DIGITAL IMAGE WARPING

–

abstract of Digital Image Warping

–

George Wolberg, IEEE Computer Society Press

R.A. van der Stelt

10 october 1996

Contents

1 Introduction

This is paper is an abstract of the book 'Digital Image Warping' written by George Wolberg. I wrote it as an assignment for my Computer Science study at the university of Leiden. It covers the basic concepts of image warping such as spatial transformations, discrete image mapping, antialiasing and scanline algorithms. It is far from complete and the book itself contains more detailed coverage of various topics I have omitted or sparsely mentioned. I mainly concentrated on chapters 3 to 6 and the first three sections of chapter 7. The rest of chapter 7 and chapter 8 are not covered by this paper. I also made some changes in the order that various topics where presented in the book. The reference section at the end of this paper contains the references to the equations, paragraphs and code examples I used in this paper.

Note that the contents of this paper is completely based on the book and the only contribution I made is the bad grammar and spelling errors.

2 Spatial Transformations

A spatial transformation (also called geometric transformation) defines a geometric relationship between points (u, v) in the input image and points (x, y) in the output image.

Spatial transformations were originally introduced to invert (correct) distortions of digital images due to camera attitude, lens abberations and to combine multiple images from the same object. This requires us to estimate the distortion model, usually by means of reference points which may be accurately marked or readily identified. These points were then used to infer the coefficients of the used spatial transformation.

2.1 Perspective and Affine Spatial Transformations

Many simple spatial transformations can be expressed in terms of the general 3x3 transformation matrix. This matrix is used to specify 2D coordinate transformations in the homogeneous coordinate system. One of the reasons that homogeneous coordinates are used is the fact that this allows us to integrate the translation function into the transformation matrix.

The possible transformations are:

Translation of all the points by adding offsets T_u and T_v to u and v.

$$(x, y, 1) = (u, v, 1) \cdot \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_u & T_v & 1 \end{pmatrix} \tag{2.1}$$

Rotation of all the points around the origin through the counterclockwise angle θ.

$$(x, y, 1) = (u, v, 1) \cdot \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.2}$$

Scaling of all the points by applying the scale factors S_u and S_v to the u and v coordinates. Note that negative scale factors specify mirroring through the axis.

$$(x, y, 1) = (u, v, 1) \cdot \begin{pmatrix} S_u & 0 & 0 \\ 0 & S_v & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.3}$$

Shearing of all the points along the u-axis (with shearing factor H_v) and along the v-axis (with shearing factor H_u). Note that the transformation matrix is singular when $H_u = 1/H_v$.

$$(x, y, 1) = (u, v, 1) \cdot \begin{pmatrix} 1 & H_u & 0 \\ H_v & 1 & 0 \\ 0 & 0 & 1 \end{pmatrix} \tag{2.4}$$

The last transformation is the perspective transformation (also called projective mapping) which has the property of foreshorting "distant" lines. Note that the real perspective projection (3D to 2D) preserves parallel lines when they are parallel to the projection plane.

$$(x', y', w') = (u, v, 1) \cdot \begin{pmatrix} 1 & 0 & a_{13} \\ 0 & 1 & a_{23} \\ 0 & 0 & 1 \end{pmatrix}, \quad x = \frac{x'}{w'}, \quad y = \frac{y'}{w'} \tag{2.5}$$

Multiple transformations can be combined by multiplying the transformation matrices. Note the matrix multiplication is generally not commutative.

A (composite) transformation matrix which consists of only the first four transformations (translate, rotate, scaling and shearing) is called an affine transformation, the homogeneous component remains constant (1, depending on a_{33}) through out the hole image. The affine transformations also preserve equidistance points while perspective transformations only preserve straight lines.

2.1.1 Inferring Perspective Transformations

Since two matrices which are (nonzero) scalar multiples of each other are equivalent in the homogeneous coordinate system, the transformation matrix can be normalized so that $a_{33} = 1$. This leaves eight degrees of freedom for a perspective transformation. The eight coefficients can be determined by establishing correspondence between four points in the input space and output space. Each of the four points correspondence $(u, v) \rightarrow (x, y)$ gives rise to the following two equations:

$$\begin{aligned} x &= a_{11}u + a_{21}v + a_{31} - a_{13}ux - a_{23}vx \\ y &= a_{12}u + a_{22}v + a_{32} - a_{13}uy - a_{23}vy \end{aligned} \tag{2.6}$$

Together this gives us a 8x8 system of equations from which the coefficients can be solved. Note that when the used points are collinear the equations have no solution. When only three points are collinear (either in the input space or output space) then the inferred transformation is affine.

