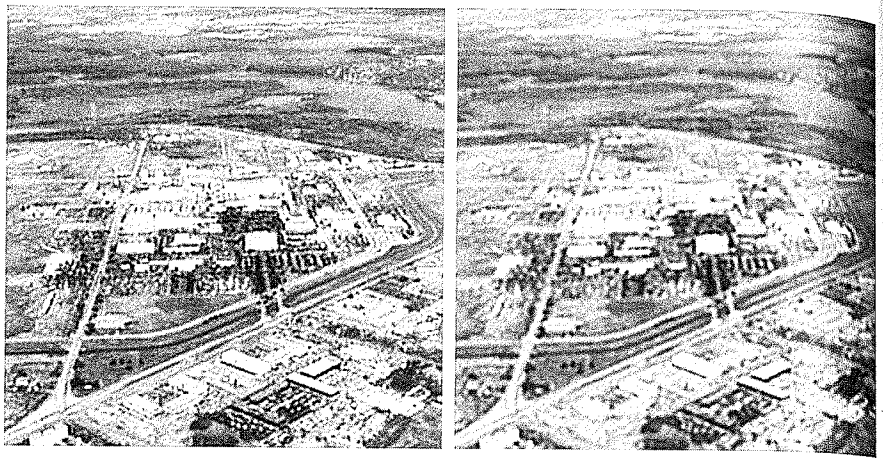


a b

FIGURE 5.31
(a) Iteratively determined constrained least squares restoration of Fig. 5.16(b), using correct noise parameters. (b) Result obtained with wrong noise parameters.

5.9 Geometric Mean Filter

It is possible to generalize slightly the Wiener filter discussed in the Section 5.8. The generalization is in the form of the so-called *geometric mean filter*:

$$\hat{F}(u,v) = \left[\frac{H^*(u,v)}{|H(u,v)|^2}\right]^{\alpha}\left[\frac{H^*(u,v)}{|H(u,v)|^2 + \beta\left[\dfrac{S_\eta(u,v)}{S_f(u,v)}\right]}\right]^{1-\alpha} G(u,v) \qquad (5.10\text{-}1)$$

with α and β being positive, real constants. The geometric mean filter consists of the two expressions in brackets raised to the powers α and $1 - \alpha$, respectively.

When $\alpha = 1$ this filter reduces to the inverse filter. With $\alpha = 0$ the filter becomes the so-called *parametric Wiener filter*, which reduces to the standard Wiener filter when $\beta = 1$. If $\alpha = 1/2$, the filter becomes a product of the two quantities raised to the same power, which is the definition of the geometric mean, thus giving the filter its name. With $\beta = 1$, as α decreases below $1/2$, the filter performance will tend more toward the inverse filter. Similarly, when α increases above $1/2$, the filter will behave more like the Wiener filter. When $\alpha = 1/2$ and $\beta = 1$, the filter also is commonly referred to as the *spectrum equalization filter*. Equation (5.10-1) is quite useful when implementing restoration filters because it really represents a family of filters combined into a single expression.

5.10 Geometric Transformations

We conclude this chapter with an introductory discussion on the use of geometric transformations for image restoration. Unlike the techniques discussed so far, geometric transformations modify the spatial relationships between pixels in an image. Geometric transformations often are called *rubber-sheet transformations*, because they may be viewed as the process of "printing" an

image on a sheet of rubber and then stretching this sheet according to some predefined set of rules.

In terms of digital image processing, a geometric transformation consists of two basic operations: (1) a *spatial transformation*, which defines the "rearrangement" of pixels on the image plane; and (2) *gray-level interpolation*, which deals with the assignment of gray levels to pixels in the spatially transformed image. We discuss in the following sections the fundamental ideas underlying these concepts, and their use in the context of image restoration.

5.11.1 Spatial Transformations

Suppose that an image f with pixel coordinates (x, y) undergoes geometric distortion to produce an image g with coordinates (x', y'). This transformation may be expressed as

$$x' = r(x, y) \tag{5.11-1}$$

and

$$y' = s(x, y) \tag{5.11-2}$$

where $r(x, y)$ and $s(x, y)$ are the spatial transformations that produced the geometrically distorted image $g(x', y')$. For example, if $r(x, y) = x/2$ and $s(x, y) = y/2$, the "distortion" is simply a shrinking of the size of $f(x, y)$ by one-half in both spatial directions.

