$\begin{aligned} u &= a_{11}x + a_{21}y + a_{31} \\ v &= a_{12}x + a_{22}y + a_{32} \end{aligned} \tag{3.10}$$

For projective transformations,

$$\begin{aligned} u &= \frac{a_{11}x + a_{21}y + a_{31}}{a_{13}x + a_{23}y + a_{33}} \\ v &= \frac{a_{12}x + a_{22}y + a_{32}}{a_{13}x + a_{23}y + a_{33}} \end{aligned} \tag{3.11}$$

3.3. POLYNOMIAL TRANSFORMATIONS

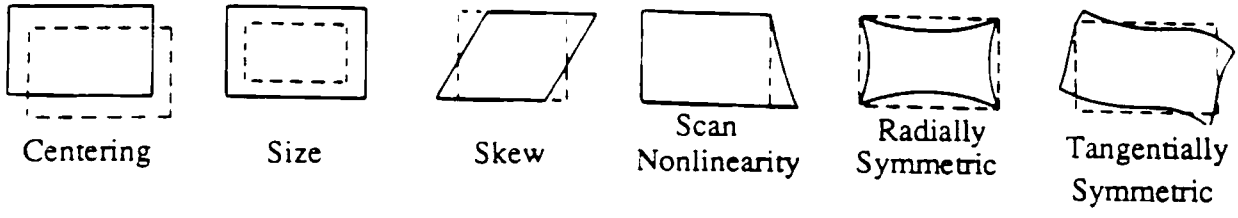
Geometric correction requires a spatial transformation to invert an unknown distortion function. The mapping functions, U and V, have been almost universally chosen to be bivariate *polynomial transformations* of the form

$$u = \sum_{i=0}^{N} \sum_{j=0}^{N-i} a_{ij} x^i y^j \qquad (3.12)$$

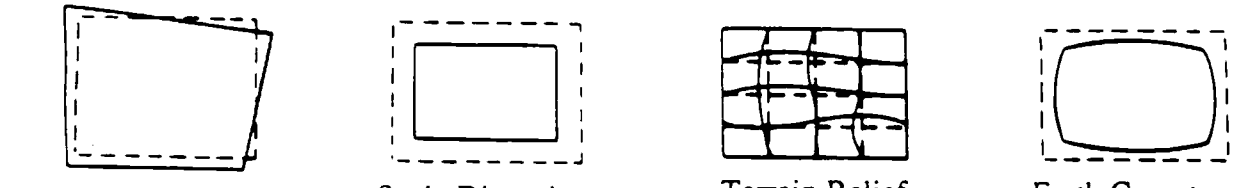
$$v = \sum_{i=0}^{N} \sum_{j=0}^{N-i} b_{ij} x^i y^j$$

where a_{ij} and b_{ij} are the constant polynomial coefficients. Since this formulation for geometric correction originated in remote sensing [Markarian 71], the discussion below will center on its use in that field. All the examples, though, have direct analogs in other related areas such as medical imaging [Singh 79] and computer vision [Rosenfeld 82].

The polynomial transformations given above are low-order global mapping functions operating on the entire image. They are intended to account for sensor-related spatial distortions such as centering, scale, skew, and pincushion effects, as well as errors due to earth curvature, viewing geometry, and camera attitude and altitude deviations. Due to dynamic operating conditions, these errors are comprised of internal and external components. The internal errors are sensor-related distortions. External errors are due to platform perturbations and scene characteristics. The effects of these errors have been categorized in [Bernstein 71] and are shown in Fig. 3.5.



TYPICAL SENSOR INTERNAL DISTORTIONS



TYPICAL EXTERNAL IMAGE DISTORTIONS

Figure 3.5: Common geometric image distortions.

These errors are characterized as low-frequency (smoothly varying) distortions. The global effects of the polynomial mapping will not account for high-frequency deformations that are local in nature. Since most sensor-related errors tend to be low-frequency, modeling the spatial transformation with low-order polynomials appears justified. Common values of N that have been used in the polynomials of Eq. (3.12) include $N=1$ [Steiner 77], $N=2$ [Nack 77], $N=3$ [Van Wie 77], and $N=4$ [Leckie 80]. In practice, a second-degree ($N=2$) approximation has been shown to be adequate [Lillestrand 72].

Rather than apply the mapping functions over the entire set of points, an *interpolation grid* is often introduced to reduce the computational complexity. This method evaluates the mapping function at a relatively sparse set of grid, or mesh, points. The spatial correspondence of points internal to the mesh is computed by bilinear interpolation from the corner points [Bernstein 76].

3.3.1. Polynomial Coefficients

Auxiliary information is needed to determine the polynomial coefficients. This information includes reseau marks, platform attitude and altitude data, and ground control points. Reseau marks are small cruciform markings inscribed on the faceplate of the sensor. Since the locations of the reseau marks can be accurately calibrated, the measured differences between their true locations and imaged (distorted) locations yields a sparse sensor distortion mapping. This accounts for the internal errors.

External errors can be directly characterized from platform attitude, altitude, and ephemerides data. However, this data is not generally precisely known. Consequently, ground control points are used to determine the external error. Ground control points (GCP) are identifiable natural landmarks detectable in a scene, whose locations and elevations are known precisely. Typical GCPs include airports, highway intersections, land-water interfaces, and geological patterns [Bernstein 71, 76].

A number of these points are located and differences between their observed and actual locations are used to characterize the external error component. Together with the internal distortion function, this serves to fully define the spatial transformation which inverts the distortions present in the input image, yielding a corrected output image. Since there are more ground control points than undetermined polynomial coefficients, a least-squared-error fit is used. For example, a second-degree approximation requires only six coefficients to be solved.

$$\begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \cdots \\ x_R \end{bmatrix} = \begin{bmatrix} 1 & u_1 & v_1 & u_1v_1 & u_1^2 & v_1^2 \\ 1 & u_2 & v_2 & u_2v_2 & u_2^2 & v_2^2 \\ 1 & u_3 & v_3 & u_3v_3 & u_3^2 & v_3^2 \\ & & & \cdots & & \\ 1 & u_R & v_R & u_Rv_R & u_R^2 & v_R^2 \end{bmatrix} \begin{bmatrix} a_{00} \\ a_{10} \\ a_{01} \\ a_{11} \\ a_{20} \\ a_{02} \end{bmatrix} \tag{3.13}$$

where $R \geq 6$. A similar equation holds for y and b_{ij}. Both of these expressions may be written in matrix notation as

$$X = WA \tag{3.14}$$

$$Y = WB$$

Using the least squares estimate theory, the best estimate for A and B is given by the pseudoinverse solution [Wong 77].

$$A = (W^T W)^{-1} W^T X \tag{3.15}$$

$$B = (W^T W)^{-1} W^T Y$$

Equation (3.15) expresses a numerical solution for estimating the polynomial coefficients. A recent paper on practical geometric correction methods can be found in [Butler 87]. Under certain simplifying conditions, it is possible to derive compact analytic solutions for the coefficients. This has the advantage of offering a stable closed form solution. An example is given in [Bizais 83] for the case in which the reference pattern exhibits point symmetry about the origin.

3.3.2. A Surface Fitting Paradigm for Geometric Correction

The problem of determining functions U and V can be conveniently posed as a surface fitting problem. Consider, for example, knowing N control points labeled $[x_i,y_i]$ in the observed image and $[u_i,v_i]$ in the reference image, where $0 \le i < N$. Deriving mapping functions U and V is equivalent to determining two smooth surfaces: one that passes through points $[x_i,y_i,u_i]$ and the other that passes through $[x_i,y_i,v_i]$ for $0 \le i < N$. Figure 3.6 shows a surface for $U(x,y)$ with control points given at the grid points.

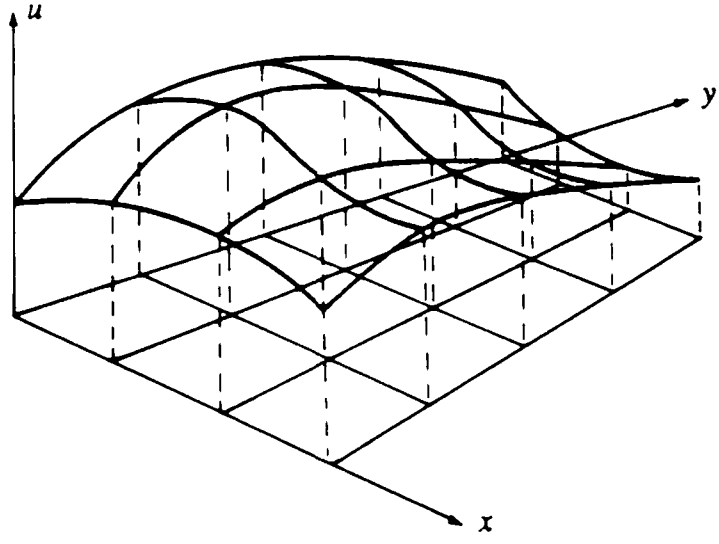


Figure 3.6: Surface $U(x,y)$.

Before an image undergoes geometric distortion, these surfaces are defined to be ramp functions. This follows from the observation that $u_i = x_i$ and $v_i = y_i$ in the absence of any deformation. Introducing geometric distortions will cause these surfaces to deviate from their initial ramp configurations. Note that as long as the surface is monotonically nondecreasing, the

resulting image does not fold back upon itself.

Given only sparse control points, it is necessary to interpolate a surface through these points and closely approximate the unknown distortion function. It is clear that global low-order polynomial mapping functions can only approximate these surfaces. Furthermore, the least-squares technique that is used to determine the coefficients average a local geometric difference over the whole image area independent of the position of the difference. As a result, local distortions cannot be handled and they instead contribute to errors at distant locations. We may, instead, interpolate the surface with a global mapping by increasing the degree of the polynomial to match the number of control points. However, the resulting polynomial is likely to exhibit excessive spatial undulations and thereby introduce further artifacts.

These problems are resolved by considering piecewise mapping functions. Rather than defining U and V via a global function, they are expressed as a union of a local functions. In this manner, the interpolated surface is composed of local surface patches, each influenced by nearby control points.

3.4. PIECEWISE POLYNOMIAL TRANSFORMATONS

The global polynomial transformations described earlier impose a single mapping function upon the whole image. They do not account for local geometric distortions such as scene elevation, atmospheric turbulence, and sensor nonlinearity. Consequently, piecewise mapping functions have been introduced to handle local deformations [Goshtasby 86, 87].

The study of piecewise interpolation has received much attention in the spline literature. The majority of the work, however, assumes that the data is available on a rectangular grid. In our application, this is generally not the case. Instead, we must consider the problem of fitting a composite surface to scattered 3-D data [Franke 79].

3.4.1. Procedure

The general procedure for performing surface interpolation on irregularly-spaced 3-D points consists of the following operations.

1) Partition each image into triangular regions by connecting neighboring control points with noncrossing line segments, forming a planar graph. This process, known as triangulation, serves to delimit local neighborhoods over which surface patches will be defined.
2) Estimate partial derivatives of U (and similarly V) with respect to x and y at each of the control points. This may be done using a local method, with data values taken from nearby control points, or a global method using all the control points. Computing the partial derivatives is necessary only if the surface patches are to join smoothly, i.e., for C^1, C^2, or smoother results[†].
3) For each triangular region, fit a smooth surface through the vertices satisfying the constraints imposed by the partial derivatives. The surface patches are generated using low-order bivariate polynomials. A linear system of equations must be solved to compute the polynomial coefficients.
4) Those regions lying outside the convex hull of the data points must extrapolate the surface from the patches lying along the boundary.
5) For each point (x,y), determine its enclosing triangle and compute an interpolated value u (similarly for v) using the polynomial coefficients derived for that triangle. This yields the (u,v) coordinates necessary to resample the input image.

3.4.2. Triangulation

Triangulation is the process of tesselating the convex hull of a set of N distinct points into triangular regions. This is done by connecting neighboring control points with noncrossing line segments, forming a planar graph. Although many configurations are possible, we are interested to achieve a partition such that points inside a triangle are closer to its three vertices than to

[†] C^1 and C^2 denote continuous first and second derivatives, respectively.

vertices of any other triangle. This is called the optimal triangulation and it avoids generating triangles with sharp angles and long edges. In this manner, only nearby data points will be used in the surface patch computations that follow. Several algorithms to obtain optimal triangulations are reviewed below.

In [Lawson 77], the author describes how to optimize an arbitrary triangulation initially created from the given data. He gives the following three criteria for optimality.

1) Max-min criterion: For each quadrilateral in the set of triangles, choose the triangulation that maximizes the minimum interior angle of the two obtained triangles. This tends to bias the tesselation against undesirable long thin triangles. Figure 3.7a shows triangle ABC selected in favor of triangle BCD under this criterion. The technique has computational complexity $O(N^{4/3})$.

2) The circle criterion: For each quadrilateral in the set of triangles, pass a circle through three of its vertices. If the fourth vertex lies outside the circle then split the quadrilateral into two triangles by drawing the diagonal which does not pass through the vertex. This is illustrated in Fig. 3.7b.

3) Thessian region criterion: For each quadrilateral in the set of triangles, construct the Thessian regions. In computational geometry, the Thessian regions are also known as Delaunay, Dirichlet, and Voronoi regions. They are the result of intersecting the perpendicular bisectors of the quadrilateral edges, as shown in Fig. 3.7c. This serves to create regions around each control point P such that points in that region are closer to P than to any other control point. Triangulation is obtained by joining adjacent Delaunay regions, a result known as Delaunay triangulation (Fig. 3.8). An $O(N^{3/2})$ triangulation algorithm using this method is described in [Green 78].

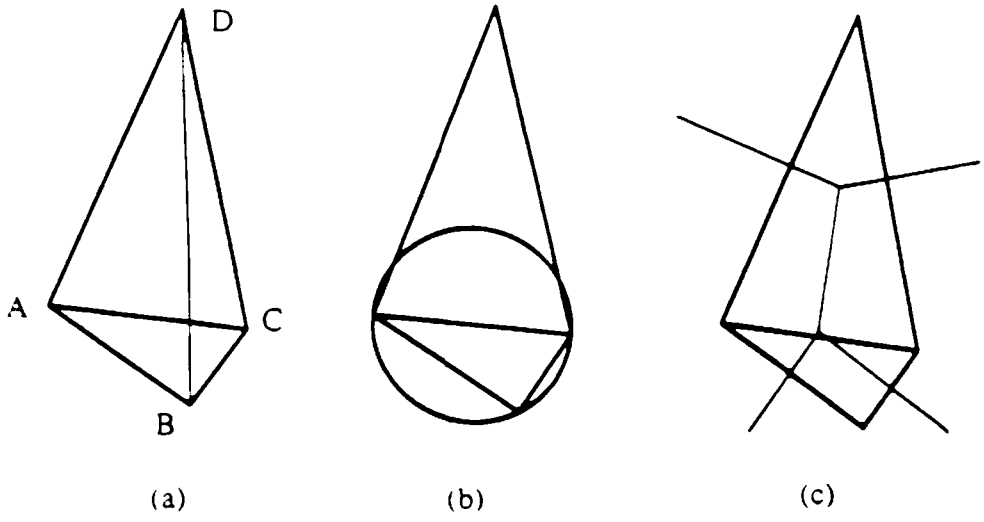
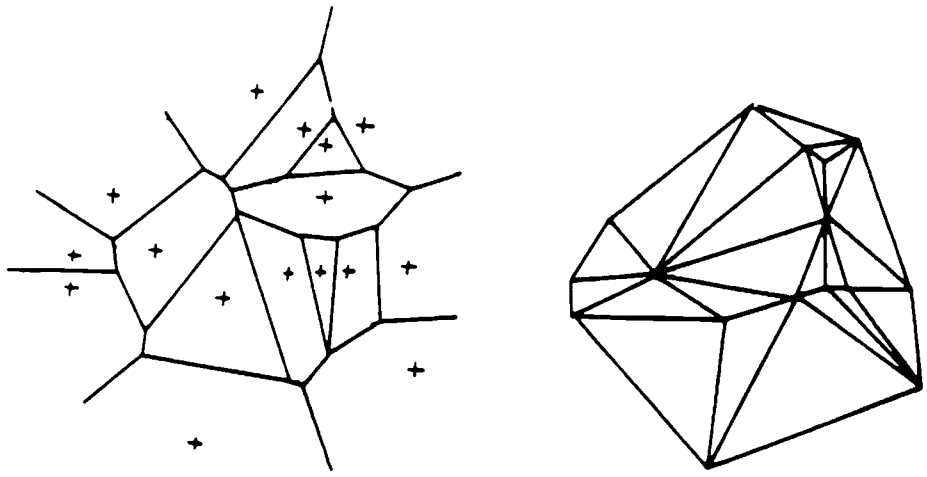


Figure 3.7: Three criteria for optimal triangulation.

An $O(N \log_2 N)$ recursive algorithm that determines the optimal triangulation is given in [Lee 80]. The method recursively splits the data into halves using the x-values of the control



(a) (b)

Figure 3.8: (a) Delaunay tesselation; (b) Triangulation.

points until each subset contains only three or four points. These small subsets are then easily triangulated using any of Lawson's three criteria. Finally, they are merged into larger subsets until all the triangular subsets are consumed, resulting in an optimal triangulation of the control points. Due to its speed and simplicity, this divide-and-conquer technique was used in [Goshtasby 87] to compute piecewise cubic mapping functions.

3.4.3. Linear Triangular Patches

Once the triangular regions are determined, the scattered 3-D data (x_i,y_i,u_i) or (x_i,y_i,v_i) are partitioned into groups of three points. Each group is fitted with a low-order bivariate polynomial to generate a surface patch. In this manner, triangulation allows only nearby control points to influence the surface patch calculations. Together, these patches comprise a composite surface defining the corresponding u or v coordinates at each point in the observed image.

We now consider the case of fitting the triangular patches with a linear interpolant, i.e., a plane. The equation of a plane through three points $[x_1,y_1,u_1]$, $[x_2,y_2,u_2]$, and $[x_3,y_3,u_3]$ is given by

$$Ax + By + Cu + D = 0 \tag{3.16}$$

where

$$A = \begin{vmatrix} y_1 & u_1 & 1 \\ y_2 & u_2 & 1 \\ y_3 & u_3 & 1 \end{vmatrix} ; \quad B = -\begin{vmatrix} x_1 & u_1 & 1 \\ x_2 & u_2 & 1 \\ x_3 & u_3 & 1 \end{vmatrix} ; \quad C = \begin{vmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix} ; \quad D = -\begin{vmatrix} x_1 & y_1 & u_1 \\ x_2 & y_2 & u_2 \\ x_3 & y_3 & u_3 \end{vmatrix}$$

As seen in Fig. 3.7b, the triangulation covers only the convex hull of the set of control points. In order to extrapolate points outside the convex hull, the planar triangles along the boundary are extended to the image border. Their extents are limited to the intersections of

neighboring planes.

3.4.4. Cubic Triangular Patches

Although piecewise linear mapping functions are continuous at the boundaries between neighboring functions, they do not provide a smooth transition across patches. In order to obtain smoother results, the patches must at least use C^1 interpolants. This is achieved by fitting the patches with higher-ordered bivariate polynomials.

This subject has received much attention in the field of computer-aided geometric design. Many algorithms have been proposed using N-degree polynomials. They include $N=2$ [Powell 77], $N=3, 4$ [Percell 76], and $N=5$ [Akima 78]. In this section, we examine the case of fitting triangular regions with cubic patches ($N=3$). A cubic patch f is a third-degree bivariate polynomial of the form

$$f(x,y) = a_1 + a_2x + a_3y + a_4x^2 + a_5xy + a_6y^2 + a_7x^3 + a_8x^2y + a_9xy^2 + a_{10}y^3 \quad (3.17)$$

The ten coefficients can be solved by determining ten constraints among them. Three relations are obtained from the coordinates of the three vertices. Six relations are derived from the partial derivatives of the patch with respect to x and y at the three vertices. Smoothly joining a patch with its neighbors requires the partial derivatives of the two patches to be the same in the direction normal to the common edge. This adds three more constraints, yielding a total of twelve relations. Since we have ten unknowns and twelve equations, the system is overdetermined and cannot be solved as given.

The solution lies in the use of the Clough-Tocher triangle, a widely known C^1 triangular interpolant [Clough 65]. Interpolation with the Clough-Tocher triangle requires the triangular region to be divided into three subtriangles. Fitting a surface patch to each subtriangle yields a total of thirty unknown parameters. Since exactly thirty constraints can be derived in this process, a linear system of thirty equations must be solved to compute a surface patch for each region in the triangulation. A full derivation of this method is given in [Goshtasby 87]. A complete review of triangular interpolants can be found in [Barnhill 77].

An interpolation algorithm offering smooth blending across patches requires partial derivative data. Since this is generally not available with the supplied data, it must be estimated. A straightforward approach to estimating the partial derivative at point P consists of fitting a second-degree bivariate polynomial through P and five of its neighbors. This allows us to determine the six parameters of the polynomial and directly compute the partial derivative. More accurate estimates can be obtained by a weighted least squares technique using more than six points [Lawson 77].

Another approach is given in [Akima 78] where the author uses P and its m nearest points $P_1, P_2, ..., P_m$, to form vector products $V_{ij} = (P - P_i) \times (P - P_j)$ with P_i and P_j being all possible combinations of the points. The vector sum V of all V_{ij}'s is then calculated. Finally, the partial derivatives are estimated from the slopes of a plane which is normal to the vector sum. A

similar approach is described in [Klucewicz 78]. Akima later improved this technique by weighting the contribution of each triangle such that small weights were assigned to large or narrow triangles when the vector sum was calculated [Akima 84]. For a comparison of methods, see [Nielson 83] and [Stead 84].

3.5. FOUR-CORNER MAPPING

The piecewise mapping functions described above are best suited for scattered data. Often, though, more efficient solutions are possible when the structure of the control points is regular. In particular, if the control points lie on a distorted rectangular grid, then triangulation is not necessary and the problem reduces to mapping one quadrilateral onto another. This spatial transformation, known as *four-corner mapping*, is pervasive in remote sensing and medical imaging where a grid of reseau marks on the sensor are imaged and registered with their known positions for calibration purposes. It is also common in computer graphics where it plays a central role in texture mapping.

3.5.1. Texture Mapping

Texture mapping is a powerful technique used to render visually realistic images in computer graphics. It consists of a series of spatial transformations: a texture plane, $[u,v]$, is transformed onto a 3-D surface, $[x',y',z']$, and then projected onto the output screen, $[x,y]$. This sequence is shown in Fig. 3.9, where f is the transformation from $[u,v]$ to $[x',y',z']$ and p is the projection from $[x',y',z']$ onto $[x,y]$. The forward mapping functions X and Y represent the composite function $p\,(f\,(u,v))$. The inverse mapping functions are U and V.

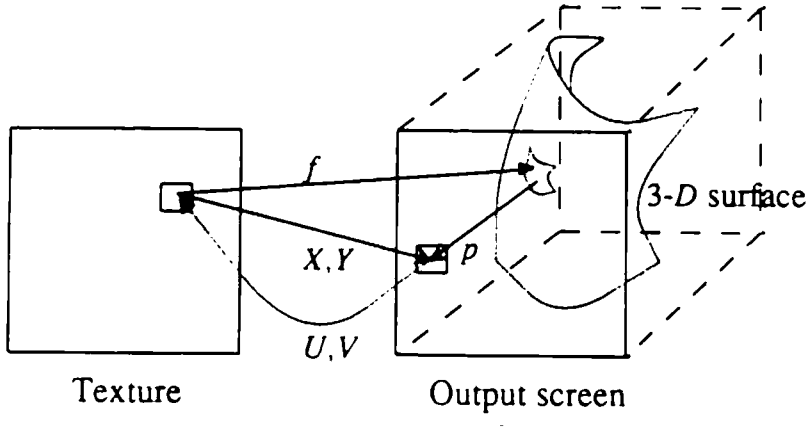


Figure 3.9: Texture mapping functions.

