IN THE CIRCUIT COURT OF THE SIXTH JUDICIAL CIRCUIT
OF THE STATE OF FLORIDA IN AND FOR PASCO COUNTY
CRC14-00216CFAES

STATE OF FLORIDA

V.

CURTIS J. REEVES

Filed For Record
Pasco County, Florida
2021 NOV 23 PM 12:02
Nikki Alvarez-Sowles
Clerk & Comptroller
Pasco County, Florida

SUPPLEMENT TO STATE'S RESPONSE TO DEFENDANT'S MOTION TO EXCLUDE PROOF AND TESTIMONY PERTAINING TO THE STATE'S FORENSIC VIDEO EXPERT ANTHONY IMEL

COMES NOW, BRUCE BARTLETT, State Attorney for the Sixth Judicial Circuit in and for Pasco County, Florida, by and through the undersigned Assistant State Attorney, hereby files this *Supplement To State's Response to the Defendant's Motion To Exclude Proof And Testimony Pertaining To The State's Forensic Video Expert Anthony Imel* as follows:

ADDITIONAL EXHIBITS

Exhibit #2 Excerpt - *Forensic Photoshop, A comprehensive Imaging Workflow for Forensic Professionals*, by Jim Hoerricks, Section 10 - Interpolation, pages 96-98.

Exhibit #3 Excerpt - *Digital Image Processing*, Third Edition by Rafael C. Gonzalez and Richard E. Woods, Chapter 2 Digital Image Fundamentals, pages 65 - 103.

Exhibit #4 Excerpt - *The Image Processing Handbook*, Fourth Edition by John C. Russ, Chapter 3: Correcting Imaging Defects, pages 197 - 205

Exhibit #5 Excerpt - *Photoshop CS3 for Forensics Professionals, A Complete Digital Imaging Course for Investigators*, pages 80 - 82.

CERTIFICATE OF SERVICE

I HEREBY CERTIFY that a copy of the foregoing <u>Supplement To</u>

State's Response To Defendant's Motion To Exclude Proof And Testimony Pertaining To The State's Forensic Video Expert Anthony Imel was furnished to Richard Escobar, Esq., Attorney for the Defendant, at 2917 West Kennedy Blvd., Suite 100, Tampa, FL 33609-3163, by U.S. Mail/Personal Service/email rescobar@escobarlaw.com, this 17th day of November, 2021.

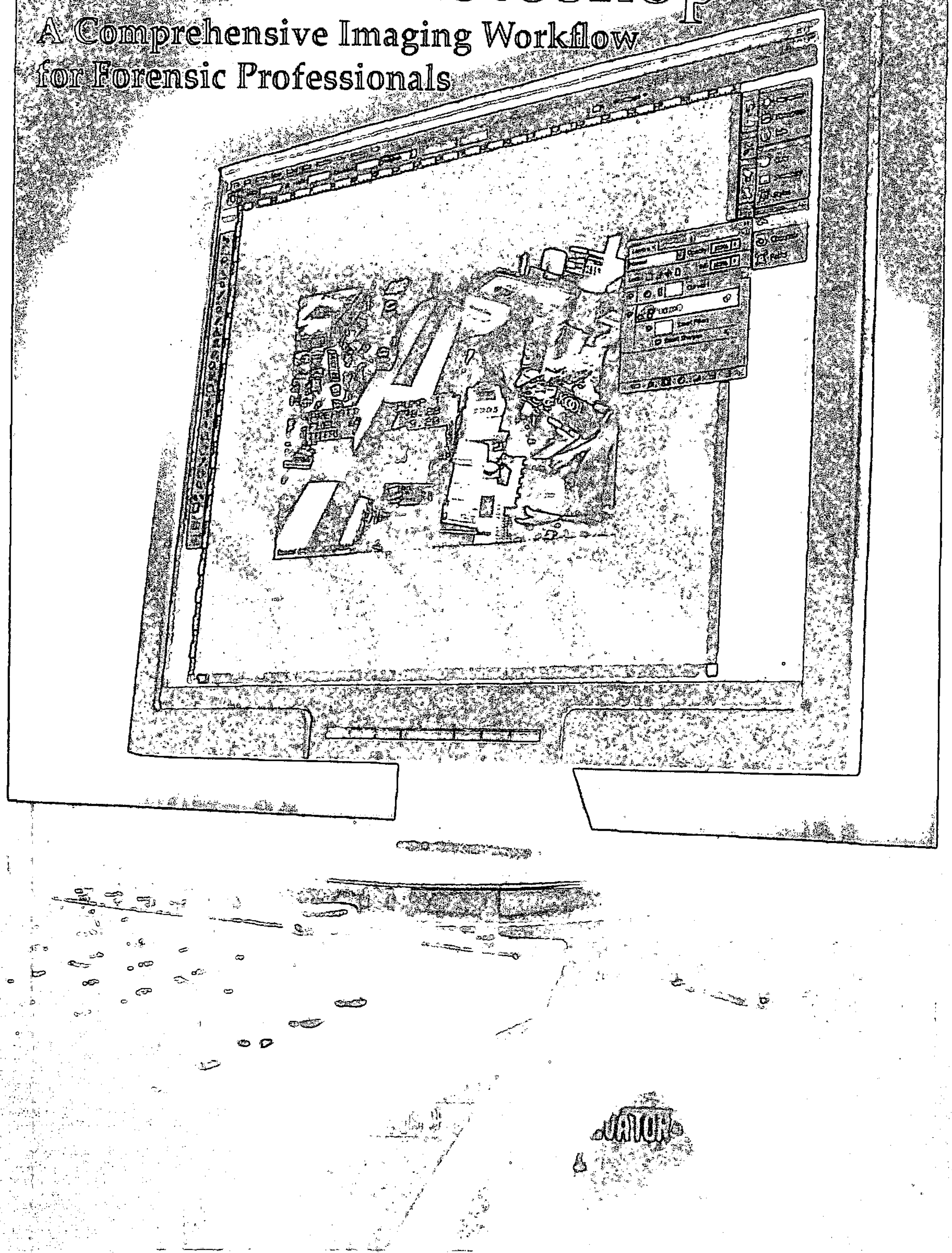
BRUCE BARTLETT, State Attorney
Sixth Judicial Circuit of Florida

By [redacted]
Assistant State Attorney
Bar No. 435988

EXHIBIT # 2

Forensic Photoshop

A Comprehensive Imaging Workflow for Forensic Professionals



from the author of the Forensic Photoshop blog

Jim Hoerricks

Interpolation

We start with a simple question: **why** interpolate? Often times, trial exhibits need to be prepared and an attorney will ask for an image to be "blown up" and put on a poster sized display board. You may be asked to resize some multi-megapixel images for use in a video presentation or for PowerPoint. Interpolation can be either an up or a down process.

It can also involve rotation (more on that later).

So **what** then is interpolation? Interpolation's purpose is to resize and/or reposition an image for final output. That output can come in a number of forms and sizes.

There are many ways to accomplish this task (as there are with anything in Photoshop). All of these are dependant on the output method and size. In my lab, I have printed everything from 4x5's to large format ink jet posters. I have even prepared files for print in newspapers and on billboards.

Interpolation as resizing

A detective once asked me how big a still frame of video could be printed. I replied that I could put it on the side of a bus, if he wanted. **How** would I do that? Let's take a look at the **Image Size** dialog box.

1. Open an image.

2. Click **Image>Image Size.**

3. Select the **Resample Image** check box.

4. Click the Resample Image list arrow, and then select an option:
Nearest Neighbor - best for quick results with low quality.
Bilinear- best for line art.
Bicubic- (default when installed) best for most purposes with high quality.
Bicubic Smoother- best for enlarging an image.
Bicubic Sharper - best for reducing an image.
You can change the default Interpolation method, click **Edit>Preferences>General.**

5. To maintain image proportions, select the **Constrain Proportions** check box.

6. Enter the desired sizes in the Pixel Dimensions or Document Size boxes. If you choose to constrain proportions in step 5, when you change one size, the other boxes will adjust automatically and in proportion with each other.

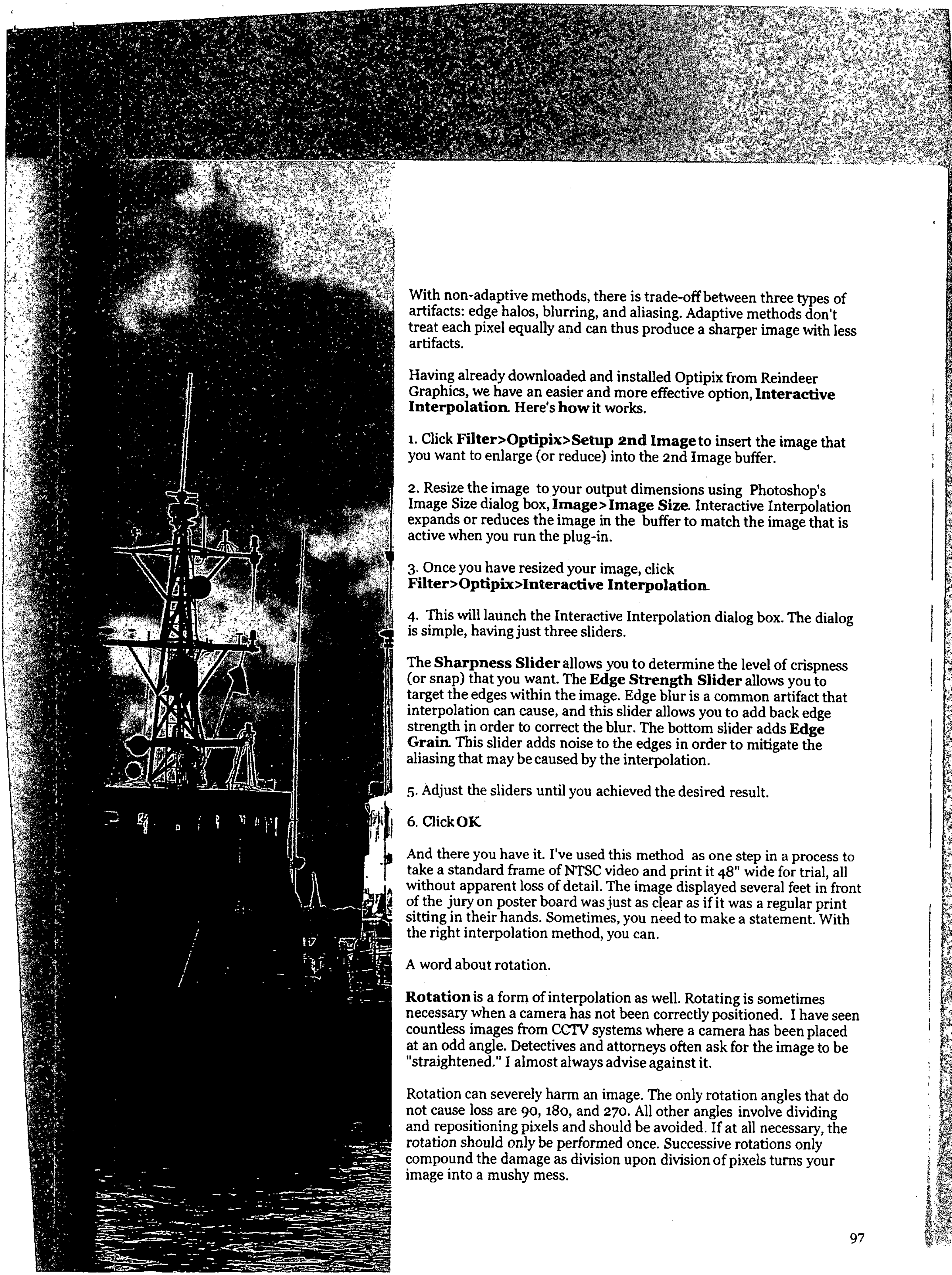
7. Click **OK**. And ... just like that you have a resized image. So, if its that easy, what's the catch?

Interpolation involves approximation. The results can vary significantly depending on the interpolation method (algorithm) chosen. Therefore, an image will always lose some quality each time interpolation is performed.

Why?

The best way that I've ever heard it described comes from an traffic collision investigator. He put it like this: "Interpolation works by utilising known values to estimate values at unknown points." To see why this is significant, let's look again at the various interpolation methods.

There are two basic types of interpolation methods, non-adaptive and adaptive. Non-adaptive methods treat all pixels equally. Examples of non-adaptive methods include nearest neighbor, bilinear, and bicubic. Adaptive methods change depending on what they are interpolating. These methods are employed in plug-ins like Genuine Fractals, which looks at content (edge vs. smooth area) whilst enlarging in order to preserve detail.



With non-adaptive methods, there is trade-off between three types of artifacts: edge halos, blurring, and aliasing. Adaptive methods don't treat each pixel equally and can thus produce a sharper image with less artifacts.

Having already downloaded and installed Optipix from Reindeer Graphics, we have an easier and more effective option, **Interactive Interpolation.** Here's **how** it works.

1. Click **Filter>Optipix>Setup 2nd Image** to insert the image that you want to enlarge (or reduce) into the 2nd Image buffer.

2. Resize the image to your output dimensions using Photoshop's Image Size dialog box, **Image>Image Size.** Interactive Interpolation expands or reduces the image in the buffer to match the image that is active when you run the plug-in.

3. Once you have resized your image, click **Filter>Optipix>Interactive Interpolation.**

4. This will launch the Interactive Interpolation dialog box. The dialog is simple, having just three sliders.

The **Sharpness Slider** allows you to determine the level of crispness (or snap) that you want. The **Edge Strength Slider** allows you to target the edges within the image. Edge blur is a common artifact that interpolation can cause, and this slider allows you to add back edge strength in order to correct the blur. The bottom slider adds **Edge Grain.** This slider adds noise to the edges in order to mitigate the aliasing that may be caused by the interpolation.

5. Adjust the sliders until you achieved the desired result.

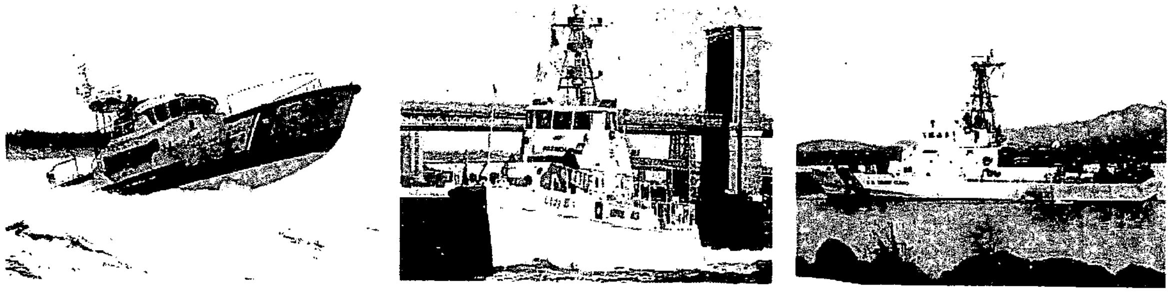
6. Click **OK.**

And there you have it. I've used this method as one step in a process to take a standard frame of NTSC video and print it 48" wide for trial, all without apparent loss of detail. The image displayed several feet in front of the jury on poster board was just as clear as if it was a regular print sitting in their hands. Sometimes, you need to make a statement. With the right interpolation method, you can.