If $r(x, y)$ and $s(x, y)$ were known analytically, recovering $f(x, y)$ from the distorted image $g(x', y')$ by applying the transformations in reverse might be possible theoretically. In practice, however, formulating a single set of analytical functions $r(x, y)$ and $s(x, y)$ that describe the geometric distortion process over the entire image plane generally is not possible. The method used most frequently to overcome this difficulty is to formulate the spatial relocation of pixels by the use of *tiepoints*, which are a subset of pixels whose location in the input (distorted) and output (corrected) images is known precisely.

Figure 5.32 shows quadrilateral regions in a distorted and corresponding corrected image. The vertices of the quadrilaterals are corresponding tiepoints.

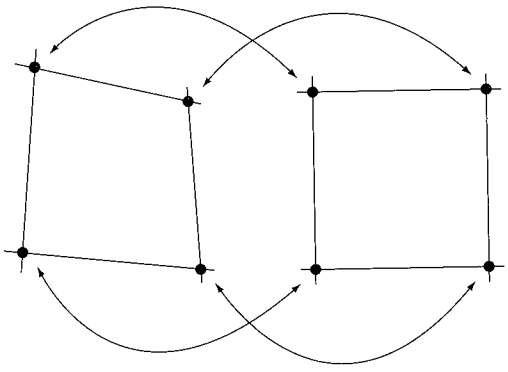


FIGURE 5.32 Corresponding tiepoints in two image segments.

Suppose that the geometric distortion process within the quadrilateral regions is modeled by a pair of bilinear equations so that

$$r(x, y) = c_1x + c_2y + c_3xy + c_4 \tag{5.11-3}$$

and

$$s(x, y) = c_5x + c_6y + c_7xy + c_8. \tag{5.11-4}$$

Then, from Eqs. (5.11-1) and (5.11-2),

$$x' = c_1x + c_2y + c_3xy + c_4 \tag{5.11-5}$$

and

$$y' = c_5x + c_6y + c_7xy + c_8. \tag{5.11-6}$$

Since there are a total of eight known tiepoints, these equations can be solved for the eight coefficients c_i, $i = 1, 2, \ldots, 8$. The coefficients constitute the geometric distortion model used to transform *all* pixels within the quadrilateral region defined by the tiepoints used to obtain the coefficients. In general, enough tiepoints are needed to generate a set of quadrilaterals that cover the entire image, with each quadrilateral having its own set of coefficients.

Once we have the coefficients, the procedure used to generate the corrected (i.e., restored) image is not difficult. If we want to find the value of the undistorted image at any point (x_0, y_0), we simply need to know where in the distorted image $f(x_0, y_0)$ was mapped. This we find out by substituting (x_0, y_0) into Eqs. (5.11-5) and (5.11-6) to obtain the geometrically distorted coordinates (x_0', y_0'). The value of the point in the undistorted image that was mapped to (x_0', y_0') is $g(x_0', y_0')$. So we obtain the restored image value simply by letting $\hat{f}(x_0, y_0) = g(x_0', y_0')$. For example, to generate $\hat{f}(0, 0)$, we substitute $(x, y) = (0, 0)$ into Eqs. (5.11-5) and (5.11-6) to obtain a pair of coordinates (x', y') from those equations. Then we let $\hat{f}(0, 0) = g(x', y')$, where x' and y' are the coordinate values just obtained. Next, we substitute $(x, y) = (0, 1)$ into Eqs. (5.11-5) and (5.11-6), obtain another pair of values (x', y'), and let $f(0, 1) = g(x', y')$ for those coordinate values. The procedure continues pixel by pixel and row by row until an array whose size does not exceed the size of image g is obtained. A column (rather than a row) scan would yield identical results. Also, a bookkeeping procedure is needed to keep track of which quadrilaterals apply at a given pixel location in order to use the proper coefficients.

Tiepoints are established by a number of different techniques, depending on the application. For instance, some image generation systems having physical artifacts (such as metallic points) embedded on the imaging sensor itself. These produce a *known* set of points (called *reseau marks*) directly on the image as it is acquired. If the image is distorted later by some other process (such as an image display or image reconstruction process), then the image can be geometrically corrected using the technique just described.