Texture mapping serves to create the appearance of complexity by simply applying image detail onto a surface, in much the same way as wallpaper. Textures are rather loosely defined. They are usually taken to be images used for mapping color onto the targeted surface. Textures are also used to perturb surface normals, thus allowing us to simulate bumps and wrinkles without the tedium of modeling them. Additional applications are included in [Heckbert 86b], a recent survey article on texture mapping.

The 3-D objects are usually modeled with planar polygons or bicubic patches. Patches are quite popular since they easily lend themselves for efficient rendering [Catmull 74, 80] and offer a natural parameterization that can be used as a curvilinear coordinate system. Polygons, on the other hand, are defined implicitly. Several parameterizations for planes and polygons are described below.

Once the surfaces are parameterized, the mapping between the input and output images is treated as a four-corner mapping. In inverse mapping, square output pixels must be projected back onto the input image for resampling purposes. In forward mapping, we project square texture pixels onto the output image via mapping functions X and Y. These operations are discussed below.

3.5.2. Mapping Rectangles to (Non)Planar Quadrilaterals

Consider the problem of mapping a rectangle onto an arbitrary quadrilateral. For consistency, we treat the rectangle as our undistorted input in the uv plane, and the quadrilateral as the distorted output in the xy plane. The mapping is defined through a piecewise function that must interpolate the coordinate assignments specified at the vertices. Thus, the approach is identical as before — a surface patch is fitted to the vertices of the tesselated regions. The only distinction is that rectangular regions are now used instead of triangular regions. This scheme, called *bilinear interpolation*, is used to evaluate the X and Y mapping functions. It is equivalent to using a bilinear patch, a nonlinear parameterization that maps rectangles onto planar or nonplanar quadrilaterals.

3.5.2.1. Bilinear Interpolation

Bilinear interpolation utilizes a linear combination of the four closest pixel values to produce a new, interpolated value. Given four points, $[u_0,v_0]$, $[u_1,v_1]$, $[u_2,v_2]$, and $[u_3,v_3]$, and their respective function values x_0, x_1, x_2, and x_3, any intermediate point $X(u,v)$ may be computed by the expression

$$X(u,v) = a_0 + a_1 u + a_2 v + a_3 uv \tag{3.18}$$

where the a_i coefficients are obtained by solving

$$[x_0\, x_1\, x_2\, x_3] = [a_0\, a_1\, a_2\, a_3] \begin{bmatrix} 1 & 1 & 1 & 1 \\ u_0 & u_1 & u_2 & u_3 \\ v_0 & v_1 & v_2 & v_3 \\ u_0v_0 & u_1v_1 & u_2v_2 & u_3v_3 \end{bmatrix} \tag{3.19}$$

Since the four points are assumed to lie on a rectangular grid, we rewrite them in the above matrix in terms of u_0, u_1, v_0, and v_2. Namely, the points are $[u_0,v_0]$, $[u_1,v_0]$, $[u_0,v_2]$, and $[u_1,v_2]$, respectively. Solving for a_i and substituting into Eq. (3.18) yields

$$X(u',v') = x_0 + (x_1-x_0)u' + (x_2-x_0)v' + (x_3-x_2-x_1+x_0)u'v' \tag{3.20}$$

where u' and $v' \in (0,1)$, and

$$u = u_0 + (u_1 - u_0)\,u'$$

$$v = v_0 + (v_1 - v_0)\,v'$$

Therefore, given a normalized coordinate $[u',v']$ and function values $[x_0,x_1,x_2,x_3]$, the point correspondence $[x,y]$ in the arbitrary quadrilateral is determined. Figure 3.10 depicts this bilinear interpolation for the X mapping function.

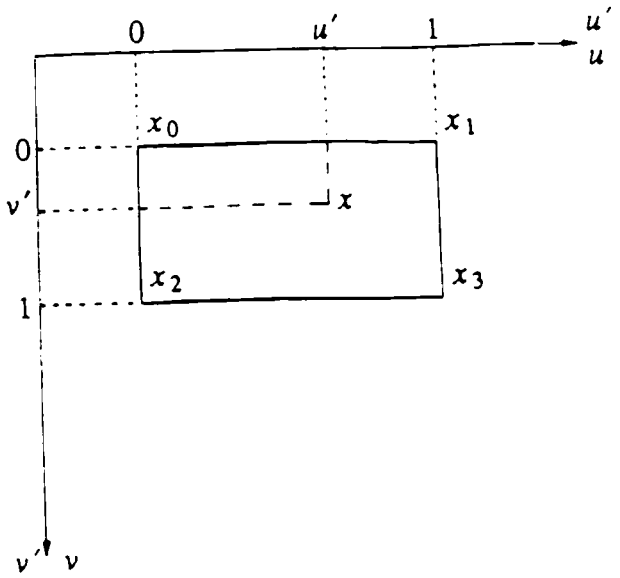


Figure 3.10: Bilinear interpolation.

presented below.

3.5.3.1. Bilinear Patch Inversion

By inverting Eq. (3.18), we can determine the normalized coordinate $[u',v']$ corresponding to the given coordinate $[x,y]$. The derivation is given below. First, we rewrite the expressions for x and y in terms of u and v, as given in Eq. (3.18).

$$x = a_0 + a_1 u + a_2 v + a_3 uv \tag{3.23a}$$

$$y = b_0 + b_1 u + b_2 v + b_3 uv \tag{3.23b}$$

Isolating u in Eq. (3.23a) gives us

$$u = \frac{x - a_0 - a_2 v}{a_1 + a_3 v} \tag{3.24}$$

In order to solve this, we must determine v. This can be done by substituting Eq. (3.24) into Eq. (3.23b). Multiplying both sides by $(a_1 + a_3 v)$ yields

$$y(a_1+a_3 v) = b_0(a_1+a_3 v) + b_1(x-a_0-a_2 v) + b_2 v(a_1+a_3 v) + b_3 v(x-a_0-a_2 v) \tag{3.25}$$

This can be rewritten as

$$c_2 v^2 + c_1 v + c_0 = 0 \tag{3.26}$$

where

$$c_0 = a_1 (b_0 - y) + b_1 (x - a_0)$$

$$c_1 = a_3 (b_0 - y) + b_3 (x - a_0) + a_1 b_2 - a_2 b_1$$

$$c_2 = a_3 b_2 - a_2 b_3$$

The inverse mapping for v thus requires the solution of a quadratic equation. Once v is determined, it is plugged into Eq. (3.24) to compute u.

3.5.3.2. Perspective Projection

A better approach is to consider the planar rectangle to be a perspective projection of the arbitrary quadrilateral. The perspective mapping of a planar quadrilateral can be expressed as

$$[uw, vw, w] = [x, y, 1] \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{3.27}$$

If a_{33} is arbitrarily chosen to be 1, eight coefficients remain to be determined. These values can be computed by solving an 8×8 system of linear equations, defined by the texture and screen coordinates of the four vertices. This is described in [Heckbert 83, 86b].

3.5.2.2. Separability

The bilinear transformation is a separable function. This property enables us to easily extend 1-D linear interpolation into two dimensions, resulting in a computationally efficient algorithm. The algorithm requires two passes, with the first pass applying 1-D linear interpolation along the horizontal direction, and the second pass interpolating along the vertical direction. For example, consider the rectangle shown in Fig. 3.11. Points x_{01} and x_{23} are interpolated in the first pass. These results are then used in the second pass to compute the final value x.

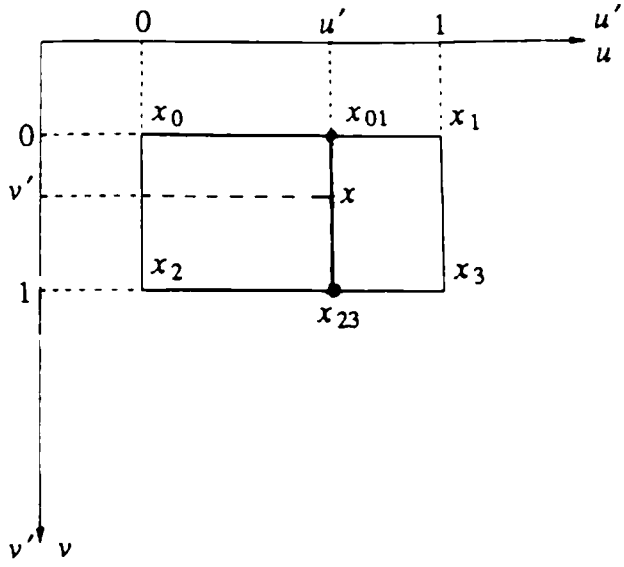


Figure 3.11: Separable bilinear interpolation.

The separable results can be shown to be identical with the solution given in Eq. (3.20). In the first (horizontal) pass, we compute

$$x_{01} = x_0 + (x_1 - x_0)u' \tag{3.21}$$
$$x_{23} = x_2 + (x_3 - x_2)u'$$

These two intermediate results are then combined in the second (vertical) pass to yield the final value

$$\begin{aligned} x &= x_{01} + (x_{23} - x_{01})v' \\ &= x_0 + (x_1 - x_0)u' + [(x_2 - x_0) + (x_3 - x_2 - x_1 + x_0)u']v' \\ &= x_0 + (x_1 - x_0)u' + (x_2 - x_0)v' + (x_3 - x_2 - x_1 + x_0)u'v' \end{aligned} \tag{3.22}$$

Notice that this result is identical with the classic solution derived in Eq. (3.20).

3.5.3. Mapping (Non)Planar Quadrilaterals to Rectangles

In remote sensing, the opposite problem is posed — given a normalized coordinate $[x', y']$ in an arbitrary (distorted) quadrilateral, find its position in the rectangle. Two solutions are

3.5.3.3. Interpolation Grid

Mapping from an arbitrary grid to a rectangular grid is an important step in performing any 2-D interpolation within an arbitrary quadrilateral. The procedure is given as follows.

1) To any point $[x,y]$ inside an interpolation region defined by four arbitrary points, a normalized coordinate $[u',v']$ is associated in a rectangular region. This makes use of the results derived above. A geometric interpretation is given in Fig. 3.12, where the normalized coordinates can be found by determining the grid lines that intersect at $[x,y]$ (point P). Given the positions labeled at the vertices, the normalized coordinates $[u',v']$ are given as

$$u' = \frac{P_{01}P_0}{P_1P_0} = \frac{P_{23}P_2}{P_3P_2} \qquad (3.28)$$

$$v' = \frac{P_{02}P_0}{P_2P_0} = \frac{P_{13}P_1}{P_3P_1}$$

2) The function values at the four quadrilateral vertices are assigned to the rectangle vertices.

3) A rectangular grid interpolation is then performed, using the normalized coordinates to index the interpolation function.

4) The result is then assigned to point $[x,y]$ in the distorted plane.

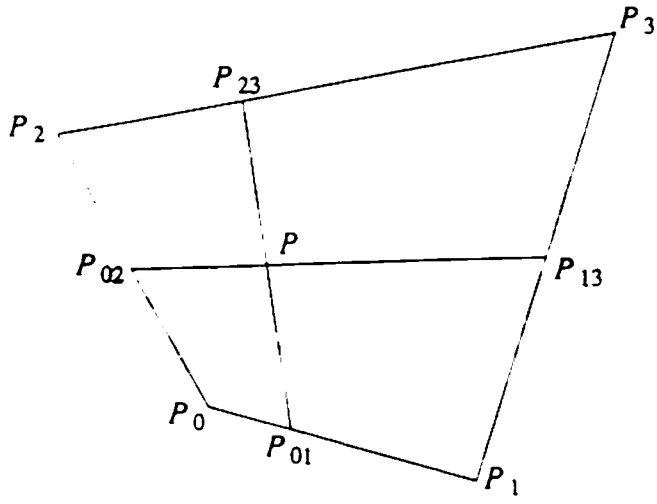


Figure 3.12: Geometric interpretation of arbitrary grid interpolation.

It is important to note that the primary benefit of this procedure is that higher-order interpolation methods (e.g., spline interpolation) which are commonly defined to operate on rectangular lattices can now be extended into the domain of non-rectangular grids. This thereby allows the generation of a continuous interpolation function for any arbitrary grid [Bizais 83].

4. SAMPLING THEORY

This section reviews the principal ideas of digital sampling and filtering theory†. Although a complete treatment of this area falls outside the scope of this paper, a brief review is appropriate in order to grasp the key issues relevant to the resampling and antialiasing stages that follow. Both stages share the common two-fold problem addressed by sampling theory:

1) Given an original input signal $g(x)$ and its sampled counterpart $g_s(x)$, are the samples of $g_s(x)$ sufficient to exactly describe $g(x)$?
2) If so, how can $g(x)$ be reconstructed from $g_s(x)$?

This problem is known as *signal reconstruction*. The solution lies in the frequency domain whereby spectral analysis is used to examine the spectrum of the sampled data.

The conclusions derived from examining the reconstruction problem will prove to be directly useful for resampling, and indicative of the filtering necessary for antialiasing. Sampling theory thereby provides an elegant mathematical framework in which to assess the quality of reconstruction, establish theoretical limits, and predict when it is not possible.

4.1. SAMPLING

Consider the imaging system discussed in section 2. For convenience, the images will be taken as one dimensional signals, i.e., a single scanline image. Recall that the continuous signal, $f(x)$, is presented to the imaging system. Due to the point spread function of the imaging device, the degraded output $g(x)$ is a bandlimited signal with attenuated high frequency components. Since visual detail directly corresponds to spatial frequency, it follows that $g(x)$ will have less detail than its original counterpart $f(x)$. The frequency content of $g(x)$ is given by its spectrum, $G(f)$, as determined by the Fourier Transform.

$$G(f) = \int_{-\infty}^{\infty} g(x)\, e^{-j2\pi fx}\, dx \tag{4.1}$$

The spectrum is shown in Fig. 4.1. Notice that the signal is bandlimited to frequency f_{max}.

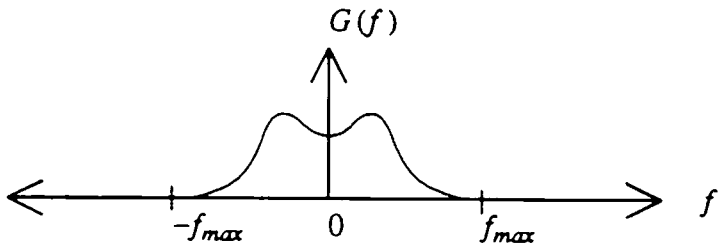


Figure 4.1: Spectrum $G(f)$.

† The reader is assumed to be familiar with elementary Fourier Transforms.

The continuous output $g(x)$ is then digitized by an ideal impulse sampler, the comb function, to get the sampled signal $g_s(x)$. The ideal 1-D sampler is given as

$$s(x) = \sum_{n=-\infty}^{\infty} \delta(x - nT_s) \tag{4.2}$$

where δ is the familiar impulse function and T_s is the sampling period. The running index n is used with δ to define the impulse train of the comb function. We now have

$$g_s(x) = g(x)s(x) \tag{4.3}$$

Taking the Fourier Transform of $g_s(x)$ yields

$$G_s(f) = G(f) * S(f) \tag{4.4}$$

$$= G(f) * \left[\sum_{n=-\infty}^{n=\infty} f_s \delta(f - nf_s) \right] \tag{4.5}$$

$$= f_s \sum_{n=-\infty}^{n=\infty} G(f - nf_s) \tag{4.6}$$

where f_s is the sampling frequency. The above equations make use of the following well-known properties of Fourier Transforms:

1) Multiplication in the spatial domain corresponds to convolution in the frequency domain. Therefore, Eq. (4.3) gives rise to a convolution in Eq. (4.4).
2) The Fourier Transform of an impulse train is itself an impulse train, justifying Eq. (4.5).
3) The spectrum of a signal sampled with frequency f_s ($T_s = 1/f_s$) yields the original spectrum replicated in the frequency domain with period f_s (Eq. 4.6).

This last property has important consequences. It yields spectrum $G_s(f)$ which, in response to a sampling period $T_s = 1/f_s$, is *periodic in frequency* with period f_s. This is depicted in Fig. 4.2. Notice then, that a small sampling period is equivalent to a high sampling frequency yielding spectra replicated far apart from each other. In the limiting case when the sampling period approaches zero ($T_s \to 0, f_s \to \infty$), only a single spectrum appears — a result consistent with the continuous case. This leads us, in the next section, to answer the central problem posed earlier regarding reconstruction of the original signal from its samples.

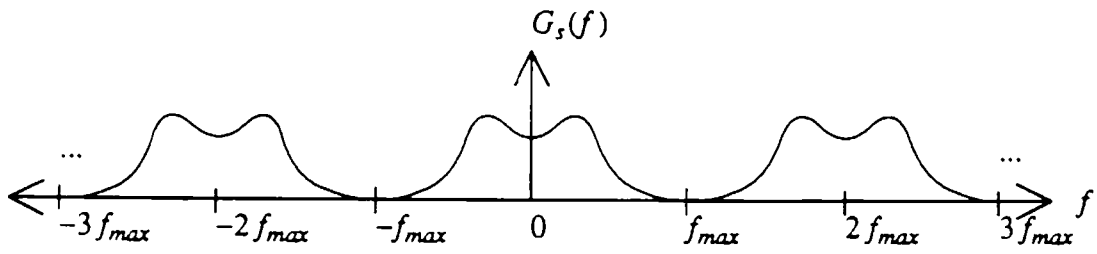


Figure 4.2: Spectrum $G_s(f)$.

4.2. RECONSTRUCTION

The above result reveals that the sampling operation has left the original input spectrum *intact*, merely replicating it periodically in the frequency domain with a spacing of f_s. This allows us to rewrite $G_s(f)$ as a sum of two terms, the low frequency (baseband) and high frequency components. The baseband expression is exactly $G(f)$, and the high frequency terms, $G_{high}(f)$, consist of the remaining replicated versions of $G(f)$.

$$G_s(f) = G(f) + G_{high}(f) \tag{4.7}$$

Exact signal reconstruction from sampled data requires us to discard the replicated spectra $G_{high}(f)$, leaving only $G(f)$, the spectrum of the signal we seek to recover. This is a crucial observation in the study of sampled-data systems.

4.2.1. Reconstruction Conditions

The only provision for exact reconstruction is that $G(f)$ be undistorted due to overlap with $G_{high}(f)$. Two conditions must hold for this to be true:

1) The signal must be bandlimited. This avoids spectra with infinite extent that are impossible to replicate without overlap.
2) The sampling frequency f_s must be greater than twice the maximum frequency f_{max}, present in the signal. This minimum sampling frequency, known as the *Nyquist rate*, is the minimum distance between spectra, each with extent of f_{max}.

The first condition merely ensures that a sufficiently large sampling frequency exists which can be used to separate replicated spectra from each other. Since all imaging systems impose a bandlimiting filter in the form of a point spread function, this condition is always satisfied for images captured through an optical system†. Note that this does not apply to synthetic images, e.g., computer generated imagery.

The second condition proves to be the most revealing statement about reconstruction. It answers the problem regarding the sufficiency of the data samples to exactly reconstruct the continuous input signal. It states that exact reconstruction is possible only when $f_s > f_{Nyquist}$, where $f_{Nyquist} = 2f_{max}$. Collectively, these two conclusions about reconstruction form the central message of sampling theory, as pioneered by Claude Shannon in his landmark papers on the subject [Shannon 48, 49]. Interestingly enough, these conditions were first discussed during the early development of television in the landmark 1934 paper by Mertz and Gray [Mertz 34]. In their work, they informally outlined these conditions as a rule-of-thumb for preventing visual artifacts in the reconstructed image.

† This does not include the shot noise that may be introduced by digital scanners.

4.2.2. Ideal Low-Pass Filter

We now turn to the second central problem: Given that it is theoretically possible to perform reconstruction, how may it be done? The answer lies with our earlier observation that sampling merely replicates the spectrum of the input signal, generating $G_{high}(f)$ in addition to $G(f)$. Therefore, the act of reconstruction requires us to completely suppress $G_{high}(f)$. This is done by multiplying $G_s(f)$ with $H(f)$, given as

$$H(f) = \begin{cases} 1 & |f| < f_{max} \\ 0 & |f| \geq f_{max} \end{cases} \tag{4.8}$$

$H(f)$ is known as an ideal low-pass filter and is depicted in Fig. 4.3, where it is shown suppressing all frequency components above f_{max}. This serves to discard the replicated spectra $G_{high}(f)$. It is ideal in the sense that the f_{max} cut-off frequency is strictly enforced as the transition point between the transmission and complete suppression of frequency components.

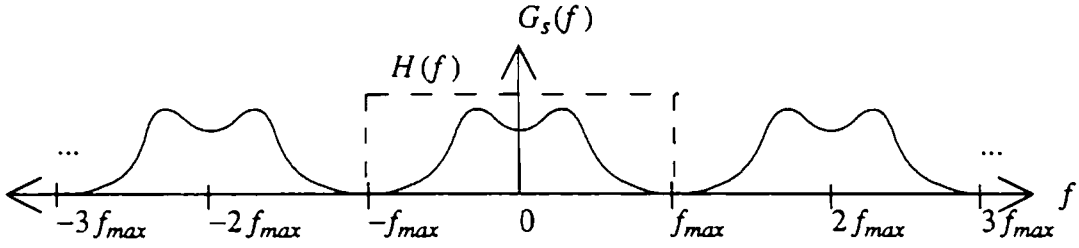


Figure 4.3: Ideal low-pass filter $H(f)$.

In the literature, there appears to be some confusion as to whether it is possible to perform exact reconstruction when sampling at *exactly* the Nyquist rate, yielding an overlap at the highest frequency component f_{max}. In that case, only the frequency can be recovered, but not the amplitude or phase. The only exception occurs if the samples are located at the minimas and maximas of the sinusoid at frequency f_{max}. Since reconstruction is possible in that exceptional instance, some sources in the literature have inappropriately included the Nyquist rate as a sampling rate which permits exact reconstruction. Nevertheless, realistic sampling techniques must sample at rates far above the Nyquist frequency in order to avoid the nonideal elements that enter into the process, e.g., sampling with a narrow pulse rather than an impulse. Therefore, this mistaken point is rather academic for natural images. This has more serious consequences for synthetic images which can indeed be sampled with a perfect comb function.

4.2.3. Sinc Function

In the spatial domain, the ideal low-pass filter is derived by computing the inverse Fourier Transform of $H(f)$. This yields the *sinc* function (Fig. 4.4), also known as a *Cardinal spline*. Note that the sinc function is only one instance of the large class of functions known as Cardinal splines. It is defined as

$$sinc(x) = \frac{sin(\pi x)}{\pi x} \tag{4.9}$$

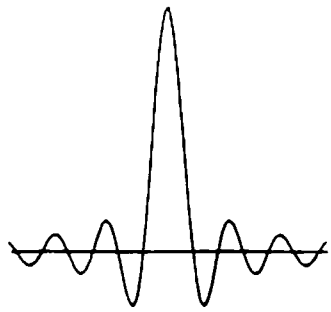


Figure 4.4: The sinc function.

Since multiplication in the frequency domain is identical to convolution in the spatial domain, $sinc\,(x)$ represents the convolution kernel used to evaluate any point x on the continuous input curve g given only the sampled data g_s.

$$\begin{aligned} g\,(x) &= sinc\,(x) * g_s(x) \\ &= \int_{-\infty}^{\infty} sinc\,(\lambda)\, g_s(x-\lambda)\, d\lambda \end{aligned} \tag{4.10}$$

Eq. (4.10) highlights an important impediment to the practical use of the ideal low-pass filter. The filter requires an infinite number of neighboring samples, i.e., an infinite filter support, in order to precisely compute the points. This is, of course, impossible owing to the finite number of data samples available. However, truncating the sinc function allows for approximate solutions to be computed at the expense of undesirable "ringing", i.e., ripple effects. These artifacts, known as the *Gibbs phenomenon*, are the overshoots and undershoots caused by reconstructing a signal with truncated frequency terms (Fig. 4.5).