2.1.2 Special Cases

The special case of mapping the unit square to an arbitrary quadrilateral can be used to interfere the coefficients directly. The following four-point correspondences are established from the uv-plane onto the xy-plane.

$$\begin{aligned} &(0, 0) \rightarrow (x_0, y_0) \\ &(1, 0) \rightarrow (x_1, y_1) \\ &(0, 1) \rightarrow (x_2, y_2) \\ &(1, 1) \rightarrow (x_3, y_3) \end{aligned} \tag{2.7}$$

In this case, the eight equations become

$$
\begin{aligned}
a_{31} &= x_0 \\
a_{11} + a_{31} - a_{13}x_1 &= x_1 \\
a_{21} + a_{31} - a_{23}x_2 &= x_2 \\
a_{11} + a_{21} + a_{31} - a_{13}x_3 - a_{23}x_3 &= x_3 \\
a_{32} &= y_0 \\
a_{12} + a_{32} - a_{13}y_1 &= y_1 \\
a_{22} + a_{32} - a_{23}y_2 &= y_2 \\
a_{12} + a_{22} + a_{32} - a_{13}y_3 - a_{23}y_3 &= y_3
\end{aligned}
\tag{2.8}
$$

The solution can take two forms, depending on whether the mapping is affine or perspective. The following definitions are used to solve the equations (2.8).

$$
\begin{array}{lllllllll}
\Delta x_1 & = & x_1 - x_3 & \Delta x_2 & = & x_2 - x_3 & \Delta x_3 & = & x_0 - x_1 + x_3 - x_2 \\
\Delta y_1 & = & y_1 - y_3 & \Delta y_2 & = & y_2 - y_3 & \Delta y_3 & = & y_0 - y_1 + y_3 - y_2
\end{array}
\tag{2.9}
$$

If $\Delta x_3 = 0$ and $\Delta y_3 = 0$, then the quadrilateral is a parallelogram. This implies that the mapping is affine. As a result, we obtain the following coefficients.

$$
\begin{pmatrix}
x_1 - x_0 & y_1 - y_0 & 0 \\
x_2 - x_0 & y_2 - y_0 & 0 \\
x_0 & y_0 & 1
\end{pmatrix}
\tag{2.10}
$$

If $\Delta x_3 \neq 0$ or $\Delta y_3 \neq 0$, then the mapping is perspective. The coefficients of the perspective transformation are

$$
\begin{aligned}
a_{13} &= \begin{vmatrix} \Delta x_3 & \Delta x_2 \\ \Delta y_3 & \Delta y_2 \end{vmatrix} \Big/ \begin{vmatrix} \Delta x_1 & \Delta x_2 \\ \Delta y_1 & \Delta y_2 \end{vmatrix} \\
a_{23} &= \begin{vmatrix} \Delta x_1 & \Delta x_3 \\ \Delta y_1 & \Delta y_3 \end{vmatrix} \Big/ \begin{vmatrix} \Delta x_1 & \Delta x_2 \\ \Delta y_1 & \Delta y_2 \end{vmatrix}
\end{aligned}
\tag{2.11}
$$

$$
\begin{array}{lllllll}
a_{11} & = & x_1 - x_0 + a_{13}x_1 & \quad a_{12} & = & y_1 - y_0 + a_{13}y_1 \\
a_{21} & = & x_2 - x_0 + a_{23}x_2 & \quad a_{22} & = & y_2 - y_0 + a_{23}y_2 \\
a_{31} & = & x_0 & \quad a_{32} & = & y_0
\end{array}
$$

The computation may be generalized to map arbitrary rectangles onto quadrilaterals by premultiplying with a scale and translation matrix.

Two other special cases can be solved using the square-to-quadrilateral case.

The first is the quadrilateral-to-square which can be found by simply computing the inverse transformation matrix of the square-to-quadrilateral case. Note that the inverse of a projective mapping can be easily computed in terms of the adjoint of the transformation matrix.

The second is the quadrilateral-to-quadrilateral can be solved by first solving the quadrilateral-to-square case and the square-to-quadrilateral case and multiplying the two found transformation matrices.

Note that these special cases are only useful when the control points are precisely known. However, when these points are known to contain errors an overdetermined system of equations should be used to solve the equations.

2.1.3 Inferring Affine Transformation

Since the last column of the affine matrix is fixed ($[0, 0, 1]$) it leaves only six degrees of freedom. Thus the correspondence $(u, v) \rightarrow (x, y)$ of three points is enough to determine the six coefficients.

$$\begin{pmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{pmatrix} = \begin{pmatrix} u_0 & v_0 & 1 \\ u_1 & v_1 & 1 \\ u_2 & v_2 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{pmatrix} \tag{2.12}$$

The solution of the six equations (2.12), which can be expressed as $X = U \cdot A$, can be found by multiplying X with the adjoint matrix of U divided by the determinant of U.

$$\begin{pmatrix} a_{11} & a_{12} & 0 \\ a_{21} & a_{22} & 0 \\ a_{31} & a_{32} & 1 \end{pmatrix} = \frac{1}{\det(U)} \cdot \begin{pmatrix} v_1 - v_2 & v_2 - v_0 & v_0 - v_1 \\ u_2 - u_1 & u_0 - u_2 & u_1 - u_0 \\ u_1 v_2 - u_2 v_1 & u_2 v_0 - u_0 v_2 & u_2 v_1 - u_1 v_0 \end{pmatrix} \cdot \begin{pmatrix} x_0 & y_0 & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{pmatrix} \tag{2.13}$$

with

$$\det(U) = u_0(v_1 - v_2) - v_0(u_1 - u_2) + (u_1 v_2 - u_2 v_1) \tag{2.14}$$

2.2 Bilinear Transformations

The general representation of a bilinear transformation is

$$(x,\ y) = (u,\ v,\ 1) \cdot \begin{pmatrix} a_3 & b_3 \\ a_2 & b_2 \\ a_1 & b_1 \\ a_0 & b_0 \end{pmatrix} \tag{2.15}$$