5.11.2 Gray-Level Interpolation

The method discussed in the preceding section steps through integer values of the coordinates (x, y) to yield the restored image $\hat{f}(x, y)$. However, depending on the values of the coefficients c_i, Eqs. (5.11-5) and (5.11-6) can yield noninteger val-

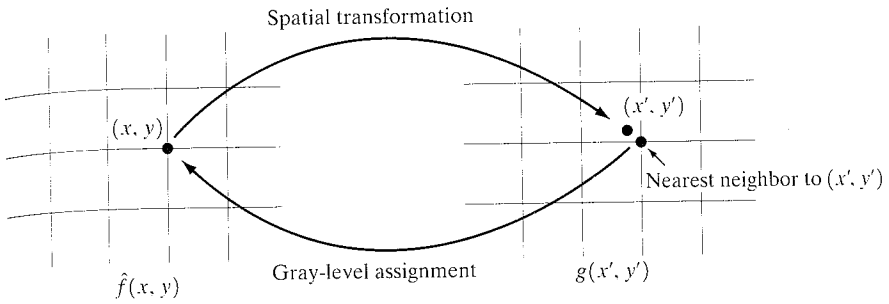


FIGURE 5.33 Gray-level interpolation based on the nearest neighbor concept.

ues for x' and y'. Because the distorted image g is digital, its pixel values are defined only at integer coordinates. Thus using noninteger values for x' and y' causes a mapping into locations of g for which no gray levels are defined. Inferring what the gray-level values at those locations should be, based only on the pixel values at integer coordinate locations, then becomes necessary. The technique used to accomplish this is called *gray-level interpolation*.

The simplest scheme for gray-level interpolation is based on a nearest neighbor approach. This method, also called *zero-order interpolation*, is illustrated in Fig. 5.33. This figure shows (1) the mapping of integer (x, y) coordinates into fractional coordinates (x', y') by means of Eqs. (5.11-5) and (5.11-6); (2) the selection of the closest integer coordinate neighbor to (x', y'); and (3) the assignment of the gray level of this nearest neighbor to the pixel located at (x, y).

Although nearest neighbor interpolation is simple to implement, this method often has the drawback of producing undesirable artifacts, such as distortion of straight edges in images of high resolution. Smoother results can be obtained by using more sophisticated techniques, such as *cubic convolution interpolation*, which fits a surface of the $\sin(z)/z$ type through a much larger number of neighbors (say, 16) in order to obtain a smooth estimate of the gray level at any desired point. Typical areas in which smoother approximations generally are required include 3-D graphics (Watt [1993]) and medical imaging (Lehman et al. [1999]). The price paid for smoother approximations is additional computational burden. For general-purpose image processing a *bilinear interpolation* approach that uses the gray levels of the four nearest neighbors usually is adequate. This approach is straightforward. Because the gray level of each of the four integral nearest neighbors of a nonintegral pair of coordinates (x', y') is known, the gray-level value at these coordinates, denoted $v(x', y')$, can be interpolated from the values of its neighbors by using the relationship

$$v(x', y') = ax' + by' + cx'y' + d \tag{5.11-7}$$

where the four coefficients are easily determined from the four equations in four unknowns that can be written using the four known neighbors of (x', y'). When these coefficients have been determined, $v(x', y')$ is computed and this

value is assigned to the location in $f(x, y)$ that yielded the spatial mapping into location (x', y'). It is easy to visualize this procedure with the aid of Fig. 5.33. The exception is that, instead of using the gray-level value of the nearest neighbor to (x', y'), we actually interpolate a value at location (x', y') and use this value for the gray-level assignment at (x, y).