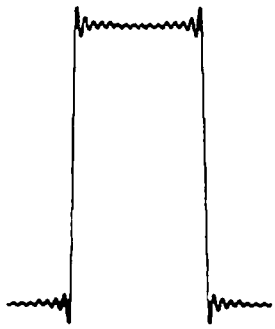


Figure 4.5: Ringing due to truncated sinc function.

In response to these difficulties, a number of approximating algorithms have been derived, offering a tradeoff between precision and computational expense. They represent nonideal reconstruction filters, allowing spurious frequencies beyond f_{max} to pass onto the output. Their descriptions are given in the resampling section.

4.3. ALIASING

If the two reconstruction conditions outlined earlier are not met, sampling theory predicts that exact reconstruction is *not* possible. This phenomenon, known as *aliasing*, occurs when signals are not bandlimited or when they are undersampled, i.e., $f_s \leq f_{Nyquist}$. In either case there will be unavoidable overlapping of spectral components, as in Fig. 4.6. Notice that the irreproducible high frequencies fold over into the low frequency range. As a result, frequencies originally beyond f_{max} will, upon reconstruction, appear in the form of much *lower* frequencies. Unlike the spurious high frequencies retained by nonideal reconstruction filters, the spectral components passed due to undersampling are more serious since they actually corrupt the components in the original signal.

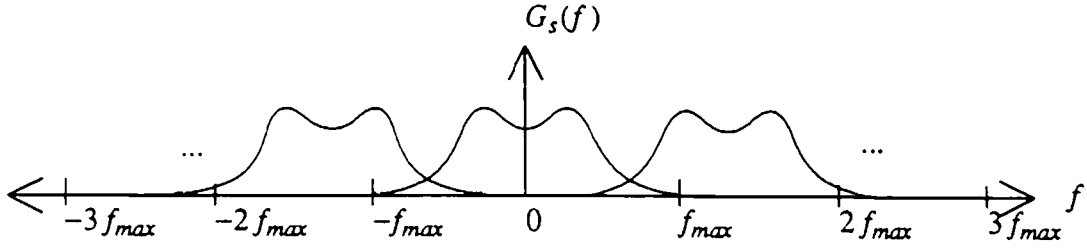


Figure 4.6: Overlapping spectral components give rise to aliasing.

Aliasing gives rise to distortions such as jagged (staircased) edges and moire patterns, the latter effect typically surfacing when the image is viewed under extreme perspective or scale change. Aliasing takes its name from the field of digital signal processing. It refers to the higher frequencies becoming aliased, and indistinguishable from, the lower frequency components in the signal if the sampling rate falls below the Nyquist frequency. In other words, undersampling causes high frequency components to appear as spurious low frequencies (Fig. 4.7). In digital images, the Nyquist rate is determined by the highest frequency that can be displayed: one cycle every two pixels. Therefore, any attempt to display higher frequencies will produce similar artifacts.

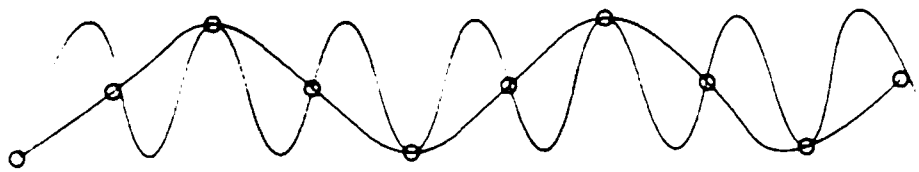


Figure 4.7: Aliasing artifacts due to undersampling.

In the computer graphics literature there is a misconception that jagged edges are always a symptom of aliasing. This is only partially true. Technically, jagged edges can arise from high frequencies introduced by inadequate reconstruction. Since these high frequencies are not corrupting the low frequency components, no aliasing is actually taking place. The confusion lies in that the suggested remedy of increasing the sampling rate is also used to eliminate aliasing.

The distinction becomes clear when we notice that the appearance of jagged edges is improved by blurring. For example, it is not uncommon to step back from an image exhibiting excessive blockiness in order to see it more clearly. This is a defocusing operation which attenuates the high frequencies admitted through nonideal reconstruction.

It is important to note that a signal may be densely sampled (far above the Nyquist rate), and continue to appear jagged if a zero-order reconstruction filter is used. In this case, the signal is clearly not aliased but rather poorly reconstructed. On the other hand, once a signal is truly undersampled, there is no postprocessing possible to improve its condition. This subtlety is pointed out in [Pavlidis 82].

4.4. ANTIALIASING

The filtering necessary to combat aliasing is known as *antialiasing*. In order to determine corrective action, we must directly address the two conditions necessary for exact signal reconstruction. The first solution calls for low-pass filtering *before* sampling. This bandlimits the signal to levels below f_{max}, thereby eliminating the offending high frequencies. Notice that the frequency at which the signal is to be sampled imposes limits on the allowable bandwidth. This is often necessary when the output sampling grid must be fixed to the resolution of an output device, e.g., screen resolution. Therefore, aliasing is often a problem that is confronted when a signal is forced to conform to an inadequate resolution due to physical constraints. As a result, it is necessary to bandlimit, or narrow, the input spectrum to conform to the allotted bandwidth as determined by the sampling frequency.

The second solution is to point sample at a higher frequency. In doing so, the replicated spectra are spaced farther apart, thereby separating the overlapping spectra tails. This approach theoretically implies sampling at a resolution determined by the highest frequencies present in the signal. Since a surface viewed obliquely can give rise to arbitrarily high frequencies, this method may require extremely high resolution. Whereas the first solution adjusts the bandwidth to accommodate the fixed sampling rate, f_s, the second solution adjusts f_s to accommodate the original bandwidth. Antialiasing by sampling at the highest frequency is clearly superior in terms of image quality. This is, of course, operating under different assumptions regarding the possibility of varying f_s. In practice, antialiasing is performed through a combination of these two approaches. That is, the sampling frequency is increased so as to reduce the amount of bandlimiting to a minimum.

The largest body of antialiasing research stems from computer graphics where high-quality rendering of complicated imagery is the central goal. The developed algorithms have primarily addressed the tradeoff issues of accuracy versus efficiency. Consequently, methods such as supersampling, adaptive sampling, stochastic sampling, pyramids, and preintegrated tables have been introduced. These techniques are described in section 6.

5. IMAGE RESAMPLING

Image resampling is the process of transforming a discrete image from one coordinate system to another. The two coordinate systems are related to each other by the mapping function of the spatial transformation. This permits the output image to be generated by the following straightforward procedure. First, the inverse mapping function is applied to the output sampling grid, projecting it onto the input. The result is a *resampling grid*, specifying the locations at which the input is to be resampled. Then, the input image is sampled at these points and the values are assigned to their respective output pixels.

The resampling process outlined above is hindered by one problem. The resampling grid does not generally coincide with the input sampling grid, taken to be the integer lattice. This is due to the fact that the range of the continuous mapping function is the set of real numbers, a superset of the integer grid upon which the input is defined. The solution therefore requires a match between the domain of the input and the range of the mapping function. This can be achieved by converting the discrete image samples into a continuous surface, a process known as *image reconstruction*. Once the input is reconstructed, it can be resampled at any position.

Conceptually, image resampling is comprised of two stages: image reconstruction followed by sampling. Although resampling takes its name from the sampling stage, image reconstruction is the implicit component in this procedure. It is achieved through an interpolation procedure, and, in fact, the terms reconstruction and interpolation are often used interchangeably.

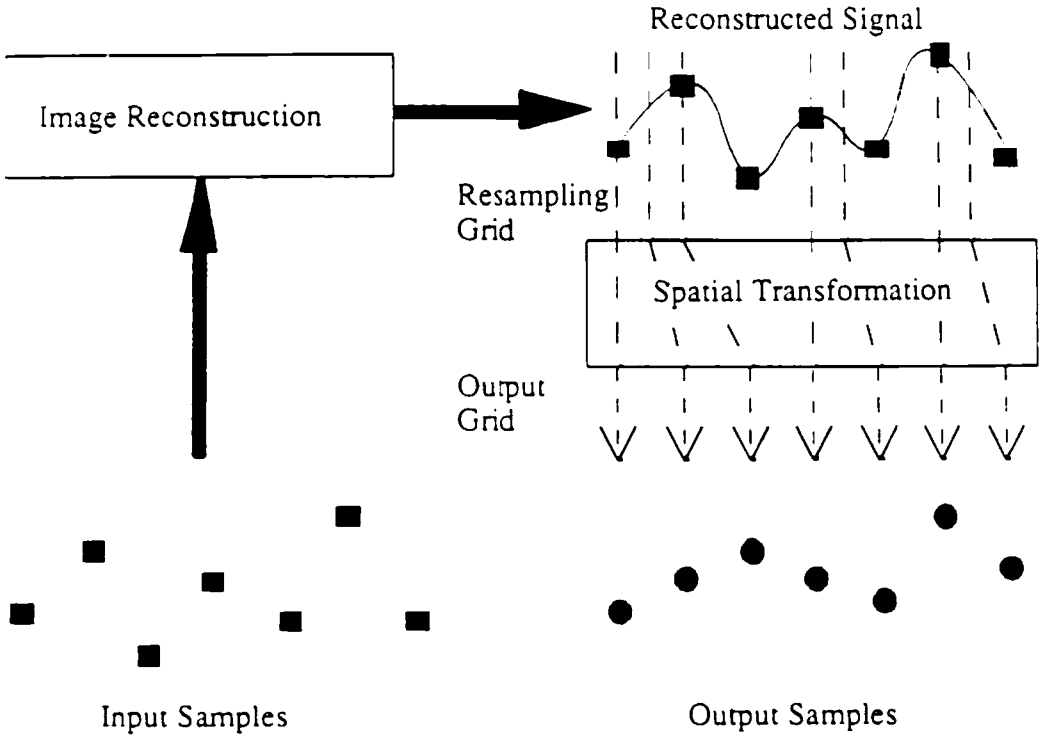


Figure 5.1: Image resampling.

The image resampling process is depicted in Fig. 5.1 for the 1-D case. A discrete input (squares) is shown passing through the image reconstruction module, yielding a continuous input signal (solid curve). Reconstruction is performed by convolving the discrete input signal with a continuous interpolating function. The reconstructed input is then modulated (multiplied) with a resampling grid (dashed arrows). Note that the resampling grid is the result of projecting the output grid onto the input through a spatial transformation. After the reconstructed signal is sampled by the resampling grid, the samples (circles) are assigned to the uniformly spaced output image.

Image magnification and minification are two typical instances of image resampling. They are illustrated in Fig. 5.2. In the top half of the figure the interval between two adjacent black and white pixels must be reconstructed in order to generate five output points. A ramp is fitted between these points and uniformly sampled at five locations to yield the intensity gradation appearing at the output. In the bottom half of the figure a scale reduction is shown. This is achieved by discarding points, a method prone to aliasing. In later sections various filters will be introduced to address the aliasing problem. These filters will be shown to be related to the interpolation functions used in reconstruction.

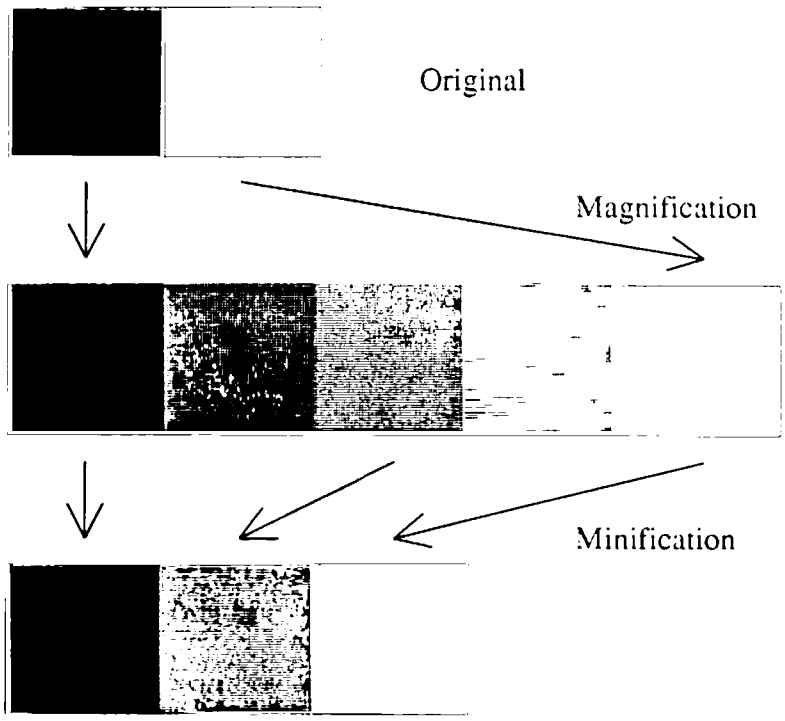


Figure 5.2: Image magnification and minification.

The interpolating function is often referred to as the interpolation *kernel*, a term used to denote the weights applied to the input signal in convolution. Another term commonly used to denote the interpolating function is *impulse response*. This relates the response of the interpolating function to a unit impulse, thereby demonstrating the influence of a sampled value upon the neighboring area.

The remainder of this section focuses on interpolation for reconstruction, the central component of image resampling. This area has received extensive treatment due to its practical significance in numerous applications. Although theoretical limits on image reconstruction are derived by sampling theory, the algorithms proposed in this section address tradeoff issues in accuracy and complexity.

5.1. INTERPOLATION

Interpolation is the process of fitting a continuous function through the discrete input samples. While sampling generates an infinite bandwidth signal from one that is bandlimited, interpolation plays an opposite role: it produces a bandlimited signal by applying a low-pass filter to the discrete signal. That is, interpolation reconstructs the signal lost in the sampling process by smoothing the data samples according to an interpolation function.

For equally spaced data, interpolation can be expressed as

$$f(x) = \sum_{k=0}^{K-1} c_k h(x - x_k) \tag{5.1}$$

where h is the interpolation kernel weighted by coefficients c_k and applied to K data samples, x_k. In all but one case that we will consider, the c_k coefficients are the data samples themselves. Note that Eq. (5.1) formulates interpolation as a convolution operation.

The computation of one interpolated point is illustrated in Fig. 5.3. The interpolating function is centered at x, the location of the point to be interpolated. The value of the interpolated point is equal to the sum of the values of the discrete input scaled by the corresponding values of the interpolation kernel. This follows from the definition of convolution.

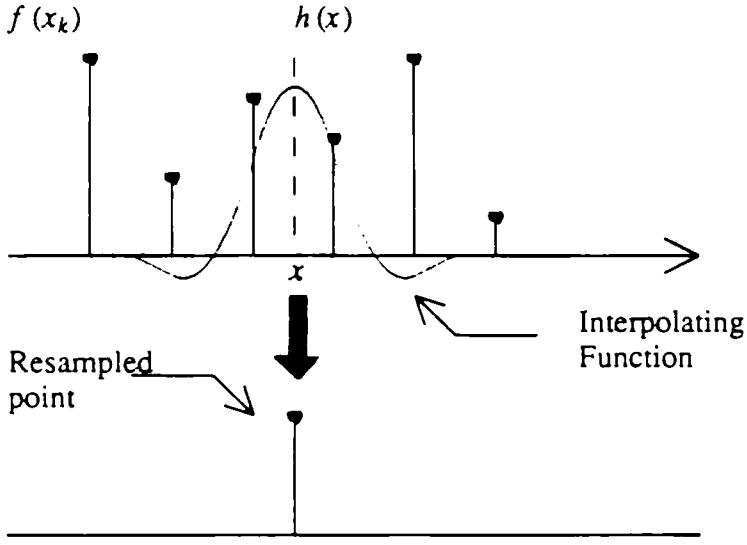


Figure 5.3: Interpolation of a single point.

The interpolation function shown in the figure extends over four points. If x is offset from the nearest point by distance d, where $0 \le d < 1$, we sample the kernel at $h(-d)$, $h(-1-d)$, $h(1-d)$, and $h(2-d)$. Since h is symmetric, it is defined only over the positive interval. Therefore, $h(d)$ and $h(1+d)$ are used in place of $h(-d)$ and $h(-1-d)$, respectively. Note that if the resampling grid is uniformly spaced, only a fixed number of points on the interpolation kernel must be evaluated. Large performance gains can be achieved by precomputing these weights and storing them in lookup tables for fast access during convolution.

Although interpolation has been posed in terms of convolution, it is rarely implemented this way. Instead, it is simpler to directly evaluate the corresponding interpolating polynomial at the resampling positions. Why then is it necessary to introduce the interpolation kernel and the convolution process into the discussion? The answer lies in the ability to compare interpolation algorithms. Whereas evaluation of the interpolation polynomial is used to implement the interpolation, analysis of the kernel is used to determine the numerical accuracy of the interpolated function. This provides us with a quantitative measure which facilitates a comparison of various interpolation methods [Schafer 73].

The accuracy of an interpolation kernel can be evaluated by analyzing its frequency domain characteristics. Of particular importance is the filter response in the passband and stopband. In this problem, the *passband* consists of all frequencies below f_{max}. The *stopband* contains all higher frequencies, arising from the sampling process†.

An ideal reconstruction filter, as described earlier, will completely suppress the stopband while leaving the passband intact. Recall that the stopband contains the offending high frequencies that, if allowed to remain, would produce aliasing artifacts. As a result, the sinc filter was devised to meet these goals and serve as the ideal reconstruction filter. Its kernel in the frequency domain applies unity gain to transmit the passband and zero gain to suppress the stopband.

The breakdown of the frequency domain into passband and stopband isolates two problems that can arise due to nonideal reconstruction filters. The first problem deals with the effects of imperfect filtering on the passband. Failure to impose unity gain on *all* frequencies in the passband will result in some combination of image smoothing or image sharpening. Smoothing, or blurring, will result when the frequency gains near the cut-off frequency start falling off. Image sharpening results when the high frequency gains are allowed to exceed unity. This follows from the direct correspondence of visual detail to spatial frequency. Furthermore, amplifying the high passband frequencies yields a sharper transition between the passband and stopband, a property shared by the sinc function.

The second problem addresses nonideal filtering on the stopband. If the stopband is allowed to persist, high frequencies will exist that will contribute to aliasing. Failure to fully suppress the stopband is a condition known as *frequency leakage*. This allows the offending

† Note that frequency ranges designated as passbands and stopbands vary among problems.

frequencies to fold over into the passband range. These distortions tend to be more serious since they are visually perceived more readily.

Due to their infinite extent, sinc filters are categorized as *infinite impulse reponse* (IIR). Practical filtering requirements, however, call for the use of *finite impulse response* (FIR) filters. In FIR filters, each output value is computed as the weighted sum of a finite number of neighboring elements. Commonly used FIR filters include the box, triangle, cubic convolution kernel, cubic B-spline, and the truncated sinc function. They serve as the interpolating functions, or kernels, described below.

5.2. INTERPOLATION KERNELS

The numerical accuracy and computational cost of interpolation algorithms are directly tied to the interpolation kernel. As a result, interpolation kernels are the target of design and analysis in the creation and evaluation of interpolation algorithms. They are subject to conditions influencing the tradeoff between accuracy and efficiency.

In this section, the analysis is applied to the 1-D case. Interpolation in 2-D will be shown to be a simple extension of the 1-D results. In addition, the data samples are assumed to be equally spaced along each dimension. This restriction imposes no serious problems since images tend to be defined on regular grids.

5.2.1. Sinc Function

Sampling theory establishes that the sinc function, or Cardinal spline, is the ideal interpolation kernel. However, since it is a filter of infinite support, it cannot be realized on standard input with finite data samples. Nevertheless, it is perfectly reasonable to consider the effects of using a truncated, and therefore finite, sinc function as the interpolation kernel.

The results of this operation are predicted by sampling theory which demonstrates that truncation in one domain leads to ringing in the other domain. Thus, truncating the sinc function in the spatial domain is equivalent to ringing in the frequency domain. Since the stopband is no longer eliminated, but rather attenuated by a ringing filter, aliasing artifacts are present. In [Ratzel 80], the author found this method to perform poorly.

The process of truncating a signal is equivalent to multiplication with a rectangle function. This function serves as a *window*, or kernel, that weighs the input signal. Ringing can be attenuated by using a different windowing function exhibiting smoother fall-off than the rectangle. The resulting *windowed* sinc function yields better results. However, since slow fall-off requires larger windows, the computation remains costly.

In spite of these problems, properties of the sinc filter may be used as heuristics for developing a superior interpolation kernel achieving accuracy and efficiency. As we will see later, the cubic convolution algorithm is an outgrowth of this goal. We now review the interpolation schemes in the order of their complexity.

5.2.2. Nearest Neighbor

The simplest interpolation algorithm from a computational standpoint is the *nearest neighbor* algorithm, where each interpolated output pixel is assigned the value of the nearest sample point in the input image. This technique, also known as the *point shift* algorithm, is given by the following interpolating polynomial.

$$f(x) = f(x_k) \qquad \frac{x_{k-1} + x_k}{2} < x \leq \frac{x_k + x_{k+1}}{2} \tag{5.2}$$

It can be achieved by convolving the image with a one-pixel width rectangle in the spatial domain. The interpolation kernel for the nearest neighbor algorithm is defined as

$$h(x) = \begin{cases} 1 & 0 \leq |x| < 1/2 \\ 0 & 1/2 \leq |x| \end{cases} \tag{5.3}$$

Various names are used to denote this simple kernel. They include the *box filter*, *sample-and-hold function*, and *Fourier window*. The kernel and its Fourier Transform are shown in Fig. 5.4.

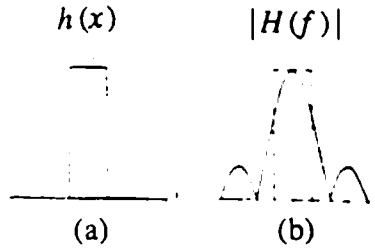


Figure 5.4: Nearest neighbor: (a) kernel, (b) Fourier Transform.

In the frequency domain, convolution with the rectangle function h is equivalent to multiplication with a sinc function. Due to the prominent side lobes and infinite extent, a sinc function makes a poor low-pass filter. Consequently, the nearest neighbor algorithm has a poor frequency domain response relative to that of the ideal low-pass filter.

The technique achieves magnification by pixel replication, and minification by sparse point sampling. For large-scale changes, nearest neighbor interpolation produces images with a blocky appearance. In addition, position errors of up to one-half pixel are possible. These problems make this technique inappropriate when sub-pixel accuracy is required.

One notable property of this algorithm is that, except for the shift, the resampled data exactly reproduces the original data if the resampling grid has the same spacing as that of the input. This means that the frequency spectra of the original and resampled images differ only by a pure linear phase shift. In general, the nearest neighbor algorithm permits zero-degree reconstruction and yields exact results only when the sampled function is piecewise constant.

Nearest neighbor interpolation was first used in remote sensing at a time when the processing time limitations of general purpose computers prohibited more sophisticated algorithms. It was found to simplify the entire mapping problem because each output point is a function of only one input sample. Furthermore, since the majority of problems involved only slight

distortions with a scale factor near one, the results were considered adequate.

Currently, this method has been superceded by more elaborate interpolation algorithms. Dramatic improvements in digital computers account for this transition. Nevertheless, the nearest neighbor algorithm continues to find widespread use in one area: frame buffer hardware zoom functions. By simply diminishing the rate at which to sample the image, and increasing the cycle period in which the sample is displayed, pixels are easily replicated on the display monitor. This scheme, also known as a sample-and-hold filter, is implemented by exploiting the roundoff features of integer arithmetic available on all computers. Although it generates images with large blocky patches, the nearest neighbor algorithm derives its primary use as a means for real-time magnification. For more sophisticated algorithms, this has only recently become realizable with the use of special-purpose hardware.

5.2.3. Linear Interpolation

Linear interpolation is a first-degree method that passes a straight line through every two consecutive points of the input signal. Given an interval (x_0,x_1), and their respective function values f_0 and f_1, the interpolating polynomial is

$$f(x) = a_1 x + a_0 \tag{5.4}$$

where a_0 and a_1 are determined by solving

$$[f_0\ f_1] = [a_1\ a_0] \begin{bmatrix} x_0 & x_1 \\ 1 & 1 \end{bmatrix}$$

This gives rise to the following interpolating polynomial.

$$f(x) = f_0 + \left[\frac{x - x_0}{x_1 - x_0}\right](f_1 - f_0) \tag{5.5}$$

Not surprisingly, we have just derived the equation of a line joining points (x_0,f_0) and (x_1,f_1). In order to evaluate this method of interpolation, we must examine the frequency response of its interpolation kernel.

