13

GEOMETRICAL IMAGE MODIFICATION

One of the most common image processing operations is geometrical modification in which an image is spatially translated, scaled in size, rotated, nonlinearly warped or viewed from a different perspective (1).

13.1. BASIC GEOMETRICAL METHODS

Image translation, size scaling and rotation can be analyzed from a unified standpoint. Let $D(j, k)$ for $0 \le j \le J - 1$ and $0 \le k \le K - 1$ denote a discrete destination image that is created by geometrical modification of a discrete source image $S(p, q)$ for $0 \le p \le P - 1$ and $0 \le q \le Q - 1$. In this derivation, the source and destination images may be different in size. Geometrical image transformations are usually based on a Cartesian coordinate system representation in which pixels are of unit dimension, and the origin $(0, 0)$ is at the center of the upper left corner pixel of an image array. The relationships between the Cartesian coordinate representations and the discrete image array of the destination image $D(j, k)$ are illustrated in Figure 13.1-1. The destination image array indices are related to their Cartesian coordinates by

$$x_j = j + \tfrac{1}{2} \tag{13.1-1a}$$

$$y_k = k + \tfrac{1}{2} \tag{13.1-1b}$$

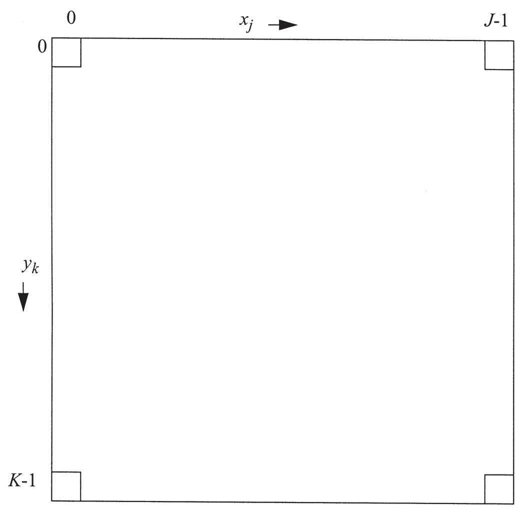


FIGURE 13.1-1. Relationship between discrete image array and Cartesian coordinate representation of a destination image $D(j, k)$.

Similarly, the source array relationship is given by

$$u_p = p + \tfrac{1}{2} \qquad (13.1\text{-}2a)$$

$$v_q = q + \tfrac{1}{2} \qquad (13.1\text{-}2b)$$

13.1.1. Translation

Translation of $S(p, q)$ with respect to its Cartesian origin to produce $D(j, k)$ involves the computation of the relative offset addresses of the two images. The translation address relationships are

$$x_j = u_p + t_x \qquad (13.1\text{-}3a)$$

$$y_k = v_q + t_y \qquad (13.1\text{-}3b)$$

where t_x and t_y are translation offset constants. There are two approaches to this computation for discrete images: forward and reverse address computation. In the forward approach, u_p and v_q are computed for each source pixel (p, q) and substituted into Eq. 13.1-3 to obtain x_j and y_k. Next, the destination array addresses (j, k) are computed by inverting Eq. 13.1-1. The composite computation reduces to

$$j' = p + t_x \tag{13.1-4a}$$

$$k' = q + t_y \tag{13.1-4b}$$

where the prime superscripts denote that j' and k' are not integers unless t_x and t_y are integers. If j' and k' are rounded to their nearest integer values, data voids can occur in the destination image. The reverse computation approach involves calculation of the source image addresses for integer destination image addresses. The composite address computation becomes

$$p' = j - t_x \tag{13.1-5a}$$

$$q' = k - t_y \tag{13.1-5b}$$

where again, the prime superscripts indicate that p' and q' are not necessarily integers. If they are not integers, it becomes necessary to interpolate pixel amplitudes of $S(p, q)$ to generate a resampled pixel estimate $\hat{S}(p, q)$, which is transferred to $D(j, k)$. The geometrical resampling process is discussed in Section 13.5.

13.1.2. Scaling

Spatial size scaling of an image can be obtained by modifying the Cartesian coordinates of the source image according to the relations

$$x_j = s_x u_p \tag{13.1-6a}$$

$$y_k = s_y v_q \tag{13.1-6b}$$

where s_x and s_y are positive-valued scaling constants, but not necessarily integer valued. If s_x and s_y are each greater than unity, the address computation of Eq. 13.1-6 will lead to magnification. Conversely, if s_x and s_y are each less than unity, minification results. The reverse address relations for the source image address are found to be

$$p' = \frac{j + \frac{1}{2}}{s_x} - \frac{1}{2} \tag{13.1-7a}$$

$$q' = \frac{k + \frac{1}{2}}{s_y} - \frac{1}{2} \tag{13.1-7b}$$

As with generalized translation, it is necessary to interpolate $S(p, q)$ to obtain $D(j, k)$.

13.1.3. Rotation

Rotation of an input image about its Cartesian origin can be accomplished by the address computation

$$x_k = u_q \cos\theta - v_p \sin\theta \tag{13.1-8a}$$

$$y_j = u_q \sin\theta + v_p \cos\theta \tag{13.1-8b}$$

where θ is the counterclockwise angle of rotation with respect to the horizontal axis of the source image. Again, interpolation is required to obtain $D(j, k)$. Rotation of a source image about an arbitrary pivot point can be accomplished by translating the origin of the image to the pivot point, performing the rotation, and then translating back by the first translation offset. Equation 13.1-8 must be inverted and substitutions made for the Cartesian coordinates in terms of the array indices in order to obtain the reverse address indices (p', q'). This task is straightforward but results in a messy expression. A more elegant approach is to formulate the address computation as a vector-space manipulation.

13.1.4. Generalized Linear Geometrical Transformations

The vector-space representations for translation, scaling and rotation are given below.

Translation:
$$\begin{bmatrix} x_j \\ y_k \end{bmatrix} = \begin{bmatrix} u_p \\ v_q \end{bmatrix} + \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{13.1-9}$$

Scaling:
$$\begin{bmatrix} x_j \\ y_k \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} u_p \\ v_q \end{bmatrix} \tag{13.1-10}$$

Rotation:
$$\begin{bmatrix} x_j \\ y_k \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} u_p \\ v_q \end{bmatrix} \tag{13.1-11}$$

Now, consider a compound geometrical modification consisting of translation, followed by scaling, followed by rotation. The address computations for this compound operation can be expressed as

$$\begin{bmatrix} x_j \\ y_k \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} u_p \\ v_q \end{bmatrix} + \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} t_x \\ t_y \end{bmatrix} \tag{13.1-12a}$$

or upon consolidation

$$\begin{bmatrix} x_j \\ y_k \end{bmatrix} = \begin{bmatrix} s_x\cos\theta & -s_y \sin\theta \\ s_x \sin\theta & s_y \cos\theta \end{bmatrix} \begin{bmatrix} u_p \\ v_q \end{bmatrix} + \begin{bmatrix} s_x t_x \cos\theta & -s_y t_y \sin\theta \\ s_x t_x \sin\theta & +s_y t_y \cos\theta \end{bmatrix} \tag{13.1-12b}$$

Equation 13.1-12*b* is, of course, linear. It can be expressed as

$$\begin{bmatrix} x_j \\ y_k \end{bmatrix} = \begin{bmatrix} c_0 & c_1 \\ d_0 & d_1 \end{bmatrix} \begin{bmatrix} u_p \\ v_q \end{bmatrix} + \begin{bmatrix} c_2 \\ d_2 \end{bmatrix} \tag{13.1-13a}$$

in one-to-one correspondence with Eq. 13.1-12*b*. Equation 13.1-13*a* can be rewritten in the more compact form

$$\begin{bmatrix} x_j \\ y_k \end{bmatrix} = \begin{bmatrix} c_0 & c_1 & c_2 \\ d_0 & d_1 & d_2 \end{bmatrix} \begin{bmatrix} u_p \\ v_q \\ 1 \end{bmatrix} \tag{13.1-13b}$$

As a consequence, the three address calculations can be obtained as a single linear address computation. It should be noted, however, that the three address calculations are not commutative. Performing rotation followed by minification followed by translation results in a mathematical transformation different than Eq. 13.1-12. The overall results can be made identical by proper choice of the individual transformation parameters.