A word about rotation.

Rotation is a form of interpolation as well. Rotating is sometimes necessary when a camera has not been correctly positioned. I have seen countless images from CCTV systems where a camera has been placed at an odd angle. Detectives and attorneys often ask for the image to be "straightened." I almost always advise against it.

Rotation can severely harm an image. The only rotation angles that do not cause loss are 90, 180, and 270. All other angles involve dividing and repositioning pixels and should be avoided. If at all necessary, the rotation should only be performed once. Successive rotations only compound the damage as division upon division of pixels turns your image into a mushy mess.



Wrapping up

The images that we see in our labs will generally require some sort of resizing. A ten megapixel image from a forensic photographer will need to be reduced in order to displayed correctly in a PowerPoint or video presentation. Low resolution digital CCTV will have to be resized in order for stills to be displayed on posters in a court room. This change in size or position is both the **what** and the **why** of the process. The **how** is relatively simple; either use Photoshop's internal resizing and rotating tools or those of a third party plug-in.

The science behind the work involves some significant computational power on the part of the program and your computer. Nearest neighbor is the easiest to explain, it just makes each pixel bigger - resulting in a blocky looking image. The others look deeper at the image and produce a result that is more pleasing to the eye. As results vary by algorithm, it's best to experiment a bit. As always, document your steps and settings.

In your testimony, try to use real-world terms to describe the process. "After reducing the noise, I began an examination to determine the most appropriate interpolation method to use with the image. That is to say, I tried several accepted methods of resizing to see which would produce an image at the requested size with the least amount of loss of detail ..."

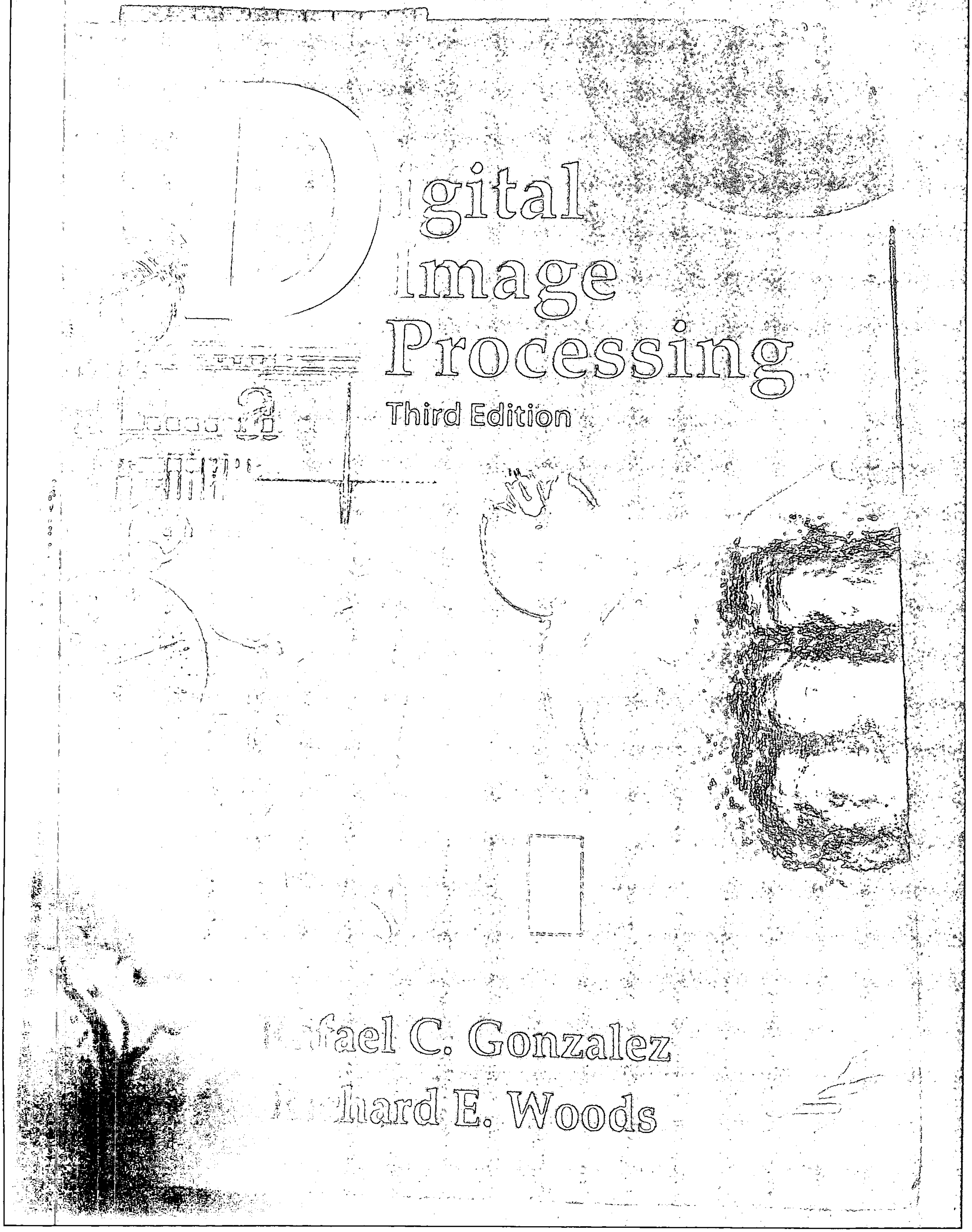
Items for Consideration:

Identify some reasons for resizing an image (up or down):

Do you resize images before placing them in presentations made in PowerPoint or Keynote? Why/why not?

Do you resize images before placing them into crime alerts or other bulletins? Why/why not?

EXHIBIT # 3



Digital Image Processing

Third Edition

Rafael C. Gonzalez
Richard E. Woods

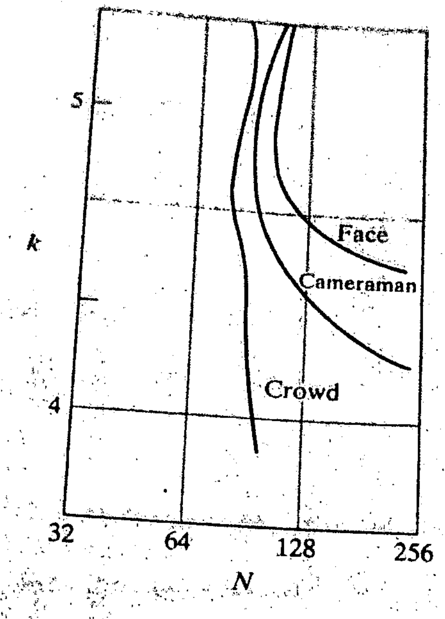


FIGURE 2.23 Typical isopreference curves for the three types of images in Fig. 2.22.

nearly independent of the number of intensity levels used (for the range of intensity levels shown in Fig. 2.23). It is of interest also to note that perceived quality in the other two image categories remained the same in some intervals in which the number of samples was increased, but the number of intensity levels actually decreased. The most likely reason for this result is that a decrease in k tends to increase the apparent contrast, a visual effect that humans often perceive as improved quality in an image.

2.4.4 Image Interpolation

Interpolation is a basic tool used extensively in tasks such as zooming, shrinking, rotating, and geometric corrections. Our principal objective in this section is to introduce interpolation and apply it to image resizing (shrinking and zooming), which are basically image *resampling* methods. Uses of interpolation in applications such as rotation and geometric corrections are discussed in Section 2.6.5. We also return to this topic in Chapter 4, where we discuss image resampling in more detail.

Fundamentally, *interpolation* is the process of using known data to estimate [illegible] unknown locations. We begin the discussion of this topic with a sim[illegible]ple. Suppose that an image of size 500 × 500 pixels has to be en[illegible] times to 750 × 750 pixels. A simple way to visualize zooming is to [illegible] imaginary 750 × 750 grid with the same pixel spacing as the original, [illegible] shrink it so that it fits exactly over the original image. Obviously, the [illegible] the shrunken 750 × 750 grid will be less than the pixel spacing [illegible] image. To perform intensity-level assignment for any point in [illegible] look for its closest pixel in the original image and assign the in[illegible] to the new pixel in the 750 × 750 grid. When we are fin[illegible] intensities to all the points in the overlay grid, we expand it to [illegible] size to obtain the zoomed image.

The method just discussed is called *nearest neighbor interpolation* because it assigns to each new location the intensity of its nearest neighbor in the original image (pixel neighborhoods are discussed formally in Section 2.5). This approach is simple but, as we show later in this section, it has the tendency to produce undesirable artifacts, such as severe distortion of straight edges. For this reason, it is used infrequently in practice. A more suitable approach is *bilinear interpolation*, in which we use the four nearest neighbors to estimate the intensity at a given location. Let (x, y) denote the coordinates of the location to which we want to assign an intensity value (think of it as a point of the grid described previously), and let $v(x, y)$ denote that intensity value. For bilinear interpolation, the assigned value is obtained using the equation

$$v(x, y) = ax + by + cxy + d \tag{2.4-6}$$

Contrary to what the name suggests, note that bilinear interpolation is not linear because of the xy term.

where the four coefficients are determined from the four equations in four unknowns that can be written using the four nearest neighbors of point (x, y). As you will see shortly, bilinear interpolation gives much better results than nearest neighbor interpolation, with a modest increase in computational burden.

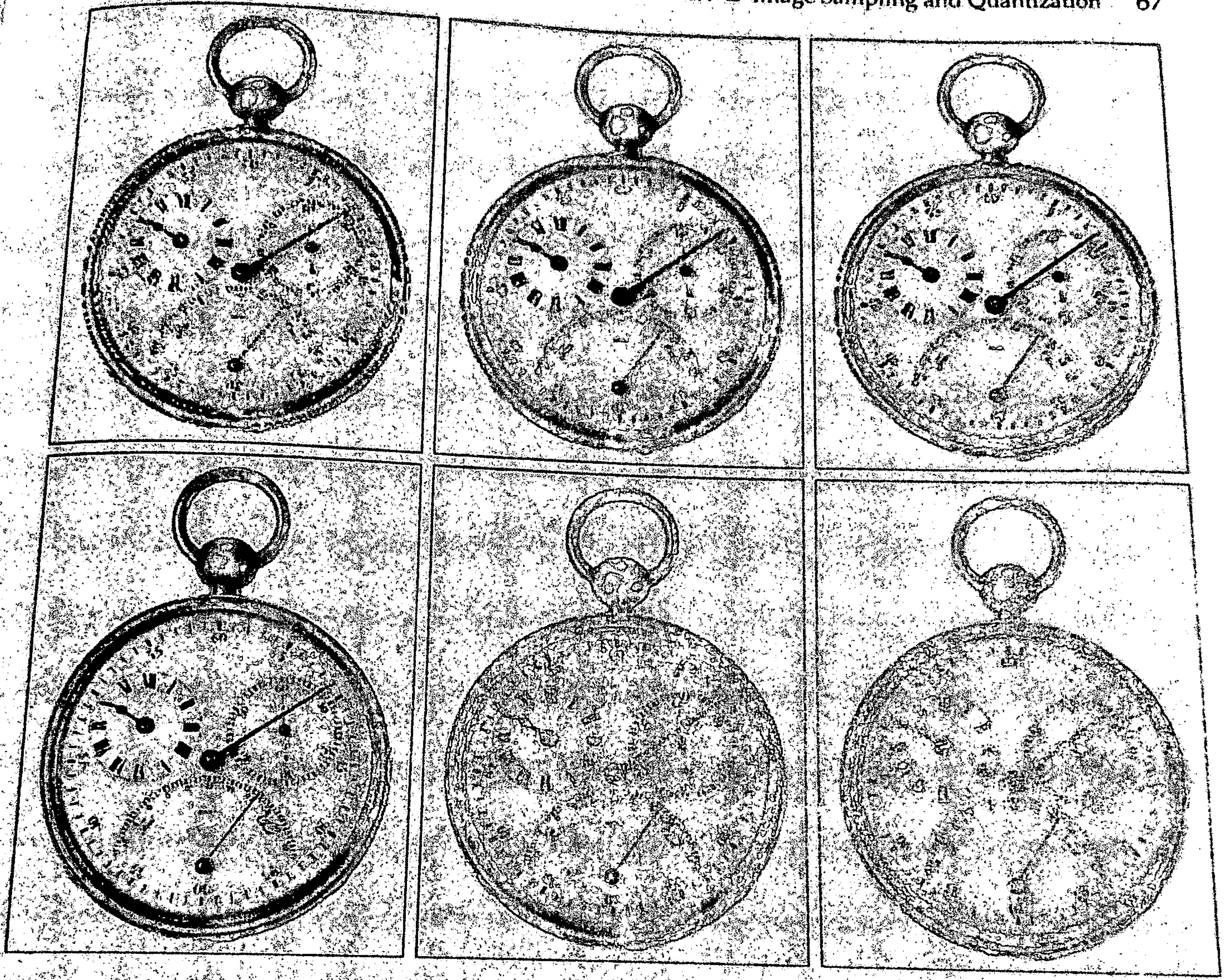
The next level of complexity is *bicubic interpolation*, which involves the sixteen nearest neighbors of a point. The intensity value assigned to point (x, y) is obtained using the equation

$$v(x, y) = \sum_{i=0}^{3}\sum_{j=0}^{3} a_{ij}x^i y^j \tag{2.4-7}$$

where the sixteen coefficients are determined from the sixteen equations in sixteen unknowns that can be written using the sixteen nearest neighbors of point (x, y). Observe that Eq. (2.4-7) reduces in form to Eq. (2.4-6) if the limits of both summations in the former equation are 0 to 1. Generally, bicubic interpolation does a better job of preserving fine detail than its bilinear counterpart. Bicubic interpolation is the standard used in commercial image editing programs, such as Adobe Photoshop and Corel Photopaint.

EXAMPLE 2.4: Comparison of interpolation approaches for image shrinking and zooming.

□ Figure 2.24(a) is the same image as Fig. 2.20(d), which was obtained by reducing the resolution of the 1250 dpi image in Fig. 2.20(a) to 72 dpi (the size shrank from the original size of 3692 × 2812 to 213 × 162 pixels) and then zooming the reduced image back to its original size. To generate Fig. 2.20(d) we used nearest neighbor interpolation both to shrink and zoom the image. As we commented before, the result in Fig. 2.24(a) is rather poor. Figures 2.24(b) and (c) are the results of repeating the same procedure but using, respectively, bilinear and bicubic interpolation for both shrinking and zooming. The result obtained by using bilinear interpolation is a significant improvement over nearest neighbor interpolation. The bicubic result is slightly sharper than the bilinear image. Figure 2.24(d) is the same as Fig. 2.20(c), which was obtained using nearest neighbor interpolation for both shrinking and zooming. We commented in discussing that figure that reducing the resolution to 150 dpi began showing degradation in the image. Figures 2.24(e) and (f) show the results of using



a b c
d e f

FIGURE 2.24 (a) Image reduced to 72 dpi and zoomed back to its original size (3692 × 2812 pixels) using nearest neighbor interpolation. This figure is the same as Fig. 2.20(d). (b) Image shrunk and zoomed using bilinear interpolation. (c) Same as (b) but using bicubic interpolation. (d)–(f) Same sequence, but shrinking down to 150 dpi instead of 72 dpi [Fig. 2.24(d) is the same as Fig. 2.20(c)]. Compare Figs. 2.24(e) and (f), especially the latter, with the original image in Fig. 2.20(a).

bilinear and bicubic interpolation, respectively, to shrink and zoom the image. In spite of a reduction in resolution from 1250 to 150, these last two images [illegible] reasonably favorably with the original, showing once again the [illegible] these two interpolation methods. As before, bicubic interpolation [illegible] sharper results. ■

It is possible to use more neighbors in interpolation, and there are more complex techniques, such as using splines and wavelets, that in some instances can yield better results than the methods just discussed. While preserving fine detail is an exceptionally important consideration in image generation for 3-D graphics (Watt [1993], Shirley [2002]) and in medical image processing (Lehmann et al. [1999]), the extra computational burden seldom is justifiable for general-purpose digital image processing, where bilinear or bicubic interpolation typically are the methods of choice.