In the spatial domain, linear interpolation is equivalent to convolving the sampled input with the following interpolation kernel.

$$h(x) = \begin{cases} 1 - |x| & 0 \le |x| < 1 \\ 0 & 1 \le |x| \end{cases} \tag{5.6}$$

Kernel h is referred to as a *triangle filter*, *roof function*, *Chateau function*, or *Bartlett window*.

This interpolation kernel corresponds to a reasonably good low-pass filter in the frequency domain. As shown in Fig. 5.5, its response is superior to that of the nearest neighbor interpolation function. In particular, the side lobes are far less prominent, indicating improved performance in the stopband. Nevertheless, a significant amount of spurious high-frequency

components continue to leak into the passband, contributing to some aliasing. In addition, the passband is moderately attenuated, resulting in image smoothing.

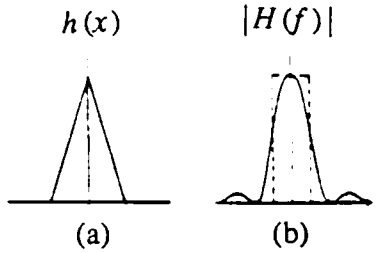


Figure 5.5: Linear interpolation: (a) kernel, (b) Fourier Transform.

Linear interpolation offers improved image quality above nearest neighbor techniques by accommodating first-degree fits. It is the most widely used interpolation algorithm for reconstruction since it produces reasonably good results at moderate cost. Often, though, higher fidelity is required and thus more sophisticated algorithms have been formulated.

Although second-degree interpolating polynomials appear to be the next step in the progression, it was shown that their filters are space-variant with phase distortion [Schafer 73]. These problems are shared by all interpolators of even-degree. This is attributed to the fact that the number of sampling points on each side of the interpolated point always differ by one. As a result, interpolating polynomials of even-degree are not considered.

5.2.4. Cubic Convolution

Cubic convolution is a third-degree interpolation algorithm originally suggested by Rifman and McKinnon [Rifman 74] as an efficient approximation to the theoretically optimum sinc interpolation function. Its interpolation kernel is derived from constraints imposed on the general cubic spline interpolation formula. The kernel is composed of piecewise cubic polynomials defined on the unit subintervals $(-2,-1)$, $(-1,0)$, $(0,1)$, and $(1,2)$. Outside the interval $(-2,2)$, the interpolation kernel is zero[†]. As a result, each interpolated point is a weighted sum of four consecutive input points. This has the desirable symmetry property of retaining two input points on each side of the interpolating region. It gives rise to a symmetric, space-invariant, interpolation kernel of the form

$$h(x) = \begin{cases} a_{30}|x|^3 + a_{20}|x|^2 + a_{10}|x| + a_{00} & 0 \le |x| < 1 \\ a_{31}|x|^3 + a_{21}|x|^2 + a_{11}|x| + a_{01} & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases} \tag{5.7}$$

The values of the coefficients can be determined by applying the following set of constraints to the interpolation kernel.

1) $h(0) = 1$ and $h(1) = h(2) = 0$.

[†] We again assume that our data points are located on the integer grid.

2) h must be continuous at $x = 0$, 1, and 2.

3) h must have a continuous first derivative at $x = 0$, 1, and 2.

The first constraint states that when h is centered on an input sample, the interpolation function is independent of neighboring samples. This permits f to actually pass through the input points. In addition, it establishes that the c_k coefficients in Eq. (5.1) are the data samples themselves. This follows from the observation that at data point x_j,

$$f(x_j) = \sum_{k=0}^{K-1} c_k h(x_j - x_k) \tag{5.8}$$

$$= \sum_{k=j-2}^{j+2} c_k h(x_j - x_k)$$

According to the first constraint listed above, $h(x_j - x_k) = 0$ unless $j = k$. Therefore, the right-hand side of Eq. (5.8) reduces to c_j. Since this equals $f(x_j)$, we see that all c_k coefficients must equal the data samples in the four-point interval.

The first two constraints provide four equations for these coefficients:

$$1 = h(0) = a_{00} \tag{5.9a}$$

$$0 = h(1^-) = a_{30} + a_{20} + a_{10} + a_{00} \tag{5.9b}$$

$$0 = h(1^+) = a_{31} + a_{21} + a_{11} + a_{01} \tag{5.9c}$$

$$0 = h(2^-) = 8a_{31} + 4a_{21} + 2a_{11} + a_{01} \tag{5.9d}$$

Three more equations are obtained from constraint (3):

$$-a_{10} = h'(0^-) = h'(0^+) = a_{10} \tag{5.9e}$$

$$3a_{30} + 2a_{20} + a_{10} = h'(1^-) = h'(1^+) = 3a_{31} + 2a_{21} + a_{11} \tag{5.9f}$$

$$12a_{31} + 4a_{21} + a_{11} = h'(2^-) = h'(2^+) = 0 \tag{5.9g}$$

The constraints given above have resulted in seven equations. However, there are eight unknown coefficients. This requires another constraint in order to obtain a unique solution. By allowing $a = a_{31}$ to be a free parameter that may be controlled by the user, the family of solutions given below may be obtained.

$$h(x) = \begin{cases} (a+2)|x|^3 - (a+3)|x|^2 + 1 & 0 \le |x| < 1 \\ a|x|^3 - 5a|x|^2 + 8a|x| - 4a & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases} \tag{5.10}$$

Additional knowledge about the shape of the desired result may be imposed upon Eq. (5.10) to yield bounds on the value of a. The heuristics applied to derive the kernel are

motivated from properties of the ideal reconstruction filter, the sinc function. By requiring h to be concave upward at $|x| = 1$, and concave downward at $x = 0$, we have

$$h''(0) = -2(a+3) < 0 \quad \rightarrow \quad a > -3 \tag{5.11a}$$

$$h''(1) = -4a > 0 \quad \rightarrow \quad a < 0 \tag{5.11b}$$

Bounding a to values between -3 and 0 makes h resemble the sinc function. In [Rifman 74], the authors use the constraint that $a = -1$ in order to match the slope of the sinc function at $x = 1$. This choice results in some amplification of the frequencies at the high-end of the passband. As stated earlier, such behavior is characteristic of image sharpening.

Other choices for a include $-1/2$ and $-3/4$. Keys selected $a = -1/2$ by making the Taylor series approximation of the interpolated function agree in as many terms as possible with the original signal [Keys 81]. He found that the resulting interpolating polynomial will exactly reconstruct a second-degree polynomial. Finally, $a = -3/4$ is used to set the second derivatives of the two cubic polynomials in h to 1 [Simon 75]. This allows the second derivative to be continuous at $x = 1$.

Of the three choices for a, the value -1 is preferable if visually enhanced results are desired. That is, the image is sharpened, making visual detail perceived more readily. However, the results are not mathematically precise, where precision is measured by the order of the Taylor series. To maximize this order, the value $a = -1/2$ is preferable. The kernel and spectrum of a cubic convolution kernel with $a = -1/2$ is shown in Fig. 5.6.

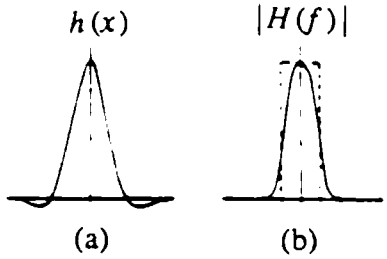


Figure 5.6: Cubic convolution: (a) kernel ($a = -1/2$), (b) Fourier Transform.

In a recent paper [Maeland 88], Maeland showed that at the Nyquist frequency the spectrum attains a value which is independent of the free parameter a. The value is equal to $(48/\pi^4)f_s$, while the value at the zero frequency is f_s. This result implies that adjusting a can alter the transition rate between the passband and stopband, but not the performance gain at the cut-off frequency. In comparing the effect of varying a, Maeland points out that cubic convolution with $a = 0$ is superior to the simple linear interpolation method when a strictly positive kernel is necessary. The role of a has also been studied in [Park 83], where a discussion is given on its optimal selection based on the frequency content of the image.

It is important to note that in the general case cubic convolution can give rise to values outside the range of the input data. Consequently, when using this method in image processing it is necessary to properly clip or rescale the results into the appropriate range for display.

5.2.5. Two-Parameter Cubic Filters

In [Mitchell 88], Mitchell and Netravali describe a variation of cubic convolution in which two parameters are used to describe a family of cubic reconstruction filters. Through a different set of constraints, the number of free parameters in Eq. (5.7) are reduced from eight to two, yielding the following two-parameter family of solutions.

$$h(x) = \frac{1}{6}\begin{cases} (-9b-6c+12)|x|^3 + (12b+6c-18)|x|^2 + (-2b+6) & 0 \le |x| < 1 \\ (-b-6c)|x|^3 + (6b+30c)|x|^2 + (-12b-48c)|x| + (8b+24c) & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases} \quad (5.12)$$

Several well-known cubic filters are derivable from Eq. (5.12) through an appropriate choice of values for (b,c). For instance, $(0,-c)$ corresponds to the cubic convolution kernel in Eq. (5.10) and $(1,0)$ is the cubic B-spline given later in Eq. (5.19).

The evaluation of these parameters is performed in the spatial domain, using the visual artifacts described in [Schreiber 85] as the criteria for judging image quality. In order to better understand the behavior of (b,c), the authors partitioned the parameter space into regions characterizing different artifacts, including blur, anisotropy, and ringing. As a result, the parameter pair $(1/3,1/3)$ is found to offer superior image quality. Another suggestion is $(3/2,-1/4)$, corresponding to a band-reject, or notch, filter. This suppresses the signal energy near the Nyquist frequency that is most responsible for conspicuous moire patterns.

Further improvements in reconstruction are possible when derivative values can be given along with the signal amplitude. This is possible for synthetic images where this information may be available. In that case, Eq. (5.1) can be rewritten as

$$f(x) = \sum_{k=0}^{K-1} \left[f_k g(x-x_k) + f'_k h(x-x_k) \right] \quad (5.13)$$

where

$$g(x) = \frac{\sin^2 \pi x}{\pi^2 x^2} \quad (5.14a)$$

$$h(x) = \frac{\sin^2 \pi x}{\pi^2 x} \quad (5.14b)$$

An approximation to the resulting reconstruction formula can be given by Hermite cubic interpolation.

$$g(x) = \begin{cases} 2|x|^3 - 3|x|^2 + 1 & 0 \le |x| < 1 \\ 0 & 1 \le |x| \end{cases} \quad (5.15a)$$

$$h(x) = \begin{cases} |x|^3 - 2x|x| + x & 0 \le |x| < 1 \\ 0 & 1 \le |x| \end{cases} \quad (5.15b)$$

5.2.6. Cubic Splines

The final reconstruction technique described here is the method of cubic spline interpolation. A *cubic spline* is a piecewise continuous third-degree polynomial. Given n points labeled (x_k,y_k) for $0 \le k < n$, the interpolating cubic spline consists of $n-1$ cubic polynomials. They pass through the supplied points, which are also known as *control points* or *knots*.

We now derive the piecewise interpolating polynomials. The k^{th} polynomial piece, f_k, is defined to pass through two consecutive input points in the fixed interval (x_k,x_{k+1}). Furthermore, f_k are joined at x_k (for $k = 1,...,n-2$) such that f_k, f_k', and f_k'' are continuous (Fig. 5.7). The interpolating polynomial f_k is given as

$$f_k(x) = a_3(x-x_k)^3 + a_2(x-x_k)^2 + a_1(x-x_k) + a_0 \tag{5.16}$$

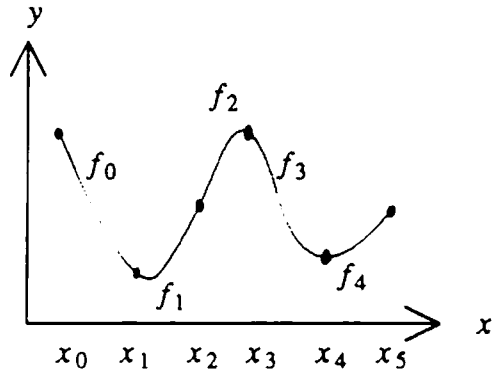


Figure 5.7: A spline consisting of 5 piecewise cubic polynomials.

The four coefficients of f_k can be defined in terms of the data points and their first (or second) derivatives. Assuming that the data samples are on the integer lattice, each spaced one unit apart, then the coefficients, defined in terms of the data samples and their first derivatives, are given below.

$$a_0 = y_k \tag{5.17a}$$

$$a_1 = y_k' \tag{5.17b}$$

$$a_2 = 3\Delta y_k - 2y_k' - y_{k+1}' \tag{5.17c}$$

$$a_3 = -2\Delta y_k + y_k' + y_{k+1}' \tag{5.17d}$$

where $\Delta y_k = y_{k+1} - y_k$.

Although the derivatives are not supplied with the data, they are derived by solving the following system of linear equations.

$$\begin{bmatrix} 2 & 4 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & . & & & & \\ & & & . & & & \\ & & & & 1 & 4 & 1 \\ & & & & & 4 & 2 \end{bmatrix} \begin{bmatrix} y'_0 \\ y'_1 \\ y'_2 \\ . \\ . \\ y'_{n-2} \\ y'_{n-1} \end{bmatrix} = \begin{bmatrix} -5y_0 + 4y_1 + y_2 \\ 3(y_2 - y_0) \\ 3(y_3 - y_1) \\ . \\ . \\ 3(y_{n-1} - y_{n-3}) \\ -y_{n-3} - 4y_{n-2} + 5y_{n-1} \end{bmatrix} \tag{5.18}$$

The not-a-knot boundary condition [de Boor 78] was used above, as reflected in the first and last rows of the matrices. It is superior to the artificial boundary conditions commonly reported in the literature, such as the natural or cyclic end conditions, which have no relevance in our application. Note that the need to solve a linear system of equations arises from global dependencies introduced by the constraints for continuous first and second derivatives at the knots. A complete derivation is given in [Wolberg 88b].

In order to compare interpolating cubic splines with other methods we must analyze the interpolation kernel. Thus far, however, the piecewise interpolating polynomials have been derived without any reference to an interpolation kernel. We seek to express the interpolating cubic spline as a convolution in a manner similar to the previous algorithms. This can be done with the use of cubic B-splines as interpolation kernels [Hou 78].

5.2.6.1. B-Splines

A *B-spline* of degree n is derived through n convolutions of the box filter, B_0. Thus, $B_1 = B_0 * B_0$ denotes a B-spline of degree 1, yielding the familiar triangle function shown in Fig. 5.5a. Interpolation by B_1 consists of a sequence of straight lines joined at the knots continuously. This is equivalent to linear interpolation.

The second-degree B-spline B_2 is produced by convolving B_0 three times. Using B_2 to interpolate data yields a sequence of parabolas which join at the knots continuously together with their slopes. The span of B_2 is limited to three points.

The *cubic B-spline* B_3 is generated from four convolutions of B_0. That is, $B_3 = B_0 * B_0 * B_0 * B_0$. The interpolation with B_3 is composed of a series of cubic polynomials which join at the knots continuously together with their slopes and curvatures, i.e., their first and second derivatives. Figure 5.8 summarizes the shapes of these low-order B-splines.

Denoting the cubic B-spline interpolation kernel as h, we have the following piecewise cubic polynomials defining the kernel.

$$h(x) = \frac{1}{6} \begin{cases} 3|x|^3 - 6|x|^2 + 4 & 0 \le |x| < 1 \\ -|x|^3 + 6|x|^2 - 12|x| + 8 & 1 \le |x| < 2 \\ 0 & 2 \le |x| \end{cases} \tag{5.19}$$

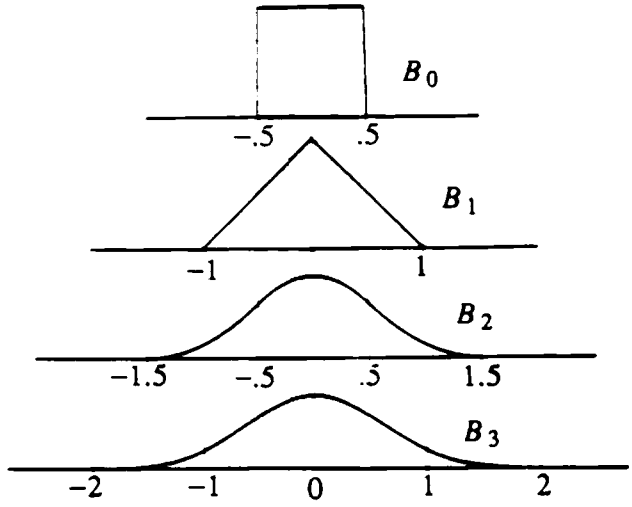


Figure 5.8: Low-order B-splines are derived from repeated box filters.

This kernel is sometimes called the *Parzen window*.

There are several properties of cubic B-splines worth noting. As in the cubic convolution method, the extent of the cubic B-spline is over four points. This allows two points on each side of the central interpolated region to be used in the convolution. Consequently, the cubic B-spline is shift-invariant as well.

Unlike cubic convolution, however, the cubic B-spline kernel is not interpolatory since it does not satisfy the necessary constraint that $h(0) = 1$ and $h(1) = h(2) = 0$. Instead, it is an approximating function which passes near the points but not necessarily through them. This is due to the fact that the kernel is strictly positive.

The positivity of the cubic B-spline kernel is actually attractive for our image processing application. When using kernels with negative lobes, e.g., the cubic convolution and truncated sinc functions, it is possible to generate negative values while interpolating positive data. Since negative intensity values are meaningless for display it is desirable to use strictly positive interpolation kernels to guarantee the positivity of the interpolated image.

There are problems, however, in directly interpolating the data with kernel h, as given in Eq. (5.19). Due to the low-pass (blur) characteristics of h, the image undergoes considerable smoothing. This is evident by examining its frequency response where the stopband is effectively suppressed at the expense of additional attenuation in the passband. This leads us to the development of an interpolation method built upon the local support of the cubic B-spline.

5.2.6.2. Interpolating B-Splines

Interpolating with cubic B-splines requires that at data point x_j, we again satisfy Eq. (5.8). Namely,

$$f(x_j) = \sum_{k=j-2}^{j+2} c_k h(x_j - x_k) \tag{5.20}$$

From Eq. (5.19), we have $h(0) = 4/6$, $h(-1) = h(1) = 1/6$, and $h(-2) = h(2) = 0$. This yields

$$f(x_j) = \frac{1}{6}(c_{j-1} + 4c_j + c_{j+1}) \tag{5.21}$$

Since this must be true for all data points, we have a chain of global dependencies for the c_k coefficients. The resulting linear system of equations is similar to that obtained for the derivatives of the cubic interpolating spline algorithm. We thus have,

$$\begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ \cdot \\ \cdot \\ f_{n-2} \\ f_{n-1} \end{bmatrix} = \begin{bmatrix} 4 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & \cdot & & & & \\ & & & \cdot & & & \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 4 \end{bmatrix} \begin{bmatrix} c_0 \\ c_1 \\ c_2 \\ \cdot \\ \cdot \\ c_{n-2} \\ c_{n-1} \end{bmatrix} \tag{5.22}$$

Labeling the three matrices above as F, K, and C, respectively, we have

$$F = K C \tag{5.23}$$

The coefficients in C may be evaluated by multiplying the known data points F with the inverse of the tridiagonal matrix K.

$$C = F K^{-1} \tag{5.24}$$

This matrix inversion has an efficient algorithm which is solvable in linear time [Press 88]. In [Lee 83], the matrix inversion step is modified to introduce high-frequency emphasis. This serves to compensate for the undesirable low-pass filter imposed by the point-spread function of the imaging system.

In all the previous methods the coefficients c_k were taken to be the data samples themselves. In the cubic spline interpolation algorithm, however, the coefficients must be determined by solving a tridiagonal matrix problem. After the interpolation coefficients have been computed, cubic spline interpolation has the same computational cost as cubic convolution.

5.3. COMPARISON OF INTERPOLATION METHODS

The quality of the popular interpolation kernels are ranked in ascending order as follows: nearest neighbor, linear, cubic convolution, cubic spline, and sinc function. These interpolation methods are compared in [Andrews 76], [Parker 83], and [Maeland 88].

The algorithms are rated according to the passband and stopband performances of their interpolation kernels. If an additional process is required to compute coefficients used together with the kernel, its effect must be evaluated as well. In [Parker 83], the authors failed to consider this when they erroneously concluded that cubic convolution is superior to cubic spline interpolation. Their conclusion was based on an inappropriate comparison of the cubic B-spline kernel with that of the cubic convolution. The fault lies in neglecting the effect of computing the coefficients in Eq. (5.1). Had the data samples been directly convolved with the cubic B-spline kernel, then the analysis would have been correct. However, in performing a matrix inversion to determine the coefficients, a certain periodic filter must be multiplied together with the spectrum of the cubic B-spline in order to produce the interpolation kernel. The resulting kernel can be easily demonstrated to be of infinite support and oscillatory, sharing the same properties as the Cardinal spline (sinc) kernel [Maeland 88]. By a direct comparison, cubic spline interpolation performs better than cubic convolution, albeit at slightly greater computational cost.

It is important to note that high quality interpolation algorithms are not always warranted for adequate reconstruction. This is due to the natural relationship that exists between the rate at which the input is sampled and the interpolation quality necessary for accurate reconstruction. If a bandlimited input is densely sampled, then its replicating spectra are spaced far apart. This diminishes the role of frequency leakage in the degradation of the reconstructed signal. Consequently, we can relax the accuracy of the interpolation kernel in the stopband. Therefore, the stopband performance necessary for adequate reconstruction can be made a function of the input sampling rate. Low sampling rates require the complexity of the sinc function, while high rates allow simpler algorithms. Although this result is intuitively obvious, it is reassuring to arrive at the same conclusion from an interpretation in the frequency domain.

The above discussion has focused on reconstructing gray-scale (color) images. Complications emerge when the attention is restricted to bi-level (binary) images. In [Abdou 82], the authors analyze several interpolation schemes for bi-level image applications. This is of practical importance for the geometric transformation of images of black-and-white documents. Subtleties are introduced due to the nonlinear elements that enter into the imaging process: quantization and thresholding. Since binary signals are not bandlimited and the nonlinear effects are difficult to analyze in the frequency domain, the analysis is performed in the spatial domain. Their results confirm the conclusions already derived regarding interpolation kernels. In addition, they arrive at useful results relating the errors introduced in the tradeoff between sampling rate and quantization.

5.4. SEPARABLE 2-D INTERPOLATION

The 1-D interpolation algorithms described above generalize quite simply to 2-D. This is accomplished by performing 1-D interpolation in each dimension. For example, the horizontal scanlines are first processed, yielding an intermediate image which then undergoes a second pass of interpolation in the vertical direction. These are the elements of a separable transformation, which allow a reconstruction filter $h(x,y)$ to be replaced by the product $h(x)h(y)$.

In 2-D, the nearest neighbor and bilinear interpolation algorithms use a 2×2 neighborhood about the desired location. The separable transform result is identical to computing these methods directly in 2-D. The proof for bilinear interpolation was given in section 3. In cubic convolution, a 4×4 neighborhood is used to achieve an approximation to the radially symmetric 2-D sinc function. Note that this is not equivalent to the result obtained through direct computation. This can be easily verified by observing that the zeros are all aligned along the rectangular grid instead of being distributed along concentric circles. Nevertheless, separable transforms provide a substantial reduction in computational complexity from $O(N^2)$ to $O(N)$ for an $N\times N$ image.