The bilinear transformation or bilinear mapping, handles the four-corner mapping problem for nonplanar quadrilaterals. It only preserves lines that are horizontal or vertical in the source image. Bilinear mapping are defined through piecewise functions that must interpolate the coordinate assignments specified at the vertices. This scheme is based on bilinear interpolation to evaluated the $X(u, v)$ and $Y(u, v)$ mapping functions.

$$X(u,v) = a_0 + a_1 u + a_2 v + a_3 uv \tag{2.16}$$

Bilinear interpolation utilizes a linear combination of the four "closest" grid values to produce a new interpolated value. Using the four grid-points (u_0, v_0), (u_1, v_0), (u_0, v_1) and (u_1, v_1) with function values x_0, x_1, x_2 and x_3 equation (2.16) can be rewritten to:

$$X(u', v') = x_0 + (x_1 - x_0)u' + (x_2 - x_0)v' + (x_3 - x_2 - x_1 + x_0)u'v' \tag{2.17}$$

with normalized coordinates

$$u' = \frac{u - u_0}{u_1 - u_0}\,, \quad v' = \frac{v - v_0}{v_1 - v_0} \tag{2.18}$$

Because the bilinear mapping is a non planar mapping the inverse of the mapping is not single valued and therefor difficult to compute.

It is important to note that higher-order interpolation methods are commonly defined to operate on rectangular lattices. The bilinear interpolation technique can be used to performing any 2D interpolation within an arbitrary quadrilateral. The procedure for mapping an arbitrary grid to a rectangular grid is given as follows:

Procedure 2.2

1) For each point P, (x, y) inside an interpolation region defined by four arbitrary points (P_1, P_2, P_3, P_4) a normalized coordinate (u', v') is associated in a rectangular region. The normalized coordinates can be found by determining the grid lines, $P_{02} - P_{13}$ and $P_{01} - P_{23}$, that intersect at point P. The normalized coordinates are given as

$$\begin{aligned} u' &= \frac{P_{01}P_0}{P_1P_0} = \frac{P_{23}P_2}{P_3P2} \\ v' &= \frac{P_{02}P_0}{P_2P_0} = \frac{P_{23}P_1}{P_3P1} \end{aligned} \tag{2.19}$$

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

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

4) The result is then assigned to point P in the distorted plane

2.3 Polynomial Transformations

Geometric correction requires a spatial transformation to invert an unknown distortion function. The (invert) mapping functions, U and V, are global polynomial transformations of the form

$$u = \sum_{i=0}^{N} \sum_{j=0}^{N-i} a_{ij} X^i y^j$$
$$v = \sum_{i=0}^{N} \sum_{j=0}^{N-i} b_{ij} x^i y^j \tag{2.20}$$

where a_{ij} and b_{ij} are the constant polynomial coefficients.
These polynomial transformations are global mapping functions operating on the entire image. They are intended to account for sensor-related spatial distortions (such as centering, scale, skew,scan nonlinearity etc) as well as errors due to viewing geometry, camera altitude etc. These errors usually are low-frequency (smoothly varying) distortions so low-order polynomial mapping functions ($N \leq 4$) should be adequate. Note that first-degree bivariate polynomials are equivalent to the 3x3 affine transformation matrix.

2.3.1 Inferring Polynomial Coefficients

In order to determine the $K = \frac{1}{2}(N+1)(N+2)$ coefficients for each mapping function (U and V) there are $M \geq K$ control points needed for which the "correct" position in the image is known. Writing out equation (2.20) for the M points gives us two overdetermined systems of linear equations. These systems can then be solved with standard least-squares techniques from linear algebra (such as pseudo-inverse, least-squares with orthogonal polynomials and the householder transformation).

Note that the least-squares technique usually only gives coefficients that best approximate the true mapping function. Especially when noisy control data is used. The approximation can be refined by throwing away the control point that has the largest deviation from the found mapping function and solving the overdetermined system of equations with $M-1$ points. By repeating this process until there are only K points left (or some error threshold is reached) the coefficients are made to more closely fit an increasingly consistent set of data.

2.3.2 Piecewise polynomial transformation

Global polynomial transformations impose a single mapping function upon a whole image. They often do not account for local geometric distortions. The weighted least-squares method for solving the polynomial coefficients is a global method that considers all control points but recomputes the coefficients at each point by using a weighted function that is biased in favour of nearby control points. It thus constitutes a hybrid global/local method, computing polynomial coefficients through a global technique, but permitting the coefficients to be spatially varying. If the control points all lie on a rectangular mesh it is possible to use bicube spline interpolation (for example B-splines or Bezier surface patches). Splines are defined globally but remain sensitive to local data in contrast to the least-squares method which averaged out local distortions.

A truly local transformation considers only nearby control points. One method of performing such a transformation using irregularly spaced control points is described by the following steps.