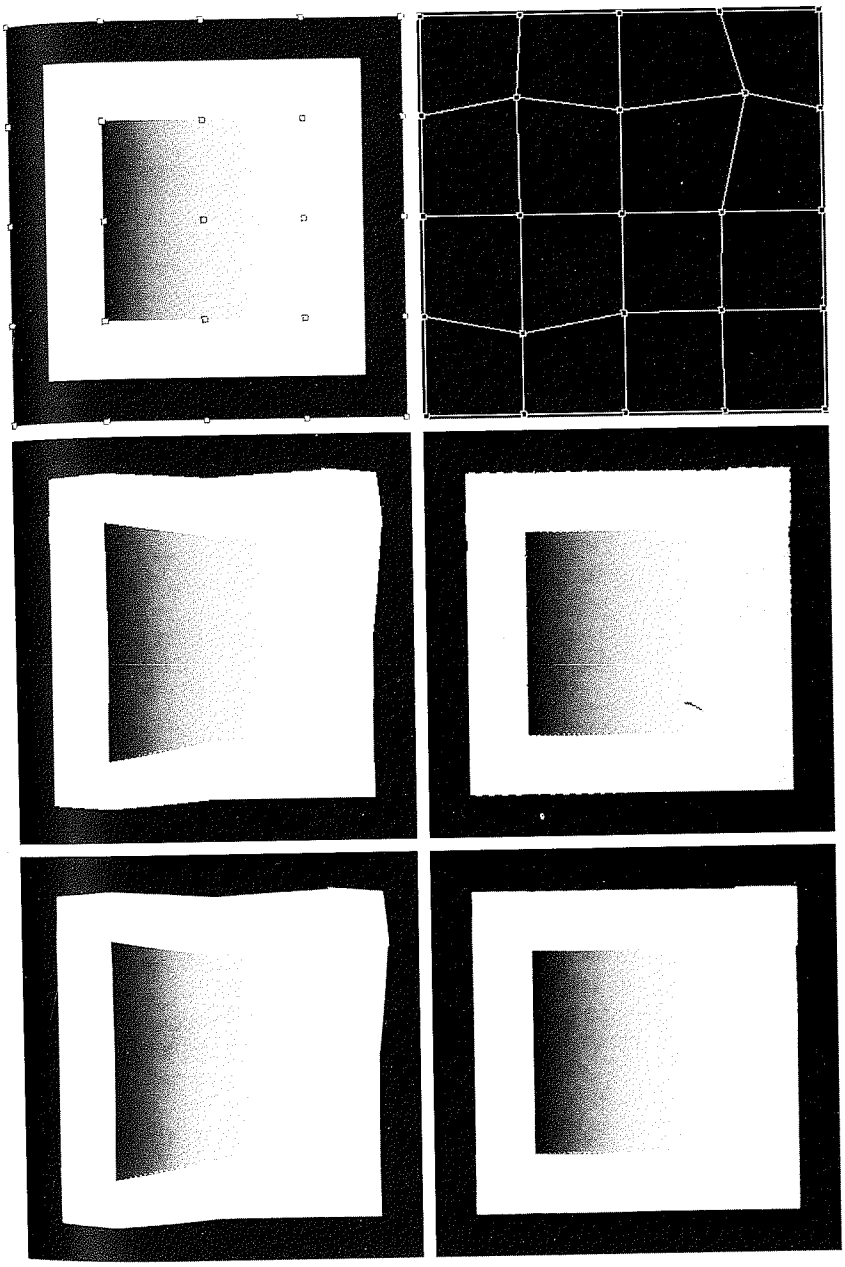
EXAMPLE 5.16: Illustration of geometric transformations.

Figure 5.34(a) shows an image with 25 regularly spaced tiepoints (highlighted to enhance visibility of the points in the picture). Figure 5.34(b) shows a simple rearrangement of the tiepoints to create geometric distortion. With reference to the procedure discussed in connection with Eqs. (5.11-5) and (5.11-6), the coefficients of these equations are a result of the mapping from the undistorted to the distorted coordinates. Once the coefficients are known, we have the model, and we can either distort an image (for demonstration purposes) or we can recover an image that was geometrically distorted under the set of conditions defined by the coefficients.

Suppose that we want to distort the image in Fig. 5.34(a). We simply substitute the value of any pixel (x_0, y_0) from that image into Eqs. (5.11-5) and (5.11-6) and generate the corresponding coordinates (x_0', y_0'), which we round off to the closest integer values. The value of the distorted image at that point is given by letting $g(x_0', y_0') = f(x_0, y_0)$, or we can use gray-level interpolation on the values of f in the neighborhood of (x_0, y_0). This is the same process described in connection with Eqs. (5.11-5) and (5.11-6). We are simply applying it in reverse.