6. ANTIALIASING

The geometric transformation of digital images is inherently a sampling process. As with all sampled data, digital images are susceptible to aliasing artifacts. This section reviews the antialiasing techniques developed to counter these deleterious effects. The largest contribution to this area stems from work in computer graphics, where visually complex images containing high spatial frequencies must be rendered onto a discrete array. In particular, antialiasing has played a critical role in the quality of texture-mapped and ray-traced images. Remote sensing and medical imaging, on the other hand, typically do not deal with large scale changes that warrant sophisticated filtering. They have therefore neglected this stage of the processing.

6.1. INTRODUCTION

Aliasing occurs when the input signal is undersampled. There are two solutions to this problem: raise the sampling rate or bandlimit the input. The first solution is ideal but may require a display resolution which is too costly or unavailable. The second solution forces the signal to conform to the low sampling rate by attenuating the high frequency components that give rise to the aliasing artifacts. In practice, some compromise is reached between these two solutions [Crow 77, 81].

6.1.1. Point Sampling

The naive approach for generating an output image is to perform *point sampling*, where each output pixel is a single sample of the input image taken independently of its neighbors (Fig. 6.1). It is clear that information is lost between the samples and that aliasing artifacts may surface if the sampling density is not sufficiently high to characterize the input. This problem is rooted in the fact that intermediate intervals between samples, which should have some influence on the output, are skipped entirely.

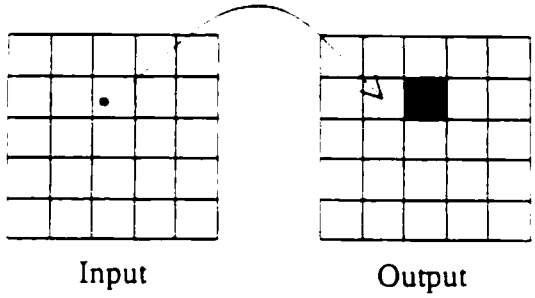


Figure 6.1: Point sampling.

Aliasing can be reduced by point sampling at a higher resolution. This raises the Nyquist limit, accounting for signals with higher bandwidths. Generally, though, the display resolution places a limit on the highest frequency that can be displayed, and thus limits the Nyquist rate to one cycle every two pixels. Any attempt to display higher frequencies will produce aliasing artifacts such as moire patterns and jagged edges. Consequently, antialiasing algorithms have been derived to bandlimit the input *before* resampling onto the output grid.

6.1.2. Area Sampling

The basic flaw in point sampling is that a discrete pixel actually represents an area, not a point. In this manner, each output pixel should be considered a window looking onto the input image. Rather than sampling a point, we must instead apply a low-pass filter (LPF) upon the projected area in order to properly reflect the information content being mapped onto the output pixel. This approach, depicted in Fig. 6.2, is called *area sampling* and the projected area is known as the *preimage*. The low-pass filter comprises the *prefiltering* stage. It serves to defeat aliasing by bandlimiting the input image prior to resampling it onto the output grid. In the general case, prefiltering can be defined by the following convolution integral.

$$g(x,y) = \int\int f(u,v)\, h(x-u, y-v)\, du\, dv \tag{6.1}$$

where f is the input image, g is the output image, h is the filter kernel, and the integration is applied to all $[u,v]$ points in the preimage.

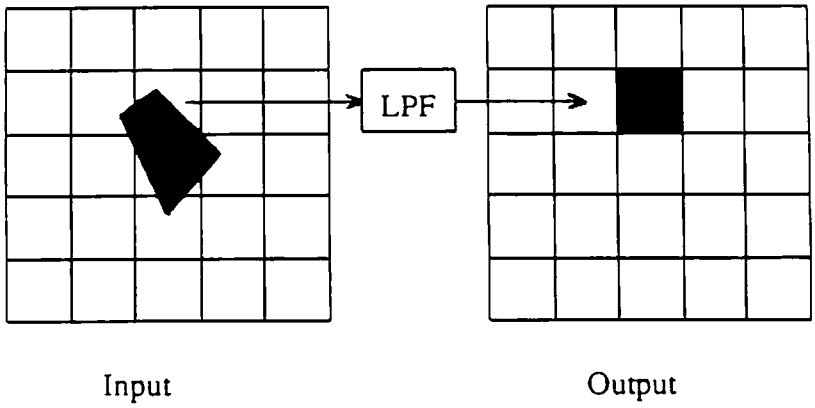


Figure 6.2: Area sampling.

Area sampling is akin to direct convolution except for one notable exception: independently projecting each output pixel onto the input image limits the extent of the filter kernel to the projected area. As we shall see, this constraint can be lifted by considering the bounding area which is the smallest region that completely bounds the pixel's convolution kernel. Depending on the size and shape of convolution kernels, these areas may overlap. Since this carries extra computational cost, most area sampling algorithms limit themselves to the restrictive definition which, nevertheless, is far superior to point sampling. The question that remains open is the manner in which the incoming data is to be filtered. There are various theoretical and practical considerations to be addressed.

6.1.3. Space-Invariant Filtering

Ideally, the sinc function should be used to filter the preimage. However, as discussed in sections 4 and 5, a finite impulse response (FIR) approximation must be used instead to form a weighted average of samples. If the mapping is affine, the filter kernel remains constant as it scans across the image. Such a filter is said to be *space-invariant*.

Fourier convolution can be used to implement space-invariant filtering by transforming the image and filter kernel into the frequency domain using an FFT, multiplying them together, and then computing the inverse FFT. For wide space-invariant kernels, this becomes the method of choice since it requires $O(N \log_2 N)$ operations instead of $O(MN)$ operations for direct convolution, where M and N are the lengths of the filter kernel and image, respectively. Since the cost of Fourier convolution is independent of the kernel width, it becomes practical when $M > \log_2 N$. This means, for example, that scaling an image can best be done in the frequency domain when excessive magnification or minification is desired. An excellent tutorial on the theory supporting digital filtering in the frequency domain can be found in [Smith 83].

6.1.4. Space-Variant Filtering

In most applications, however, *space-variant* filters are required, where the kernel varies with position. This is necessary for many common operations such as perspective mappings, nonlinear warps, and texture mapping. In such cases, space-variant FIR filters are used to convolve the preimage. Proper filtering requires a large number of preimage samples in order to compute each output pixel. There are various sampling strategies used to collect these samples. They can be broadly categorized into two classes: regular sampling and irregular sampling.

6.2. REGULAR SAMPLING

The process of using a regular sampling grid to collect image samples is called *regular sampling*. It is also known as *uniform sampling*, which is slightly misleading since an irregular sampling grid can also generate a uniform distribution of samples. Regular sampling includes point sampling, as well as the supersampling and adaptive sampling techniques described below.

6.2.1. Supersampling

The process of using more than one regularly-spaced sample per pixel is known as *supersampling*. Each output pixel value is evaluated by computing a weighted average of the samples taken from their respective preimages. For example, if the supersampling grid is three times denser than the output grid (i.e., there are nine grid points per pixel area), each output pixel will be an average of the nine samples taken from its projection in the input image. If, say, three samples hit a green object and the remaining six samples hit a blue object, the composite color in the output pixel will be one-third green and two-thirds blue.

Supersampling reduces aliasing by bandlimiting the input signal. The purpose of the high-resolution supersampling grid is to refine the estimate of the preimages seen by the output pixels. The samples then enter the prefiltering stage, consisting of a low-pass filter. This permits the input to be resampled onto the (relatively) low-resolution output grid without any offending high frequencies introducing aliasing artifacts (Fig. 6.3).

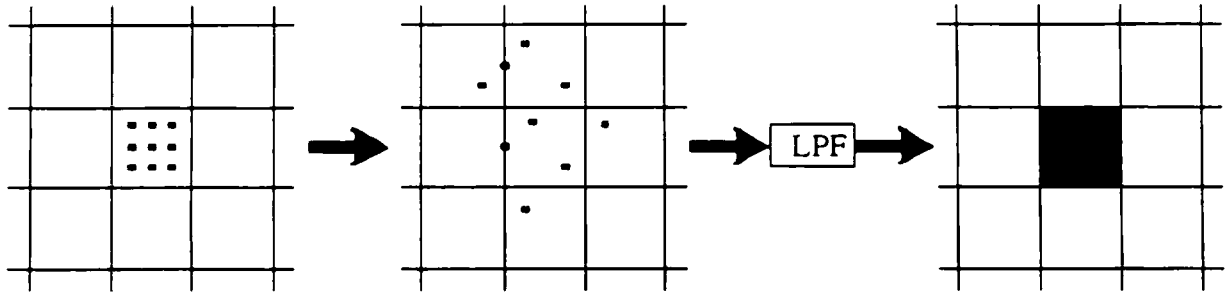


Figure 6.3: Supersampling.

There are two problems associated with straightforward supersampling. The first problem is that the increased frequency of the prefiltered image continues to be fixed. Therefore, there will always be sufficiently high frequencies that will alias. The second problem is cost. In our example, supersampling will take nine times longer than point sampling. Although there is a clear need for the additional computation, the dense placement of samples can be optimized. Adaptive sampling is introduced to address these drawbacks.

6.2.2. Adaptive Sampling

In *adaptive sampling*, the samples are distributed more densely in areas of high intensity variance. In this manner, supersamples are collected only in regions that warrant their use. Early work in adaptive sampling for computer graphics is described in [Whitted 80]. The strategy is to subdivide areas between previous samples when an edge, or some other high frequency pattern, is present. Two approaches to adaptive sampling have been described in the literature. The first approach allows sampling density to vary as a function of local image variance [Lee 85, Kajiya 86]. A second approach introduces two levels of sampling densities: a regular pattern for most areas and a higher-density pattern for regions demonstrating high frequencies. The regular pattern simply consists of one sample per output pixel. The high density pattern involves local supersampling at a rate of 4 to 16 samples per pixel. Typically, these rates are adequate for suppressing aliasing artifacts.

A sampling strategy is required to determine where supersampling is necessary. In [Mitchell 87], the author describes a method in which the image is divided into small square supersampling cells, each containing eight or nine of the low-density samples. The entire cell is supersampled if its samples exhibit excessive variation. In [Lee 85], the variance of the samples are used to indicate high frequency. It is well-known, however, that variance is a poor measure of visual perception of local variation. Another alternative is to use contrast, which more closely models the nonlinear response of the human eye to rapid fluctuations in light intensities [Caelli 81]. Contrast is given as

$$C = \frac{I_{max} - I_{min}}{I_{max} + I_{min}} \tag{6.2}$$

Adaptive sampling reduces the number of samples required for a given image quality. The problem with this technique, however, is that the variance measurement is itself based on point samples, and so this method can fail as well. This is particularly true for sub-pixel objects that do not cross pixel boundaries. Nevertheless, adaptive sampling presents a far more reliable and cost-effective alternative to supersampling.

6.2.3. Reconstruction from Regular Samples

Each output pixel is evaluted as an average of the preimage samples. The low-pass filters shown in Figs. 6.2 and 6.3 are actually reconstruction filters used to interpolate the output point. They share the identical function of the reconstruction filters discussed in section 5: they bandlimit the sampled signal (suppress the replicated spectra) so that the resampling process does not itself introduce aliasing. The careful reader will notice that reconstruction serves two roles:

1) Reconstruction filters interpolate the input samples to compute values at nonintegral positions. These values are the preimage samples that are assigned to the supersampling grid.
2) The very same filters are used to interpolate a new value from the dense set of samples collected in step (1). The result is applied to the output pixel.

When reconstruction filters are applied to interpolate new values from regularly-spaced samples, errors may appear as observable derivative discontinuities across pixel boundaries. In antialiasing, reconstruction errors are more subtle. Consider an object of constant intensity which is entirely embedded in pixel p, i.e., a sub-pixel sized object. We will assume that the popular triangle filter is used as the reconstruction kernel. As the object moves away from the center of p, the computed intensity for p decreases as it moves towards the edge. Upon crossing the pixel boundary, the object begins to contribute to the adjacent pixel, no longer having an influence on p. If this motion were animated, the object would appear to flicker as it crossed the image. This artifact is due to the limited range of the filter. This suggests that a wider filter is required, in order to reflect the object's contribution to neighboring pixels.

One ad hoc solution is to use a square pyramid with a base width of 2×2 pixels. This approach was used in [Blinn 76], an early paper on texture mapping. In general, by varying the width of the filter a compromise is reached between passband transmission and stopband attenuation. This underscores the need for high-quality reconstruction filters to prevent aliasing in image resampling.

Despite the apparent benefits of supersampling and adaptive sampling, all regular sampling methods share a common problem: information is discarded in a coherent way. This produces coherent aliasing artifacts that are easily perceived. Since spatially correlated errors are a consequence of the regularity of the sampling grid, the use of irregular sampling grids has been proposed to address this problem.

6.3. IRREGULAR SAMPLING

Irregular sampling is the process of using an irregular sampling grid in which to sample the input image. This process is also referred to as *nonuniform sampling* and *stochastic sampling*. As before, the term nonuniform sampling is a slight misnomer since irregular sampling can be used to produce a uniform distribution of samples. The name stochastic sampling is more appropriate since it denotes the fact that the irregularly-spaced locations are determined probabilistically via a Monte Carlo technique.

The motivation for irregular sampling is that coherent aliasing artifacts can be rendered incoherent, and thus less conspicuous. By collecting irregularly-spaced samples, the energies of the offending high frequencies are made to appear as featureless noise of the correct average intensity, an artifact that is much less objectionable than aliasing. This claim is supported by evidence from work in color television encoding [Lim 77], image noise measurement [Sakrison 77], dithering [Lim 69, Ulichney 87], and the distribution of retinal cells in the human eye [Yellott 83].

6.3.1. Stochastic Sampling

Although the mathematical properties of stochastic sampling have received a great deal of attention, this technique has only recently been advocated as a new approach to antialiasing for images. In particular, it has played an increasing role in ray tracing where the rays (point samples) are now stochastically distributed to perform a Monte Carlo evaluation of integrals in the rendering equation. This is called *distributed ray tracing* and has been used with great success in computer graphics to simulate motion blur, depth of field, penumbrae, gloss, and translucency [Cook 84, 86].

There are three common forms of stochastic sampling discussed in the literature: Poisson sampling, jittered sampling, and point-diffusion sampling.

6.3.2. Poisson Sampling

Poisson sampling uses an irregular sampling grid that is stochastically generated to yield a uniform distribution of sample points. This approximation to uniform sampling can be improved with the addition of a minimum-distance constraint between sample points. The result, known as the *Poisson-disk distribution*, has been suggested as the optimal sampling pattern to mask aliasing artifacts. This is motivated by evidence that the Poisson-disk distribution is found among the sparse retinal cells outside the foveal region of the eye. It has been suggested that this spatial organization serves to scatter aliasing into high-frequency random noise [Yellott 83].

A Poisson-disk sampling pattern and its Fourier Transform are shown in Fig. 6.4. Theoretical arguments can be given in favor of this sampling pattern, in terms of its spectral characteristics. An ideal sampling pattern, it is argued, should have a broad noisy spectrum with minimal low-frequency energy. A perfectly random pattern such as white noise is an example of such a signal where all frequency components have equal magnitude. This is equivalent to the

Missing 60-67

$$A = V_x^2 + V_y^2$$

$$B = -2(U_x V_x + U_y V_y)$$

$$C = U_x^2 + U_y^2$$

where

$$(U_x, V_x) = \left(\frac{\partial u}{\partial x}, \frac{\partial v}{\partial x} \right)$$

$$(U_y, V_y) = \left(\frac{\partial u}{\partial y}, \frac{\partial v}{\partial y} \right)$$

Once the ellipse parameters are determined, samples in the texture space may be tested for point-inclusion in the ellipse by incrementally computing Q for new values of u and v. In texture space the contours of Q are concentric ellipses. Points inside the ellipse satisfy $Q(u,v) < F$ for some threshold F.

$$F = U_x V_y - U_y V_x \tag{6.7}$$

This means that point-inclusion testing for ellipses can be done with one function evaluation rather than the four needed for quadrilaterals (four line equations).

If a point is found to satisfy $Q < F$, then the sample value is weighted with the appropriate lookup table entry. In screen space, the lookup table is indexed by r, the radius of the circle upon which the point lies. In texture space, though, Q is related to r^2. Rather than indexing with r, which would require us to compute $r = \sqrt{Q}$ at each pixel, the kernel values are stored into the lookup table so that they may be indexed by Q directly. Initializing the lookup table in this manner results in large computational efficiency. Thus, instead of determining which concentric circle the texture point maps onto in screen space, we determine which concentric ellipse the point lies upon in texture space and use it to index the appropriate weight in the lookup table.

Explicitly treating preimages as ellipses permits the function Q to take on a dual role: point-inclusion testing and lookup table index. The EWA is thereby able to achieve high-quality filtering at substantially lower cost. After all the points in the ellipse have been scanned, the sum of the weighted values is divided by the sum of the weights (for normalization) and assigned to the output pixel.

All direct convolution methods have a computational cost proportional to the number of input pixels accessed. This cost is exacerbated in [Feibush 80] and [Gangnet 82] where the collected input samples must be mapped into screen space to be weighted with the kernel. By achieving identical results without this costly mapping, the EWA is the most cost-effective high-quality filtering method.

6.5. PREFILTERING

The direct convolution methods described above impose minimal constraints on the filter area (quadrilateral, ellipse) and filter kernel (precomputed lookup table entries). Additional speedups are possible if further constraints are imposed. Pyramids and preintegrated tables are introduced to approximate the convolution integral with a constant number of accesses. This compares favorably against direct convolution which requires a large number of samples that grow proportionately to preimage area. As we shall see, though, the filter area will be limited to squares or rectangles, and the kernel will consist of a box filter. Subsequent advances have extended their use to more general cases with only marginal increases in cost.

6.5.1. Pyramids

Pyramids are multi-resolution data structures commonly used in image processing and computer vision. They are generated by successively bandlimiting and subsampling the original image to form a hierarchy of images at ever decreasing resolutions. The original image serves as the base of the pyramid, and its coarsest version resides at the apex. Thus, in a lower resolution version of the input, each pixel represents the average of some number of pixels in the higher resolution version.

The resolution of successive levels typically differ by a power of two. This means that successively coarser versions each have one quarter of the total number of pixels as their adjacent predecessors. The memory cost of this organization is modest: $1 + 1/4 + 1/16 + \cdots = 4/3$ times that needed for the original input. This requires only 33% more memory.

To filter a preimage, one of the pyramid levels is selected based on the size of its bounding square box. That level is then point sampled and assigned to the respective output pixel. The primary benefit of this approach is that the cost of the filter is constant, requiring the same number of pixel accesses independent of the filter size. This performance gain is the result of the filtering that took place while creating the pyramid. Furthermore, if preimage areas are adequately approximated by squares, the direct convolution methods amount to point sampling a pyramid. This approach was first applied to texture mapping in [Catmull 74] and described in [Dungan 78].

There are several problems with the use of pyramids. First, the appropriate pyramid level must be selected. A coarse level may yield excessive blur while the adjacent finer level may be responsible for aliasing due to insufficient bandlimiting. Second, preimages are constrained to be squares. This proves to be a crude approximation for elongated preimages. For example, when a surface is viewed obliquely the texture may be compressed along one dimension. Using the largest bounding square will include the contributions of many extraneous samples and result in excessive blur. These two issues were addressed by Williams and Crow, respectively, along with extensions proposed by other researchers.

Williams proposed a pyramid organization called *mip map* to store color images at multiple resolutions in a convenient memory organization [Williams 83]. The acronym "mip" stands for

"multum in parvo," a Latin phrase meaning "many things in a small place." The scheme supports trilinear interpolation, where both intra- and inter-level interpolation can be computed using three normalized coordinates: u, v, and d. Both u and v are spatial coordinates used to access points within a pyramid level. The d coordinate is used to index, and interpolate between, different levels of the pyramid. This is depicted in Fig. 6.7.

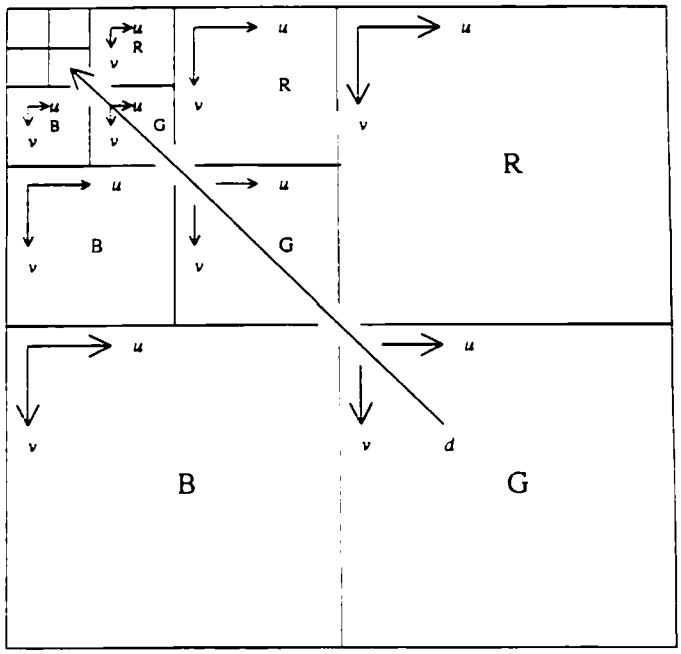


Figure 6.7: Mip Map memory organization.

The quadrants touching the east and south borders contain the original red, green, and blue (RGB) components of the color image. The remaining upper-left quadrant contains all the lower resolution copies of the original. The memory organization depicted in Fig. 6.7 clearly supports the earlier claim that memory cost is 4/3 times that required for the original input. Each level is shown indexed by the $[u,v,d]$ coordinate system, where d is shown slicing through the pyramid levels. Since corresponding points in different pyramid levels have indices which are related by some power of two, simple binary shifts can be used to access these points across the multi-resolution copies. This is a particularly attractive feature for hardware implementation.

The primary difference between mip maps and ordinary pyramids is the trilinear interpolation scheme possible with the $[u,v,d]$ coordinate system. By allowing a continuum of points to be accessed, mip maps are referred to as pyramidal parametric data structures. In Williams' implementation, a box filter (Fourier window) was used to create the mip maps, and a triangle filter (Bartlett window) was used to perform intra- and inter-level interpolation. The value of d must be chosen to balance the tradeoff between aliasing and blurring. Heckbert suggests

$$d^2 = \mathrm{MAX}\left[\left[\frac{\partial u}{\partial x}\right]^2 + \left[\frac{\partial v}{\partial x}\right]^2, \left[\frac{\partial u}{\partial y}\right]^2 + \left[\frac{\partial v}{\partial y}\right]^2\right] \tag{6.8}$$

where d is proportional to the span of the preimage area, and the partial derivatives can be

computed from the surface projection [Heckbert 83].

6.5.2. Summed-Area Tables

An alternative to pyramidal filtering was proposed by Crow in [Crow 84]. It extends the filtering possible in pyramids by allowing rectangular areas, oriented parallel to the coordinate axes, to be filtered in constant time. The central data structure is a preintegrated buffer of intensities, known as the *summed-area table*. This table is generated by computing a running total of the input intensities as the image is scanned along successive scanlines. For every position P in the table, we compute the sum of intensities of pixels contained in the rectangle between the origin and P. The sum of all intensities in any rectangular area of the input may easily be recovered by computing a sum and two differences of values taken from the table. For example, consider the rectangles R_0, R_1, R_2, and R shown in Fig. 6.8. The sum of intensities in rectangle R can be computed by considering the sum at $[x\,1, y\,1]$, and discarding the sums of rectangles R_0, R_1, and R_2. This corresponds to removing all area lying below and to the left of R. The resulting area is rectangle R, and its sum S is given as

$$S = T[x\,1, y\,1] - T[x\,1, y\,0] - T[x\,0, y\,1] + T[x\,0, y\,0] \tag{6.9}$$

where $T[x,y]$ is the value in the summed-area table indexed by coordinate pair $[x,y]$.