To obtain the reverse address calculation, it is necessary to invert Eq. 13.1-13*b* to solve for (u_p, v_q) in terms of (x_j, y_k). Because the matrix in Eq. 13.1-13*b* is not square, it does not possess an inverse. Although it is possible to obtain (u_q, v_p) by a pseudoinverse operation, it is convenient to augment the rectangular matrix as follows:

$$\begin{bmatrix} x_j \\ y_k \\ 1 \end{bmatrix} = \begin{bmatrix} c_0 & c_1 & c_2 \\ d_0 & d_1 & d_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_q \\ 1 \end{bmatrix} \tag{13.1-14}$$

This three-dimensional vector representation of a two-dimensional vector is a special case of a *homogeneous coordinates* representation (2–4).

The use of homogeneous coordinates enables a simple formulation of concatenated operators. For example, consider the rotation of an image by an angle θ about a pivot point (x_c, y_c) in the image. This can be accomplished by

$$\begin{bmatrix} x_j \\ y_k \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & x_c \\ 0 & 1 & y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & -x_c \\ 0 & 1 & -y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_q \\ 1 \end{bmatrix} \tag{13.1-15}$$

which reduces to a single 3×3 transformation:

$$\begin{bmatrix} x_j \\ y_k \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & -x_c\cos\theta + y_c\sin\theta + x_c \\ \sin\theta & \cos\theta & -x_c\sin\theta - y_c\cos\theta + y_c \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_q \\ 1 \end{bmatrix} \tag{13.1-16}$$

The reverse address computation for the special case of Eq. 13.1-16, or the more general case of Eq. 13.1-13, can be obtained by inverting the 3×3 transformation matrices by numerical methods. Another approach, which is more computationally efficient, is to initially develop the homogeneous transformation matrix in reverse order as

$$\begin{bmatrix} u_p \\ v_q \\ 1 \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 \\ b_0 & b_1 & b_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_j \\ y_k \\ 1 \end{bmatrix} \tag{13.1-17}$$

where for translation

$$a_0 = 1 \tag{13.1-18a}$$

$$a_1 = 0 \tag{13.1-18b}$$

$$a_2 = -t_x \tag{13.1-18c}$$

$$b_0 = 0 \tag{13.1-18d}$$

$$b_1 = 1 \tag{13.1-18e}$$

$$b_2 = -t_y \tag{13.1-18f}$$

and for scaling

$$a_0 = 1/s_x \tag{13.1-19a}$$

$$a_1 = 0 \tag{13.1-19b}$$

$$a_2 = 0 \tag{13.1-19c}$$

$$b_0 = 0 \tag{13.1-19d}$$

$$b_1 = 1/s_y \tag{13.1-19e}$$

$$b_2 = 0 \tag{13.1-19f}$$

and for rotation

$$a_0 = \cos\theta \tag{13.1-20a}$$

$$a_1 = \sin\theta \tag{13.1-20b}$$

$$a_2 = 0 \tag{13.1-20c}$$

$$b_0 = -\sin\theta \tag{13.1-20d}$$

$$b_1 = \cos\theta \tag{13.1-20e}$$

$$b_2 = 0 \tag{13.1-20f}$$

Address computation for a rectangular destination array $D(j, k)$ from a rectangular source array $S(p, q)$ of the same size results in two types of ambiguity: some pixels of $S(p, q)$ will map outside of $D(j, k)$; and some pixels of $D(j, k)$ will not be mappable from $S(p, q)$ because they will lie outside its limits. As an example, Figure 13.1-2 illustrates rotation of an image by 45° about its center. If the desire of the mapping is to produce a complete destination array $D(j, k)$, it is necessary to access a sufficiently large source image $S(p, q)$ to prevent mapping voids in $D(j, k)$. This is accomplished in Figure 13.1-2*d* by embedding the original image of Figure 13.1-2*a* in a zero background that is sufficiently large to encompass the rotated original.

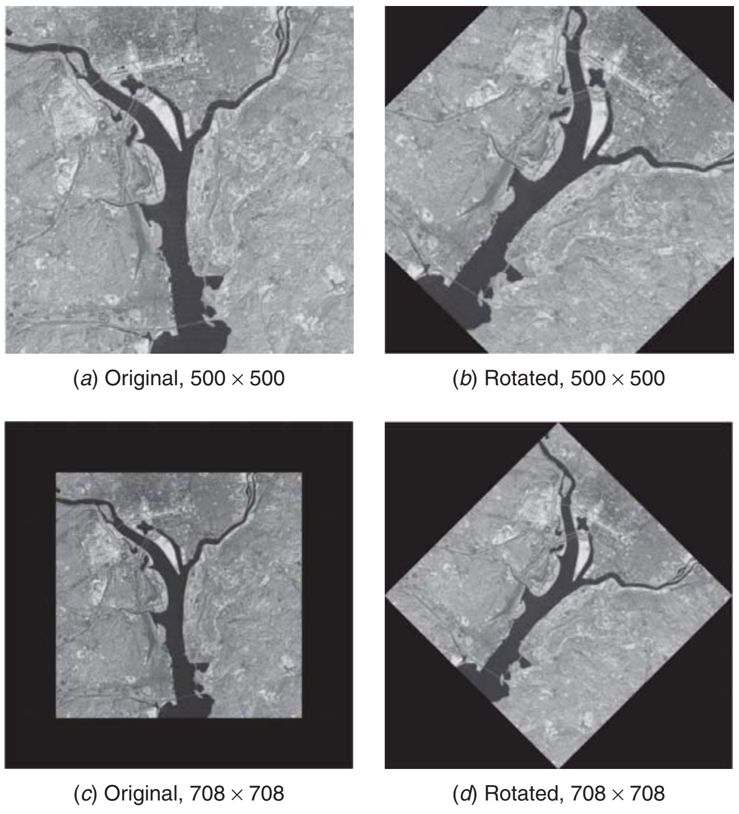

(*a*) Original, 500 × 500
(*b*) Rotated, 500 × 500
(*c*) Original, 708 × 708
(*d*) Rotated, 708 × 708

FIGURE 13.1-2. Image rotation by -45° on the `washington_ir` image about its center.

13.1.5. Affine Transformation

The geometrical operations of translation, size scaling and rotation are special cases of a geometrical operator called an *affine transformation*. It is defined by Eq. 13.1-13*b*, in which the constants c_i and d_i are general weighting factors. The affine transformation is not only useful as a generalization of translation, scaling and rotation. It provides a means of image shearing in which the rows or columns are successively uniformly translated with respect to one another. Figure 13.1-3 illustrates image shearing of rows of an image. In this example, $c_0 = d_1 = 1.0$, $c_1 = 0.1$, $d_0 = 0.0$ and $c_2 = d_2 = 0.0$.

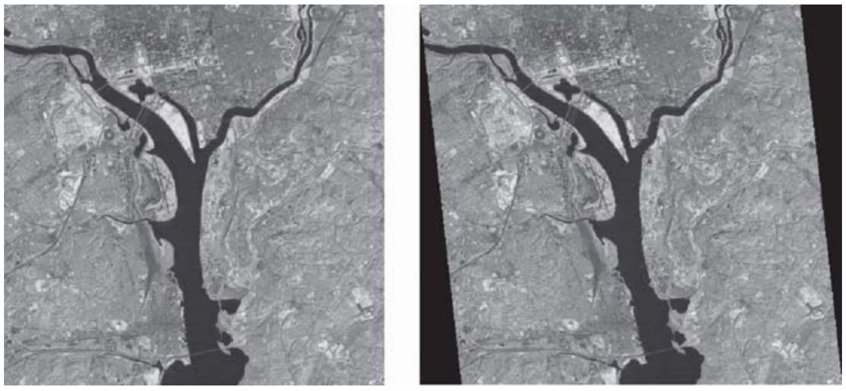


FIGURE 13.1-3. Horizontal image shearing on the `washington_ir` image.

13.1.6. Separable Rotation

The address mapping computations for translation and scaling are separable in the sense that the horizontal output image coordinate x_j depends only on u_p, and y_k depends only on v_q. Consequently, it is possible to perform these operations separably in two passes. In the first pass, a one-dimensional address translation is performed independently on each row of an input image to produce an intermediate array $I(j, q)$. In the second pass, columns of the intermediate array are processed independently to produce the final result $D(j, k)$.