2.5 Some Basic Relationships between Pixels

In this section, we consider several important relationships between pixels in a digital image. As mentioned before, an image is denoted by $f(x, y)$. When referring in this section to a particular pixel, we use lowercase letters, such as p and q.

2.5.1 Neighbors of a Pixel

A pixel p at coordinates (x, y) has four *horizontal* and *vertical* neighbors whose coordinates are given by

$$(x + 1, y), (x - 1, y), (x, y + 1), (x, y - 1)$$

This set of pixels, called the *4-neighbors* of p, is denoted by $N_4(p)$. Each pixel is a unit distance from (x, y), and some of the neighbor locations of p lie outside the digital image if (x, y) is on the border of the image. We deal with this issue in Chapter 3.

The four *diagonal* neighbors of p have coordinates

$$(x + 1, y + 1), (x + 1, y - 1), (x - 1, y + 1), (x - 1, y - 1)$$

and are denoted by $N_D(p)$. These points, together with the 4-neighbors, are called the *8-neighbors* of p, denoted by $N_8(p)$. As before, some of the neighbor locations in $N_D(p)$ and $N_8(p)$ fall outside the image if (x, y) is on the border of the image.

2.5.2 Adjacency, Connectivity, Regions, and Boundaries

Let V be the set of intensity values used to define adjacency. In a binary image, $V = \{1\}$ if we are referring to adjacency of pixels with value 1. In a gray-scale image, the idea is the same, but set V typically contains more elements. For example, in the adjacency of pixels with a range of possible intensity values 0 to 255, set V could be any subset of these 256 values. We consider three types of adjacency:

(a) *4-adjacency*. Two pixels p and q with values from V are 4-adjacent if q is in the set $N_4(p)$.

(b) *8-adjacency*. Two pixels p and q with values from V are 8-adjacent if q is in the set $N_8(p)$.

(c) *m-adjacency* (mixed adjacency). Two pixels p and q with values from V are m-adjacent if

  (i) q is in $N_4(p)$, *or*

  (ii) q is in $N_D(p)$ *and* the set $N_4(p) \cap N_4(q)$ has no pixels whose values are from V.

We use the symbols $\cap$ and $\cup$ to denote set intersection and union, respectively. Given sets A and B, recall that their *intersection* is the set of elements that are members of both A and B. The *union* of these two sets is the set of elements that are members of A, of B, or of both. We discuss sets in more detail in Section 2.6.4.

Mixed adjacency is a modification of 8-adjacency. It is introduced to eliminate the ambiguities that often arise when 8-adjacency is used. For example, consider the pixel arrangement shown in Fig. 2.25(a) for $V = \{1\}$. The three pixels at the top of Fig. 2.25(b) show multiple (ambiguous) 8-adjacency, as indicated by the dashed lines. This ambiguity is removed by using m-adjacency, as shown in Fig. 2.25(c).

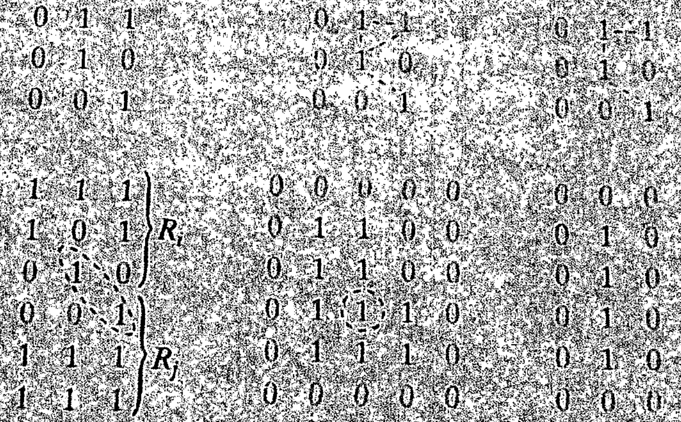
A *(digital) path* (or *curve*) from pixel p with coordinates (x, y) to pixel q with coordinates (s, t) is a sequence of distinct pixels with coordinates

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n)$$

where $(x_0, y_0) = (x, y)$, $(x_n, y_n) = (s, t)$, and pixels (x_i, y_i) and (x_{i-1}, y_{i-1}) are adjacent for $1 \leq i \leq n$. In this case, n is the *length* of the path. If $(x_0, y_0) = (x_n, y_n)$, the path is a *closed* path. We can define 4-, 8-, or m-paths depending on the type of adjacency specified. For example, the paths shown in Fig. 2.25(b) between the top right and bottom right points are 8-paths, and the path in Fig. 2.25(c) is an m-path.

Let S represent a subset of pixels in an image. Two pixels p and q are said to be *connected* in S if there exists a path between them consisting entirely of pixels in S. For any pixel p in S, the *set* of pixels that are connected to it in S is called a *connected component* of S. If it only has one connected component, then set S is called a *connected set*.

Let R be a subset of pixels in an image. We call R a *region* of the image if R is a connected set. Two regions, R_i and R_j are said to be *adjacent* if their union forms a connected set. Regions that are not adjacent are said to be *disjoint*. We consider 4- and 8-adjacency when referring to regions. For our definition to make sense, the type of adjacency used must be specified. For example, the two regions (of 1s) in Fig. 2.25(d) are adjacent only if 8-adjacency is used (according to the definition in the previous paragraph, a 4-path between the two regions does not exist, so their union is not a connected set).



[illegible]URE 2.25 (a) An arrangement of pixels. (b) Pixels that are 8-adjacent (adjacency is [illegible] by dashed lines; note the ambiguity). (c) m-adjacency. (d) Two regions (of 1s) that [illegible]cent if 8-adjecency is used. (e) The circled point is part of the boundary of the [illegible] pixels only if 8-adjacency between the region and background is used. (f) The [illegible]undary of the 1-valued region does not form a closed path, but its outer [illegible] does.

Suppose that an image contains K disjoint regions, R_k, $k = 1, 2, \ldots, K$, none of which touches the image border.[†] Let R_u denote the union of all the K regions, and let $(R_u)^c$ denote its complement (recall that the *complement* of a set S is the set of points that are not in S). We call all the points in R_u the *foreground*, and all the points in $(R_u)^c$ the *background* of the image.

The *boundary* (also called the *border* or *contour*) of a region R is the set of points that are adjacent to points in the complement of R. Said another way, the border of a region is the set of pixels in the region that have at least one background neighbor. Here again, we must specify the connectivity being used to define adjacency. For example, the point circled in Fig. 2.25(e) is not a member of the border of the 1-valued region if 4-connectivity is used between the region and its background. As a rule, adjacency between points in a region and its background is defined in terms of 8-connectivity to handle situations like this.

The preceding definition sometimes is referred to as the *inner border* of the region to distinguish it from its *outer border*, which is the corresponding border in the background. This distinction is important in the development of border-following algorithms. Such algorithms usually are formulated to follow the outer boundary in order to guarantee that the result will form a closed path. For instance, the inner border of the 1-valued region in Fig. 2.25(f) is the region itself. This border does not satisfy the definition of a closed path given earlier. On the other hand, the outer border of the region does form a closed path around the region.

If R happens to be an entire image (which we recall is a rectangular set of pixels), then its boundary is defined as the set of pixels in the first and last rows and columns of the image. This extra definition is required because an image has no neighbors beyond its border. Normally, when we refer to a region, we are referring to a subset of an image, and any pixels in the boundary of the region that happen to coincide with the border of the image are included implicitly as part of the region boundary.

The concept of an *edge* is found frequently in discussions dealing with regions and boundaries. There is a key difference between these concepts, however. The boundary of a finite region forms a closed path and is thus a "global" concept. As discussed in detail in Chapter 10, edges are formed from pixels with derivative values that exceed a preset threshold. Thus, the idea of an edge is a "local" concept that is based on a measure of intensity-level discontinuity at a point. It is possible to link edge points into edge segments, and sometimes these segments are linked in such a way that they correspond to boundaries, but this is not always the case. The one exception in which edges and boundaries correspond is in binary images. Depending on the type of connectivity and edge operators used (we discuss these in Chapter 10), the edge extracted from a binary region will be the same as the region boundary.

[†] We make this assumption to avoid having to deal with special cases. This is done without loss of generality because if one or more regions touch the border of an image, we can simply pad the image with a 1-pixel-wide border of background values.

This is intuitive. Conceptually, until we arrive at Chapter 10, it is helpful to think of edges as intensity discontinuities and boundaries as closed paths.

2.5.3 Distance Measures

For pixels p, q, and z, with coordinates (x, y), (s, t), and (v, w), respectively, D is a *distance function* or *metric* if

(a) $D(p, q) \geq 0 \quad (D(p, q) = 0 \quad \text{iff} \quad p = q)$,
(b) $D(p, q) = D(q, p)$, and
(c) $D(p, z) \leq D(p, q) + D(q, z)$.

The *Euclidean distance* between p and q is defined as

$$D_e(p, q) = \left[(x - s)^2 + (y - t)^2\right]^{\frac{1}{2}} \tag{2.5-1}$$

For this distance measure, the pixels having a distance less than or equal to some value r from (x, y) are the points contained in a disk of radius r centered at (x, y).

The D_4 *distance* (called the *city-block distance*) between p and q is defined as

$$D_4(p, q) = |x - s| + |y - t| \tag{2.5-2}$$

In this case, the pixels having a D_4 distance from (x, y) less than or equal to some value r form a diamond centered at (x, y). For example, the pixels with D_4 distance ≤ 2 from (x, y) (the center point) form the following contours of constant distance:

$$\begin{array}{ccccc} & & 2 & & \\ & 2 & 1 & 2 & \\ 2 & 1 & 0 & 1 & 2 \\ & 2 & 1 & 2 & \\ & & 2 & & \end{array}$$

The pixels with $D_4 = 1$ are the 4-neighbors of (x, y).

The D_8 *distance* (called the *chessboard distance*) between p and q is defined as

$$D_8(p, q) = \max(|x - s|, |y - t|) \tag{2.5-3}$$

In this case, the pixels with D_8 distance from (x, y) less than or equal to some value r form a square centered at (x, y). For example, the pixels with D_8 distance ≤ 2 from (x, y) (the center point) form the following contours of constant distance:

$$\begin{array}{ccccc} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{array}$$

[illegible] pixels with $D_8 = 1$ are the 8-neighbors of (x, y).

Note that the D_4 and D_8 distances between p and q are independent of any paths that might exist between the points because these distances involve only the coordinates of the points. If we elect to consider m-adjacency, however, the D_m distance between two points is defined as the shortest m-path between the points. In this case, the distance between two pixels will depend on the values of the pixels along the path, as well as the values of their neighbors. For instance, consider the following arrangement of pixels and assume that p, p_2, and p_4 have value 1 and that p_1 and p_3 can have a value of 0 or 1:

$$\begin{array}{cc} p_3 & p_4 \\ p_1 & p_2 \\ p & \end{array}$$

Suppose that we consider adjacency of pixels valued 1 (i.e., $V = \{1\}$). If p_1 and p_3 are 0, the length of the shortest m-path (the D_m distance) between p and p_4 is 2. If p_1 is 1, then p_2 and p will no longer be m-adjacent (see the definition of m-adjacency) and the length of the shortest m-path becomes 3 (the path goes through the points $pp_1p_2p_4$). Similar comments apply if p_3 is 1 (and p_1 is 0); in this case, the length of the shortest m-path also is 3. Finally, if both p_1 and p_3 are 1, the length of the shortest m-path between p and p_4 is 4. In this case, the path goes through the sequence of points $pp_1p_2p_3p_4$.

2.6 An Introduction to the Mathematical Tools Used in Digital Image Processing

Before proceeding, you may find it helpful to download and study the review material available in the Tutorials section of the book Web site. The review covers introductory material on matrices and vectors, linear systems, set theory, and probability.

This section has two principal objectives: (1) to introduce you to the various mathematical tools we use throughout the book; and (2) to help you begin developing a "feel" for how these tools are used by applying them to a variety of basic image-processing tasks, some of which will be used numerous times in subsequent discussions. We expand the scope of the tools and their application as necessary in the following chapters.

2.6.1 Array versus Matrix Operations

An *array* operation involving one or more images is carried out on a *pixel-by-pixel* basis. We mentioned earlier in this chapter that images can be viewed equivalently as matrices. In fact, there are many situations in which operations between images are carried out using matrix theory (see Section 2.6.6). It is for this reason that a clear distinction must be made between array and matrix operations. For example, consider the following 2 × 2 images:

$$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix}$$

The *array product* of these two images is

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

On the other hand, the *matrix product* is given by

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

We assume array operations throughout the book, unless stated otherwise. For example, when we refer to raising an image to a power, we mean that each individual pixel is raised to that power; when we refer to dividing an image by another, we mean that the division is between corresponding pixel pairs, and so on.

2.6.2 Linear versus Nonlinear Operations

One of the most important classifications of an image-processing method is whether it is linear or nonlinear. Consider a general operator, H, that produces an output image, $g(x, y)$, for a given input image, $f(x, y)$:

$$H\left[f(x, y)\right] = g(x, y) \tag{2.6-1}$$

H is said to be a *linear operator* if

$$\begin{aligned} H\left[a_i f_i(x, y) + a_j f_j(x, y)\right] &= a_i H\left[f_i(x, y)\right] + a_j H\left[f_j(x, y)\right] \\ &= a_i g_i(x, y) + a_j g_j(x, y) \end{aligned} \tag{2.6-2}$$

where a_i, a_j, $f_i(x, y)$, and $f_j(x, y)$ are arbitrary constants and images (of the same size), respectively. Equation (2.6-2) indicates that the output of a linear operation due to the sum of two inputs is the same as performing the operation on the inputs individually and then summing the results. In addition, the output of a linear operation to a constant times an input is the same as the output of the operation due to the original input multiplied by that constant. The first property is called the property of *additivity* and the second is called the property of *homogeneity*.