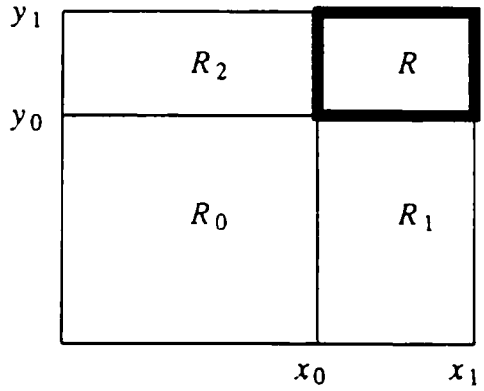


Figure 6.8: Summed-area table calculation.

Since $T[x\,1, y\,0]$ and $T[x\,0, y\,1]$ both contain R_0, the sum of R_0 was subtracted twice in Eq. (6.9). As a result, $T[x\,0, y\,0]$ was added back to restore the sum. Once S is determined it is divided by the area of the rectangle. This gives the average intensity over the rectangle, a process equivalent to filtering with a Fourier window (box filtering).

There are two problems with the use of summed-area tables. First, the filter area is restricted to rectangles. This is addressed in [Glassner 86], where an adaptive, iterative technique is proposed for obtaining arbitrary filter areas by removing extraneous regions from the rectangular bounding box. Second, the summed-area table is restricted to box filtering. This, of course, is attributed to the use of unweighted averages that keeps the algorithm simple. In [Perlin 85] and [Heckbert 86a], the summed-area table is generalized to support more sophisticated filtering by

repeated integration.

It is shown that by repeatedly integrating the summed-area table n times, it is possible to convolve an orthogonally oriented rectangular region with an n^{th}-order box filter (B-spline). Kernels for small n are shown in Fig. 5.8. The output value is computed by using $n + 1$ weighted samples from the preintegrated table. Since this result is independent of the size of the rectangular region, this method offers a great reduction in computation over that of direct convolution. Perlin called this a *selective image filter* because it allows each sample to be blurred by different amounts.

Repeated integration has rather high memory costs relative to pyramids. This is due to the number of bits necessary to retain accuracy in the large summations. Nevertheless, it allows us to filter rectangular or elliptical regions, rather than just squares as in pyramid techniques. Since pyramid and summed-area tables both require a setup time, they are best suited for input that is intended to be used repeatedly, i.e., a stationary background scene. In this manner, the initialization overhead can be amortized over each use. However, if the texture is only to be used once, the direct convolution methods raise a challenge to the cost-effectiveness offered by pyramids and summed-area tables.

6.6. FREQUENCY CLAMPING

The antialiasing methods described above all attempt to bandlimit the input by convolving input samples with a filter in the spatial domain. An alternative to this approach is to transform the input to the frequency domain, apply an appropriate low-pass filter to the spectrum, and then compute the inverse transform to display the bandlimited result. This was, in fact, already suggested as a viable technique for space-invariant filtering in which the low-pass filter can remain constant throughout the image. Norton, Rockwood, and Skolmoski explore this approach for space-variant filtering, where each pixel may require different bandlimiting to avoid aliasing [Norton 82].

The authors propose a simple technique for clamping, or suppressing, the offending high frequencies at each point in the image. This clamping function technique requires some a priori knowledge about the input image. In particular, the input should not be given as an array of samples but rather it should be represented by a Fourier series, i.e., a sum of bandlimited terms of increasing frequencies. When the frequency of a term exceeds the Nyquist rate at a given pixel, that term is forced to the local average value. This method has been successfully applied in a real-time visual system for flight simulators. It is used to solve the aliasing problem for textures of clouds and water, patterns which are convincingly generated using only a few low-order Fourier terms.

6.7. ANTIALIASED LINES AND TEXT

A large body of work has been directed towards efficient antialiasing methods for eliminating the jagged appearance of lines and text in raster images. These two applications have

attracted a lot of attention due to their practical importance in the ever growing workstation and personal computer markets. While *images* of lines and text can be handled with the algorithms described above, antialiasing techniques have been developed which embed the filtering process directly within the drawing routines.

Shaded (gray) pixels for lines can be generated, for example, with the use of a lookup table indexed by the distance between each pixel center and the line (or curve). Since arbitrary kernels can be stored in the lookup table at no extra cost, this approach shares the same merits as [Feibush 80]. Conveniently, the point-line distance can be computed incrementally by the same Bresenham algorithm used to determine which pixels must be turned on. This algorithm is described in [Gupta 81].

In [Turkowski 82], the CORDIC rotation algorithm is used to calculate the point-line distance necessary for indexing into the kernel lookup table. Other related papers describing the use of lookup tables and bitmaps for efficient antialiasing of lines and polygons can be found in [Pitteway 80], [Fiume 83], and [Abram 85]. Recent work in this area is described in [Chen 88]. For a description of recent advances in antialiased text, the reader is referred to [Naiman 87].

6.8. DISCUSSION

This section has reviewed methods to combat the aliasing artifacts that may surface upon performing geometric transformations on digital images. Aliasing becomes apparent when the mapping of input pixels onto the output is many-to-one. Sampling theory suggests theoretical limitations and provides insight into the solution. In the majority of cases, increasing display resolution is not a parameter that the user is free to adjust. Consequently, the approaches have dealt with bandlimiting the input so that it may conform to the available output resolution.

All contributions in this area fall into one of two categories: direct convolution and prefiltering. Direct convolution calls for increased sampling to accurately resolve the input preimage that maps onto the current output pixel. A low-pass filter is applied to these samples, generating a single bandlimited output value. This approach raises two issues: sampling techniques and efficient convolution. The first issue has been addressed by the work on regular and irregular sampling, including the recent advances in stochastic sampling. The second issue has been treated by algorithms which embed the filter kernels in lookup tables and provide fast access to the appropriate weights. Despite all possible optimizations, the computational complexity of this approach is inherently coupled with the number of samples taken over the preimage. Thus, larger preimages will incur higher sampling and filtering costs.

A cheaper approach providing lower quality results is obtained through prefiltering. By precomputing pyramids and summed-area tables, filtering is possible with only a constant number of computations, independent of the preimage area. Combining the partially filtered results contained in these data structures produces large performance gains. The cost, however, is in terms of constraints on the filter kernel and approximations to the preimage area. Designing efficient filtering techniques that support arbitrary preimage areas and filter kernels remains a great challenge. It is a subject that will continue to receive much attention.

7. SEPARABLE GEOMETRIC TRANSFORMATION ALGORITHMS

Separable geometric transformation algorithms, also known as *scanline algorithms*, spatially transform 2-D images by decomposing the mapping into a sequence of orthogonal 1-D transformations. This implies that the mapping function is separable, i.e., each dimension can be resampled independently of the other. For instance, 2-pass scanline algorithms typically apply the first pass to the image rows and the second pass to the columns. Although separable algorithms cannot handle all possible mapping functions, they can be shown to work particularly well for a wide class of common transformations, including affine and perspective mappings.

7.1. INTRODUCTION

Geometric transformations have traditionally been formulated as either forward or inverse mappings operating entirely in 2-D. Their advantages and drawbacks have already been described in section 3. We briefly restate these features in order to motivate the case for separable geometric transformation algorithms.

7.1.1. Forward Mapping

Forward mappings deposit input pixels into an output accumulator array. A distinction is made here based on the order in which pixels are fetched and stored. In forward mappings, the input arrives in scanline order (row by row) but the results are free to leave in any order, projecting into arbitrary areas in the output. In the general case, this means that no output pixel is guaranteed to be totally computed until the entire input has been scanned. Therefore, a full 2-D accumulator array must be retained throughout the duration of the mapping. Since the square input pixels project onto quadrilaterals at the output, costly intersection tests are needed to properly compute their overlap with the discrete output cells. Furthermore, an adaptive algorithm must be used to determine when supersampling is necessary in order to avoid blocky appearances upon one-to-many mappings.

7.1.2. Inverse Mapping

Inverse mappings are more commonly used to perform spatial transformations. By operating in scanline order at the output, square output pixels are projected onto arbitrary quadrilaterals. In this case, the projected areas lie in the input and are not generated in scanline order. Each preimage must be sampled and convolved with a low-pass filter to compute an intensity at the output. In section 6 we reviewed clever approaches to efficiently approximate this computation. While either forward or inverse mappings can be used to realize *arbitrary* mapping functions, there are many transformations that are well-suited for alternate techniques that yield further computational savings. These are the mappings that can be implemented with separable algorithms. They include affine and perspective mappings onto bivariate surfaces.

7.1.3. Separable Mapping

There are several advantages to decomposing a mapping into a series of 1-D transforms. First, the resampling problem is made simpler since reconstruction, area sampling, and filtering can now be done entirely in 1-D. Second, this lends itself naturally to digital hardware implementation. Note that no sophisticated digital filters are necessary to deal explicitly with the 2-D case. Third, the mapping can be done in scanline order both in scanning the input image and in producing the projected image. In this manner, an image may be processed in the same format in which it is stored in the framebuffer: rows and columns. This leads to efficient data access and large savings in I/O time. The approach is amenable to stream-processing techniques such as pipelining, and facilitates the design of hardware that works at real-time video rates.

7.2. 2-PASS TRANSFORMS

Consider a spatial transformation specified by forward mapping functions X and Y such that

$$[x, y] = T(u,v) = [X(u,v), Y(u,v)] \tag{7.1}$$

The transformation T is said to be *separable* if $T(u,v) = F(u)G(v)$. Since it is understood that G is applied only after F, the mapping $T(u,v)$ is said to be 2-*pass transformable*, or simply 2-*passable*. Functions F and G are called the 2-*pass functions*, each operating along different axes. Consequently, the forward mapping in Eq. (7.1) can be rewritten as a succession of two 1-D mappings F and G, the horizontal and vertical transformations, respectively.

7.2.1. Catmull and Smith, 1980

The most general presentation of the 2-pass technique appears in the seminal work described by Catmull and Smith in [Catmull 80]. This paper tackles the problem of mapping a 2-D image onto a 3-D surface and then projecting the result onto the 2-D screen for viewing. The contribution of this work lies in the decomposition of these steps into a sequence of computationally cheaper mapping operations. In particular, it is shown that a 2-D resampling problem can be replaced with two orthogonal 1-D resampling stages. This is depicted in Fig. 7.1.

7.2.1.1. First Pass

In the first pass, each horizontal scanline (row) is resampled according to spatial transformation $F(u)$, generating an intermediate image I in scanline order. All pixels in I have the same x-coordinates that they will assume in the final output; only their y-coordinates now remain to be computed. Since each scanline will generally have a different transformation, function $F(u)$ will usually differ from row to row. Consequently, F can be considered to be a function of both u and v. In fact, it is clear that mapping function F is identical to X, generating x-coordinates from points in the $[u,v]$ plane. To remain consistent with earlier notation, we rewrite $F(u,v)$ as $F_v(u)$ to denote that F is applied to horizontal scanlines, each having constant v. Therefore, the first pass is expressed as

$$[x, v] = [F_v(u), v] \tag{7.2}$$

where $F_v(u) = X(u,v)$. This relation maps all $[u,v]$ points onto the $[x,v]$ plane.

7.2.1.2. Second Pass

In the second pass, each vertical scanline (column) in I is resampled according to spatial transformation $G(v)$, generating the final image in scanline order. The second pass is more complicated than the first pass because the expression for G is often difficult to derive. This is due to the fact that we must invert $[x,v]$ to get $[u,v]$ so that G can directly access $Y(u,v)$. In doing so, new y-coordinates can be computed for each point in I.

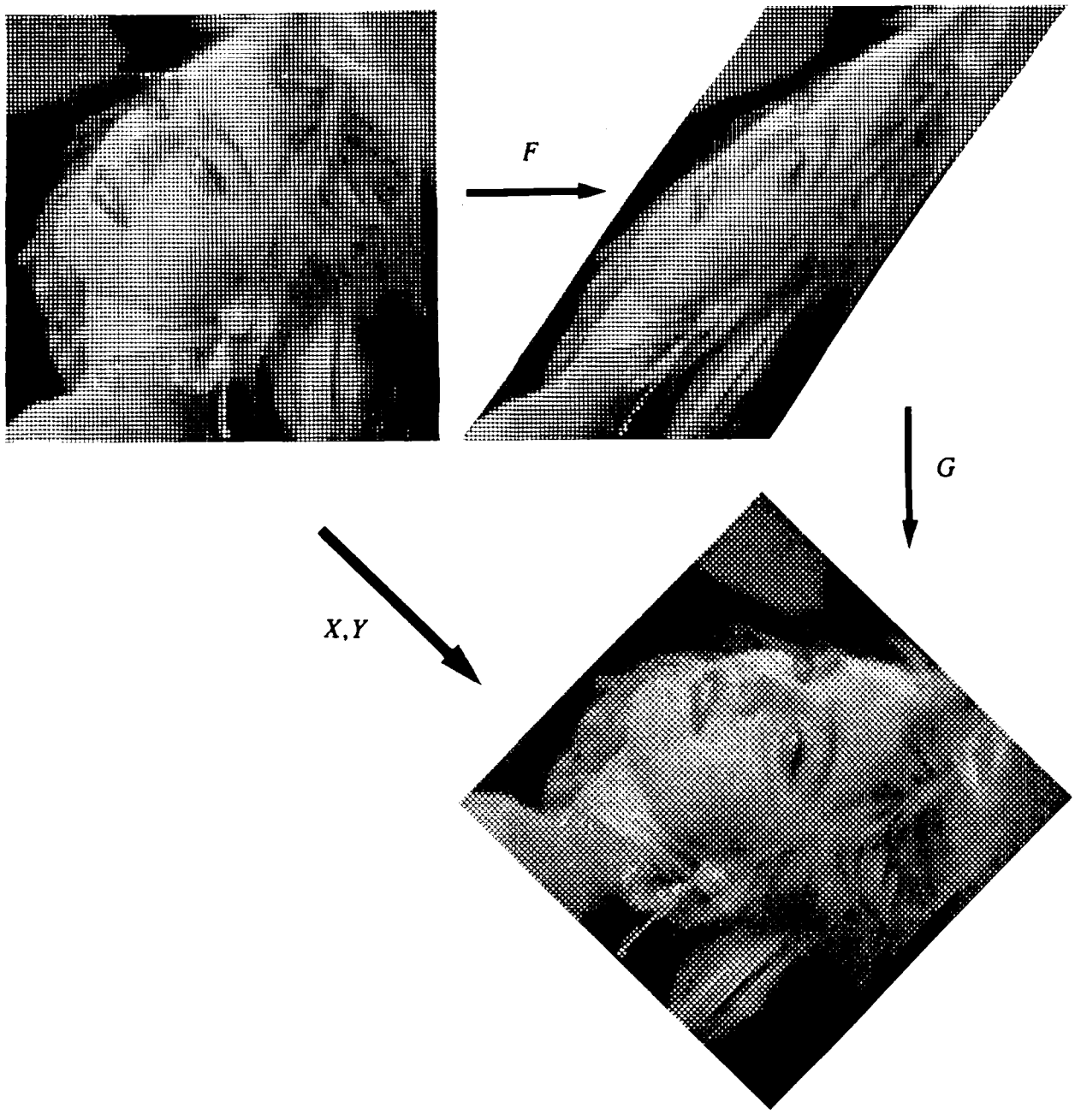


Figure 7.1: 2-pass geometric transformation.

Inverting f requires us to solve the equation $X(u,v) - \tilde{x} = 0$ for u to obtain $u = H_x(v)$ for vertical scanline (column) $\tilde{x}$. Note that $\tilde{x}$ contains all the pixels along the column at x. Function H, known as the *auxiliary function*, represents the u-coordinates of the inverse projection of $\tilde{x}$, the column we wish to resample. Thus, for every column in I, we compute $H_x(v)$ and use it together with the available v-coordinates to index into mapping function Y. This specifies the vertical spatial transformation necessary for resampling the column. The second pass is therefore expressed as

$$[x, y] = [x, G_x(v)] \tag{7.3}$$

where $G_x(v)$ refers to the evaluation of $G(x,v)$ along vertical scanlines with constant x. It is given by

$$G_x(v) = Y(H_x(v), v) \tag{7.4}$$

The relation in Eq. (7.3) maps all points in I from the $[x,v]$ plane onto the $[x,y]$ plane, the coordinate system of the final image.

7.2.1.3. 2-Pass Algorithm

In summary, the 2-pass algorithm has three steps. They correspond directly to the evaluation of scanline functions F and G, as well as the auxiliary function H.

1) The horizontal scanline function is defined as $F_v(u) = X(u,v)$. Each row is resampled according to this spatial transformation, yielding intermediate image I.

2) The auxiliary function $H_x(v)$ is derived for each vertical scanline $\tilde{x}$ in I. It is defined as the solution to $\tilde{x} = X(u,v)$ for u, if such a solution can be derived. Sometimes a closed form solution for H is not possible and numerical techniques such as the Newton-Raphson iteration method must be used. As we shall see later, computing H is the principal difficulty with the 2-pass algorithm.

3) Once $H_x(v)$ is determined, the second pass plugs it into the expression for $Y(u,v)$ to evaluate the target y-coordinates of all pixels in column x in image I. The vertical scanline function is defined as $G_x(v) = Y(H_x(v),v)$. Each column in I is resampled according to this spatial transformation, yielding the final image.

7.2.1.4. An Example: Rotation

The above procedure is demonstrated on the simple case of rotation. The rotation matrix is given as

$$[x, y] = [u, v] \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{7.5}$$

We want to transform every pixel in the original image in scanline order. If we scan a row by varying u and holding v constant, we immediately notice that the transformed points are not being generated in scanline order. This presents difficulties in antialiasing filtering and fails to achieve our goals of scanline input and output.

Alternatively, we may evaluate the scanline by holding v constant in the output as well, and only evaluating the new x values. This is given as

$$[x, v] = [u\cos\theta - v\sin\theta, v] \tag{7.6}$$

This results in a picture that is skewed and scaled along the horizontal scanlines.

The next step is to transform this intermediate result by holding x constant and computing y. However, the equation $y = u\sin\theta + v\cos\theta$ cannot be applied since the variable u is referenced instead of the available x. Therefore, it is first necessary to express u in terms of x. Recall that $x = u\cos\theta - v\sin\theta$, so

$$u = \frac{x + v\sin\theta}{\cos\theta} \tag{7.7}$$

Substituting this into $y = u\sin\theta + v\cos\theta$ yields

$$y = \frac{x\sin\theta + v}{\cos\theta} \tag{7.8}$$

The output picture is now generated by computing the y-coordinates of the pixels in the intermediate image, and resampling in vertical scanline order. This completes the 2-pass rotation. An example of this procedure for a 45° clockwise rotation is shown in Fig. 7.1.

The stages derived above are directly related to the general procedure described earlier. The three expressions for F, G, and H are explicitly listed below.

1) The first pass is defined by Eq. (7.6). In this case, $F_v(u) = u\cos\theta - v\sin\theta$.
2) The auxiliary function H is given in Eq. (7.7). It is the result of isolating u from the expression for x in mapping function $X(u,v)$. In this case, $H_x(v) = (x + v\sin\theta) / \cos\theta$.
3) The second pass then plugs $H_x(v)$ into the expression for $Y(u,v)$, yielding Eq. (7.8). In this case, $G_x(v) = (x\sin\theta + v) / \cos\theta$.

7.2.1.5. Bottleneck Problem

After completing the first pass, it is sometimes possible for the intermediate image to collapse into a narrow area. If this area is much less than that of the final image, then there is insufficient data left to accurately generate the final image in the second pass. This phenomenon, referred to as the *bottleneck problem* in [Catmull 80], is the result of a many-to-one mapping in the first pass followed by a one-to-many mapping in the second pass.

The bottleneck problem occurs, for instance, upon rotating an image clockwise by 90°. Since the top row will map to the rightmost column, all of the points in the scanline will collapse onto the rightmost point. Similar operations on all the other rows will yield a diagonal line as the intermediate image. No possible separable solution exists for this case when implemented in this order. This unfortunate result can be readily observed by noting that the $\cos\theta$ term in the denominator of Eq. (7.7) approaches zero as θ approaches 90°, thereby giving rise to an undeterminable inverse.

The solution to this problem lies in considering all the possible orders in which a separable algorithm can be implemented. Four variations are possible to generate the intermediate image:

1) Transform u first.
2) Transform v first.

3) Rotate the input image by 90° and transform u first.

4) Rotate the input image by 90° and transform v first.

In each case, the area of the intermediate image can be calculated. The method that produces the largest intermediate area is used to implement the transformation. If a 90° rotation is required, it is conveniently implemented by reading horizontal scanlines and writing them in vertical scanline order.

In our example, methods (3) and (4) will yield the correct result. This applies equally to rotation angles near 90°. For instance, an 87° rotation is best implemented by first rotating the image by 90° as noted above and then applying a −3° rotation using the 2-pass technique. These difficulties are resolved more naturally in a recent paper, described later, that demonstrates a separable technique for implementing arbitrary spatial lookup tables [Wolberg 88c].

7.2.1.6. Foldover Problem

The 2-pass algorithm is particularly well-suited for mapping images onto surfaces with closed form solutions to auxiliary function H. For instance, texture mapping onto rectangles that undergo perspective projection was first shown to be 2-passable in [Catmull 80]. This was independently discovered by Evans and Gabriel at Ampex Corporation where the result was implemented in hardware. The product was a real-time video effects generator called ADO (Ampex Digital Optics). It has met with great success in the television broadcasting industry where it is routinely used to map images onto rectangles in 3-space and move them around fluidly.

The process is more complicated for surfaces of higher order, e.g., bilinear, biquadratic, and bicubic patches. Since these surfaces are often nonplanar, they may be self-occluding. This has the effect of making F or G become multi-valued at points where the image folds upon itself, a problem known as *foldover*.

Foldover can occur in either of the two passes. In the vertical pass, the solution for *single* folds in G is to compute the depth of the vertical scanline endpoints. At each column, the endpoint which is furthest from the viewer is transformed first. The subsequent closer points along the vertical scanline will obscure the distant points and remain visible. Generating the image in this back-to-front order becomes more complicated for surfaces with more than one fold. In the general case, this becomes a hidden surface problem.

This problem can be avoided by restricting the mappings to be nonfolded, or single-valued. This simplification reduces the warp to one that resembles those used in remote sensing. In particular, it is akin to mapping images onto distorted planar grids where the spatial transformation is specified by a polynomial transformation. For instance, the nonfolded biquadratic patch can be shown to correct common lens aberrations such as the barrel and pincushion distortions depicted in Fig. 3.5.

Once we restrict patches to be nonfolded, only one solution is valid. This means that only one u on each horizontal scanline can map to the current vertical scanline. We cannot attempt to

use classic techniques to solve for H because n solutions may be obtained for an n^{th}-order surface patch. Instead, we find a solution $u = H_x(0)$ for the first horizontal scanline. Since we are assuming smooth surface patches, the next adjacent scanline can be expected to lie in the vicinity. The Newton-Raphson iteration method can be used to solve for $H_x(1)$ using the solution from $H_x(0)$ as a first approximation (starting value). This exploits the spatial coherence of surface elements to solve the inverse problem at hand.

The complexity of this problem can be reduced at the expense of additional memory. The need to evaluate H can be avoided altogether if we make use of earlier computations. Recall that the values of u that we now need in the second pass were already computed in the first pass. Thus, by introducing an auxiliary framebuffer to store these u's, H becomes available by trivial lookup table access.

In practice, there may be many u's mapping onto the unit interval between x and $x+1$. Since we are only interested in the inverse projection of integer values of x, we compute x for a dense set of equally spaced u's. When the integer values of two successive x's differ, we take one of the two following approaches.

1) Iterate on the interval of their projections u_i and u_{i+1}, until the computed x is an integer.

2) Approximate u by $u = u_i + a\,(u_{i+1} - u_i)$ where $a = x - x_i$.

The computed u is then stored in the auxiliary framebuffer at location x.

7.2.2. Fraser, Schowengerdt, and Briggs, 1985

Fraser, Schowengerdt, and Briggs demonstrate the 2-pass approach for geometric correction applications [Fraser 85]. They address the problem of accessing data along vertical scanlines. This issue becomes significant when processing large multichannel images such as Landsat multispectral data. Accessing pixels along columns can be inefficient and can lead to major performance degradation if the image cannot be entirely stored in main memory. Note that paging will also contribute to excessive time delays. Consequently, the intermediate image should be transposed, making rows become columns and columns become rows. This allows the second pass to operate along easily accessible rows.