Procedure 2.3.2

1) Triangulate the image by connecting neighbouring control points with noncrossing line segments.

2) Estimate the partial derivatives of U (and V) with respect to x and y at each of the control points (only necessary when the patches are to be joined smoothly, i.e for C^1, C^2, etc).

3) For each triangular region, find smooth surface patch through the vertices by solving the coefficients of low-order bivariate polynomials (with the constraints imposed by the partial derivatives).

4) Regions lying outside the convex hull of the control points must extrapolate the surface patches from neighbouring triangles.

5) For each point (x, y) the enclosing surfaces patches are used to determine the values of u and v which give the mapped coordinate (u, v) of (x, y).

3 Discrete mapping

3.1 Forward Mapping

Forward mapping consists of copying each input pixel onto the output image at positions determined by the X and Y mapping functions. Due to the fact that discrete input pixels are mapped to real output positions, it is inappropriate to implement the spatial transformation as a point-to-point mapping. Doing so can give rise to two types of problems; holes (due to magnification) and overlaps (due to minification). These problems can be avoided by using a four-corner mapping paradigm which considers input pixels as square patches that may be transformed into arbitrary quadrilaterals in the output image. This method requires an accumulator array and costly intersections to integrate the contributions of the input pixels to each output pixel. A approximation to the four-corner mapping paradigm can be achieved by mapping more points on the pixel-square to the output pixel. The number of points is determined by the area covered by the quadrilateral in the output image. This method is familiar to the adaptive supersampling used for antialiasing.

3.2 Inverse Mapping

Inverse mapping consists of projecting each output coordinate into the input image by the U and V (inverse) mapping functions. Because the discrete output pixels are mapped to real-valued position in the input image, an interpolation stage is needed to retrieve input values at undefined input positions. Unlike the point-to-point forward mapping scheme, the inverse mapping scheme guarantees that all output pixels are computed. Note that the holes now occurs in the input image consisting of large amounts of input data discarded while evaluating the output.

3.3 Image Reconstructing

From signal analysis we know that pointsampling a continuous signal introduces infinite high frequencies in the fourier spectrum of the sampling. In fact when the signal is sampled with the Dirac-function[1] then the fourier spectrum of the signal consists of the repetition of the original spectrum at regular intervals. The distance and thus the overlap between the replicated spectrums is inversely proportional on the sample frequency of the sampling function. Signal analysis also tells us that correct reconstruction of a point sampled signal is only possible when there is no overlap between the replicate spectrums in the fourier spectrum of the sampled signal. This leads to a minimal sampling frequency of twice the highest frequency of the original signal, this minimal sampling rate is also known as the Nyquist frequency. An other consequence is that signals with infinite high frequency components need to be bandlimited before being sampled. Since high frequency components are directly related to high visible detail in the image, bandlimiting results in blurring of the image.

3.3.1 Convolution Kernels

Bandlimiting of a signal is achieved by multiplying the fourier spectrum of the signal with a block-function which has the value 1 in the lower frequency range (pass-band) and 0 in

[1] the Dirac-function is also known as the impulse-function, see equation (4.2.2) on page 98 of the book "Digital Image Warping"

the higher frequency range (the stop-band). Multiplying in the frequency domain constitutes a convolution in the spatial domain. Unfortunately the fourier transformation (also known as the convolution kernel) of the block function is an infinite wide sinc-function which is impractical to use as a filter for a finite image. Different convolution kernels can be used to approximate the block-filter. These convolution kernels are typically evaluated by analyzing their performance in the passband and stopband. An ideal filter will have unity gain in the passband and zero gain in the stopband in order to transmit and suppress the signal's spectrum in these respective frequency ranges. Unfortunately, ideal (or superior nonideal) filters generally have wide extent in the spatial domain. An extensive description of commonly used filters (and their performance in the frequency domain) is given in chapter 5 [DIW].

3.4 Aliasing

Aliasing 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. Aliasing is often a problem when an image is forced to conform to an inadequate resolution due to physical constraints.
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 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 distinction between reconstruction and aliasing artifacts becomes clear when we notice that the appearance of jagged edges is improved by blurring. Contrarily, once an image is truly undersampled, there is no postprocessing possible to improve its condition.

3.5 Image Resampling

Image resampling is the process of transforming a sampled image from one coordinate system to another. The two coordinate systems are related to each other by the mapping function of the spatial transformation.

To prevent aliasing artifacts when performing an image warp, it is necessarily to first reconstruct the original signal of the (sampled) input image and to bandlimit the warped signal before resampling it onto the output pixels.