As a simple example, suppose that H is the sum operator, Σ; that is, the function of this operator is simply to sum its inputs. To test for linearity, we start with the left side of Eq. (2.6-2) and attempt to prove that it is equal to the right side:

$$\begin{aligned} \sum\left[a_i f_i(x, y) + a_j f_j(x, y)\right] &= \sum a_i f_i(x, y) + \sum a_j f_j(x, y) \\ &= a_i \sum f_i(x, y) + a_j \sum f_j(x, y) \\ &= a_i g_i(x, y) + a_j g_j(x, y) \end{aligned}$$

These are array summ
tions, not the sums of
the elements of the
images. As such, the s
of a single image is th
image itself.

where the first step follows from the fact that summation is distributive. So, an expansion of the left side is equal to the right side of Eq. (2.6-2), and we conclude that the sum operator is linear.

On the other hand, consider the max operation, whose function is to find the maximum value of the pixels in an image. For our purposes here, the simplest way to prove that this operator is nonlinear, is to find an example that fails the test in Eq. (2.6-2). Consider the following two images

$$f_1 = \begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} \quad \text{and} \quad f_2 = \begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}$$

and suppose that we let $a_1 = 1$ and $a_2 = -1$. To test for linearity, we again start with the left side of Eq. (2.6-2):

$$\begin{aligned} \max\left\{(1)\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix} + (-1)\begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}\right\} &= \max\left\{\begin{bmatrix} -6 & -3 \\ -2 & -4 \end{bmatrix}\right\} \\ &= -2 \end{aligned}$$

Working next with the right side, we obtain

$$\begin{aligned} (1)\max\left\{\begin{bmatrix} 0 & 2 \\ 2 & 3 \end{bmatrix}\right\} + (-1)\max\left\{\begin{bmatrix} 6 & 5 \\ 4 & 7 \end{bmatrix}\right\} &= 3 + (-1)7 \\ &= -4 \end{aligned}$$

The left and right sides of Eq. (2.6-2) are not equal in this case, so we have proved that in general the max operator is nonlinear.

As you will see in the next three chapters, especially in Chapters 4 and 5, linear operations are exceptionally important because they are based on a large body of theoretical and practical results that are applicable to image processing. Nonlinear systems are not nearly as well understood, so their scope of application is more limited. However, you will encounter in the following chapters several nonlinear image processing operations whose performance far exceeds what is achievable by their linear counterparts.

2.6.3 Arithmetic Operations

Arithmetic operations between images are array operations which, as discussed in Section 2.6.1, means that arithmetic operations are carried out between corresponding pixel pairs. The four arithmetic operations are denoted as

$$\begin{aligned} s(x, y) &= f(x, y) + g(x, y) \\ d(x, y) &= f(x, y) - g(x, y) \\ p(x, y) &= f(x, y) \times g(x, y) \\ v(x, y) &= f(x, y) \div g(x, y) \end{aligned} \tag{2.6-3}$$

It is understood that the operations are performed between corresponding pixel pairs in f and g for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$

where, as usual, M and N are the row and column sizes of the images. Clearly, s, d, p, and v are images of size $M \times N$ also. Note that image arithmetic in the manner just defined involves images of the same size. The following examples are indicative of the important role played by arithmetic operations in digital image processing.

EXAMPLE 2.5: Addition (averaging) of noisy images for noise reduction.

□ Let $g(x, y)$ denote a corrupted image formed by the addition of noise, $\eta(x, y)$, to a noiseless image $f(x, y)$; that is,

$$g(x, y) = f(x, y) + \eta(x, y) \tag{2.6-4}$$

where the assumption is that at every pair of coordinates (x, y) the noise is uncorrelated[†] and has zero average value. The objective of the following procedure is to reduce the noise content by adding a set of noisy images, $\{g_i(x, y)\}$. This is a technique used frequently for image enhancement.

If the noise satisfies the constraints just stated, it can be shown (Problem 2.20) that if an image $\bar{g}(x, y)$ is formed by averaging K different noisy images,

$$\bar{g}(x, y) = \frac{1}{K}\sum_{i=1}^{K} g_i(x, y) \tag{2.6-5}$$

then it follows that

$$E\{\bar{g}(x, y)\} = f(x, y) \tag{2.6-6}$$

and

$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2_{\eta(x,y)} \tag{2.6-7}$$

where $E\{\bar{g}(x, y)\}$ is the expected value of $\bar{g}$, and $\sigma^2_{\bar{g}(x,y)}$ and $\sigma^2_{\eta(x,y)}$ are the variances of $\bar{g}$ and η, respectively, all at coordinates (x, y). The standard deviation (square root of the variance) at any point in the average image is

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}}\sigma_{\eta(x,y)} \tag{2.6-8}$$

As K increases, Eqs. (2.6-7) and (2.6-8) indicate that the variability (as measured by the variance or the standard deviation) of the pixel values at each location (x, y) decreases. Because $E\{\bar{g}(x, y)\} = f(x, y)$, this means that $\bar{g}(x, y)$ approaches $f(x, y)$ as the number of noisy images used in the averaging process increases. In practice, the images $g_i(x, y)$ must be *registered* (aligned) in order to avoid the introduction of blurring and other artifacts in the output image.

[†] The variance of a random variable z with mean m is defined as $E[(z - m)^2]$, where $E\{\cdot\}$ is [illegible] value of the argument. The covariance of two random variables z_i and z_j is defined as [illegible] $m_j)]$. If the variables are *uncorrelated*, their covariance is 0.

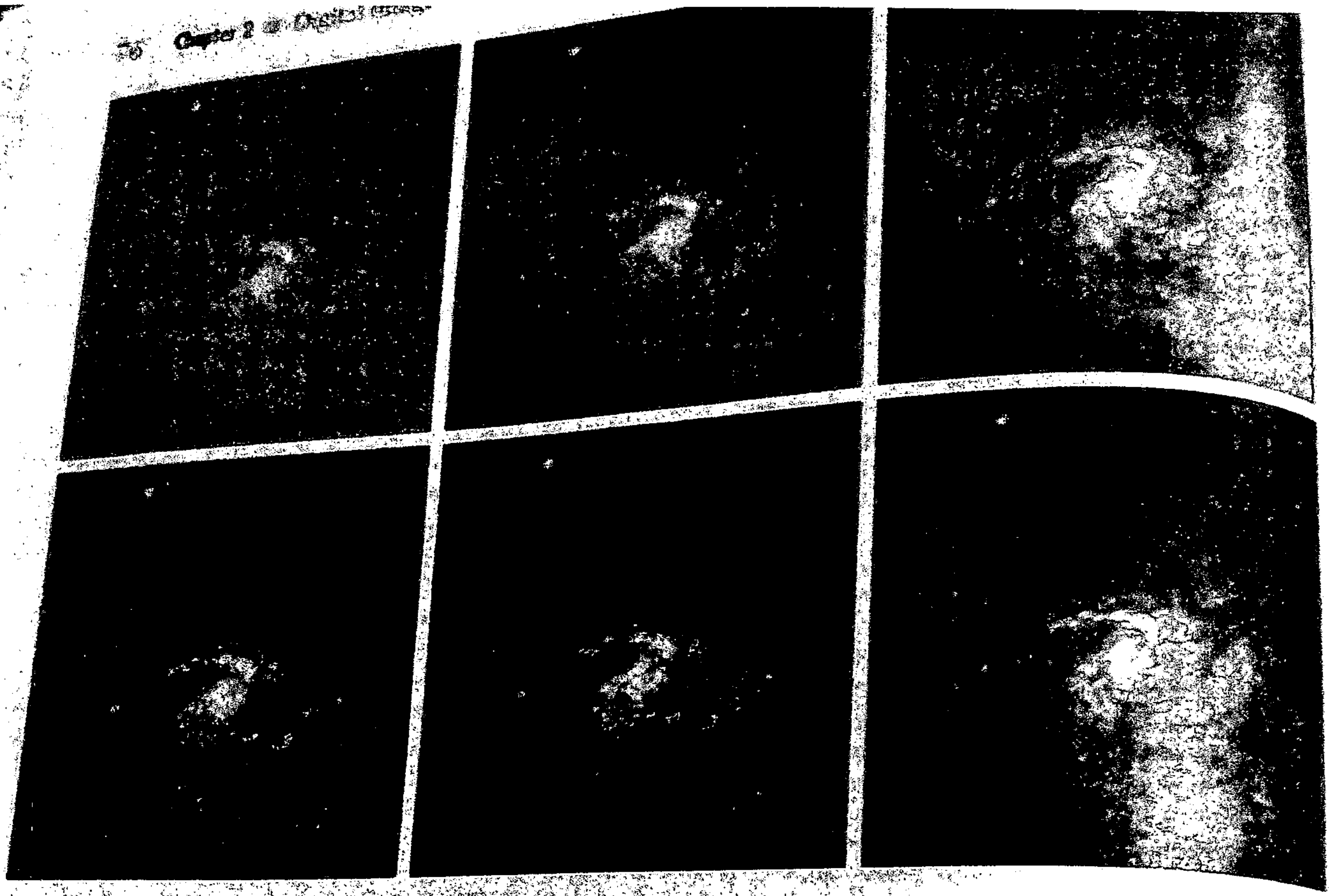


FIGURE 2.26 (a) Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise. (b)–(f) Res averaging 5, 10, 20, 50, and 100 noisy images, respectively. (Original image courtesy of NAS

An important application of image averaging is in the field of astronom where imaging under very low light levels frequently causes sensor noise t render single images virtually useless for analysis. Figure 2.26(a) shows an image in which corruption was simulated by adding to it Gaussian noise wit zero mean and a standard deviation of 64 intensity levels. This image, typica noisy images taken under low light conditions, is useless for all practical pu poses. Figures 2.26(b) through (f) show the results of averaging 5, 10, 20, 50, an 100 images, respectively. We see that the result in Fig. 2.26(e), obtained wit $K = 50$, is reasonably clean. The image Fig. 2.26(f), resulting from averagin 100 noisy images, is only a slight improvement over the image in Fig. 2.26(e

The images shown in this example are from a galaxy pair called NGC 3314, taken by NASA's Hubble Space Telescope. NGC 3314 lies about 140 million light-years from Earth, in the direction of the southern-hemisphere constellation Hydra. The bright stars forming a pinwheel shape near the center of the front galaxy were formed from interstellar gas and dust.

Addition is a discrete version of continuous integration. In astronomica observations, a process equivalent to the method just described is to use the in- tegrating capabilities of CCD (see Section 2.3.3) or similar sensors for nois reduction by observing the same scene over long periods of time. Cooling als is used to reduce sensor noise. The net effect, however, is analogous to averagin a set of noisy digital images.

EXAMPLE 2.6: Image subtraction for enhancing differences.

■ A frequent application of image subtraction is in the enhancement of *differences* between images. For example, the image in Fig. 2.27(b) was obtained by setting to zero the least-significant bit of every pixel in Fig. 2.27(a). Visually, these images are indistinguishable. However, as Fig. 2.27(c) shows, subtracting one image from the other clearly shows their differences. Black (0) values in this difference image indicate locations where there is no difference between the images in Figs. 2.27(a) and (b).

As another illustration, we discuss briefly an area of medical imaging called *mask mode radiography*, a commercially successful and highly beneficial use of image subtraction. Consider image differences of the form

$$g(x, y) = f(x, y) - h(x, y) \tag{2.6-9}$$

Change detection via image subtraction is used also in image segmentation, which is the topic of Chapter 10.

In this case $h(x, y)$, the *mask*, is an X-ray image of a region of a patient's body captured by an intensified TV camera (instead of traditional X-ray film) located opposite an X-ray source. The procedure consists of injecting an X-ray contrast medium into the patient's bloodstream, taking a series of images called *live images* [samples of which are denoted as $f(x, y)$] of the same anatomical region as $h(x, y)$, and subtracting the mask from the series of incoming live images after injection of the contrast medium. The net effect of subtracting the mask from each sample live image is that the areas that are different between $f(x, y)$ and $h(x, y)$ appear in the output image, $g(x, y)$, as enhanced detail. Because images can be captured at TV rates, this procedure in essence gives a movie showing how the contrast medium propagates through the various arteries in the area being observed.

Figure 2.28(a) shows a mask X-ray image of the top of a patient's head prior to injection of an iodine medium into the bloodstream, and Fig. 2.28(b) is a sample of a live image taken after the medium was injected. Figure 2.28(c) is

2.27 (a) Infrared image of the Washington, D.C. area. (b) Image obtained by setting to zero the least bit of every pixel in (a). (c) Difference of the two images, scaled to the range [0, 255] for clarity.

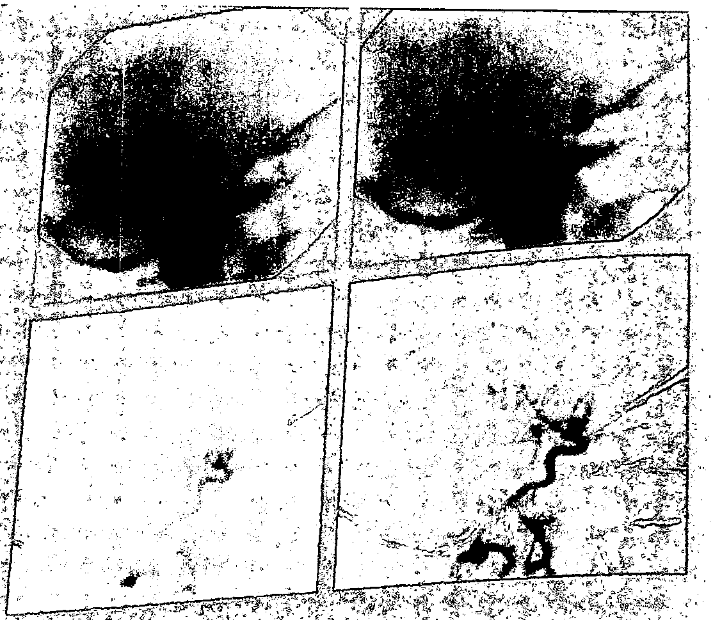


FIGURE 2.28 Digital subtraction angiography. (a) Mask image. (b) A live image. (c) Difference between (a) and (b). (d) Enhanced difference image. (Figures (a) and (b) courtesy of The Image Sciences Institute, University Medical Center, Utrecht, The Netherlands.)

the difference between (a) and (b). Some fine blood vessel structures are visible in this image. The difference is clear in Fig. 2.28(d), which was obtained by enhancing the contrast in (c) (we discuss contrast enhancement in the next chapter). Figure 2.28(d) is a clear "map" of how the medium is propagating through the blood vessels in the subject's brain. ■

EXAMPLE 2.7: Using image multiplication and division for shading correction.

☐ An important application of image multiplication (and division) is *shading correction*. Suppose that an imaging sensor produces images that can be modeled as the product of a "perfect image," denoted by $f(x, y)$, times a shading function, $h(x, y)$; that is, $g(x, y) = f(x, y)h(x, y)$. If $h(x, y)$ is known, we can obtain $f(x, y)$ by multiplying the sensed image by the inverse of $h(x, y)$ (i.e., dividing g by h). If $h(x, y)$ is not known, but access to the imaging system is possible, we can obtain an approximation to the shading function by imaging a target of constant intensity. When the sensor is not available, we often can estimate the shading pattern directly from the image, as we discuss in Section 9.6. Figure 2.29 shows an example of shading correction.