A fast transposition algorithm is introduced that operates directly on a multichannel image, manipulating the data by a general 3-D permutation. The three dimensions include the row, column, and channel indices. The transposition algorithm uses a bit-reversed indexing scheme akin to that used in the Fast Fourier Transform (FFT) algorithm. Transposition is executed "in place," with no temporary buffers, by interchanging all elements having corresponding bit-reversed index pairs.

7.2.3. Fant, 1986

The central benefit of separable algorithms is the reduction in complexity of 1-D resampling algorithms. When the input is restricted to be one-dimensional, efficient solutions are made possible for the image reconstruction and antialiasing components of resampling. Fant presents a detailed description of such an algorithm that is well-suited for hardware implementation [Fant 86].

The process treats the input and output as streams of pixels that are consumed and generated at rates determined by the spatial mapping. The input is assumed to be mapped onto the output along a single direction, i.e., with no folds. As each input pixel arrives, it is weighted by its partial contribution to the current output pixel and integrated into an accumulator. In terms of the input and output streams, one of three conditions is possible:

1) The current input pixel is entirely consumed without completing an output pixel.
2) The input is entirely consumed while completing the output pixel.
3) The output pixel will be completed without entirely consuming the current input pixel. In this case, a new input value is interpolated from the neighboring input pixels at the position where the input was no longer consumed. It is used as the next element in the input stream.

If conditions (2) or (3) apply, the output computation is complete and the accumulator value is stored into the output array. The accumulator is then reset to zero in order to receive new input contributions for the next output pixel. Since the input is unidirectional, a one-element accumulator is sufficient. The process continues to cycle until the entire input stream is consumed.

The algorithm described in [Fant 86] is the principal 1-D resampling method used in separable transformations. It is demonstrated in the example below. Consider the input arrays shown in Fig. 7.2. The first array specifies the values of $F_v(u)$ for $u=0,1,...,4$. These represent new x-coordinates for their respective input pixels. For instance, the leftmost pixel will start at $x=.6$ and terminate at $x=2.3$. The next input pixel begins to influence the output at $x=2.3$ and proceeds until $x=3.2$. This continues until the last input pixel is consumed, filling the output between $x=3.3$ and $x=3.9$.

The second array specifies the distribution range that each input pixel assumes in the output. It is simply the difference between adjacent coordinates. Note that this requires the first array to have an additional element to define the length of the last input pixel. Large values correspond to stretching, and small values reflect compression. They determine the rate at which input is consumed to generate the output stream.

The input intensity values are given in the third array. Their contributions to the output stream is marked by connecting segments. The output values are labeled A_0 through A_3 and are defined below. For clarity, the following notation is used: interpolated input values are written within square brackets ([]), weights denoting contributions to output pixels are written within an extra level of parentheses, and input intensity values are printed in boldface.

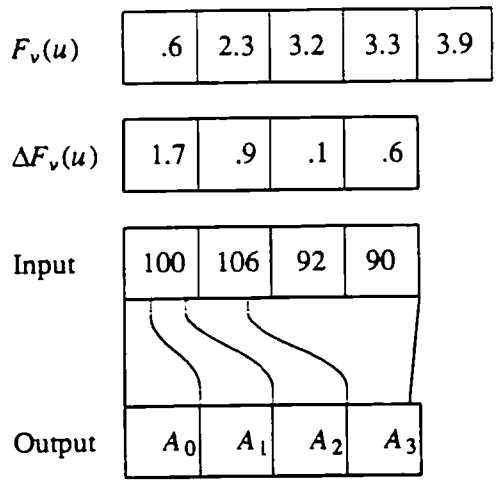


Figure 7.2: Resampling example.

$$A_0 = (\mathbf{100})((.4)) = 40$$

$$A_1 = \left[(\mathbf{100})\left[1-\frac{.4}{1.7}\right] + (\mathbf{106})\left[\frac{.4}{1.7}\right]\right]((1)) = 101$$

$$A_2 = \left[(\mathbf{100})\left[1-\frac{1.4}{1.7}\right] + (\mathbf{106})\left[\frac{1.4}{1.7}\right]\right]((.3)) + (\mathbf{106})((.7)) = 106$$

$$A_3 = \left[(\mathbf{106})\left[1-\frac{.7}{.9}\right] + (\mathbf{92})\left[\frac{.7}{.9}\right]\right]((.2)) + (\mathbf{92})((.1)) + (\mathbf{96})((.6)) = 82$$

The algorithm demonstrates both image reconstruction and antialiasing. When we are not positioned at pixel boundaries in the input stream, linear interpolation is used to reconstruct the discrete input. When more than one input element contributes to an output pixel, the weighted results are integrated in an accumulator to achieve antialiasing. These two cases are both represented in the above equations, as denoted by the expressions between square brackets and double parentheses, respectively.

7.2.4. Smith, 1987

The 2-pass algorithm has been shown to apply to a wide class of transformations of general interest. These mappings include the perspective projection of rectangles, bivariate patches, and superquadrics. Smith has discussed them in detail in [Smith 87].

The paper emphasizes the mathematical consequence of decomposing mapping functions X and Y into a sequence of F followed by G. Smith distinguishes X and Y as the *parallel warp*, and F and G as the *serial warp*, where *warp* refers to resampling. He shows that an n^{th}-order serial warp is equivalent to an $(n^2+n)^{th}$-order parallel warp. This higher-order polynomial mapping is quite different in form from the parallel polynomial warp. Smith also proves that the serial equivalent of a parallel warp is generally more complicated than a polynomial warp. This is due to the fact that the solution to H is typically not a polynomial.

7.3. ROTATION

The earliest separable geometric techniques can be traced back to the application of image rotation. Several of these algorithms are reviewed below.

7.3.1. Braccini and Marino, 1980

Braccini and Marino use a variant of the Bresenham line-drawing algorithm to rotate and shear images [Braccini 80]. While this does not qualify as a separable technique, it is included here because it is similar in spirit. In particular, the algorithm demonstrates the decomposition of the rotation matrix into simpler operations which can be efficiently computed.

Consider a straight line with slope n/m, where n and m are both integers. The line is rotated by an angle θ from the horizontal. The expressions for $\cos\theta$ and $\sin\theta$ can be given in terms of n and m as follows:

$$\cos\theta = \frac{m}{\sqrt{(n^2 + m^2)}} \tag{7.9}$$

$$\sin\theta = \frac{n}{\sqrt{(n^2 + m^2)}}$$

These terms can be substituted into the rotation matrix R to yield

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{7.10}$$

$$= \frac{m}{\sqrt{(n^2 + m^2)}} \begin{bmatrix} 1 & n/m \\ -(n/m) & 1 \end{bmatrix}$$

The matrix in Eq. (7.10) is equivalent to generating a digital line with slope n/m, an operation conveniently implemented by the Bresenham line-drawing algorithm [Foley 82]. The scale factor that is applied to the matrix amounts to resampling the input pixels, an operation which can be formulated in terms of the Bresenham algorithm as well. This is evident by noting that the distribution of n input pixels onto m output pixels is equivalent to drawing a line with slope n/m. The primary advantage of this formulation is that it exploits the computational benefits of the Bresenham algorithm: an incremental technique using only simple integer arithmetic computations.

The rotation algorithm is thereby implemented by depositing the input pixels along a digital line. Both the position of points along the line and the resampling of the input array are determined using the Bresenham algorithm. Due to the inherent jaggedness of digital lines, holes may appear between adjacent lines. Therefore, an extra pixel is drawn at each bend in the line to fill any gap that may otherwise be present. Clearly, this is a crude attempt to avoid holes, a problem inherent in this forward mapping approach.

The above procedure has been used for rotation and scale changes. It has been generalized

into a 2-pass technique to realize all affine transformations. This is achieved by using different angles and scale factors along each of the two image axes. Further nonlinear extensions are possible if the parameters are allowed to vary depending upon spatial position, e.g., space-variant mapping.

7.3.2. Weiman, 1980

Weiman describes a rotation algorithm based on cascading simpler 1-D scale and shear operations [Weiman 80]. These transformations are determined by decomposing the rotation matrix R into four submatrices.

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{7.11}$$
$$= \begin{bmatrix} 1 & \tan\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\sin\theta\cos\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & \cos\theta \end{bmatrix} \begin{bmatrix} 1/\cos\theta & 0 \\ 0 & 1 \end{bmatrix}$$

This formulation represents a separable algorithm in which 1-D scaling and shearing are performed along both image axes. As in the Braccini-Marino algorithm, an efficient line-drawing algorithm is used to resample the input pixels and perform shearing. Instead of using the incremental Bresenham algorithm, Weiman uses a periodic code algorithm devised by Rothstein. By averaging over all possible cyclic shifts in the code, the transformed image is shown to be properly filtered. In this respect, the Weiman algorithm is superior to that in [Braccini 80].

7.3.3. Paeth, 1986 / Tanaka, et. al., 1986

The most significant algorithm to be proposed for image rotation was proposed independently in [Paeth 86] and [Tanaka 86]. They demonstrate that rotation can be implemented by cascading three shear transformations.

$$R = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \tag{7.12}$$
$$= \begin{bmatrix} 1 & 0 \\ -\tan(\theta/2) & 1 \end{bmatrix} \begin{bmatrix} 1 & \sin\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ -\tan(\theta/2) & 1 \end{bmatrix}$$

The algorithm first skews the image along the horizontal direction by displacing each row. The result is then skewed along the vertical direction. Finally, an additional skew in the horizontal direction yields the rotated image. This sequence is illustrated in Fig. 7.3.

The primary advantage to the 3-pass shear transformation algorithm is that it avoids a costly scale operation. In this manner, it differs significantly from the 2-pass Catmull-Smith algorithm which combined scaling and shearing in each pass, and the 4-pass Weiman algorithm which further decomposed the scale/shear sequence. By not introducing a scale operation, the algorithm avoids complications in sampling, filtering, and the associated degradations. Note, for instance, that this method is not susceptible to the bottleneck problem.

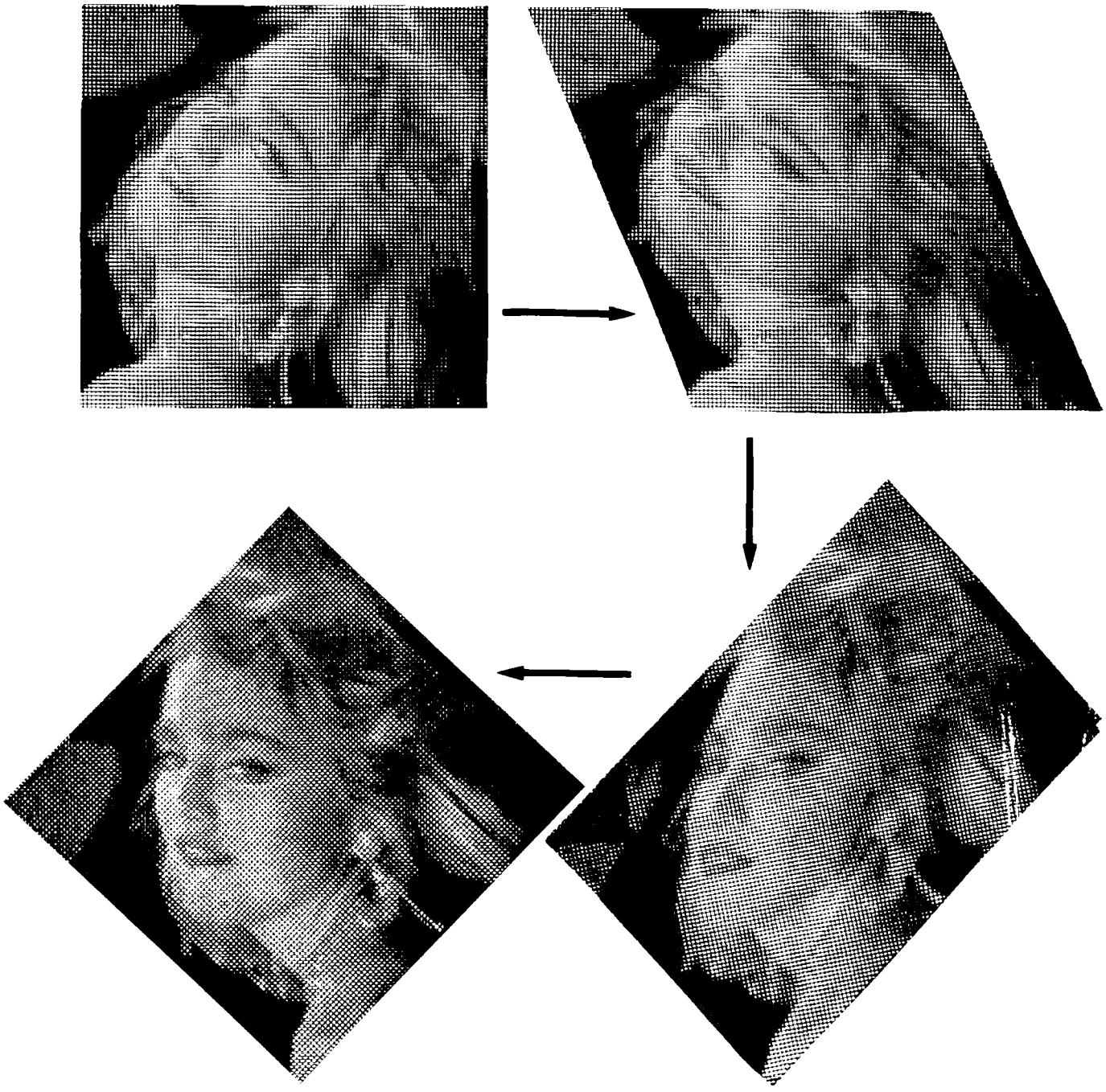


Figure 7.3: 3-pass shear transformation algorithm.

Simplifications are based in the particularly efficient means available to realize a shear transformation. The skewed output is the result of displacing each scanline differently. The displacement is generally not integral, but remains constant for all pixels on a given scanline. This allows intersection testing to be computed once for each scanline, noting that each input pixel can overlap at most two output pixels in the skewed image. The result is used to weigh each input intensity as it contributes to the output. Since the filter support is limited to two pixels, a simple box filter is adequate. Furthermore, the sum of the pixel intensities along any scanline can be shown to remain unchanged after the shear operation. Thus, the algorithm produces no

visible spatial-variant artifacts or holes. Finally, images on bitmap displays can be rotated using conventional hardware supporting *bitblt*, the bit block transfer operation useful for translations.

7.4. MORE SEPARABLE MAPPINGS

Additional separable geometric transformations are described in this section. They rely on the simplifications of 1-D processing to perform perspective projections, mappings among arbitrary planar shapes, and spatial lookup tables.

7.4.1. Perspective Projection: Robertson, 1987

The perspective projection of 3-D surfaces has been shown to be reducible into a series of fast 1-D resampling operations [Robertson 87]. In the traditional approach, this task has proved to be computationally expensive due to the problems in determining visibility and performing hidden-point removal. With the introduction of this algorithm, the problem can be decomposed into efficient separable components that can each be implemented at rates approaching real-time.

The procedure begins by rotating the image into alignment with the frontal (nearest) edge of the viewing window. Each horizontal scanline is then compressed so that all pixels which lie in a line of sight from the viewpoint are aligned into columns in the intermediate image. That is, each resulting column comprises a line of sight between the viewpoint and the surface.

Occlusion of a pixel can now only be due to another pixel in that column that lies closer to the viewer. This simplifies the perspective projection and hidden-pixel removal stages. These operations are performed along the vertical scanlines. By processing each column in back-to-front order, hidden-pixel removal is executed trivially.

Finally, the intermediate image undergoes a horizontal pass to apply the horizontal projection. This pass is complicated by the need to invert the previously applied horizontal compression. The difficulty arises since the image has already undergone hidden-pixel removal. Consequently, it is not directly known which surface point has been mapped to the current projected point. This can be uniquely determined only after additional calculations. The resulting image is the perspective transformation of the input, performed at rates which make real-time interactive manipulation possible.

7.4.2. Warping Among Arbitrary Planar Shapes: Wolberg, 1988

The advantages of 1-D resampling have been exploited for use in warping images among arbitrary planar shapes [Wolberg 88a]. The algorithm addresses the following inadequately solved problem: mapping between two images which are delimited by arbitrary, closed, planar, curves, e.g., hand-drawn curves.

Unlike many other problems treated in image processing or computer graphics, the stretching of an arbitrary shape onto another, and the associated mapping, is a problem not addressed in a tractable fashion in the literature. The lack of attention to this class of problems can be easily explained. In image processing, there is a well-defined 2-D rectilinear coordinate system.

Correcting for distortions amounts to mapping the four corners of a nonrectangular patch onto the four corners of a rectangular patch. In computer graphics, a parameterization exists for the 2-D image, the 3-D object, and the 2-D screen. Consequently, warping amounts to a change of coordinate system (2-D to 3-D) followed by a projection onto the 2-D screen. The problems considered in this work fail to meet the above properties. They are neither parameterized nor are they well suited for four-corner mapping.

The algorithm treats an image as a collection of interior layers. Informally, the layers are extracted in a manner similar to peeling an onion. A radial path emanates from each boundary point, crossing interior layers until the innermost layer, the skeleton, is reached. Assuming correspondences may be established between the boundary points of the source and target images, the warping problem is reduced to mapping between radial paths in both images. Note that the layers and the radial paths actually comprise a sampling grid.

This algorithm uses a generalization of polar coordinates. The extension lies in that radial paths are not restricted to terminate at a single point. Rather, a fully connected skeleton obtained from a thinning operation may serve as terminators of radial paths directed from the boundary. This permits the processing of arbitrary shapes.

The 1-D resampling operations are introduced in three stages. First, the radial paths in the source image must be resampled so that they all take on the same length. Then these normalized lists, which comprise the columns in our intermediate image, are resampled in the horizontal direction. This serves to put them in direct correspondence to their counterparts in the target image. Finally, each column is resampled to lengths that match those of the radial paths in the target image. In general, these lengths will vary due to asymmetric image boundaries.

The final image is generated by wrapping the resampled radial paths onto the target shape. This procedure is identical to the previous peeling operation except that values are now deposited onto the traversed pixels.

7.4.3. Spatial Lookup Tables: Wolberg and Boult, 1988

Sampling an arbitrary forward mapping function yields a 2-D *spatial lookup table*. This specifies the output coordinates for all input pixels. A separable technique to implement this utility is of great practical importance. The chief complications arise from the bottleneck and foldover problems described earlier. These difficulties are addressed in [Wolberg 88c].

Wolberg and Boult propose a 2-pass algorithm akin to that in [Catmull 80], with additional memory and data structures introduced to guard against the bottleneck and foldover problems. The solution lies in careful implementation of function F. Rather than blindly calculating the intermediate image without regard for any possible many-to-one mappings, the filter that is used to perform antialiasing also determines when aliasing is present. While integrating pixels into a single-element accumulator array, their y-coordinates are inspected. If they do not all lie within a single pixel in the final image, then the bottleneck problem is present. This is an accurate measure of bottleneck superior to that in [Catmull 80]. For instance, they suggest that the area

of the intermediate image be used to detect bottleneck problems. This, however, is a global measure which may fail to highlight severe compression in local areas. Although this proves to be satisfactory for mappings onto low-order surface patches, it is inadequate for arbitrary mappings.

The points subjected to the bottleneck are stored in a list and their processing is deferred until the second pass. Thus, the intermediate image actually consists of a combination of properly filtered pixels and pointers to lists of pixels. The second pass then processes this data along vertical scanlines. This approach is shown to resolve the foldover problem as well.

7.5. DISCUSSION

Scanline algorithms all share a common theme: simple interpolation, antialiasing, and data access are made possible when operating along a single dimension. Using a 2-pass transform as an example, the first pass represents a forward mapping. Since the data is assumed to be unidirectional, a single-element accumulator is sufficient for filtering purposes. This is in contrast to a full 2-D accumulator array for standard forward mappings. The second pass is actually a hybrid mapping function, requiring an inverse mapping to allow a new forward mapping to proceed. Namely, auxiliary function H must be solved before G, the second-pass forward mapping, can be evaluated.

A benefit of this approach is that clipping along one dimension is possible. For instance, there is no need to compute H for a particular column that is known in advance to be clipped. This results in some timesavings. The principal difficulty, however, is the bottleneck problem which exists as a form of aliasing. This is avoided in some applications, such as rotation, where it has been shown that no scaling is necessary in any of the 1-D passes. More generally, special attention must be provided to counteract this degradation. This has been demonstrated for the case of arbitrary spatial lookup tables.

8. SUMMARY

Geometric transformation techniques for digital images are a subject of widespread interest. They are of practical importance to the remote sensing, medical imaging, computer vision, and computer graphics communities. Typical applications can be grouped into two classes: geometric correction and geometric distortion. Geometric correction refers to distortion compensation of imaging sensors, decalibration, and geometric normalization. It is applied to remote sensing, medical imaging, and computer vision. Geometric distortion refers to texture mapping, a powerful computer graphics tool for realistic image synthesis.

All geometric transformations have three principal components: spatial transformation, image resampling, and antialiasing. They have each received considerable attention. However, due to domain-dependent assumptions and constraints, they have rarely received uniform treatment. For instance, in remote sensing work where there is usually no severe scale change, image reconstruction is more sophisticated than antialiasing. However, in computer graphics where there is often more dramatic image compression, antialiasing plays a more significant role. This has served to obscure the single underlying set of principles that govern all geometric transformations for digital images. The goal of this paper has been to survey the numerous contributions to this field, with special emphasis given to the presentation of a single coherent framework.

Various formulations of spatial transformations have been reviewed, including affine and perspective mappings, polynomial transformations, piecewise polynomial transformations, and four-corner mapping. The role of these mapping functions in geometric correction and geometric distortion was discussed. For instance, polynomial transformations were introduced to extend the class of mappings beyond affine transformations. Thus, in addition to performing the common translate, scale, rotate, and shear operations, it is possible to invert pincushion and barrel distortions. For more local control, piecewise polynomial transformations are widespread. It was shown that by establishing several correspondence points, an entire mapping function can be generated through the use of local interpolants. This is actually a surface reconstruction problem. There continues to be a great deal of activity in this area as evidenced by recent papers on multigrid relaxation algorithms to iteratively propagate constraints throughout the surface. Consequently, the tools of this field of mathematics can be applied directly to spatial transformations.

Image resampling has been shown to primarily consist of image reconstruction, an interpolation process. Various interpolation methods have been reviewed, including the (truncated) sinc function, nearest neighbor, linear interpolation, cubic convolution, 2-parameter cubic filters, and cubic splines. By analyzing the responses of their filter kernels in the frequency domain, a comparison of interpolation methods was presented. In particular, the quality of interpolation is assessed by examining the performance of the interpolation kernel in the passbands and stopbands. A review of sampling theory has been included to provide the necessary background for a comprehensive understanding of image resampling and antialiasing.

Antialiasing has recently attracted much attention in the computer graphics community. The earliest antialiasing algorithms were restrictive in terms of the preimage shape and filter

kernel that they supported. For example, box filtering over rectangular preimages were common. Later developments obtained major performance gains by retaining these restrictions but permitting the number of computations to be independent of the preimage area. Subsequent improvements offered fewer restrictions at lower cost. In these instances the preimage areas were extended to ellipses and the filter kernels, now stored in lookup tables, were allowed to be arbitrary. The design of efficient filters that operate over an arbitrary input area and accommodate arbitrary filter kernels remains a great challenge.

Development of superior filters used another line of attack: advanced sampling strategies. They include supersampling, adaptive sampling, and stochastic sampling. These techniques draw upon recent results on perception and the human visual system. The suggested sampling patterns that are derived from the blue noise criteria offer promising results. Their critics, however, point to the excessive sampling densities required to reduce noise levels to unobjectionable limits. Determining minimum sampling densities which satisfy some subjective criteria requires additional work.