The image resampling process thus compromises two stages: image reconstruction followed by sampling the warped image. There are four basic elements to ideal resampling: reconstruction, warping, refiltering and sampling.
Table 3.5

Stage	Mathematical definition
Discrete Input	$f(y), \quad u \in Z$
Reconstructed Input	$f_c(u) \quad = f(u) * r(u) = \sum_{k \in Z} f(k) r(u-k)$
Warped Signal	$g_c(x) \quad = f_c(m^{-1}(x))$
Continous Output	$g'_c(x) \quad = g_c(x) * h(x) = \int g_c(t) h(x-t) dt$
Discrete Output	$g(x) \quad = g'_c(x) s(x)$

The process begins with $f(u)$, the discrete input defined over integer values of u. It is reconstructed into $f_c(u)$ through convolution with reconstruction filter $r(u)$. The continuous input $f_c(u)$ is then warped according to mapping function m. The forward map is given as $x = m(u)$ and the inverse map is $u = m^{-1}(x)$.
The spatial transformation leaves us with $g_c(x)$, the continuous warped output. Depending on the mapping function $m^{-1}(x)$, $g_c(x)$ may have arbitrarily high frequencies. Therefor, it is bandlimited by function $h(x)$ to conform to the Nyquist rate of the output. The bandlimited result is $g'_c(x)$. This function is sampled by $s(x)$, the output sampling grid. Note that $s(x)$ is not required to sample the output at the same density as that of the input.

There are only two filtering components to the entire resampling process: reconstruction and refiltering. They can be combined into a single filter as follows:

$$\begin{aligned} g'_c(x) &= \int f_c(m^{-1}(t))h(x-t)dt \\ &= \int \left\{ \sum_{k \in Z} f(k)r(m^{-1}(t)-k) \right\} h(x-t)dt \\ &= \sum_{k \in Z} f(k)\rho(x,k) \end{aligned} \tag{3.1}$$

where

$$\rho(x,k) = \int r(m^{-1}(t)-k)h(x-t)dt \tag{3.2}$$

is the resampling filter that specifies the weight of the input sample at location k for an output sample at location x.

Substituting $t = m(u)$ the resampling filter can be expressed in terms of an integral in the input space.

$$\rho(x,k) = \int r(u-k)h(x-m(u)) \left| \frac{\partial m}{\partial u} \right| du \tag{3.3}$$

where $|\partial m/\partial u|$ is the determinant of the Jacobian matrix interrelating the input and output coordinate systems.

In the input-space form, the resampling filter is expressed in terms of a reconstruction filter and a warped prefilter. In the output-space form however, its is expressed in terms of a warped reconstruction filter and a prefilter. Therefor the actual warping is incorporated into either the reconstruction filter or the prefilter but not both.

4 Antialiasing

The basic flaw in point sampling is that a discrete pixel actually represents an area, not a point. This can be corrected by applying 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 is called area sampling. The low-pass filter comprises the prefiltering stage of the image resampling process. It serves to defeat aliasing by bandlimiting the input image prior to pointsampling it onto the output grid.

The prefiltering is defined by the convolution integral

$$g(x,y) = \int\int f(u,v)h(x-u,y-v)dudv \tag{4.1}$$

Since the (ideal) sinc function can't be used as prefilter (because of its infinite range) an non-ideal FIR[2] filter must be used. If however the filter kernel is space-invariant (remains constant as it scans across the image) the image and filter kernel can be transformed into the frequency domain by using the FFT[3]fourier convolution. In the frequency domain the image and the filter kernel only need to be multiplied after which the result can be transformed back to the spatial domain using 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 $M > \log_2 N$.

However, in many common image warping operations (such as perspective mapping, nonlinear warps, texture mapping, etc) space-variant filters are required, where the kernel varies with position. In such cases space-variant FIR filters require large number of preimage samples in order to compute each output pixel.

4.1 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. Note that supersampling is a form of areasampling and therefor reduces aliasing by bandlimiting the input signal. Since the band filtering of high frequencies is limited by the density of the supersampling grid an balance must be found between the cost of supersampling versus preventing aliasing.

A possible solution is adaptive supersampling where the samples are distributed more densely in areas of high intensity variance. Note that estimating intensity variance by using samples is prone to the same aliasing artifacts of sampling itself.

The value assigned to the pixel can be computed by taking the average of the samples. If however an other interpolation filter is used it is advisable to extent the base of the convolution kernel to the neighbouring pixels to probably include the samples at the boundary of the pixel-square.

[2]FIR stands for "Finite Impulse Response".

[3]FFT stands for "Fast Fourier Transform" which is a fast implementation of the "Discrete Fourier Transform" algorithm.

4.2 Irregular 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 could in theory prevent this problem.

Irregular sampling (also called stochastic sampling) is the process of using an irregular sampling grid in which to sample the input image. It is also used 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.

There are free common forms of stochastic sampling, Poisson sampling, jittered sampling, and pointdiffusion sampling.

4.2.1 Poisson Sampling

Poisson sampling uses an irregular sampling grid that is stochastically generated to yield an uniform distribution of sample points. This approximation to uniform sampling can be improved with the addition of a minimum-distance constraint between sample points. This Poisson-disk distribution is likely to be the optimal sampling pattern since it can also be found among the sparse retinal cells outside the foveal region of the eye. It appears that this spatial organization serves to scatter aliasing into high-frequency random noise. An ideal sampling pattern therefor should have a broad noisy spectrum with minimal low-frequency. Such a pattern exhibits no coherence which can give rise to structured aliasing artifacts. Distribution which satisfy these conditions are known as blue noise (in contrast with white noise which is a random signal where all frequency components have equal magnitude).