Another common use of image multiplication is in *masking*, also called *region of interest* (ROI), operations. The process, illustrated in Fig. 2.30, consists simply of multiplying a given image by a mask image that has 1s in the ROI and 0s elsewhere. There can be more than one ROI in the mask image, and the shape of the ROI can be arbitrary, although rectangular shapes are used frequently for ease of implementation. ☐

A few comments about implementing image arithmetic operations are in order before we leave this section. In practice, most images are displayed using 8 bits (even 24-bit color images consist of three separate 8-bit channels). Thus, we expect image values to be in the range from 0 to 255. When images

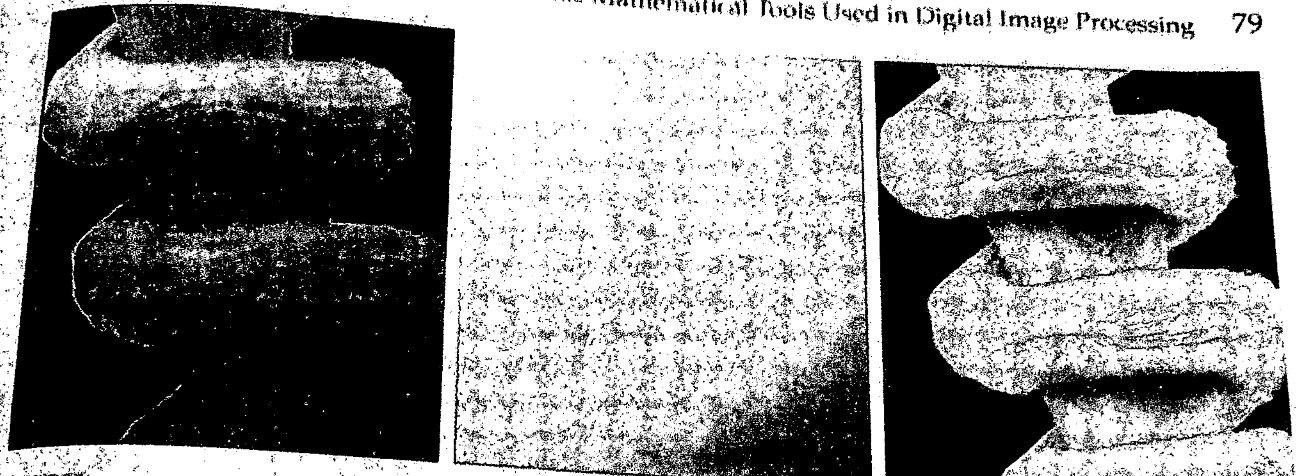
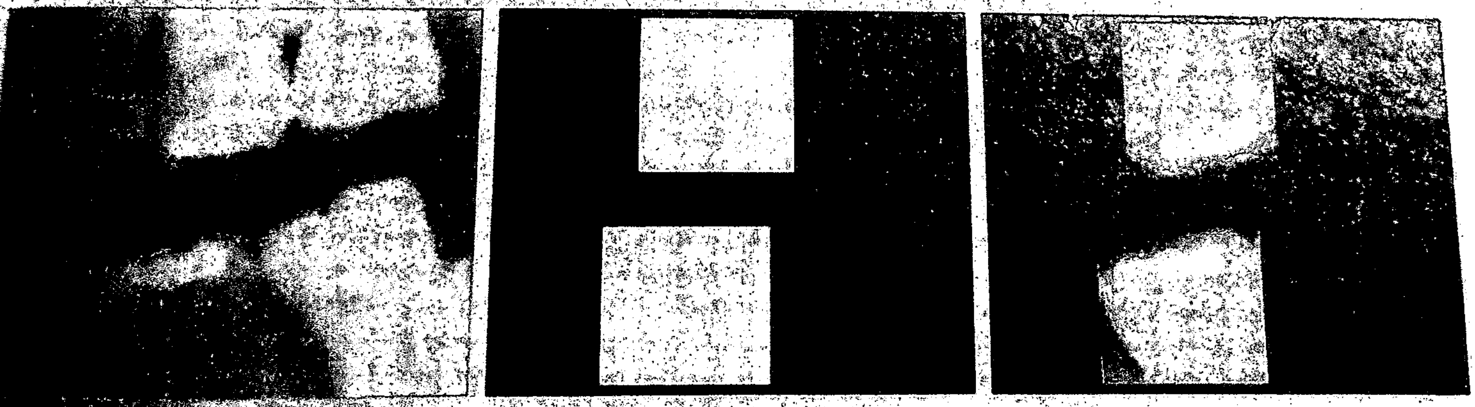


FIGURE 2.29 Shading correction. (a) Shaded SEM image of a tungsten filament and support, magnified approximately 130 times. (b) The shading pattern. (c) Product of (a) by the reciprocal of (b). (Original image courtesy of Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

are saved in a standard format, such as TIFF or JPEG, conversion to this range is automatic. However, the approach used for the conversion depends on the system used. For example, the values in the difference of two 8-bit images can range from a minimum of −255 to a maximum of 255, and the values of a sum image can range from 0 to 510. Many software packages simply set all negative values to 0 and set to 255 all values that exceed this limit when converting images to 8 bits. Given an image f, an approach that guarantees that the full range of an arithmetic operation between images is "captured" into a fixed number of bits is as follows. First, we perform the operation

$$f_m = f - \min(f) \tag{2.6-10}$$



2.30 (a) Digital dental X-ray image. (b) ROI mask for isolating teeth with fillings (white corresponds t
corresponds to 0). (c) Product of (a) and (b).

which creates an image whose minimum value is 0. Then, we perform the operation

$$f_s = K\left[f_m / \max(f_m) \right] \tag{2.6-11}$$

which creates a scaled image, f_s, whose values are in the range $[0, K]$. When working with 8-bit images, setting $K = 255$ gives us a scaled image whose intensities span the full 8-bit scale from 0 to 255. Similar comments apply to 16-bit images or higher. This approach can be used for all arithmetic operations. When performing division, we have the extra requirement that a small number should be added to the pixels of the divisor image to avoid division by 0.

2.6.4 Set and Logical Operations

In this section, we introduce briefly some important set and logical operations. We also introduce the concept of a fuzzy set.

Basic set operations

Let A be a set composed of *ordered pairs* of real numbers. If $a = (a_1, a_2)$ is an *element* of A, then we write

$$a \in A \tag{2.6-12}$$

Similarly, if a is not an element of A, we write

$$a \notin A \tag{2.6-13}$$

The set with no elements is called the *null* or *empty set* and is denoted by the symbol $\varnothing$.

A set is specified by the contents of two braces: $\{\cdot\}$. For example, when we write an expression of the form $C = \{w \mid w = -d, d \in D\}$, we mean that set C is the set of elements, w, such that w is formed by multiplying each of the elements of set D by -1. One way in which sets are used in image processing is to let the elements of sets be the *coordinates* of pixels (ordered pairs of integers) representing regions (objects) in an image.

If every element of a set A is also an element of a set B, then A is said to be a *subset* of B, denoted as

$$A \subseteq B \tag{2.6-14}$$

The *union* of two sets A and B, denoted by

$$C = A \cup B \tag{2.6-15}$$

is the set of elements belonging to either A, B, *or* both. Similarly, the *intersection* of two sets A and B, denoted by

$$D = A \cap B \tag{2.6-16}$$

is the set of elements belonging to both A *and* B. Two sets A and B are said to be *disjoint* or *mutually exclusive* if they have no common elements, in which case,

$$A \cap B = \varnothing$$

The *set universe*, U, is the set of all elements in a given application. By definition, all set elements in a given application are members of the universe defined for that application. For example, if you are working with the set of real numbers, then the set universe is the real line, which contains all the real numbers. In image processing, we typically define the universe to be the rectangle containing all the pixels in an image.

The *complement* of a set A is the set of elements that are not in A:

$$A^c = \{w \mid w \notin A\} \tag{2.6-18}$$

The *difference* of two sets A and B, denoted $A - B$, is defined as

$$A - B = \{w \mid w \in A, w \notin B\} = A \cap B^c \tag{2.6-19}$$

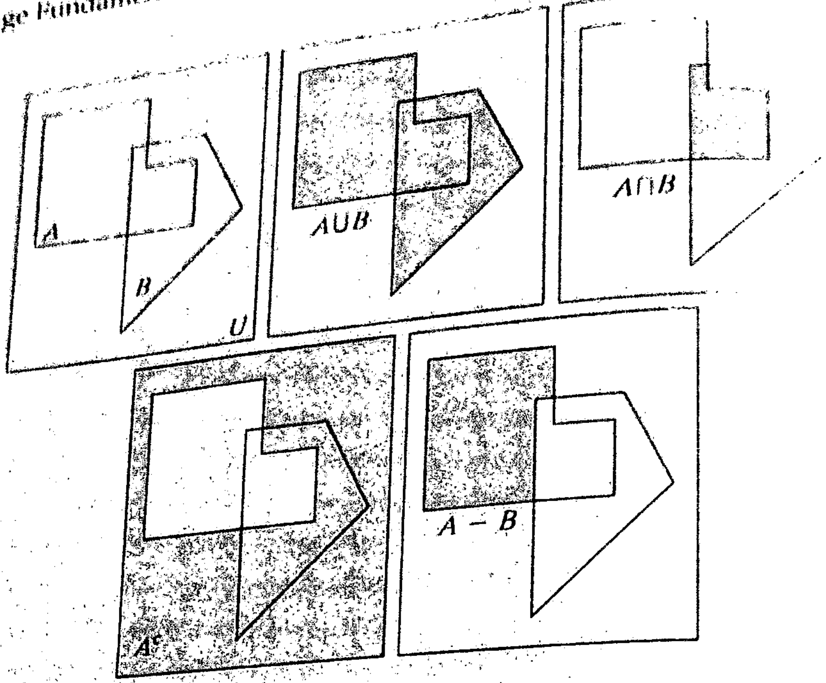
We see that this is the set of elements that belong to A, but not to B. We could, for example, define A^c in terms of U and the set difference operation: $A^c = U - A$.

Figure 2.31 illustrates the preceding concepts, where the universe is the set of coordinates contained within the rectangle shown, and sets A and B are the sets of coordinates contained within the boundaries shown. The result of the set operation indicated in each figure is shown in gray.[†]

In the preceding discussion, set membership is based on position (coordinates). An implicit assumption when working with images is that the intensity of all pixels in the sets is the same, as we have not defined set operations involving intensity values (e.g., we have not specified what the intensities in the intersection of two sets is). The only way that the operations illustrated in Fig. 2.31 can make sense is if the images containing the sets are binary, in which case we can talk about set membership based on coordinates, the assumption being that all member of the sets have the same intensity. We discuss this in more detail in the following subsection.

When dealing with gray-scale images, the preceding concepts are not applicable, because we have to specify the intensities of all the pixels resulting from a set operation. In fact, as you will see in Sections 3.8 and 9.6, the union and intersection operations for gray-scale values usually are defined as the max and min of corresponding pixel pairs, respectively, while the complement is defined as the pairwise differences between a constant and the intensity of every pixel in an image. The fact that we deal with corresponding pixel pairs tells us that gray-scale set operations are array operations, as defined in Section 2.6.1. The following example is a brief illustration of set operations involving gray-scale images. We discuss these concepts further in the two sections mentioned above.

[†] The operations in Eqs. (2.6-12)–(2.6-19) are the basis for the algebra of sets, which starts with properties such as the commutative laws: $A \cup B = B \cup A$ and $A \cap B = B \cap A$, and from these develops a broad theory based on set operations. A treatment of the algebra of sets is beyond the scope of the present discussion, but you should be aware of its existence.

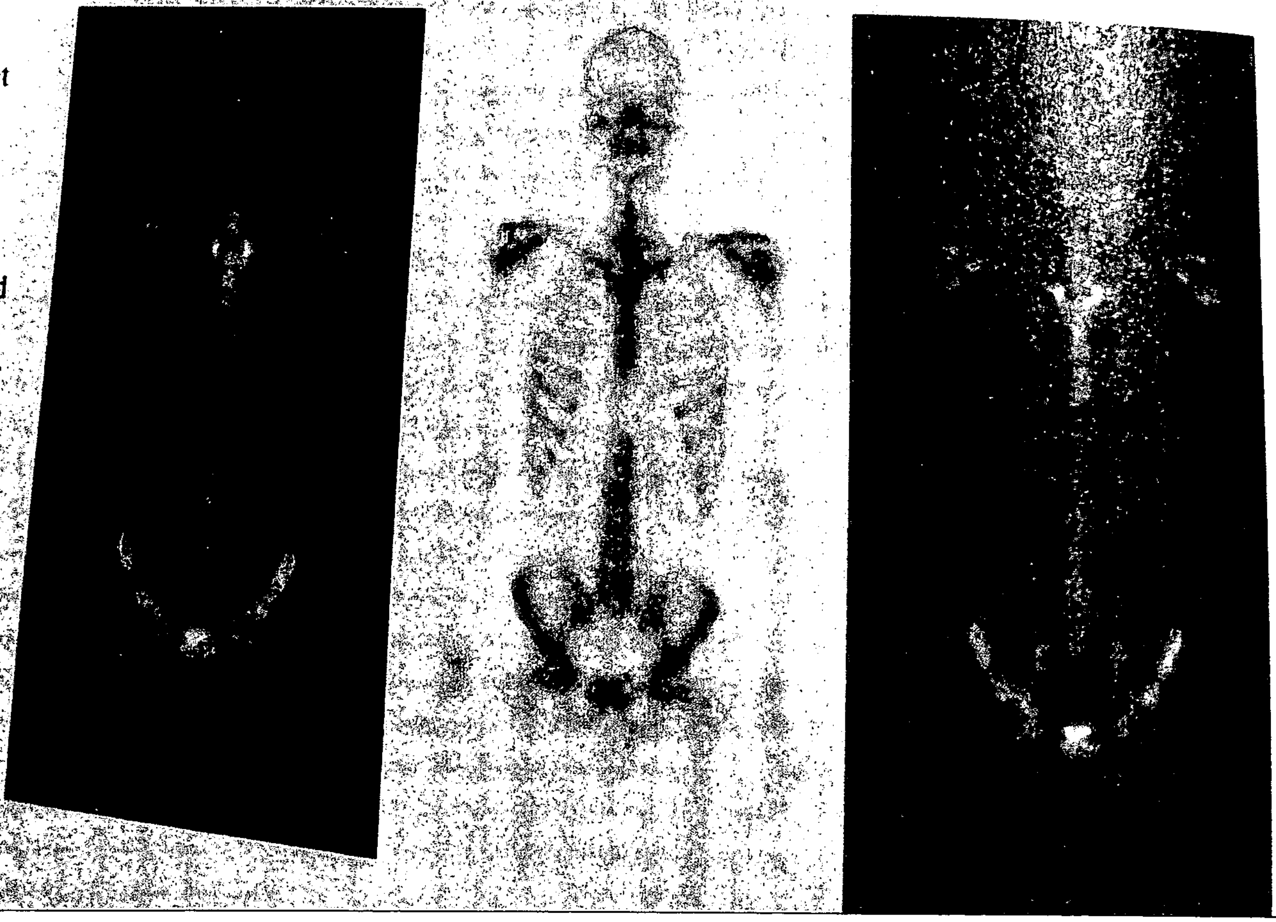


a b c
d e

FIGURE 2.31 (a) Two sets of coordinates, A and B, in 2-D space. (b) The union of A and B. (c) The intersection of A and B. (d) The complement of A. (e) The difference between A and B. In (b)–(e) the shaded areas represent the members of the set operation indicated.