Referring to Eq. 13.1-8, it is observed that the address computation for rotation is of a form such that x_j is a function of both u_p and v_q; and similarly for y_k. One might then conclude that rotation cannot be achieved by separable row and column processing, but Catmull and Smith (5) have demonstrated otherwise. In the first pass of the Catmull and Smith procedure, each row of $S(p, q)$ is mapped into the corresponding row of the intermediate array $I(j, q)$ using the standard row address computation of Eq. 13.1-8*a*. Thus

$$x_j = u_q \cos\theta - v_q \sin\theta \tag{13.1-21}$$

Then, each column of $I(j, q)$ is processed to obtain the corresponding column of $D(j, k)$ using the address computation

$$y_k = \frac{x_j \sin\theta + v_q}{\cos\theta} \tag{13.1-22}$$

Substitution of Eq. 13.1-21 into Eq. 13.1-22 yields the proper composite *y*-axis transformation of Eq. 13.1-8*b*. The "secret" of this separable rotation procedure is the ability to invert Eq. 13.1-21 to obtain an analytic expression for u_p in terms of x_j. In this case,

$$u_p = \frac{x_j + v_q \sin\theta}{\cos\theta} \tag{13.1-23}$$

when substituted into Eq. 13.1-21, gives the intermediate column warping function of Eq. 13.1-22.

The Catmull and Smith two-pass algorithm can be expressed in vector-space form as

$$\begin{bmatrix} x_j \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & 0 \\ \tan\theta & \dfrac{1}{\cos\theta} \end{bmatrix} \begin{bmatrix} \cos\theta & -\sin\theta \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_q \end{bmatrix} \tag{13.1-24}$$

The separable processing procedure must be used with caution. In the special case of a rotation of 90°, all of the rows of $S(p, q)$ are mapped into a single column of $I(p, k)$, and hence the second pass cannot be executed. This problem can be avoided by processing the columns of $S(p, q)$ in the first pass. In general, the best overall results are obtained by minimizing the amount of spatial pixel movement. For example, if the rotation angle is + 80°, the original should be rotated by +90° by conventional row–column swapping methods, and then that intermediate image should be rotated by –10° using the separable method.

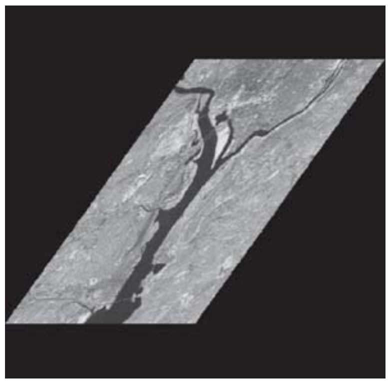
Figure 13.1-4 provides an example of separable rotation of an image by 45°. Figure 13.1-4*a* is the original, Figure 13.1-4*b* shows the result of the first pass and Figure 13.1-4*c* presents the final result.

Separable, two-pass rotation offers the advantage of simpler computation compared to one-pass rotation, but there are some disadvantages to two-pass rotation. Two-pass rotation causes loss of high spatial frequencies of an image because of the intermediate scaling step (6), as seen in Figure 13.1-4*b*. Also, there is the potential of increased aliasing error (6,7), as discussed in Section 13.5.

Several authors (6,8,9) have proposed a three-pass rotation procedure in which there is no scaling step and hence no loss of high-spatial-frequency content with proper interpolation. The vector-space representation of this procedure is given by

$$\begin{bmatrix} x_j \\ y_k \end{bmatrix} = \begin{bmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ \sin\theta & 1 \end{bmatrix} \begin{bmatrix} 1 & -\tan(\theta/2) \\ 0 & 1 \end{bmatrix} \begin{bmatrix} u_p \\ v_q \end{bmatrix} \qquad (13.1\text{-}25)$$

(*a*) Original


(*b*) First-pass result

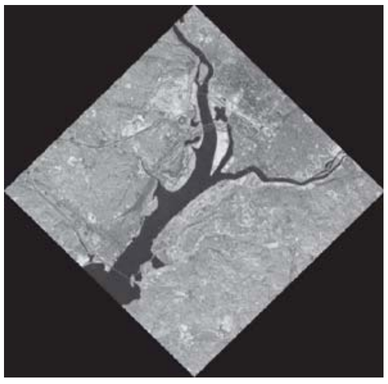

(*c*) Second-pass result

FIGURE 13.1-4. Separable two-pass image rotation on the `washington_ir` image.

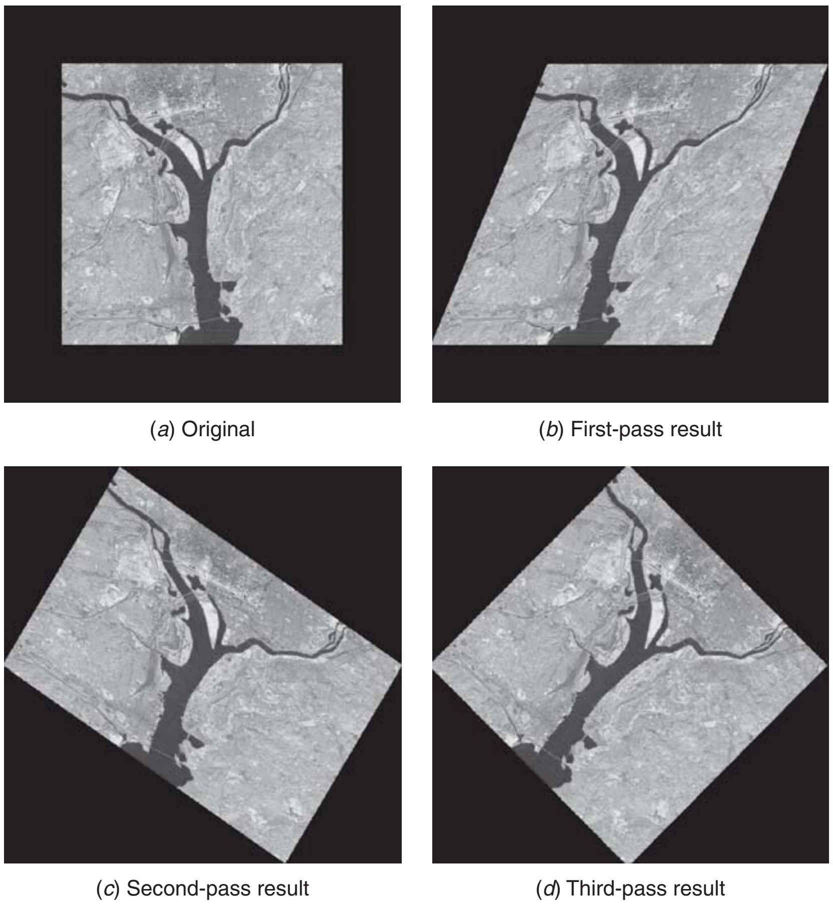


FIGURE 13.1-5. Separable three-pass image rotation on the `washington_ir` image.

This transformation is a series of image shearing operations without scaling. Figure 13.1-5 illustrates three-pass rotation for rotation by 45°.

13.1.7. Polar Coordinate Conversion

Certain imaging sensors, such as a scanning radar sensor and an ultrasound sensor, generate pie-shaped images in the spatial domain inset into a zero-value background. Some algorithms process such data by performing a *Cartesian-to-polar* coordinate conversion, manipulating the polar domain data and then performing an inverse *polar-to-Cartesian* coordinate conversion. Figure 13.1-6 illustrates the geometry of the Cartesian-to-polar conversion process. Upon completion of the conversion, the destination image will contain linearly scaled versions of the rho and theta polar domain values.

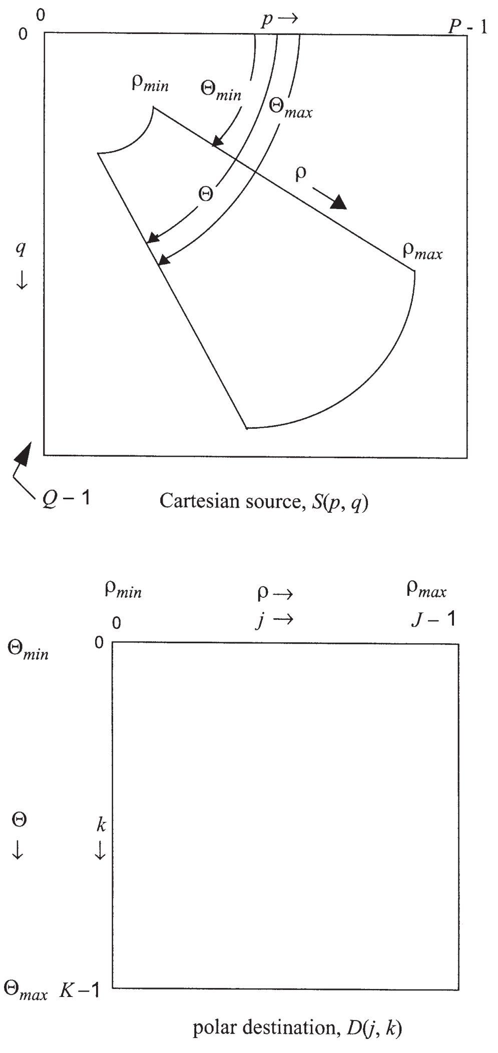


FIGURE 13.1-6. Relationship of source and destination images for Cartesian-to-polar conversion.