Unfortunately the Poisson-disk sampling patterns are difficult to generate. This is due to the minimal distance constrained which needs to be checked when the sample points are generated.

4.2.2 Jittered Sampling

Jittered sampling is an approximation of a Poisson-disk distribution. A jittered sampling pattern is created by randomly perturbing each sample point on a regular sampling pattern. The result is inferior to that of the optimal Poisson-disk distribution. This is caused by the granularity of the distribution which corresponds to increased low-frequency energy in the distribution. However since the magnitude of the noise is directly proportional to the sampling rate, improved image quality is achieved with increased sample density.

4.2.3 Point-Diffusion Sampling

The point-diffusion sampling algorithm has been proposed as a computational efficient technique to generate Poisson-disk samples. It is based on the Floyed-Steinberg error-diffusion algorithm used for dithering grayscale images into bitmaps. The dithering algorithm is described in chapter 6 page 176. The distribution of points generated by the algorithm to simulate gary scale satisfies the blue-noise criteria.

4.2.4 Reconstruction of Irregular Sampling

The supersampling and adaptive sampling techniques used with regular sampling can also be used with stochastic sampling. In general, irregular sampling requires rather high sampling densities and thus adaptive sampling plays a natural role in this process. It serves to dramatically reduce the noise level while avoiding needless computation.

Once the irregularly spaced samples are collected, they must pass through a reconstruction filter to be resampled at the resolution of the output image. Reconstruction is made difficult by the irregular distribution of the samples. One common approach is to use weighted-average filters:

$$f(x) = \frac{\sum_{k=-\infty}^{\infty} h(x - x_k) f(x_k)}{\sum_{k=-\infty}^{\infty} h(x - x_k)} \tag{4.2}$$

Other more efficient reconstruction filters such as multi-stage filters and direct convolution techniques are described in chapter 6 page 177-180 [DIW].

4.3 Prefiltering with Pyramids and Mip-Maps

Pyramids are data-structures containing multiple copies of the same image at different resolutions. The original image forms the base of the pyramid and the "higher"layers are formed by increasingly lower resolution copies of the orignal image. When the resolution of successive layers differ by a power of two then the additional memory cost of the pyramid can be shown to be 1/3 (33%) of the memory requirement of the original image.

A image is prefilter by pointsampling one of the layers of the pyramid. Unfortunately the choice of the "correct" layer to be pointsampled is not as straightforward due to the jump in resolutions between the layers and due to the forshorting caused by perspective mapping.

Mip-maps[4] are pyramids which address these problems by interpolating between the layers as well as inside the layers. This is achieved by storing the three color-planes of the image in the way as is depicted in figure (1).

The u and v coordinates are spatial coordinates used to access points within a pyramid level. The d coordinate is used to index, and interpolate between different layers of the mip-map. The value of d must be chosen to balance the tradeoff between aliasing and blurring. A possible choice is

$$d^2 = \max\left(\left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial x}\right)^2, \left(\frac{\partial u}{\partial y}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2\right) \tag{4.3}$$

where d is proportional to the span of the preimage area, and the partial derivatives can be computed from the surface projection. Since corresponding points in different layers have indices which are related by some power of two, simple binary shifts can be used to access points across the multi-resolution copies (allowing efficient hardware implementation).

Pyramid require setup time, which means that the are best suited for input that is intended to be used repeatedly (i.e, texture maps). However, if the texture is only to be used once, direct convolution methods are more cost-effective then pyramids.

[4] "mip" stands for "multum in parvo" which is Latin for "many things in a small place".