EXAMPLE 2.8: Set operations involving image intensities.

■ Let the elements of a gray-scale image be represented by a set A whose elements are triplets of the form (x, y, z), where x and y are spatial coordinates and z denotes intensity, as mentioned in Section 2.4.2. We can define the *complement* of A as the set $A^c = \{(x, y, K - z) | (x, y, z) \in A\}$, which simply denotes the set of pixels of A whose intensities have been subtracted from a constant K. This constant is equal to $2^k - 1$, where k is the number of intensity bits used to represent z. Let A denote the 8-bit gray-scale image in Fig. 2.32(a), and suppose that we want to form the negative of A using set



a b c

FIGURE 2.32 Set operations involving gray-scale images. (a) Original image. (b) Image negative obtained using set complementation. (c) The union of (a) and a constant image. (Original image courtesy of G.E. Medical Systems.)

operations. We simply form the set $A_n = A^c = \{(x, y, 255 - z) \mid (x, y, z) \in A\}$. Note that the coordinates are carried over, so A_n is an image of the same size as A. Figure 2.32(b) shows the result.

The union of two gray-scale sets A and B may be defined as the set

$$A \cup B = \left\{ \max_z(a, b) \mid a \in A, b \in B \right\}$$

That is, the union of two gray-scale sets (images) is an array formed from the maximum intensity between pairs of spatially corresponding elements. Again, note that coordinates carry over, so the union of A and B is an image of the same size as these two images. As an illustration, suppose that A again represents the image in Fig. 2.32(a), and let B denote a rectangular array of the same size as A, but in which all values of z are equal to 3 times the mean intensity, m, of the elements of A. Figure 2.32(c) shows the result of performing the set union, in which all values exceeding $3m$ appear as values from A and all other pixels have value $3m$, which is a mid-gray value. □

Logical operations

When dealing with binary images, we can think of *foreground* (1-valued) and *background* (0-valued) sets of pixels. Then, if we define regions (objects) as being composed of foreground pixels, the set operations illustrated in Fig. 2.31 become operations between the coordinates of objects in a binary image. When dealing with binary images, it is common practice to refer to union, intersection, and complement as the OR, AND, and NOT *logical* operations, where "logical" arises from logic theory in which 1 and 0 denote true and false, respectively.

Consider two regions (sets) A and B composed of foreground pixels. The OR of these two sets is the set of elements (coordinates) belonging either to A or B or to both. The AND operation is the set of elements that are common to A and B. The NOT operation of a set A is the set of elements not in A. Because we are dealing with images, if A is a given set of foreground pixels, NOT(A) is the set of all pixels in the image that are not in A, these pixels being background pixels and possibly other foreground pixels. We can think of this operation as turning all elements in A to 0 (black) and all the elements not in A to 1 (white). Figure 2.33 illustrates these operations. Note in the fourth row that the result of the operation shown is the set of foreground pixels that belong to A but not to B, which is the definition of set difference in Eq. (2.6-19). The last row in the figure is the XOR (exclusive OR) operation, which is the set of foreground pixels belonging to A or B, but not both. Observe that the preceding operations are between regions, which clearly can be irregular and of different sizes. This is as opposed to the gray-scale operations discussed earlier, which are array operations and thus require sets whose spatial dimensions are the same. That is, gray-scale set operations involve complete images, as opposed to regions of images.

We need be concerned in theory only with the cability to implement the AND, OR, and NOT logic operators because these three operators are *functionally*

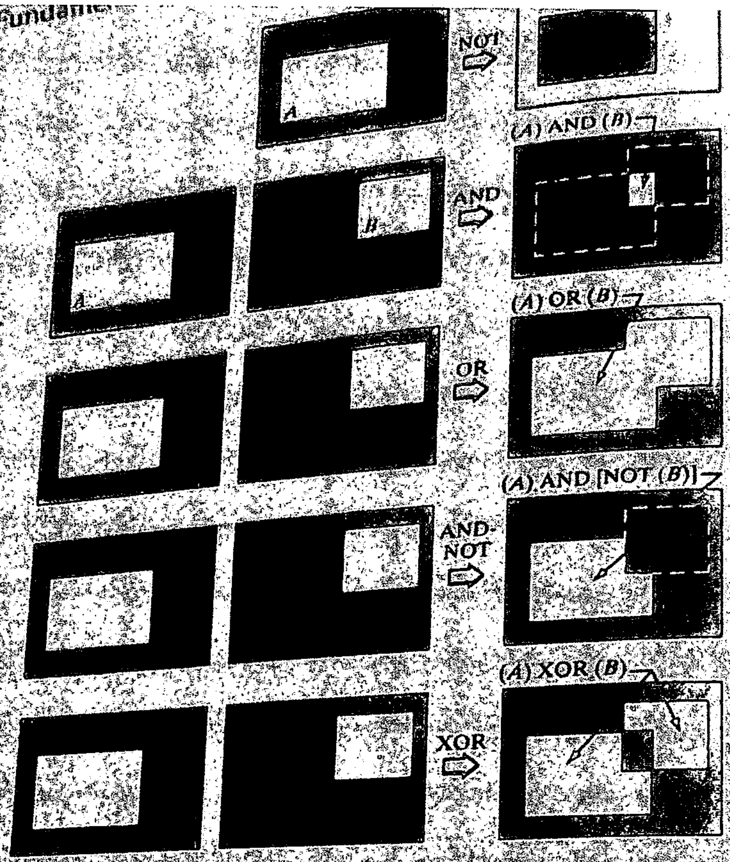


FIGURE 2.33 Illustration of logical operations involving foreground (white) pixels. Black represents binary 0s and white binary 1s. The dashed lines are shown for reference only. They are not part of the result.

complete. In other words, any other logic operator can be implemented by using only these three basic functions, as in the fourth row of Fig. 2.33, where we implemented the set difference operation using AND and NOT. Logic operations are used extensively in image morphology, the topic of Chapter 9.

Fuzzy sets

The preceding set and logical results are *crisp* concepts, in the sense that elements either are or are not members of a set. This presents a serious limitation in some applications. Consider a simple example. Suppose that we wish to categorize all people in the world as being young or not young. Using crisp sets, let U denote the set of all people and let A be a subset of U, which we call the *set of young people*. In order to form set A, we need a *membership function* that assigns a value of 1 or 0 to every element (person) in U. If the value assigned to an element of U is 1, then that element is a member of A; otherwise it is not. Because we are dealing with a bi-valued logic, the membership function simply defines a threshold at or below which a person is considered young, and above which a person is considered not young. Suppose that we define as young any person of age 20 or younger. We see an immediate difficulty. A person whose age is 20 years and 1 sec would not be a member of the set of young people. This limitation arises regardless of the age threshold we use to classify a person as being young. What we need is more flexibility in what we mean by "young," that is, we need a *gradual* transition from young to not young. The theory of *fuzzy sets* implements this concept by that are gradual between the limit values of 1 (definitely young) to 0 (definitely not young). Using fuzzy sets, we can make a statement such as a person being 50% young (in the middle of the transition between young and not young). In other words, age is an imprecise concept, and fuzzy logic provides the tools to deal with such concepts. We explore fuzzy sets in detail in Section 3.8.

2.6.5 Spatial Operations

Spatial operations are performed directly on the pixels of a given image. We classify spatial operations into three broad categories: (1) single-pixel operations, (2) neighborhood operations, and (3) geometric spatial transformations.

Single-pixel operations

The simplest operation we perform on a digital image is to alter the values of its individual pixels based on their intensity. This type of process may be expressed as a transformation function, T, of the form:

$$s = T(z) \tag{2.6-20}$$

where z is the intensity of a pixel in the original image and s is the (mapped) intensity of the corresponding pixel in the processed image. For example, Fig. 2.34 shows the transformation used to obtain the negative of an 8-bit image, such as the image in Fig. 2.32(b), which we obtained using set operations. We discuss in Chapter 3 a number of techniques for specifying intensity transformation functions.

Neighborhood operations

Let S_{xy} denote the set of coordinates of a neighborhood centered on an arbitrary point (x, y) in an image, f. Neighborhood processing generates a corresponding pixel at the same coordinates in an output (processed) image, g, such that the value of that pixel is determined by a specified operation involving the pixels in the input image with coordinates in S_{xy}. For example, suppose that the specified operation is to compute the average value of the pixels in a rectangular neighborhood of size $m \times n$ centered on (x, y). The locations of pixels

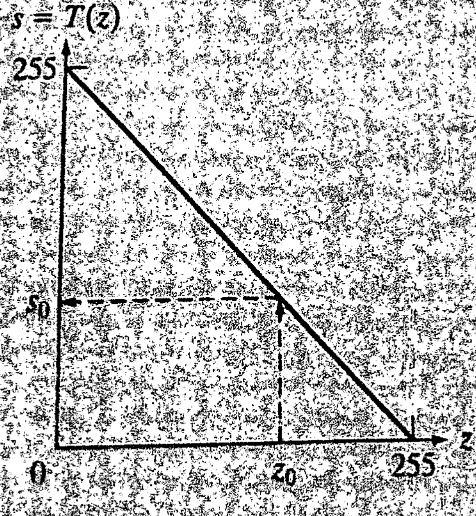
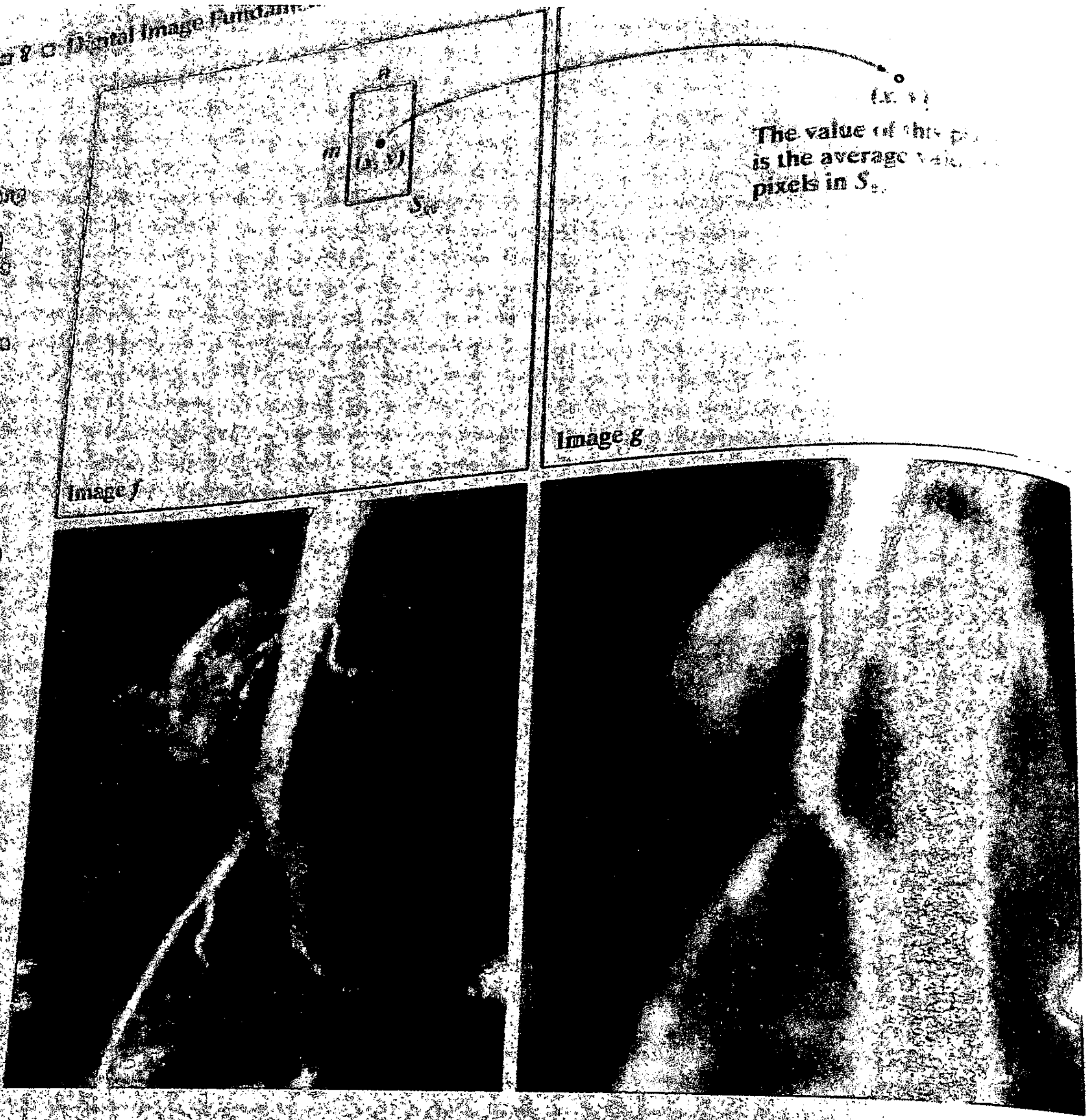


FIGURE 2.34 Intensity transformation function used to obtain the negative of an 8-bit image. The dashed arrows show transformation of an arbitrary input intensity value z_0 into its corresponding output value s_0.



a b
c d

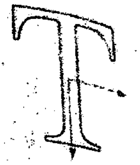
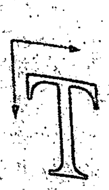
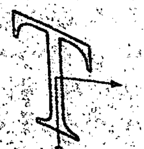
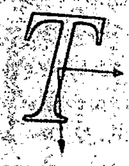
FIGURE 2.35 Local averaging using neighborhood processing. The procedure is illustrated in (a) and (b) for a rectangular neighborhood. (c) The aortic angiogram discussed in Section 1.3.2. (d) The result of using Eq. (2.6-21) with $m = n = 41$. The images are of size 790 × 686 pixels.

in this region constitute the set S_{xy}. Figures 2.35(a) and (b) illustrate the process. We can express this operation in equation form as

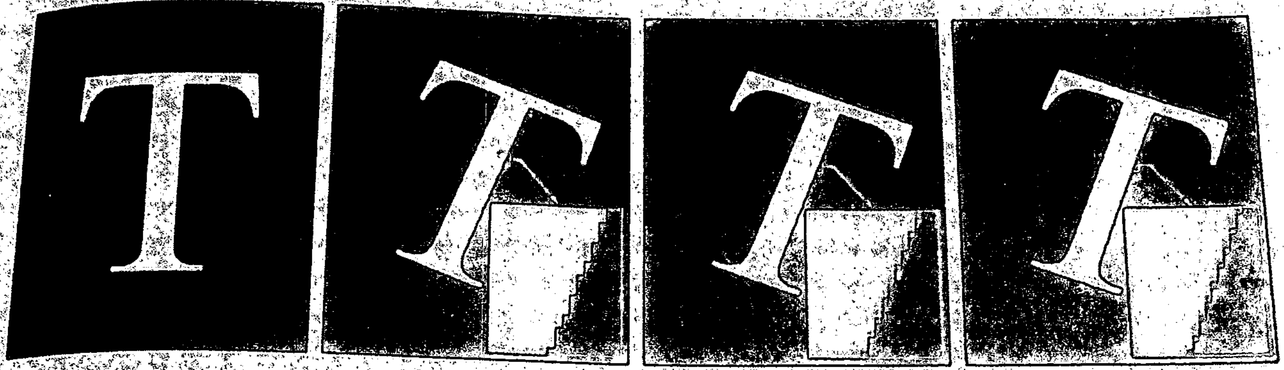
$$g(x, y) = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} f(r, c) \tag{2.6-21}$$

where r and c are the row and column coordinates of the pixels whose coordinates are members of the set S_{xy}. Image g is created by varying the coordinates (x, y) so that the center of the neighborhood moves from pixel to pixel in image f, and repeating the neighborhood operation at each new location. For instance, the image in Fig. 2.35(d) was created in this manner using a neighborhood of size 41 × 41. The net effect is to perform local blurring in the original image. This type of process is used, for example, to eliminate small details and thus render "blobs" corresponding to the largest regions of an image. We

TABLE 2.2 Affine transformations based on Eq. (2.6-23).