The final section has discussed various separable algorithms introduced to obtain large performance gains. These algorithms have been shown to apply over a wide range of transformations, including perspective projection of rectangles, bivariate patches, and superquadrics. Hardware products, such as the Ampex ADO and Quantel Mirage, are based on these techniques to produce real-time video effects for the television industry. Recent progress has been made in scanline algorithms that avoid the bottleneck problem, a degradation that is particular to the separable method. These modifications have been demonstrated on the special case of rotation and the arbitrary case of spatial lookup tables.

Despite the relatively short history of geometric transformation techniques for digital images, a great deal of progress has been made. This has been accelerated within the last decade through the proliferation of fast and cost-effective digital hardware. Algorithms which were too costly to consider in the early development of this area, are either commonplace or are receiving increased attention. Future work in the areas of reconstruction and antialiasing will most likely integrate models of the human visual system to achieve higher quality images. This has been demonstrated in a recent study of a family of filters defined by piecewise cubic polynomials, as well as recent work in stochastic sampling. Related problems that deserve attention include new adaptive filtering techniques, irregular sampling algorithms, and reconstruction from irregular samples. In addition, work remains to be done on efficient separable schemes to integrate sophisticated reconstruction and antialiasing filters into a system supporting more general spatial transformations. This is likely to have great impact on the various diverse communities which have contributed to this broad area.

9. ACKNOWLEDGEMENTS

I wish to thank Profs. Terry Boult and Steven Feiner for their careful review of this paper, and for their contributions to its content and form. I also thank Prof. Gerald Maguire for many insightful discussions. The author was supported by an NSF Graduate Fellowship.

10. REFERENCES

[Abdou 82] Abdou, I.E. and K.Y. Wong, "Analysis of Linear Interpolation Schemes for Bi-Level Image Applications," *IBM J. Res. Develop.*, vol. 26, no. 6, pp. 667-680, November 1982.

[Abram 85] Abram, G., L. Westover, and T. Whitted, "Efficient Alias-free Rendering using Bit-masks and Look-up Tables," *Computer Graphics*, (SIGGRAPH '85 Proceedings), vol. 19, no. 3, pp. 53-59, July 1985.

[Akima 78] Akima, H., "A Method of Bivariate Interpolation and Smooth Surface Fitting for Irregularly Distributed Data Points," *ACM Trans. Math. Software*, vol. 4, pp. 148-159, 1978.

[Akima 84] Akima, H., "On Estimating Partial Derivatives for Bivariate Interpolation of Scattered Data," *Rocky Mountain J. Math.*, vol. 14, pp. 41-52, 1984.

[Andrews 76] Andrews, H.C. and C.L. Patterson, "Digital Interpolation of Discrete Images," *IEEE Trans. Computers*, vol. C-25, pp. 196-202, 1977.

[Barnhill 77] Barnhill, R.E., "Representation and Approximation of Surfaces," *Mathematical Software III*, Ed. by J.R. Rice, Academic Press, London, pp. 69-120, 1977.

[Bernstein 71] Bernstein, R. and H. Silverman, "Digital Techniques For Earth Resource Image Data Processing," *Proc. Amer. Inst. Aeronautics and Astronautics 8th Annu. Meeting*, vol. C21, AIAA paper no. 71-978, October 1971.

[Bernstein 76] Bernstein, R., "Digital Image Processing of Earth Observation Sensor Data," *IBM J. Res. Develop.*, vol. 20, pp. 40-57, January 1976.

[Bizais 83] Bizais, Y., I.G. Zubal, R.W. Rowe, G.W. Bennett, and A.B. Brill, "2-D Fitting and Interpolation Applied to Image Distortion Analysis," *Pictorial Data Analysis*, Ed. by R.M. Haralick, NATO ASI Series, vol. F4, 1983, pp. 321-333.

[Blinn 76] Blinn, J.F and M.E. Newell, "Texture and Reflection in Computer Generated Images," *Comm. ACM*, vol. 19, no. 10, pp. 542-547, October 1976.

[Braccini 80] Braccini, C., and G. Marino, "Fast Geometrical Manipulations of Digital Images," *Computer Graphics and Image Processing*, vol. 13, pp. 127-141, 1980.

[Butler 87] Butler, D.A. and P.K. Pierson, "Correcting Distortion in Digital Images," *Proc. Vision* '87, pp. 6-49-6-69, June 1987.

[Caelli 81] Caelli, T., *Visual Perception: Theory and Practice*, Pergamon Press, Oxford, 1981.

[Catmull 74] Catmull, E., "A Subdivision Algorithm for Computer Display of Curved Surfaces," Ph.D. Dissertation, Dept. of Computer Science, University of Utah, Tech. Report UTEC-CSc-74-133, December 1974.

[Catmull 80] Catmull, E. and A.R. Smith, "3-D Transformations of Images in Scanline

Order," *Computer Graphics*, (SIGGRAPH '80 Proceedings), vol. 14, no. 3, pp. 279-285, July 1980.

[Chen 88] Chen, Y.-T., P.D. Fisher, and M.D. Olinger, "The Application of Area Antialiasing on Raster Image Displays," *Graphics Interface* '88, pp. 211-216, 1988.

[Clough 65] Clough, R.W. and J.L. Tocher, "Finite Element Stiffness Matrices for Analysis of Plates in Bending," *Proc. Conf. on Matrix Methods in Structural Mechanics*, pp. 515-545, 1965.

[Cook 84] Cook, R.L., T. Porter, and L. Carpenter, "Distributed Ray Tracing," *Computer Graphics*, (SIGGRAPH '84 Proceedings), vol. 18, no. 3, pp. 137-145, July 1984.

[Cook 86] Cook, R.L., "Stochastic Sampling in Computer Graphics," *ACM Trans. on Graphics*, vol. 5, no. 1, pp. 51-72, January 1986.

[Crow 77] Crow, F.C., "The Aliasing Problem in Computer-Generated Shaded Images," *Comm. ACM*, vol. 20, no. 11, pp. 799-805, November 1977.

[Crow 81] Crow, F.C., "A Comparison of Antialiasing Techniques," *IEEE Computer Graphics and Applications*, vol. 1, no. 1, pp. 40-48, January 1981.

[Crow 84] Crow, F.C., "Summed-Area Tables for Texture Mapping," *Computer Graphics*, (SIGGRAPH '84 Proceedings), vol. 18, no. 3, pp. 207-212, July 1984.

[Cutrona 60] Cutrona, L.J., E.N. Leith, C.J. Palermo, and L.J. Porcello, "Optical Data Processing and Filtering Systems," *IRE Trans. Inf. Theory*, vol. IT-6, pp. 386-400, 1960.

[de Boor 78] de Boor, Carl, *A Practical Guide to Splines*, Springer-Verlag, NY, 1978.

[Dippe 85] Dippe, M.A.Z and E.H. Wold, "Antialiasing Through Stochastic Sampling," *Computer Graphics*, (SIGGRAPH '85 Proceedings), vol. 19, no. 3, pp. 69-78, July 1985.

[Dungan 78] Dungan, W. Jr., A. Stenger, and G. Sutty, "Texture Tile Considerations for Raster Graphics," *Computer Graphics*, (SIGGRAPH '78 Proceedings), vol. 12, no. 3, pp. 130-134, August 1978.

[Fant 86] Fant, K.M., "A Nonaliasing, Real-Time Spatial Transform Technique," *IEEE Computer Graphics and Applications*, vol. 6, no. 1, pp. 71-80, January 1986.

[Feibush 80] Feibush, E.A., M. Levoy, and R.L. Cook, "Synthetic Texturing Using Digital Filters," *Computer Graphics*, (SIGGRAPH '80 Proceedings), vol. 14, no. 3, pp. 294-301, July 1980.

[Fiume 83] Fiume, E. and A. Fournier, "A Parallel Scan Conversion Algorithm with Anti-Aliasing for a General-Purpose Ultracomputer," *Computer Graphics*, (SIGGRAPH '83 Proceedings), vol. 17, no. 3, pp. 141-150, July 1983.

[Floyd 75] Floyd, R.W and L. Steinberg, "Adaptive Algorithm for Spatial Grey Scale," *SID Intl. Sym. Dig. Tech. Papers*, pp. 36-37, 1975.

[Foley 82] Foley, J.D. and A. Van Dam, *Fundamentals of Interactive Computer Graphics*, Addison-Wesley, Reading, MA, 1982.

[Franke 79] Franke, R., "A Critical Comparison of Some Methods for Interpolation of Scattered Data," *Naval Postgraduate School Technical Report*, NPS-53-79-003, 1979.

[Fraser 85] Fraser, D., R.A. Schowengerdt, and I. Briggs, "Rectification of Multichannel Images in Mass Storage Using Image Transposition," *Computer Vision, Graphics, and Image Processing*, vol. 29, no. 1, pp. 23-36, January 1985.

[Gangnet 82] Gangnet, M., D. Perny, P. Coueignoux, "Perspective Mapping of Planar Textures," *Eurographics* '82,, pp., 57-71, 1982.

[Glassner 86] Glassner, A., "Adaptive Precision in Texture Mapping," (SIGGRAPH '86 Proceedings), vol. 20, no. 4, pp. 297-306, July 1986.

[Goshtasby 86] Goshtasby, A., "Piecewise Linear Mapping Functions for Image Registration," *Pattern Recognition*, vol. 19, no. 6, pp. 459-466, 1986.

[Goshtasby 87] Goshtasby, A., "Piecewise Cubic Mapping Functions for Image Registration," *Pattern Recognition*, vol. 20, no. 5, pp. 525-533, 1987.

[Green 78] Green, P.J. and R. Sibson, "Computing Dirichlet Tessellations in the Plane," *Computer Journal*, vol. 21, pp. 168-173, 1978.

[Greene 86] Greene, N. and P.S. Heckbert, "Creating Raster Omnimax Images from Multiple Perspective Views Using the Elliptical Weighted Average Filter," *IEEE Computer Graphics and Applications*, vol. 6, no. 6, pp. 21-27, June 1986.

[Gupta 81] Gupta, S. and R.F. Sproull, "Filtering Edges for Gray-Scale Displays," *Computer Graphics*, (SIGGRAPH '81 Proceedings), vol. 15, no. 3, pp. 1-5, August 1981.

[Haralick 76] Haralick, R.M., "Automatic Remote Sensor Image Processing," *Topics in Applied Physics, Vol.* 11: *Digital Picture Analysis*, Ed. by A. Rosenfeld, Springer-Verlag, 1976, pp. 5-63.

[Heckbert 83] Heckbert, P., "Texture Mapping Polygons in Perspective," Tech. Memo No. 13, NYIT Computer Graphics Lab, April 1983.

[Heckbert 86a] Heckbert, P., "Filtering by Repeated Filtering," *Computer Graphics*, (SIGGRAPH '86 Proceedings), vol. 20, no. 4, pp. 315-321, July 1986.

[Heckbert 86b] Heckbert, P., "Survey of Texture Mapping," *IEEE Computer Graphics and Applications*, vol. 6, no. 11, pp. 56-67, November 1986.

[Horner 87] Horner, J.L., *Optical Signal Processing*, Academic Press, NY, 1987.

[Hou 78] Hou, H.S. and H.C. Andrews, "Cubic Splines for Image Interpolation and Digital Filtering," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-26, pp. 508-517, 1987.

[Jarvis 76] Jarvis, J.F., C.N. Judice, and W.H. Ninke, "A Survey of Techniques for the Display of Continuous-Tone Pictures on Bilevel Displays," *Computer Graphics and Image Processing*, vol. 5, pp. 13-40, 1976.

[Kajiya 86] Kajiya, J.T., "The Rendering Equation," *Computer Graphics*, (SIGGRAPH '86 Proceedings), vol. 20, no. 4, pp. 143-150, July 1986.

[Keys 81] Keys, R.G., "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-29, pp. 1153-1160, 1981.

[Klucewicz 78] Klucewicz, I.M., "A Piecewise C^1 Interpolant to Arbitrarily Spaced Data," *Computer Graphics and Image Processing*, vol. 8, pp. 92-112, 1978.

[Knowlton 72] Knowlton, K. and L. Harmon, "Computer-Produced Grey Scales," *Computer Graphics and Image Processing*, vol. 1, pp. 1-20, 1972.

[Lawson 77] Lawson, C.L., "Software for C^1 Surface Interpolation," *Mathematical Software III*, Ed. by J.R. Rice, Academic Press, London, pp. 161-194, 1977.

[Leckie 80] Leckie, D.G., "Use of Polynomial Transformations for Registration of Airborne Digital Line Scan Images," *14th Intl. Sym. Remote Sensing of Environment*, pp. 635-641, 1980.

[Lee 80] Lee, D.T. and B.J. Schachter, "Two Algorithms for Constructing a Delaunay Triangulation," *Intl. J. Computer Info. Sci.*, vol. 9, pp. 219-242, 1980.

[Lee 83] Lee, C.-H., "Restoring Spline Interpolation of CT Images," *IEEE Trans. Medical Imaging*, vol. MI-2, no. 3, pp. 142-149, September 1983.

[Lee 85] Lee, M., R.A. Redner, and S.P. Uselton, "Statistically Optimized Sampling for Distributed Ray Tracing," *Computer Graphics*, (SIGGRAPH '85 Proceedings), vol. 19, no. 3, pp. 61-67, July 1985.

[Lee 87] Lee, D., T. Pavlidis, and G.W. Wasilkowski, "A Note on the Trade-off Between Sampling and Quantization in Signal Processing," *J. Complexity*, vol. 3, no. 4, pp. 359-371, December 1987.

[Lillestrand 72] Lillestrand, R.L., "Techniques for Changes in Urban Development from Aerial Photography," *IEEE Trans. Computers*, vol. C-21, pp. 546-549, 1972.

[Lim 69] Lim, J.O., "Design of Dither Waveforms for Quantized Visual Signals," *Bell System Tech. J.*, vol. 48, pp. 2555-2582, 1969.

[Lim 77] Lim, J.O., "Digital Coding of Color Video Signals — A Review," *IEEE Trans. Comm.*, vol. COMM-25, no. 11, pp. 1349-1382, November 1977.

[Maeland 88] Maeland, E., "On the Comparison of Interpolation Methods," *IEEE Trans. Medical Imaging*, vol. MI-7, no. 3, pp. 213-217, September 1988.

[Markarian 71] Markarian, H., R. Bernstein, D.G. Ferneyhough, L.E. Gregg, and F.S. Sharp, "Implementation of Digital Techniques for Correcting High Resolution Images," *Proc. Amer. Inst. Aeronautics and Astronautics 8th Annu. Meeting*,

vol. C21, AIAA paper no. 71-326, pp. 285-304, October 1971.

[Mertz 34] Mertz, P. and F. Gray, "A Theory of Scanning and its Relation to the Characteristics of the Transmitted Signal in Telephotography and Television," *Bell System Tech. J.*, vol. 13, pp. 464-515, July 1934.

[Mitchell 87] Mitchell, D., "Generating Antialiased Images at Low Sampling Densities," *Computer Graphics*, (SIGGRAPH '87 Proceedings), vol. 21, no. 4, pp. 65-72, July 1987.

[Mitchell 88] Mitchell, D. and A. Netravali, "Reconstruction Filters in Computer Graphics," *Computer Graphics*, (SIGGRAPH '88 Proceedings), vol. 22, no. 4, pp. 221-228, August 1988.

[Nack 77] Nack, M.L., "Rectification and Registration of Digital Images and the Effect of Cloud Detection," *Proc. Machine Processing of Remotely Sensed Data*, pp. 12-23, 1977.

[Naiman 87] Naiman, A. and A. Fournier, "Rectangular Convolution for Fast Filtering of Characters," *Computer Graphics*, (SIGGRAPH '87 Proceedings), vol. 21, no. 4, pp. 233-242, July 1987.

[Netravali 80] Netravali, A.N. and J.O. Limb, "Picture Coding: A Review," *Proc. IEEE* vol. 68, pp. 366-406, 1980.

[Nielson 83] Nielson, G.M. and R. Franke, "Surface Construction Based Upon Triangulations," *Surfaces in Computer Aided Geometric Design*, Ed. by R.E. Barnhill and W. Boehm, North-Holland Publishing Co., Amsterdam, pp. 163-177, 1983.

[Norton 82] 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.

[Oka 87] Oka, M., K. Tsutsui, A. Ohba, Y. Kurauchi, and T. Tagao, "Real-Time Manipulation of Texture-Mapped Surfaces," *Computer Graphics*, (SIGGRAPH '87 Proceedings), vol. 21, no. 4, pp. 181-188, July 1987.

[Paeth 86] Paeth, A.W., "A Fast Algorithm for General Raster Rotation," *Graphics Interface* '86, pp. 77-81, May 1986.

[Park 83] Park, S.K. and R.A. Schowengerdt, "Image Reconstruction by Parametric Cubic Convolution," *Computer Vision, Graphics, and Image Processing*, vol. 23, pp. 258-272, 1983.

[Parker 83] Parker, J.A., R.V. Kenyon, and D.E. Troxel, "Comparison of Interpolating Methods for Image Resampling," *IEEE Trans. Medical Imaging*, vol. MI-2, no. 1, pp. 31-39, March 1983.

[Pavlidis 82] Pavlidis, T., *Algorithms for Graphics and Image Procesing*, Computer Science Press, Rockville, MD, 1982.

[Percell 76] Percell, P., "On Cubic and Quartic Clough-Tocher Finite Elements," *SIAM J. Numerical Analysis*, vol. 13, pp. 100-103, 1976.

[Perlin 85] Perlin, K., "Course Notes," *SIGGRAPH '85 State of the Art in Image Synthesis seminar notes*, July 1985.

[Pitteway 80] Pitteway, M.L.V and D.J. Watkinson, "Bresenham's Algorithm with Grey Scale," *Comm. ACM*, vol. 23, no. 11, pp. 625-626, November 1980.

[Powell 77] Powell, M.J.D. and M.A Sabin, "Piecewise Quadratic Approximations on Triangles," *ACM Trans. Mathematical Software*, vol. 3, pp. 316-325, 1977.

[Press 88] Press, W.H., Flannery, B.P., Teukolsky, S.A., and Vetterling, W.T., *Numerical Recipes in C*, Cambridge University Press, Cambridge, 1988.

[Ratzel 80] Ratzel, J.N., "The Discrete Representation of Spatially Continuous Images," Ph.D. Dissertation, Massachusetts Institute of Technology, Cambridge, 1980.

[Rifman 74] Rifman, S.S. and D.M. McKinnon, "Evaluation of Digital Correction Techiques for ERTS Images — Final Report, Report 20634-6003-TU-00, TRW Systems, Redondo Beach, Calif., July 1974.

[Robertson 87] Robertson, P.K., "Fast Perspective Views of Images Using One-Dimensional Operations," *IEEE Computer Graphics and Applications*, vol. 7, no. 2, pp. 47-56, February 1987.

[Rogers 76] Rogers, D.F. and J.A. Adams, *Mathematical Elements for Computer Graphics*, McGraw-Hill, NY, 1976.

[Rosenfeld 82] Rosenfeld, A. and A.C. Kak, *Digital Picture Processing, Vol.* 2, Academic Press, NY, 1982.

[Sakrison 77] Sakrison, D.J., "On the Role of the Observer and a Distortion Measure in Image Transmission," *IEEE Trans. Comm.*, vol. COMM-25, no. 11, pp. 1251-1267, November 1977.

[Schafer 73] Schafer, R.W. and L.R. Rabiner, "A Digital Signal Processing Approach to Interpolation," *Proc. IEEE*, vol. 61, pp. 692-702, June 1973.

[Schreiber 85] Schreiber, W.F. and D.E. Troxel, "Transformation Between Continuous and Discrete Representations of Images: A Perceptual Approach," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. PAMI-7, no. 2, pp. 178-186, March 1985.

[Shannon 48] Shannon, C., "A Mathematical Theory of Communication," *Bell System Tech. J.*, vol. 27, pp. 379-423, July 1948, and vol. 27, pp. 623-656, October 1948.

[Shannon 49] Shannon, C., "Communication in the Presence of Noise," *Proc. Inst. Radio Eng.*, vol. 37, no. 1, pp. 10-21, January 1949.

[Simon 75] Simon, K.W., "Digital Image Reconstruction and Resampling for Geometric Manipulation," *Proc. IEEE Symp. on Machine Processing of Remotely Sensed Data*, pp. 3A-1-3A-11, 1975.

[Singh 79] Singh, M., W. Frei, T. Shibita, G.H. Huth, and N.E. Telfer, "A Digital Technique For Accurate Change Detection in Nuclear Medicine Images — with Application to Myocardial Perfussion Studies Using Thallium 201," *IEEE Trans. Nuclear Science*, vol. NS-26, pp. 565-575, February 1979.

[Smith 83] Smith, A.R., "Digital Filtering Tutorial for Computer Graphics," parts 1 and 2, *SIGGRAPH '83 Introduction to Computer Animation seminar notes*, pp. 244-272, July 1983.

[Smith 87] Smith, A.R., "Planar 2-Pass Texture Mapping and Warping," *Computer Graphics*, (SIGGRAPH '87 Proceedings), vol. 21, no. 4, pp. 263-272, July 1987.

[Stead 84] Stead, S.E., "Estimation of Gradients from Scattered Data," *Rocky Mountain J. Math.*, vol. 14, pp. 265-279, 1984.

[Steiner 77] Steiner, D. and M.E. Kirby, "Geometrical Referencing of Landsat Images by Affine Transformation and Overlaying of Map Data," *Photogrametria*, vol. 33, pp. 41-75, 1977.

[Stoffel 81] Stoffel, J.C. and J.F. Moreland, "A Survey of Electronic Techniques for Pictorial Image Reproduction," *IEEE Trans. Comm.*, vol. COMM-29, no. 12, pp. 1898-1925, December 1981.

[Tanaka 86] Tanaka, A., M. Kameyama, S. Kazama, and O. Watanabe, "A Rotation Method for Raster Image Using Skew Transformation," *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 272-277, June 1986.

[Turkowski 82] Turkowski, K., "Anti-Aliasing Through the Use of Coordinate Transformations," *ACM Trans. on Graphics*, vol. 1, no. 3, pp. 215-234, July 1982.

[Ulichney 87] Ulichney, R., *Digital Halftoning*, MIT Press, 1987.

[Van Wie 77] Van Wie, P., and M. Stein, "A Landsat Digital Image Rectification System," *IEEE Trans. Geoscience Electronics*, vol. GE-15, pp. 130-17, 1977.

[Weiman 80] Weiman, C.F.R., "Continuous Anti-Aliased Rotation and Zoom of Raster Images," *Computer Graphics*, (SIGGRAPH '80 Proceedings), vol. 14, no. 3, pp. 286-293, July 1980.

[Whitted 80] Whitted, T., "An Improved Illumination Model for Shaded Display," *Comm. ACM*, vol. 23, no. 6, pp. 343-349, June 1980.

[Williams 83] Williams, L., "Pyramidal Parametrics," *Computer Graphics*, (SIGGRAPH '83 Proceedings), vol. 17, no. 3, pp. 1-11, July 1983.

[Wolberg 88a] Wolberg, G., "Image Warping Among Arbitrary Planar Shapes," *New Trends in Computer Graphics* (Proc. Computer Graphics Intl. '88), Ed. by N. Magnenat-Thalmann and D. Thalmann, Springer-Verlag, pp. 209-218, 1988.

[Wolberg 88b] Wolberg, G., "Cubic Spline Interpolation: A Review," Columbia University Computer Science Tech. Report CUCS-389-88, November 1988.

[Wolberg 88c] Wolberg, G. and T. Boult, "Image Warping With Spatial Lookup Tables,"

Columbia University Computer Science Tech. Report CUCS-398-88, December 1988.

[Wong 77] Wong, R.Y., "Sensor Transformation," *IEEE Trans. Syst. Man Cybern.*, vol. SMC-7, pp. 836-840, Dec. 1977.

[Yellott 83] Yellott, J.I Jr., "Spectral Consequences of Photoreceptor Sampling in the Rhesus Retina," *Science*, vol. 221, pp. 382-385, 1983.