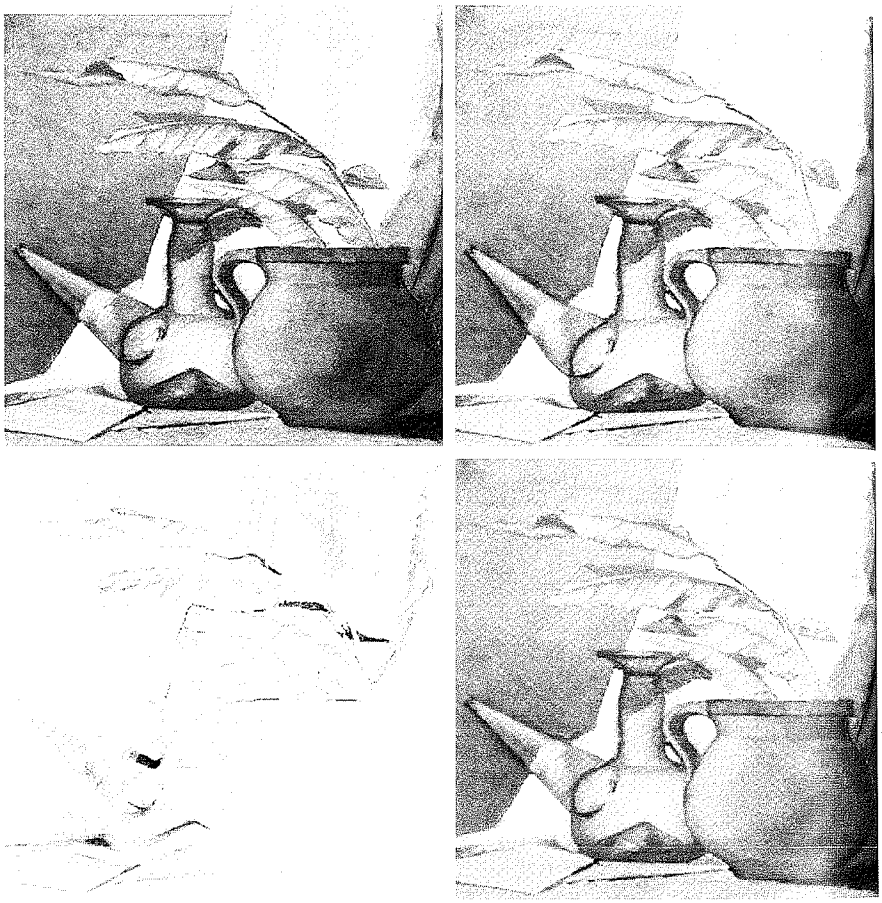
The result of distorting Fig. 5.34(a) by the method just discussed is shown in Fig. 5.34(c), where the nearest neighbor gray-level assignment scheme was used. Note that this is fairly severe distortion. If this were the given image, we would use the method discussed in connection with Eqs. (5.11-5) and (5.11-6), and one of the gray-level assignment techniques discussed in Section 5.11.2. The result of this procedure is shown in Fig. 5.34(d). The nearest neighbor gray-level assignment method was employed again. Note that the geometric correction was reasonable, but there is a significant number of errors in gray-level assignments, especially along the boundaries between the gray and black regions. Figures 5.34(e) and (f) show the same sequence of experiments, but using bilinear gray-level interpolation instead. The improvements are particularly visible in the boundaries between the gray and black regions.

The images just discussed are so regular and have such few gray levels in the sharp boundaries that almost any type of geometric distortion will cause significant degradation. When images have more texture, geometric correction errors tend to be less noticeable. For example, consider Fig. 5.35. Figure 5.35(b) is the result of geometrically distorting Fig. 5.35(a) in the same manner as Fig. 5.34(e). This distortion in Fig. 5.35(b) is not nearly as noticeable. The differences between Figs. 5.35(a) and (b) are not insignificant, as the difference image in Fig. 5.35(c) shows. They simply are not as visible because of the variety of texture in this image. Finally, Fig. 5.35(d) shows the geometrically corrected image. For all practical purposes, this image is of the same quality as the original.



a b
c d
e f

FIGURE 5.34 (a) Image showing tiepoints. (b) Tiepoints after geometric distortion. (c) Geometrically distorted image, using nearest neighbor interpolation. (d) Restored result. (e) Image distorted using bilinear interpolation. (f) Restored image.



a b
c d

FIGURE 5.35 (a) An image before geometric distortion. (b) Image geometrically distorted using the same parameters as in Fig. 5.34(e). (c) Difference between (a) and (b). (d) Geometrically restored image.