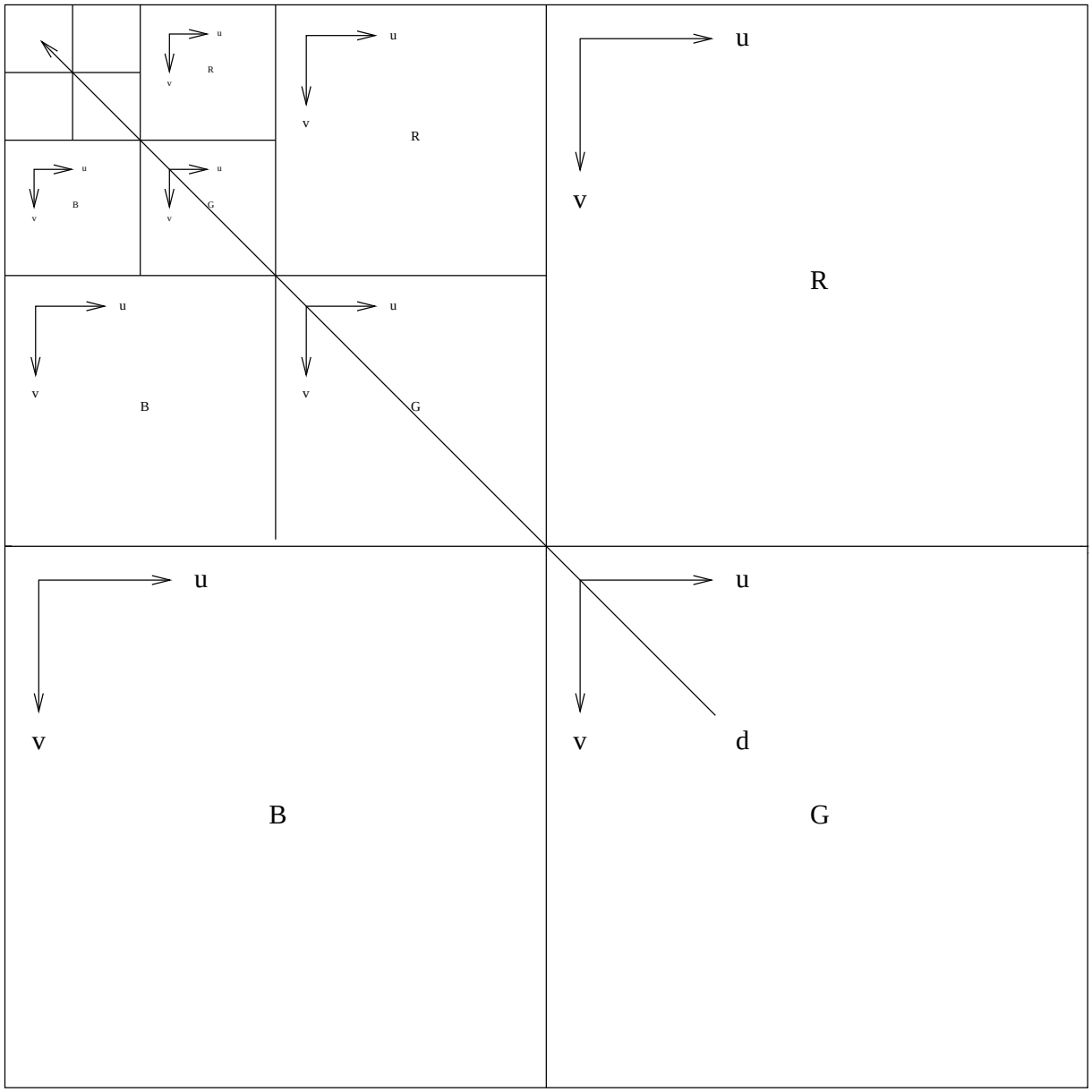

Figure 1: *mip-map memory organization*

5 Scanline Algorithms

Scanline algorithms comprise a special class of geometric transformation techniques that operate only along rows and columns. There are several advantages to decomposing a mapping into a series of 1D transforms. First, the resampling problem is made simpler since reconstruction, area sampling and filtering can now be done entirely in 1D. Secondly, this lends itself naturally to digital hardware implementation. Note that no sophisticated digital filters are necessary to deal with the 2D case. Third the mapping can be done in scanline order both in scanning the input and in producing the projected image.

5.1 Incremental Algorithms

Texture mapping is a powerful technique used to add visual detail to synthetic images. It consist of a series of spatial transformations: a texture plane $[u, v]$, is transformed onto a 3D surface, $[x, y, z]$, and then projected onto the output screen, $[x, y]$. The surfaces of 3D objects are usually modeled with planar polygons or bicubic patches. The mapping between the texture map and the surface patches is usually treated as a four-corner mapping. Consider an input square texture in the uv-plane mapped onto a planar quadrilateral in the xyz-coordinate system using inverse mapping. The mapping can be specified by designating texture coordinates to the quadrilateral. The Gouraud shading algorithm is used to determine the correspondence for all interior quadrilateral points. The algorithm is also used to enhance realism by shading polygonal surfaces to approximate curved surfaces.
Gouraud shading interpolates the intensities all along a polygon, given only the true values at the vertices. It does so while operating in scanline order. For each scanline, the intensities I_a and I_b at the endpoints x_a and x_b are computed. This is achieved through linear interpolation between the intensities of the appropriate polygon vertices. Then, beginning with I_a, the intensity values along successive scanline positions are computed incremental

$$I_{x+1} = I_x + dI \ , \qquad dI = \frac{I_b - I_a}{x_b - x_a} \tag{5.1}$$

This algorithm can be used to compute the (u, v) coordinates by computing the u and v values instead of the intensities. The (u, v) coordinates are then used as an index in the input texture.

The following segment of C code is a simplified example of processing a single scanline.
Code 5.1

```
dx = 1.0/(x1-x0);             /* normalization factor */
du = (u1-u0)*dx;              /* constant increment for u */
dv = (v1-v0)*dx;              /* constant increment for v */
dz = (z1-z0)*dx;              /* constant increment for z */
for (x = x0; x < x1; x++) {   /* visit all scanline pixels */
        if (z < zbuf[x]) {    /* is new point closer */
                zbuf[x] = z;  /* update z-buffer */
                scr[x] = tex(u,v);
        }               /* write texture value to screen */
        u += du;              /* increment u */
```

```
        v += dv;              /* increment v */
        z += dz;              /* increment z */
}
```

Note that the z-coordinates are stored in a z-buffer to check for possible occluding polygons (that where rendered first).