Cartesian-to-Polar. The Cartesian-to-polar conversion process involves an inverse address calculation for each destination image address. For each (*j*, *k*), compute

$$\rho(j) = \frac{j[\rho_{max} - \rho_{min}]}{J-1} + \rho_{min} \qquad (13.1\text{-}26a)$$

$$\theta(k) = \frac{k[\theta_{max} - \theta_{min}]}{K-1} + \theta_{min} \qquad (13.1\text{-}26b)$$

And for each $\rho(j)$ and $\theta(k)$ compute

$$p' = [\rho(j)]\cos\{\theta(k)\} \qquad (13.1\text{-}26c)$$

$$q' = [\rho(j)]\sin\{\theta(k)\} \qquad (13.1\text{-}26d)$$

For each (p', q') non-integer source address, interpolate its nearest neighbours, and transfer the interpolated pixel to $D(j, k)$.

Polar-to-Cartesian. The polar-to-Cartesian conversion process also involves an inverse address calculation. For each (*j*, *k*), compute

$$\rho(j) = \sqrt{j^2 + k^2} \qquad (13.1\text{-}27a)$$

$$\theta(k) = \operatorname{atan}\left\{\frac{k}{j}\right\} \qquad (13.1\text{-}27b)$$

And for each $\rho(j)$ and $\theta(k)$ compute

$$p' = \frac{[\rho(j) - \rho_{min}][P-1]}{\rho_{max} - \rho_{min}} \qquad (13.1\text{-}27c)$$

$$q' = \frac{[\theta(k) - \theta_{min}][Q-1]}{\theta_{max} - \theta_{min}} \qquad (13.1\text{-}27d)$$

Then interpolate $S(p', q')$ to obtain $D(j, k)$.

13.2. SPATIAL WARPING

The address computation procedures described in the preceding section can be extended to provide nonlinear spatial warping of an image. In the literature, this process is often called *rubber-sheet stretching* (16,17). Let

$$x = X(u, v) \tag{13.2-1a}$$

$$y = Y(u, v) \tag{13.2-1b}$$

denote the generalized forward address mapping functions from an input image to an output image. The corresponding generalized reverse address mapping functions are given by

$$u = U(x, y) \tag{13.2-2a}$$

$$v = V(x, y) \tag{13.2-2b}$$

For notational simplicity, the (j, k) and (p, q) subscripts have been dropped from these and subsequent expressions. Consideration is given next to some examples and applications of spatial warping.

The reverse address computation procedure given by the linear mapping of Eq. 13.1-17 can be extended to higher dimensions. A second-order polynomial warp address mapping can be expressed as

$$u = a_0 + a_1 x + a_2 y + a_3 x^2 + a_4 xy + a_5 y^2 \tag{13.2-3a}$$

$$v = b_0 + b_1 x + b_2 y + b_3 x^2 + b_4 xy + b_5 y^2 \tag{13.2-3b}$$

In vector notation,

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} a_0 & a_1 & a_2 & a_3 & a_4 & a_5 \\ b_0 & b_1 & b_2 & b_3 & b_4 & b_5 \end{bmatrix} \begin{bmatrix} 1 \\ x \\ y \\ x^2 \\ xy \\ y^2 \end{bmatrix} \tag{13.2-3c}$$

For first-order address mapping, the weighting coefficients (a_i, b_i) can easily be related to the physical mapping as described in Section 13.1. There is no simple physical counterpart for second address mapping. Typically, second-order and higher-order address mapping are performed to compensate for spatial distortion caused by a physical imaging system. For example, Figure 13.2-1 illustrates the effects of imaging a rectangular grid with an electronic camera that is subject to nonlinear pincushion or barrel distortion.

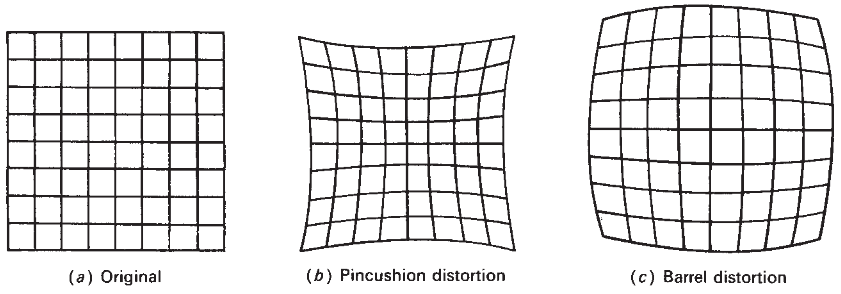


FIGURE 13.2-1. Geometric distortion.

Figure 13.2-2 presents a generalization of the problem. An ideal image $S(j, k)$ is subject to an unknown physical spatial distortion. The observed image is measured over a rectangular array $O(p, q)$. The objective is to perform a spatial correction warp to produce a corrected image array $\hat{S}(j, k)$. Assume that the address mapping from the ideal image space to the observation space is given by

$$u = O_u\{x, y\} \qquad (13.2\text{-}4a)$$

$$v = O_v\{x, y\} \qquad (13.2\text{-}4b)$$

where $O_u\{x, y\}$ and $O_v\{x, y\}$ are physical mapping functions. If these mapping functions are known, then Eq. 13.2-4 can, in principle, be inverted to obtain the proper corrective spatial warp mapping. If the physical mapping functions are not known, Eq. 13.2-3 can be considered as an estimate of the physical mapping functionsbased on the weighting coefficients (a_i, b_i). These polynomial weighting coefficients

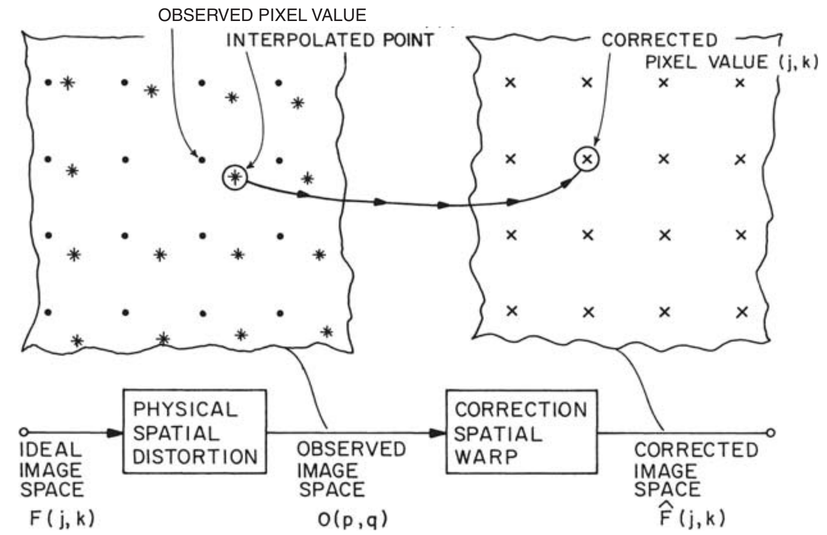

OBSERVED PIXEL VALUE
INTERPOLATED POINT
CORRECTED PIXEL VALUE (j, k)
PHYSICAL SPATIAL DISTORTION
CORRECTION SPATIAL WARP
IDEAL IMAGE SPACE F(j, k)
OBSERVED IMAGE SPACE O(p, q)
CORRECTED IMAGE SPACE F̂(j, k)

FIGURE 13.2-2. Spatial warping concept.

are normally chosen to minimize the mean-square error between a set of observation coordinates (u_m, v_m) and the polynomial estimates (u, v) for a set $(1 \leq m \leq M)$ of known data points (x_m, y_m) called *control points*. It is convenient to arrange the observation space coordinates into the vectors

$$\mathbf{u}^T = [u_1, u_2, \ldots, u_M] \tag{13.2-5a}$$

$$\mathbf{v}^T = [v_1, v_2, \ldots, v_M] \tag{13.2-5b}$$

Similarly, let the second-order polynomial coefficients be expressed in vector form as

$$\mathbf{a}^T = [a_0, a_1, \ldots, a_5] \tag{13.2-6a}$$

$$\mathbf{b}^T = [b_0, b_1, \ldots, b_5] \tag{13.2-6b}$$

The mean-square estimation error can be expressed in the compact form

$$\mathcal{E} = (\mathbf{u} - \mathbf{A}\mathbf{a})^T(\mathbf{u} - \mathbf{A}\mathbf{a}) + (\mathbf{v} - \mathbf{A}\mathbf{b})^T(\mathbf{v} - \mathbf{A}\mathbf{b}) \tag{13.2-7}$$

where

$$\mathbf{A} = \begin{bmatrix} 1 & x_1 & y_1 & x_1^2 & x_1 y_1 & y_1^2 \\ 1 & x_2 & y_2 & x_2^2 & x_2 y_2 & y_2^2 \\ & & & & & \\ & & & & & \\ 1 & x_M & y_M & x_M^2 & x_M y_M & y_M^2 \end{bmatrix} \tag{13.2-8}$$