Summary

The principal results in this chapter are based on the assumption that image degradation can be modeled as a linear, position-invariant process followed by additive noise that is not correlated with image values. Even when these assumptions are not entirely valid, it often is possible to obtain useful results by using the methods developed in the preceding sections.

Some of the restoration techniques derived in this chapter are based on various criteria of optimality. The use of the word *optimal* in this context refers strictly to a mathematical concept, not to optimal response of the human visual system. In fact, the present lack of knowledge about visual perception precludes a general formulation of the image restoration problem that takes into account observer preferences and capabilities. In

view of these limitations, the advantage of the concepts introduced in this chapter is the development of fundamental approaches that have reasonably predictable behavior and are supported by a solid body of knowledge.

As in Chapters 3 and 4, certain restoration tasks, such as random noise reduction, are carried out in the spatial domain using convolution masks. The frequency domain was found ideal for reducing periodic noise and for modeling some important degradations, such as blur caused by motion during image acquisition. We also found the frequency domain to be a useful tool for formulating restoration filters, such as the Wiener and constrained least squares filters.

As mentioned in Chapter 4, the frequency domain offers an intuitive, solid base for experimentation. Once an approach (filter) has been found to perform satisfactorily for a given application, implementation usually is carried out via the design of a digital filter that approximates the frequency-domain solution, but runs much faster in a computer or in a dedicated hardware/firmware system. Digital filter design is beyond the scope of this book, but references relevant to this topic are included in the section that follows.

References and Further Reading

For additional reading on the linear model of degradation presented in Section 5.1, see Castleman [1996] and Pratt [1991]. The book by Peebles [1993] provides an intermediate-level coverage of noise probability density functions and their properties (Section 5.2). The book by Papoulis [1991] is more advanced and covers these concepts in more detail. References for Section 5.3 are Umbaugh [1998], Boie and Cox [1992], Hwang and Haddad [1995], Wilburn [1998], and Eng and Ma [2001]. The general area of adaptive filter design is good background for the adaptive filters discussed in Section 5.3. The book by Haykin [1996] is a good introduction to this topic. The filters in Section 5.4 are direct extensions of the material in Chapter 4. For additional reading on the material of Section 5.5, see Rosenfeld and Kak [1982] and Pratt [1991].

The topic of estimating the degradation function (Section 5.6) is an area of considerable current interest. Some of the early techniques for estimating the degradation function are given in Andrews and Hunt [1977], Rosenfeld and Kak [1982], Bates and McDonnell [1986], and Stark [1987]. Since the degradation function seldom is known exactly, a number of techniques have been proposed over the years, in which specific aspects of restoration are emphasized. For example, Geman and Reynolds [1992], and Hurn and Jennison [1996], deal with issues of preserving sharp transitions in gray levels in an attempt to emphasize sharpness, while Boyd and Meloche [1998] are concerned with restoring thin objects in degraded images. Examples of techniques that deal with image blur are Yitzhaky et al. [1998], Harikumar and Bresler [1999], Mesarović [2000], and Giannakis and Heath [2000]. Restoration of sequences of images also is of considerable interest. The book by Kokaram [1998] provides a good foundation in this area.

The filtering approaches discussed in Sections 5.7 through 5.10 have been explained in various way over the years in numerous books and articles on image processing. There are two major approaches underpinning the development of these filters. One is based on a general formulation using matrix theory, as introduced by Andrews and Hunt [1977]. This approach is elegant and general, but it is difficult for newcomers to the field because it lacks intuitiveness. Approaches based directly on frequency domain filtering (the approach we followed in this chapter) usually are easier to follow by those who first encounter restoration, but lack the unifying mathematical rigor of the matrix approach. Both approaches arrive at the same results, but our experience in teaching this material in a variety of settings indicates that students first entering this field favor the latter approach by a significant margin. Complementary readings for our coverage of the