There are several problems using this algorithm. First the textured polygon shows undesirable discontinuities along horizontal lines passing through the vertices. This is due to a sudden change in du and dv as we move past a vertex and is an artifact of the linear interpolation. Second, the algorithm does not produce the foreshorting due to perspective transformation. In fact, this approach is a 2-pass implementation of the bilinear mapping which is only exact for affine mapping and not perspective mapping. A simple solution would be to subdivide the polygon to achieve better approximation of the perspective mapping. This of course requires more costly computations of the vertices texture coordinates. It is also difficult to determine how much subdivision is necessary.

5.2 Incremental Perspective Transformations

Since a perspective transformation is a ratio of two linear interpolants, it becomes possible to achieve theoretically correct results by introducing the homogeneous coordinate w. It should be interpolated along side u and v and used to divided u and v to get the correct values. The following code contains the necessary adjustments to make the scanline approach work for perspective mapping.
Code 5.2

```
dx = 1.0/(x1-x0);             /* normalization factor */
du = (u1-u0)*dx;              /* constant increment for u */
dv = (v1-v0)*dx;              /* constant increment for v */
dz = (z1-z0)*dx;              /* constant increment for z */
dw = (w1-w0)*dx;              /* constant increment for w */
for (x = x0; x < x1; x++) {   /* visit all scanline pixels */
        if (z < zbuf[x]) {    /* is new point closer */
                zbuf[x] = z;  /* update z-buffer */
                scr[x] = tex(u/w,v/w);
        }                /* write texture value to buffer */
        u += du;              /* increment u */
        v += dv;              /* increment v */
        z += dz;              /* increment z */
}
```

However, the division operations needed for perspective mapping are expensive and undermine some of the computational gains. Of course, approximation of the division operation can be implemented using a truncated Taylor series approximation with a lookup table.

The used Taylor approximation is

$$\frac{1}{w} = \frac{1}{w_0} - \frac{\delta}{w_0^2} \tag{5.2}$$

with $1/w_0$ a value stored in the lookup table and $\delta = (w - w_0)$.

Other incremental algorithms use quadratic or cubic interpolation to approximate the nonuniformity introduced by perspective transformations. These algorithms are described in chapter 7, page 199-204 [DIW].

5.3 Separable Algorithms

The incremental scanline algorithms all exploit the computational savings made possible by forward differences. While they may be fast at computing the transformation, they neglect filtering issues between scanlines. Rather than attempting to approximate the transformations along only one direction, separable algorithms decompose their mapping functions along orthogonal directions, i.e. rows and columns.

Although separable algorithms cannot handle all possible mapping functions they do work well for a wide class of common transformations, including affine and perspective mapping. The geometric transformations that are best suited for this approach are those that can be shown to be separable (each dimension can be resampled independently of the other).

5.3.1 Example: Rotation

An example of a separable transformation is the affine rotation matrix

$$R = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \tag{5.3}$$

A rotation algorithm based completely on simpler 1D scale and shear operations is given by decomposing the rotation matrix R into four submatrices.

$$R = \begin{pmatrix} 1 & \tan\theta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\sin\theta\cos\theta & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ 0 & \cos\theta \end{pmatrix} \begin{pmatrix} 1/\cos\theta & 0 \\ 0 & 1 \end{pmatrix} \tag{5.4}$$

This formulation represents a 4-pass separable algorithm in which 1D scaling and shearing are performed along both image axes. An efficient line-drawing algorithm can be used to resample the input pixels and perform shearing.

A 2-pass decomposition of R is given by

$$R = \begin{pmatrix} \cos\theta & 0 \\ -\sin\theta & 1 \end{pmatrix} \begin{pmatrix} 1 & \tan\theta \\ 0 & 1/\cos\theta \end{pmatrix} \tag{5.5}$$

It combines the scaling and shearing in each pass.

An important 3-pass algorithm for rotation is given by the decomposition

$$R = \begin{pmatrix} 1 & 0 \\ -\tan\frac{1}{2}\theta & 1 \end{pmatrix} \begin{pmatrix} 1 & \sin\theta \\ 0 & 1 \end{pmatrix} \begin{pmatrix} 1 & 0 \\ -\tan\frac{1}{2}\theta & 1 \end{pmatrix} \tag{5.6}$$

The algorithm is implemented by three shear transformations without any scaling. It first skews the image along the horizontal direction by displacing each row. The result is than

skewed along the vertical direction. Finally, an additional skew in the horizontal direction yields the rotated image. The primary advantage to the 3-pass shear transformation algorithm is that it avoids a costly scale operation. By not introducing a scale operation, the algorithm avoids complications in sampling, filtering, and the associated degradations. Simplifications are based on efficient realizations of the 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. Note that each pixel can overlap at most two pixels so a simple triangle filter (linear interpolation) is adequate. Further more, the sum of the pixel intensities along any scanline can be shown to remain unchanged after shear operation. Thus the algorithm produces no visible spatial-variant artifacts or holes.

A References

[DIW]

This paper is an abstract of the book “Digital Image Warping” written by Goerge Wolberg of the Department of Computer Science at Columbia University, New York. It was published by the IEEE Computer Society Press in 1992 (third edittion 1994, ISBN 0-8186-8944-7).

A.1 Paragraphs

A.2 Procedures

A.3 Tables

A.4 Figures

A.5 Code examples

A.6 Equations