From Appendix 1, it has been determined that the error will be minimum if

$$\mathbf{a} = \mathbf{A}^{-}\mathbf{u} \tag{13.2-9a}$$

$$\mathbf{b} = \mathbf{A}^{-}\mathbf{v} \tag{13.2-9b}$$

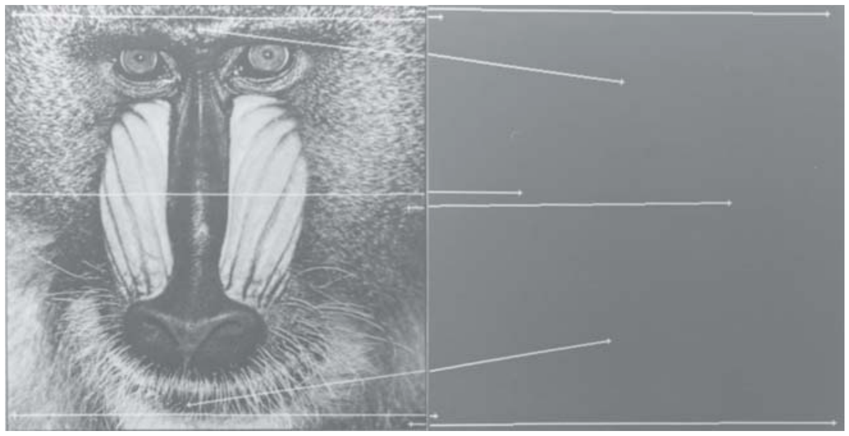
where $\mathbf{A}^{-}$ is the generalized inverse of $\mathbf{A}$. If the number of control points is chosen greater than the number of polynomial coefficients, then

$$\mathbf{A}^{-} = [\mathbf{A}^T\mathbf{A}]^{-1}\mathbf{A} \tag{13.2-10}$$

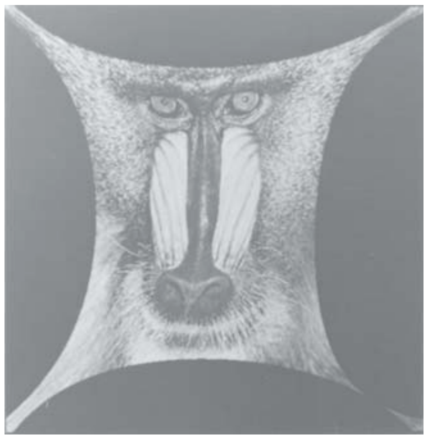
provided that the control points are not linearly related. Following this procedure, the polynomial coefficients (a_i, b_i) can easily be computed, and the address mapping of Eq. 13.2-1 can be obtained for all (j, k) pixels in the corrected image. Of course, proper interpolation is necessary.

Equation 13.2-3 can be extended to provide a higher-order approximation to the physical mapping of Eq. 13.2-3. However, practical problems arise in computing the pseudoinverse accurately for higher-order polynomials. For most applications, second-order polynomial computation suffices. Figure 13.2-3 presents an example of second-order polynomial warping of an image. In this example, the mapping of control points is indicated by the graphics overlay.

The spatial warping techniques discussed in this section have application for two types of geometrical image manipulation: *image mosaicing* and *image blending*. Image mosaicing involves the spatial combination of a set of partially overlapped images to create a larger image of a scene. Image blending is a process of creating a set of images between a temporal pair of images such that the created images form a smooth spatial interpolation between the reference image pair. References 11 to 15 provide details of image mosaicing and image blending algorithms.



(*a*) Source control points (*b*) Destination control points



(*c*) Warped

FIGURE 13.2-3. Second-order polynomial spatial warping on the `mandrill_mon` image.

13.3. PERSPECTIVE TRANSFORMATION

Most two-dimensional images are views of three-dimensional scenes from the physical perspective of a camera imaging the scene. It is often desirable to modify an observed image so as to simulate an alternative viewpoint. This can be accomplished by use of a *perspective transformation*.

Figure 13.3-1 shows a simple model of an imaging system that projects points of light in three-dimensional object space to points of light in a two-dimensional image plane through a lens focused for distant objects. Let (X, Y, Z) be the continuous domain coordinate of an object point in the scene, and let (x, y) be the continuous domain-projected coordinate in the image plane. The image plane is assumed to be at the center of the coordinate system. The lens is located at a distance f to the right of the image plane, where f is the focal length of the lens. By use of similar triangles, it is easy to establish that

$$x = \frac{fX}{f - Z} \tag{13.3-1a}$$

$$y = \frac{fY}{f - Z} \tag{13.3-1b}$$

Thus, the projected point (x, y) is related nonlinearly to the object point (X, Y, Z). This relationship can be simplified by utilization of homogeneous coordinates, as introduced to the image processing community by Roberts (1).

Let

$$\mathbf{v} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \tag{13.3-2}$$

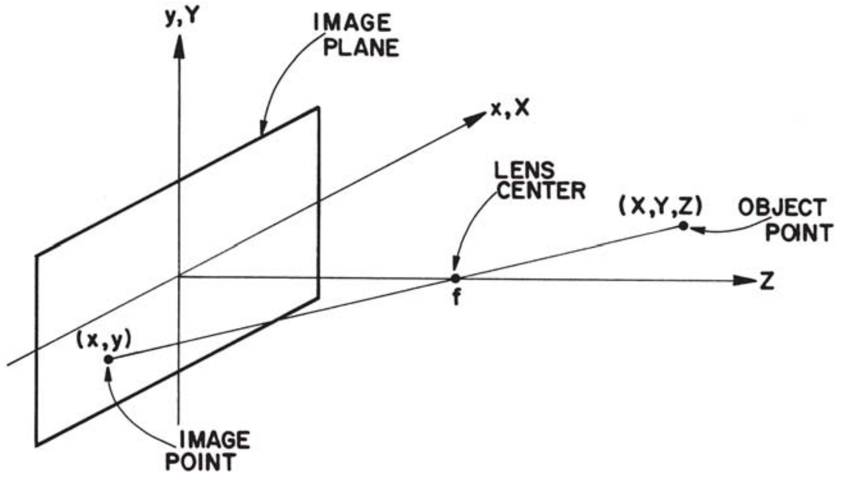


FIGURE 13.3-1. Basic imaging system model.

be a vector containing the object point coordinates. The homogeneous vector $\tilde{\mathbf{v}}$ corresponding to $\mathbf{v}$ is

$$\tilde{\mathbf{v}} = \begin{bmatrix} sX \\ sY \\ sZ \\ s \end{bmatrix} \tag{13.3-3}$$

where s is a scaling constant. The Cartesian vector $\mathbf{v}$ can be generated from the homogeneous vector $\tilde{\mathbf{v}}$ by dividing each of the first three components by the fourth. The utility of this representation will soon become evident.

Consider the following perspective transformation matrix:

$$\mathbf{P} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1/f & 1 \end{bmatrix} \tag{13.3-4}$$

This is a modification of the Roberts (1) definition to account for a different labeling of the axes and the use of column rather than row vectors. Forming the vector product

$$\tilde{\mathbf{w}} = \mathbf{P}\tilde{\mathbf{v}} \tag{13.3-5a}$$

yields

$$\tilde{\mathbf{w}} = \begin{bmatrix} sX \\ sY \\ sZ \\ s - sZ/f \end{bmatrix} \tag{13.3-5b}$$

The corresponding image plane coordinates are obtained by normalization of $\tilde{\mathbf{w}}$ to obtain

$$\mathbf{w} = \begin{bmatrix} \dfrac{fX}{f-Z} \\ \dfrac{fY}{f-Z} \\ \dfrac{fZ}{f-Z} \end{bmatrix} \tag{13.3-6}$$

It should be observed that the first two elements of $\mathbf{w}$ correspond to the imaging relationships of Eq. 13.3-1.