		Coordinate Equations	Example
Identity	$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix}c_x & 0 & 0\\0 & c_y & 0\\0 & 0 & 1\end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix}\cos\theta & \sin\theta & 0\\-\sin\theta & \cos\theta & 0\\0 & 0 & 1\end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\sin\theta + w\cos\theta$	
Translation	$\begin{bmatrix}1 & 0 & 0\\0 & 1 & 0\\t_x & t_y & 1\end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix}1 & 0 & 0\\s_v & 1 & 0\\0 & 0 & 1\end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix}1 & s_h & 0\\0 & 1 & 0\\0 & 0 & 1\end{bmatrix}$	$x = v$ $y = s_h v + w$	

each location, (v, w), computing the spatial location, (x, y), of the corresponding pixel in the output image using Eq. (2.6-23) directly. A problem with the forward mapping approach is that two or more pixels in the input image can be transformed to the same location in the output image, raising the question of how to combine multiple output values into a single output pixel. In addition, it is possible that some output locations may not be assigned a pixel at all. The second approach, called *inverse mapping*, scans the output pixel locations and, at each location, (x, y), computes the corresponding location in the input image using $(v, w) = T^{-1}(x, y)$. It then interpolates (using one of the techniques discussed in Section 2.4.4) among the nearest input pixels to determine the intensity of the output pixel value. Inverse mappings are more efficient to implement than forward mappings and are used in numerous commercial implementations of spatial transformations (for example, MATLAB uses this approach).



a b c d

FIGURE 2.36 (a) A 300 dpi image of the letter T. (b) Image rotated 21° using nearest neighbor interpolation to assign intensity values to the spatially transformed pixels. (c) Image rotated 21° using bilinear interpolation. (d) Image rotated 21° using bicubic interpolation. The enlarged sections show edge detail for the three interpolation approaches.

EXAMPLE 2.9: Image rotation and intensity interpolation.

□ The objective of this example is to illustrate image rotation using an affine transform. Figure 2.36(a) shows a 300 dpi image and Figs. 2.36(b)–(d) are the results of rotating the original image by 21°, using nearest neighbor, bilinear, and bicubic interpolation, respectively. Rotation is one of the most demanding geometric transformations in terms of preserving straight-line features. As we see in the figure, nearest neighbor interpolation produced the most jagged edges and, as in Section 2.4.4, bilinear interpolation yielded significantly improved results. As before, using bicubic interpolation produced slightly sharper results. In fact, if you compare the enlarged detail in Figs. 2.36(c) and (d), you will notice in the middle of the subimages that the number of vertical gray "blocks" that provide the intensity transition from light to dark in Fig. 2.36(c) is larger than the corresponding number of blocks in (d), indicting that the latter is a sharper edge. Similar results would be obtained with the other spatial transformations in Table 2.2 that require interpolation (the identity transformation does not, and neither does the translation transformation if the increments are an integer number of pixels). This example was implemented using the inverse mapping approach discussed in the preceding paragraph. ▣

Image registration is an important application of digital image processing used to align two or more images of the same scene. In the preceding discussion, the form of the transformation function required to achieve a desired geometric transformation was known. In image registration, we have available the input and output images, but the specific transformation that produced the output image from the input generally is unknown. The problem, then, is to es-[illegible] the transformation function and then use it to register the two images. [illegible] terminology, the input image is the image that we wish to transform, [illegible] we call the *reference* image is the image against which we want to [illegible] the input.

For example, it may be of interest to align (register) two or more images taken at approximately the same time, but using different imaging systems, such as an MRI (magnetic resonance imaging) scanner and a PET (positron emission tomography) scanner. Or, perhaps the images were taken at different times using the same instrument, such as satellite images of a given location taken several days, months, or even years apart. In either case, combining the images or performing quantitative analysis and comparisons between them requires compensating for geometric distortions caused by differences in viewing angle, distance, and orientation; sensor resolution; shift in object positions; and other factors.

One of the principal approaches for solving the problem just discussed is to use *tie points* (also called *control points*), which are corresponding points whose locations are known precisely in the input and reference images. There are numerous ways to select tie points, ranging from interactively selecting them to applying algorithms that attempt to detect these points automatically. In some applications, imaging systems have physical artifacts (such as small metallic objects) embedded in the imaging sensors. These produce a set of *known* points (called *reseau marks*) directly on all images captured by the system, which can be used as guides for establishing tie points.

The problem of estimating the transformation function is one of modeling. For example, suppose that we have a set of four tie points each in an input and a reference image. A simple model based on a bilinear approximation is given by

$$x = c_1 v + c_2 w + c_3 vw + c_4 \tag{2.6-24}$$

and

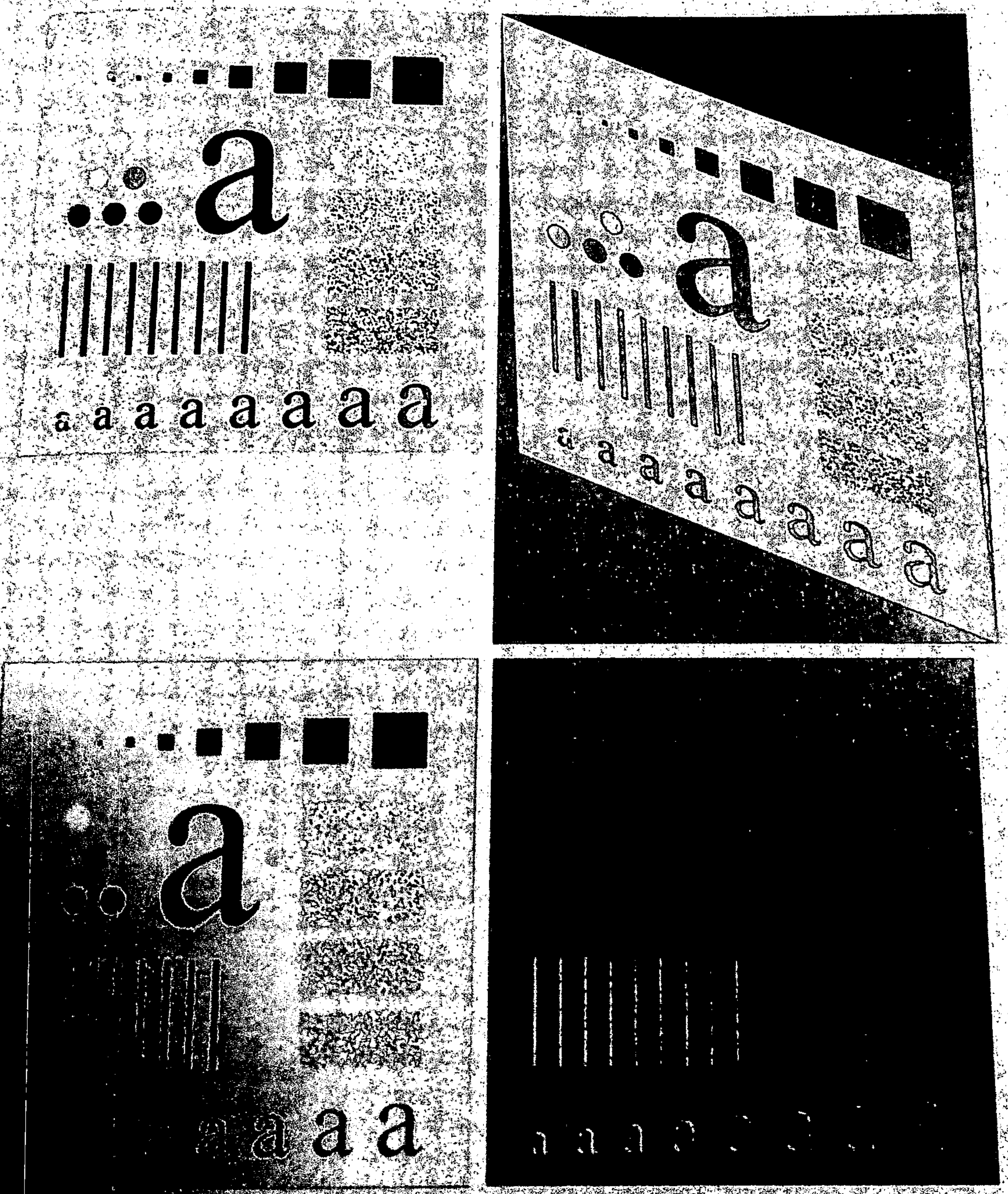
$$y = c_5 v + c_6 w + c_7 vw + c_8 \tag{2.6-25}$$

where, during the estimation phase, (v, w) and (x, y) are the coordinates of tie points in the input and reference images, respectively. If we have four pairs of corresponding tie points in both images, we can write eight equations using Eqs. (2.6-24) and (2.6-25) and use them to solve for the eight unknown coefficients, $c_1, c_2, \ldots, c_8$. These coefficients constitute the model that transforms the pixels of one image into the locations of the pixels of the other to achieve registration.

Once we have the coefficients, Eqs. (2.6-24) and (2.6-25) become our vehicle for transforming all the pixels in the input image to generate the desired new image, which, if the tie points were selected correctly, should be registered with the reference image. In situations where four tie points are insufficient to obtain satisfactory registration, an approach used frequently is to select a larger number of tie points and then treat the quadrilaterals formed by groups of four tie points as subimages. The subimages are processed as above, with all the pixels within a quadrilateral being transformed using the coefficients determined from those tie points. Then we move to another set of four tie points and repeat the procedure until all quadrilateral regions have been processed. Of course, it is possible to use regions that are more complex than quadrilaterals and employ more complex models squares algorithms. In general, the number of control points and sophistication of the model required to solve a problem is dependent on the severity of the geometric distortion. Finally, keep in mind that the transformation defined by Eqs. (2.6-24) and (2.6-25), or any other model for that matter, simply maps the spatial coordinates of the pixels in the input image. We still need to perform intensity interpolation using any of the methods discussed previously to assign intensity values to those pixels.

EXAMPLE 2.10: Image registration.

◼ Figure 2.37(a) shows a reference image and Fig. 2.37(b) shows the same image, but distorted geometrically by vertical and horizontal shear. Our objective is to use the reference image to obtain tie points and then use the tie points to register the images. The tie points we selected (manually) are shown as small white squares near the corners of the images (we needed only four tie



a b
c d

FIGURE 2.37 Image registration. (a) Reference image. (b) Input (geometrically distorted image). Corresponding tie points are shown as small white squares near the corners. (c) Registered image (note the errors in the border). (d) Difference between (a) and (c), showing more registration errors.

points because the distortion is linear shear in both directions). Figure 2.37(c) shows the result of using these tie points in the procedure discussed in the preceding paragraphs to achieve registration. We note that registration was not perfect, as is evident by the black edges in Fig. 2.37(c). The difference image in Fig. 2.37(d) shows more clearly the slight lack of registration between the reference and corrected images. The reason for the discrepancies is error in the manual selection of the tie points. It is difficult to achieve perfect matches for tie points when distortion is so severe. ◻

2.6.6 Vector and Matrix Operations

Consult the Tutorials section in the book Web site for a brief tutorial on vectors and matrices.

Multispectral image processing is a typical area in which vector and matrix operations are used routinely. For example, you will learn in Chapter 6 that color images are formed in RGB color space by using red, green, and blue component images, as Fig. 2.38 illustrates. Here we see that *each* pixel of an RGB image has three components, which can be organized in the form of a *column vector*

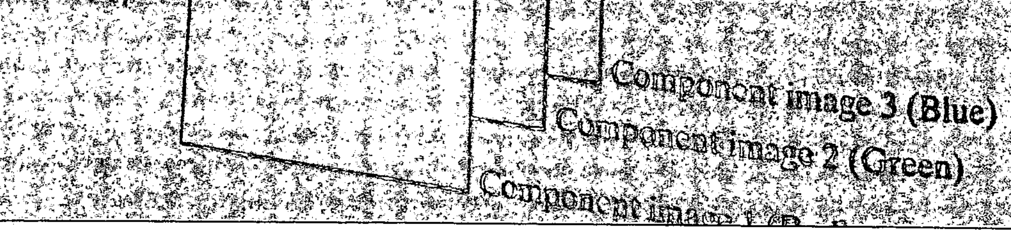
$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \tag{2.6-26}$$

where z_1 is the intensity of the pixel in the red image, and the other two elements are the corresponding pixel intensities in the green and blue images, respectively. Thus an RGB color image of size $M \times N$ can be represented by three component images of this size, or by a total of MN 3-D vectors. A general multispectral case involving n component images (e.g., see Fig. 1.10) will result in n-dimensional vectors. We use this type of vector representation in parts of Chapters 6, 10, 11, and 12.

Once pixels have been represented as vectors we have at our disposal the tools of vector-matrix theory. For example, the *Euclidean distance*, D, between a pixel vector $\mathbf{z}$ and an arbitrary point $\mathbf{a}$ in n-dimensional space is defined as the vector product

$$\begin{aligned} D(\mathbf{z}, \mathbf{a}) &= \left[(\mathbf{z} - \mathbf{a})^T(\mathbf{z} - \mathbf{a})\right]^{\frac{1}{2}} \\ &= \left[(z_1 - a_1)^2 + (z_2 - a_2)^2 + \cdots + (z_n - a_n)^2\right]^{\frac{1}{2}} \end{aligned} \tag{2.6-27}$$

FIGURE 2.38 Formation of a vector from corresponding pixel values in three RGB component images.



We see that this is a generalization of the 2-D Euclidean distance defined in Eq. (2.5-1). Equation (2.6-27) sometimes is referred to as a *vector norm*, denoted by $\|\mathbf{z} - \mathbf{a}\|$. We will use distance computations numerous times in later chapters.

Another important advantage of pixel vectors is in linear transformations, represented as

$$\mathbf{w} = \mathbf{A}(\mathbf{z} - \mathbf{a}) \tag{2.6-28}$$

where $\mathbf{A}$ is a matrix of size $m \times n$ and $\mathbf{z}$ and $\mathbf{a}$ are column vectors of size $n \times 1$. As you will learn later, transformations of this type have a number of useful applications in image processing.