It is possible to project a specific image point (x_i, y_i) back into three-dimensional object space through an inverse perspective transformation

$$\tilde{\mathbf{v}} = \mathbf{P}^{-1}\tilde{\mathbf{w}} \tag{13.3-7a}$$

where

$$\mathbf{P}^{-1} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 1/f & 1 \end{bmatrix} \tag{13.3-7b}$$

and

$$\tilde{\mathbf{w}} = \begin{bmatrix} sx_i \\ sy_i \\ sz_i \\ s \end{bmatrix} \tag{13.3-7c}$$

In Eq. 13.3-7c, z_i is regarded as a free variable. Performing the inverse perspective transformation yields the homogeneous vector

$$\tilde{\mathbf{w}} = \begin{bmatrix} sx_i \\ sy_i \\ sz_i \\ s + sz_i/f \end{bmatrix} \tag{13.3-8}$$

The corresponding Cartesian coordinate vector is

$$\mathbf{w} = \begin{bmatrix} \dfrac{fx_i}{f - z_i} \\ \dfrac{fy_i}{f - z_i} \\ \dfrac{fz_i}{f - z_i} \end{bmatrix} \tag{13.3-9}$$

or equivalently,

$$x = \frac{fx_i}{f - z_i} \tag{13.3-10a}$$

$$y = \frac{fy_i}{f - z_i} \tag{13.3-10b}$$

$$z = \frac{fz_i}{f - z_i} \tag{13.3-10c}$$

Equation 13.3-10 illustrates the many-to-one nature of the perspective transformation. Choosing various values of the free variable z_i results in various solutions for (X, Y, Z), all of which lie along a line from (x_i, y_i) in the image plane through the lens center. Solving for the free variable z_i in Eq. 13.3-l0*c* and substituting into Eqs. 13.3-10*a* and 13.3-10*b* gives

$$X = \frac{x_i}{f}(f - Z) \tag{13.3-11a}$$

$$Y = \frac{y_i}{f}(f - Z) \tag{13.3-11b}$$

The meaning of this result is that because of the nature of the many-to-one perspective transformation, it is necessary to specify one of the object coordinates, say *Z*, in order to determine the other two from the image plane coordinates (x_i, y_i). Practical utilization of the perspective transformation is considered in the next section.

13.4. CAMERA IMAGING MODEL

The imaging model utilized in the preceding section to derive the perspective transformation assumed, for notational simplicity, that the center of the image plane was coincident with the center of the world reference coordinate system. In this section, the imaging model is generalized to handle physical cameras used in practical imaging geometries (18). This leads to two important results: a derivation of the fundamental relationship between an object and image point; and a means of changing a camera perspective by digital image processing.

Figure 13.4-1 shows an electronic camera in world coordinate space. This camera is physically supported by a gimbal that permits panning about an angle θ (horizontal movement in this geometry) and tilting about an angle ϕ (vertical movement). The gimbal center is at the coordinate (X_G, Y_G, Z_G) in the world coordinate system. The gimbal center and image plane center are offset by a vector with coordinates (X_o, Y_o, Z_o).

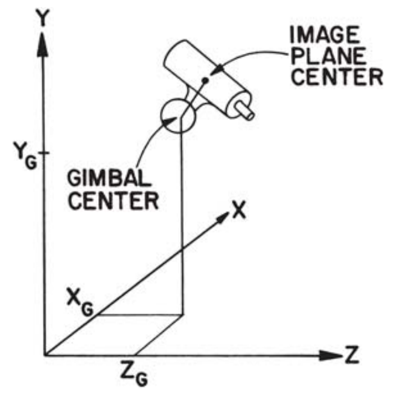


FIGURE 13.4-1. Camera imaging model.

If the camera were to be located at the center of the world coordinate origin, not panned nor tilted with respect to the reference axes, and if the camera image plane was not offset with respect to the gimbal, the homogeneous image model would be as derived in Section 13.3; that is

$$\tilde{\mathbf{w}} = \mathbf{P}\tilde{\mathbf{v}} \tag{13.4-1}$$

where $\tilde{\mathbf{v}}$ is the homogeneous vector of the world coordinates of an object point, $\tilde{\mathbf{w}}$ is the homogeneous vector of the image plane coordinates and $\mathbf{P}$ is the perspective transformation matrix defined by Eq. 13.3-4. The camera imaging model can easily be derived by modifying Eq. 13.4-1 sequentially using a three-dimensional extension of translation and rotation concepts presented in Section 13.1.

The offset of the camera to location (X_G, Y_G, Z_G) can be accommodated by the translation operation

$$\tilde{\mathbf{w}} = \mathbf{P}\mathbf{T}_G\tilde{\mathbf{v}} \tag{13.4-2}$$

where

$$\mathbf{T}_G = \begin{bmatrix} 1 & 0 & 0 & -X_G \\ 0 & 1 & 0 & -Y_G \\ 0 & 0 & 1 & -Z_G \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13.4-3}$$

Pan and tilt are modeled by a rotation transformation

$$\tilde{\mathbf{w}} = \mathbf{P}\mathbf{R}\mathbf{T}_G\tilde{\mathbf{v}} \tag{13.4-4}$$

where $\mathbf{R} = \mathbf{R}_\phi \mathbf{R}_\theta$ and

$$\mathbf{R}_\theta = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \sin\theta & \cos\theta & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13.4-5}$$

and

$$\mathbf{R}_\phi = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\phi & -\sin\phi & 0 \\ 0 & \sin\phi & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13.4-6}$$

The composite rotation matrix then becomes

$$\mathbf{R} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 & 0 \\ \cos\phi\sin\theta & \cos\phi\cos\theta & -\sin\phi & 0 \\ \sin\phi\sin\theta & \sin\phi\cos\theta & \cos\phi & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13.4-7}$$

Finally, the camera-to-gimbal offset is modeled as

$$\tilde{\mathbf{w}} = \mathbf{P}\mathbf{T}_C\mathbf{R}\mathbf{T}_G\tilde{\mathbf{v}} \tag{13.4-8}$$

where

$$\mathbf{T}_C = \begin{bmatrix} 1 & 0 & 0 & -X_o \\ 0 & 1 & 0 & -Y_o \\ 0 & 0 & 1 & -Z_o \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{13.4-9}$$

Equation 13.4-8 is the final result giving the complete camera imaging model transformation between an object and an image point. The explicit relationship between an object point (X, Y, Z) and its image plane projection (x, y) can be obtained by performing the matrix multiplications analytically and then forming the Cartesian coordinates by dividing the first two components of $\tilde{\mathbf{w}}$ by the fourth. Upon performing these operations, one obtains

$$x = \frac{f[(X - X_G)\cos\theta - (Y - Y_G)\sin\theta - X_0]}{-(X - X_G)\sin\theta\sin\phi - (Y - Y_G)\cos\theta\sin\phi - (Z - Z_G)\cos\phi + Z_0 + f} \tag{13.4-10a}$$

$$y = \frac{f\,[(X - X_G)\sin\theta\cos\phi + (Y - Y_G)\cos\theta\cos\phi - (Z - Z_G)\sin\phi - Y_0]}{-(X - X_G)\sin\theta\sin\phi - (Y - Y_G)\cos\theta\sin\phi - (Z - Z_G)\cos\phi + Z_0 + f} \tag{13.4-10b}$$

Equation 13.4-10 can be used to predict the spatial extent of the image of a physical scene on an imaging sensor.

Another important application of the camera imaging model is to form an image by postprocessing such that the image appears to have been taken by a camera at a different physical perspective. Suppose that two images defined by $\tilde{\mathbf{w}}_1$ and $\tilde{\mathbf{w}}_2$ are formed by taking two views of the same object with the same camera. The resulting camera model relationships are then

$$\tilde{\mathbf{w}}_1 = \mathbf{P}\mathbf{T}_C\mathbf{R}_1\mathbf{T}_{G1}\tilde{\mathbf{v}} \tag{13.4-11a}$$

$$\tilde{\mathbf{w}}_2 = \mathbf{P}\mathbf{T}_C\mathbf{R}_2\mathbf{T}_{G2}\tilde{\mathbf{v}} \tag{13.4-11b}$$

Because the camera is identical for the two images, the matrices $\mathbf{P}$ and $\mathbf{T}_C$ are invariant in Eq. 13.4-11. It is now possible to perform an inverse computation of Eq. 13.4-11*b* to obtain

$$\tilde{\mathbf{v}} = [\mathbf{T}_{G1}]^{-1}[\mathbf{R}_1]^{-1}[\mathbf{T}_C]^{-1}[\mathbf{P}]^{-1}\tilde{\mathbf{w}}_1 \tag{13.4-12}$$

and by substitution into Eq. 13.4-11*b*, it is possible to relate the image plane coordinates of the image of the second view to that obtained in the first view. Thus

$$\tilde{\mathbf{w}}_2 = \mathbf{P}\mathbf{T}_C\mathbf{R}_2\mathbf{T}_{G2}[\mathbf{T}_{G1}]^{-1}[\mathbf{R}_1]^{-1}[\mathbf{T}_C]^{-1}[\mathbf{P}]^{-1}\tilde{\mathbf{w}}_1 \tag{13.4-13}$$

As a consequence, an artificial image of the second view can be generated by performing the matrix multiplications of Eq. 13.4-13 mathematically on the physical image of the first view. Does this always work? No, there are limitations. First, if some portion of a physical scene were not "seen" by the physical camera, perhaps it was occluded by structures within the scene, then no amount of processing will recreate the missing data. Second, the processed image may suffer severe degradations resulting from undersampling if the two camera aspects are radically different. Nevertheless, this technique has valuable applications.

13.5. GEOMETRICAL IMAGE RESAMPLING

As noted in the preceding sections of this chapter, the reverse address computation process usually results in an address result lying between known pixel values of an input image. Thus, it is necessary to estimate the unknown pixel amplitude from its known neighbors. This process is related to the image reconstruction task, as described in Chapter 4, in which a space-continuous display is generated from an array of image samples. However, the geometrical resampling process is usually not spatially regular. Furthermore, the process is discrete to discrete; only one output pixel is produced for each input address.

In this section, consideration is given to the general geometrical resampling process in which output pixels are estimated by interpolation of input pixels. The special, but common case, of image magnification by an integer zooming factor is also discussed. In this case, it is possible to perform pixel estimation by convolution.

13.5.1. Interpolation Methods

The simplest form of resampling interpolation is to choose the amplitude of an output image pixel to be the amplitude of the input pixel nearest to the reverse address. This process, called *nearest-neighbor interpolation*, can result in a spatial offset error by as much as $1/\sqrt{2}$ pixel units. The resampling interpolation error can be significantly reduced by utilizing all four nearest neighbors in the interpolation. A common approach, called *bilinear interpolation*, is to interpolate linearly along each row of an image and then interpolate that result linearly in the columnar direction. Figure 13.5-1 illustrates the process. The estimated pixel is easily found to be

$$\begin{aligned}\hat{F}(p', q') = {} & (1-b)[(1-a)F(p, q) + aF(p+1, q)] \\ & + b[(1-a)F(p, q+1) + aF(p+1, q+1)] \end{aligned} \tag{13.5-1}$$

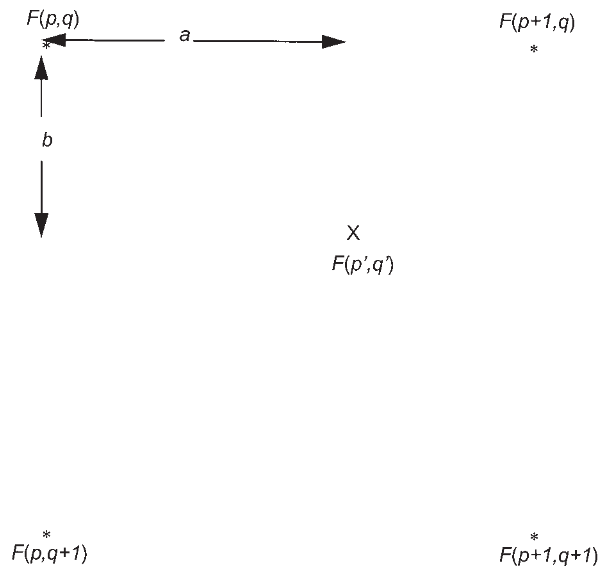


FIGURE 13.5-1. Bilinear interpolation.

where $a = p' - p$ and $b = q' - q$. Although the horizontal and vertical interpolation operations are each linear, in general, their sequential application results in a nonlinear surface fit between the four neighboring pixels.

The expression for bilinear interpolation of Eq. 13.5-1 can be generalized for any interpolation function $R\{x\}$ that is zero-valued outside the range of ± 1 sample spacing. With this generalization, interpolation can be considered as the summing of four weighted interpolation functions as given by

$$F(p', q') = F(p, q)R\{-b\}R\{a\} + F(p+1, q)R\{-b\}R\{-(1-a)\}$$
$$+ F(p, q+1)R\{1-b\}R\{a\} + F(p+1, q+1)R\{1-b\}R\{-(1-a)\} \quad (13.5\text{-}2)$$

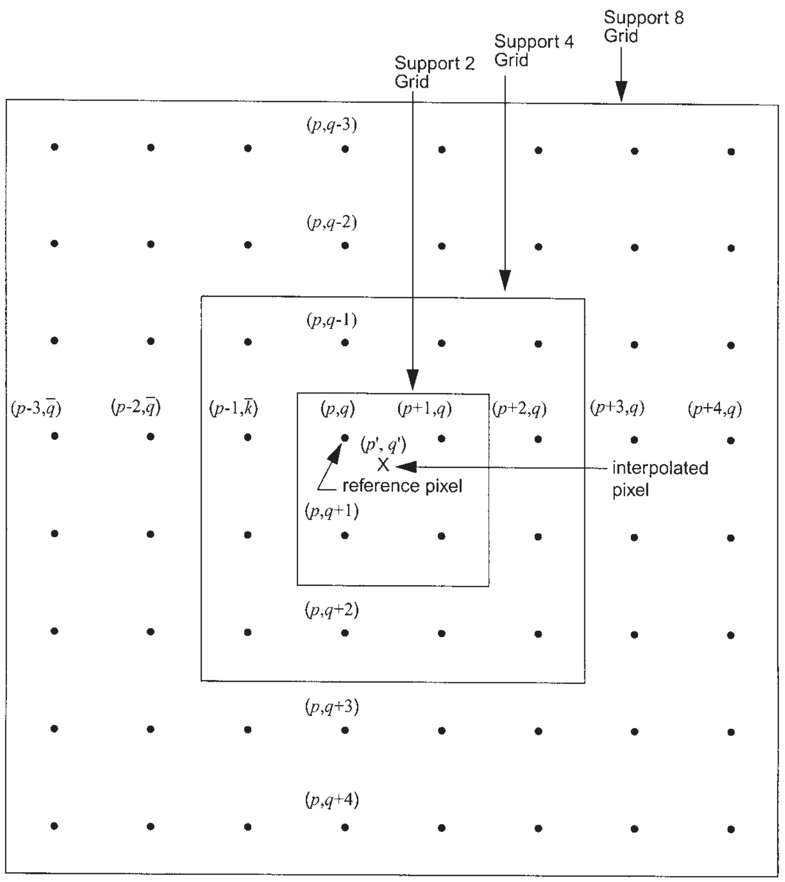


FIGURE 13.5-2. Support 2, 4 and 8 interpolation.

In the special case of linear interpolation, $R\{x\} = R_1\{x\}$, where $R_1\{x\}$ is defined in Eq. 4.3-2. Making this substitution, it is found that Eq. 13.5-2 is equivalent to the bilinear interpolation expression of Eq. 13.5-1.

Figure 13.5-2 defines a generalized interpolation neighborhood for support 2, 4 and 8 interpolation in which the pixel $F(p, q)$ is the nearest neighbor to the pixel to be interpolated.

Typically, for reasons of computational complexity, resampling interpolation is limited to a 4×4 pixel neighborhood. For this case, the interpolated pixel may be expressed in the compact form

$$F(p', q') = \sum_{m=-1}^{2} \sum_{n=-1}^{2} F(p+m, q+n) R_C\{(m-a)\} R_C\{-(n-b)\} \qquad (13.5\text{-}3)$$

where $R_C(x)$ denotes a bicubic interpolation function such as a cubic B-spline or cubic interpolation function, as defined in Section 4.3-2.

13.5.2. Convolution Methods

When an image is to be magnified by an integer zoom factor, pixel estimation can be implemented efficiently by convolution (19). As an example, consider image magnification by a factor of 2:1. This operation can be accomplished in two stages. First, the input image is transferred to an array in which rows and columns of zeros are interleaved with the input image data as follows:

$$\begin{bmatrix} A & B \\ C & D \end{bmatrix} \qquad \begin{bmatrix} A & 0 & B \\ 0 & 0 & 0 \\ C & 0 & D \end{bmatrix}$$

input image neighborhood

zero-interleaved neighborhood

Next, the zero-interleaved neighborhood image is convolved with one of the discrete interpolation kernels listed in Figure 13.5-3. Figure 13.5-4 presents the magnification results for several interpolation kernels. The inevitable visual trade-off between the interpolation error (the jaggy line artifacts) and the loss of high spatial frequency detail in the image is apparent from the examples.

This discrete convolution operation can easily be extended to higher-order magnification factors. For N:1 magnification, the core kernel is a $N \times N$ peg array. For large kernels it may be more computationally efficient in many cases, to perform the interpolation indirectly by Fourier domain filtering rather than by convolution.

For color images, the geometrical image modification methods discussed in this chapter can be applied separately to the red, green and blue components of the color image. Vrhel (20) has proposed converting a color image to luma/chroma (or lightness/chrominance) color coordinates and performing the geometrical modification in the converted color space. Large support interpolation is then performed on the luma or lightness component, and nearest neighbor interpolation is performed on the luma/chrominance components. After the geometrical processing is completed, conversion to RGB space is performed. This type of processing takes advantage of the tolerance of the human visual system to chroma or chrominance errors compared to luma/lightness errors.

Peg

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Pyramid

$$\frac{1}{4}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

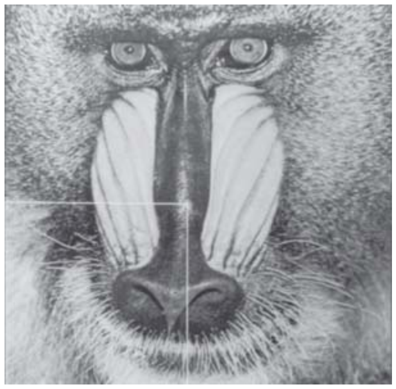
Bell

$$\frac{1}{16}\begin{bmatrix} 1 & 3 & 3 & 1 \\ 3 & 9 & 9 & 3 \\ 3 & 9 & 9 & 3 \\ 1 & 3 & 3 & 1 \end{bmatrix}$$

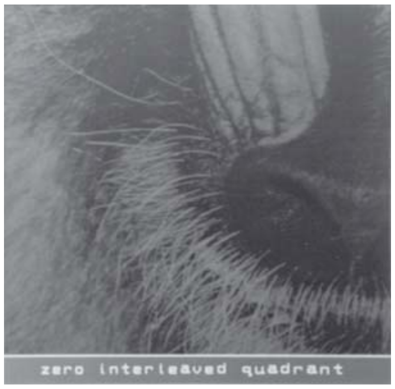
Cubic B-spline

$$\frac{1}{64}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

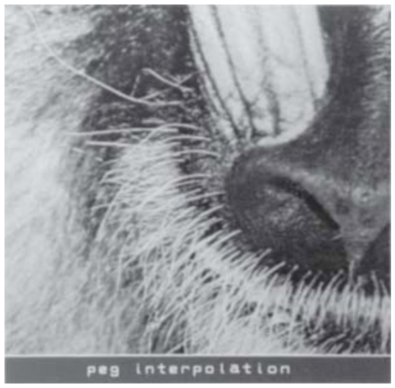
FIGURE 13.5-3. Interpolation kernels for 2:1 magnification.



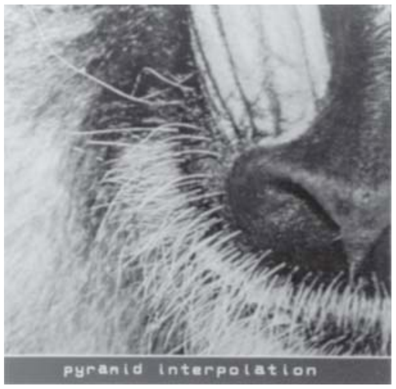
(*a*) Original



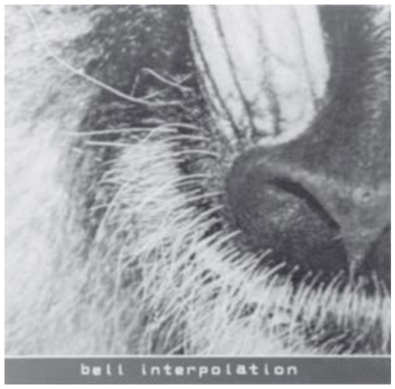
(*b*) Zero interleaved quadrant



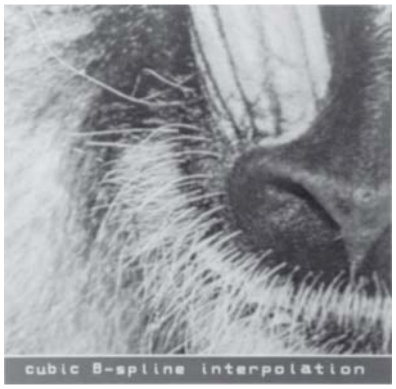
(*c*) Peg



(*d*) Pyramid



(*e*) Bell



(*f*) Cubic B-spline

FIGURE 13.5-4. Image interpolation on the `mandrill_mon` image for 2:1 magnification.

REFERENCES

1. G. Wolberg, *Digital Image Warping*, IEEE Computer Society Press, Washington DC, 1990.
2. L. G. Roberts, “Machine Perception of Three-Dimensional Solids,” in *Optical and Electro-Optical Information Processing*, J. T. Tippett et al., Eds., MIT Press, Cambridge, MA, 1965.
3. D. F. Rogers, *Mathematical Elements for Computer Graphics*, 2nd ed., McGraw-Hill, New York, 1989.
4. J. D. Foley et al., *Computer Graphics: Principles and Practice in C*, 2nd ed., Addison-Wesley, Reading, MA, 1996.
5. E. Catmull and A. R. Smith, “3-D Transformation of Images in Scanline Order,” *Computer Graphics*, *SIGGRAPH '80 Proc.,* **14**, 3, July 1980, 279–285.
6. M. Unser, P. Thevenaz and L. Yaroslavsky, “Convolution-Based Interpolation for Fast, High-Quality Rotation of Images, *IEEE Trans. Image Processing*, **IP-4**, 10, October 1995, 1371–1381.
7. D. Fraser and R. A. Schowengerdt, “Avoidance of Additional Aliasing in Multipass Image Rotations,” *IEEE Trans. Image Processing*, **IP-3**, 6, November 1994, 721–735.
8. A. W. Paeth, “A Fast Algorithm for General Raster Rotation,” *in Proc. Graphics Interface ‘86-Vision Interface*, 1986, 77–81.
9. P. E. Danielson and M. Hammerin, “High Accuracy Rotation of Images, in *CVGIP: Graphical Models and Image Processing*, **54**, 4, July 1992, 340–344.
10. M. R. Spillage and J. Liu, Schaum’s Mathematical Handbook of Formulas and Tables, 2nd ed., McGraw-Hill 1998.
11. D. L. Milgram, “Computer Methods for Creating Photomosaics,” *IEEE Trans. Computers*, **24**, 1975, 1113–1119.
12. D. L. Milgram, “Adaptive Techniques for Photomosaicing,” *IEEE Trans. Computers*, **26**, 1977, 1175–1180.
13. S. Peleg, A. Rav-Acha and A. Zomet, “Mosaicing on Adaptive Manifolds,” *IEEE Trans. Pattern Analysis and Machine Intelligence*, **22**, 10, October 2000, 1144–1154.
14. H. Nicolas, “New Methods for Dynamic Mosaicking,” *IEEE Trans. Image Processing,* **10**, 8, August 2001, 1239–1251.
15. R. T. Whitaker, “A Level-Set Approach to Image Blending, *IEEE Trans. Image Processing,* **9**, 11, November 2000, 1849–1861.
16. R. Bernstein, “Digital Image Processing of Earth Observation Sensor Data,” *IBM J. Research and Development*, **20**, 1, 1976, 40–56.
17. D. A. O’Handley and W. B. Green, “Recent Developments in Digital Image Processing at the Image Processing Laboratory of the Jet Propulsion Laboratory,” *Proc. IEEE*, **60**, 7, July 1972, 821–828.
18. K. S. Fu, R. C. Gonzalez and C. S. G. Lee, *Robotics: Control, Sensing, Vision and Intelligence*, McGraw-Hill, New York, 1987.

19. W. K. Pratt, “Image Processing and Analysis Using Primitive Computational Elements,” in *Selected Topics in Signal Processing*, S. Haykin, Ed., Prentice Hall, Englewood Cliffs, NJ, 1989.

20. M. Vrhel, “Color Image Resolution Conversion,” *IEEE Trans. Image Processing,* **14**, 3, March 2005, 328–333.