As noted in Eq. (2.4-2), entire images can be treated as matrices (or, equivalently, as vectors), a fact that has important implication in the solution of numerous image processing problems. For example, we can express an image of size $M \times N$ as a vector of dimension $MN \times 1$ by letting the first row of the image be the first N elements of the vector, the second row the next N elements, and so on. With images formed in this manner, we can express a broad range of linear processes applied to an image by using the notation

$$\mathbf{g} = \mathbf{Hf} + \mathbf{n} \tag{2.6-29}$$

where $\mathbf{f}$ is an $MN \times 1$ vector representing an input image, $\mathbf{n}$ is an $MN \times 1$ vector representing an $M \times N$ noise pattern, $\mathbf{g}$ is an $MN \times 1$ vector representing a processed image, and $\mathbf{H}$ is an $MN \times MN$ matrix representing a linear process applied to the input image (see Section 2.6.2 regarding linear processes). It is possible, for example, to develop an entire body of generalized techniques for image restoration starting with Eq. (2.6-29), as we discuss in Section 5.9. We touch on the topic of using matrices again in the following section, and show other uses of matrices for image processing in Chapters 5, 8, 11, and 12.

2.6.7 Image Transforms

All the image processing approaches discussed thus far operate directly on the pixels of the input image; that is, they work directly in the *spatial domain*. In some cases, image processing tasks are best formulated by transforming the input images, carrying the specified task in a *transform domain*, and applying the inverse transform to return to the spatial domain. You will encounter a number of different transforms as you proceed through the book. A particularly important class of 2-D linear transforms, denoted $T(u, v)$, can be expressed in the general form

$$T(u, v) = \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) r(x, y, u, v) \tag{2.6-30}$$

[illegible] is the input image, $r(x, y, u, v)$ is called the *forward transforma-* [illegible] and Eq. (2.6-30) is evaluated for $u = 0, 1, 2, \ldots, M - 1$ and [illegible] $N - 1$. As before, x and y are spatial variables, while M and N

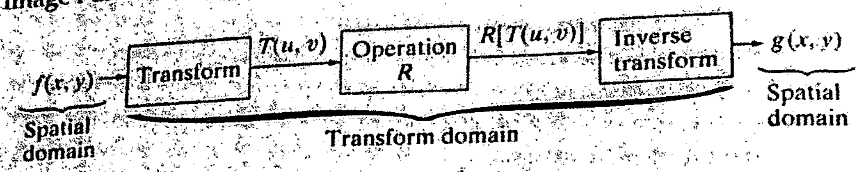


FIGURE 2.39 General approach for operating in the linear transform domain.

are the row and column dimensions of f. Variables u and v are called the *transform variables*. $T(u, v)$ is called the *forward transform* of $f(x, y)$. Given $T(u, v)$, we can recover $f(x, y)$ using the *inverse transform* of $T(u, v)$,

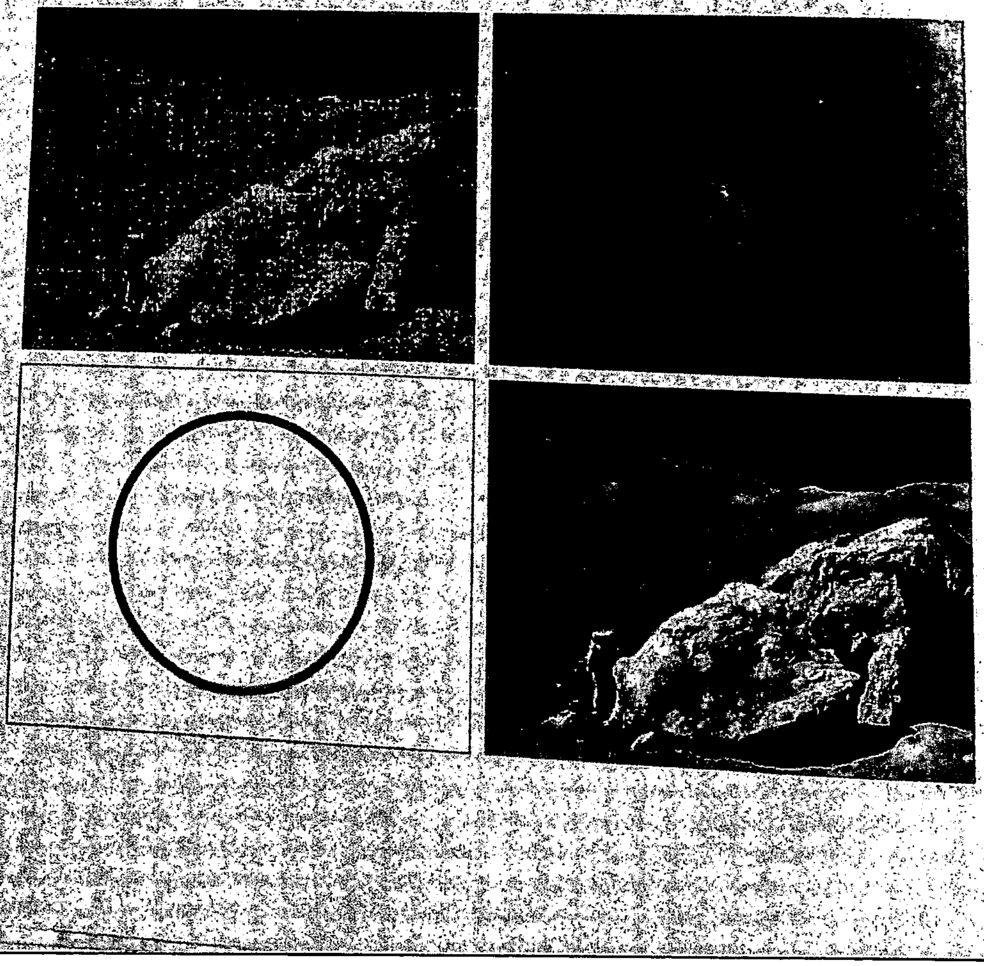
$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} T(u, v)s(x, y, u, v) \qquad (2.6\text{-}31)$$

for $x = 0, 1, 2, \ldots, M - 1$ and $y = 0, 1, 2, \ldots, N - 1$, where $s(x, y, u, v)$ is called the *inverse transformation kernel*. Together, Eqs. (2.6-30) and (2.6-31) are called a *transform pair*.

Figure 2.39 shows the basic steps for performing image processing in the linear transform domain. First, the input image is transformed, the transform is then modified by a predefined operation, and, finally, the output image is obtained by computing the inverse of the modified transform. Thus, we see that the process goes from the spatial domain to the transform domain and then back to the spatial domain.

EXAMPLE 2.11: Image processing in the transform domain.

■ Figure 2.40 shows an example of the steps in Fig. 2.39. In this case the transform used was the Fourier transform, which we mention briefly later in this section and discuss in detail in Chapter 4. Figure 2.40(a) is an image corrupted



a b
c d

FIGURE 2.40 (a) Image corrupted by sinusoidal interference. (b) Magnitude of the Fourier transform showing the bursts of energy responsible for the interference. (c) Mask used to eliminate the energy bursts. (d) Result of computing the inverse of the modified Fourier transform. (Original image courtesy of NASA.)

by sinusoidal interference, and Fig. 2.40(b) is the magnitude of its Fourier transform, which is the output of the first stage in Fig. 2.39. As you will learn in Chapter 4, sinusoidal interference in the spatial domain appears as bright bursts of intensity in the transform domain. In this case, the bursts are in a circular pattern that can be seen in Fig. 2.40(b). Figure 2.40(c) shows a mask image (called a *filter*) with white and black representing 1 and 0, respectively. For this example, the operation in the second box of Fig. 2.39 is to multiply the mask by the transform, thus eliminating the bursts responsible for the interference. Figure 2.40(d) shows the final result, obtained by computing the inverse of the modified transform. The interference is no longer visible, and important detail is quite clear. In fact, you can even see the *fiducial marks* (faint crosses) that are used for image alignment. ☐

The forward transformation kernel is said to be *separable* if

$$r(x, y, u, v) = r_1(x, u)r_2(y, v) \tag{2.6-32}$$

In addition, the kernel is said to be *symmetric* if $r_1(x, y)$ is functionally equal to $r_2(x, y)$, so that

$$r(x, y, u, v) = r_1(x, u)r_1(y, v) \tag{2.6-33}$$

Identical comments apply to the inverse kernel by replacing r with s in the preceding equations.

The 2-D Fourier transform discussed in Example 2.11 has the following forward and inverse kernels:

$$r(x, y, u, v) = e^{-j2\pi(ux/M+vy/N)} \tag{2.6-34}$$

and

$$s(x, y, u, v) = \frac{1}{MN}e^{j2\pi(ux/M+vy/N)} \tag{2.6-35}$$

respectively, where $j = \sqrt{-1}$, so these kernels are complex. Substituting these kernels into the general transform formulations in Eqs. (2.6-30) and (2.6-31) gives us the *discrete Fourier transform pair*:

$$T(u, v) = \sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x, y)e^{-j2\pi(ux/M+vy/N)} \tag{2.6-36}$$

and

$$f(x, y) = \frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1} T(u, v)e^{j2\pi(ux/M+vy/N)} \tag{2.6-37}$$

[illegible] ations are of fundamental importance in digital image processing, [illegible] vote most of Chapter 4 to deriving them starting from basic principles [illegible] then using them in a broad range of applications.

[illegible] difficult to show that the Fourier kernels are separable and symmetric [illegible] oblem 2.25), and that separable and symmetric kernels allow 2-D [illegible] to be computed using 1-D transforms (Problem 2.26). When the forward and inverse kernels of a transform pair satisfy these two conditions, and $f(x, y)$ is a square image of size $M \times M$, Eqs. (2.6-30) and (2.6-31) can be expressed in matrix form:

$$\mathbf{T} = \mathbf{AFA} \tag{2.6-38}$$

where $\mathbf{F}$ is an $M \times M$ matrix containing the elements of $f(x, y)$ [see Eq. (2.4-2)], $\mathbf{A}$ is an $M \times M$ matrix with elements $a_{ij} = r_1(i, j)$, and $\mathbf{T}$ is the resulting $M \times M$ transform with values $T(u, v)$ for $u, v = 0, 1, 2, \ldots, M - 1$.

To obtain the inverse transform, we pre- and post-multiply Eq. (2.6-38) by an inverse transformation matrix $\mathbf{B}$:

$$\mathbf{BTB} = \mathbf{BAFAB} \tag{2.6-39}$$

If $\mathbf{B} = \mathbf{A}^{-1}$,

$$\mathbf{F} = \mathbf{BTB} \tag{2.6-40}$$

indicating that $\mathbf{F}$ [whose elements are equal to image $f(x, y)$] can be recovered completely from its forward transform. If $\mathbf{B}$ is not equal to $\mathbf{A}^{-1}$, then use of Eq. (2.6-40) yields an approximation:

$$\hat{\mathbf{F}} = \mathbf{BAFAB} \tag{2.6-41}$$

In addition to the Fourier transform, a number of important transforms, including the Walsh, Hadamard, discrete cosine, Haar, and slant transforms, can be expressed in the form of Eqs. (2.6-30) and (2.6-31) or, equivalently, in the form of Eqs. (2.6-38) and (2.6-40). We discuss several of these and some other types of image transforms in later chapters.

2.6.8 Probabilistic Methods

Consult the Tutorials section in the book Web site for a brief overview of probability theory.

Probability finds its way into image processing work in a number of ways. The simplest is when we treat intensity values as random quantities. For example, let z_i, $i = 0, 1, 2, \ldots, L - 1$, denote the values of all possible intensities in an $M \times N$ digital image. The probability, $p(z_k)$, of intensity level z_k occurring in a given image is estimated as

$$p(z_k) = \frac{n_k}{MN} \tag{2.6-42}$$

where n_k is the number of times that intensity z_k occurs in the image and MN is the total number of pixels. Clearly,

$$\sum_{k=0}^{L-1} p(z_k) = 1 \tag{2.6-43}$$

Once we have $p(z_k)$, we can determine a number of important image characteristics. For example, the mean (average) intensity is given by

$$m = \sum_{k=0}^{L-1} z_k \, p(z_k) \tag{2.6-44}$$

Similarly, the variance of the intensities is

$$\sigma^2 = \sum_{k=0}^{L-1} (z_k - m)^2 p(z_k) \qquad (2.6\text{-}45)$$

The variance is a measure of the spread of the values of z about the mean, so it is a useful measure of image contrast. In general, the nth moment of random variable z about the mean is defined as

$$\mu_n(z) = \sum_{k=0}^{L-1} (z_k - m)^n p(z_k) \qquad (2.6\text{-}46)$$

We see that $\mu_0(z) = 1$, $\mu_1(z) = 0$, and $\mu_2(z) = \sigma^2$. Whereas the mean and variance have an immediately obvious relationship to visual properties of an image, higher-order moments are more subtle. For example, a positive third moment indicates that the intensities are biased to values higher than the mean, a negative third moment would indicate the opposite condition, and a zero third moment would tell us that the intensities are distributed approximately equally on both sides of the mean. These features are useful for computational purposes, but they do not tell us much about the appearance of an image in general.

The units of the variance are in intensity values squared. When comparing contrast values, we usually use the standard deviation, σ (square root of the variance), instead because its dimensions are directly in terms of intensity values.

EXAMPLE 2.12: Comparison of standard deviation values as measures of image intensity contrast.

■ Figure 2.41 shows three 8-bit images exhibiting low, medium, and high contrast, respectively. The standard deviations of the pixel intensities in the three images are 14.3, 31.6, and 49.2 intensity levels, respectively. The corresponding variance values are 204.3, 997.8, and 2424.9, respectively. Both sets of values tell the same story but, given that the range of possible intensity values in these images is [0, 255], the standard deviation values relate to this range much more intuitively than the variance. ◻

As you will see in progressing through the book, concepts from probability play a central role in the development of image processing algorithms. For example, in Chapter 3 we use the probability measure in Eq. (2.6-42) to derive intensity transformation algorithms. In Chaper 5, we use probability and matrix formulations to develop image restoration algorithms. In Chapter 10, probability is used for image segmentation, and in Chapter 11 we use it for texture description. In Chapter 12, we derive optimum object recognition techniques based on a probabilistic formulation.

a b c

FIGURE 2.41 Images exhibiting (a) low contrast, (b) medium contrast, and (c) high contrast.

Thus far, we have addressed the issue of applying probability to a single random variable (intensity) over a single 2-D image. If we consider sequences of images, we may interpret the third variable as time. The tools needed to handle this added complexity are *stochastic* image processing techniques (the word *stochastic* is derived from a Greek word meaning roughly "to aim at a target," implying randomness in the outcome of the process). We can go a step further and consider an *entire* image (as opposed to a point) to be a spatial random event. The tools needed to handle formulations based on this concept are techniques from *random fields*. We give one example in Section 5.8 of how to treat entire images as random events, but further discussion of stochastic processes and random fields is beyond the scope of this book. The references at the end of this chapter provide a starting point for reading about these topics.

Summary

The material in this chapter is primarily background for subsequent discussions. Our treatment of the human visual system, although brief, provides a basic idea of the capabilities of the eye in perceiving pictorial information. The discussion on light and the electromagnetic spectrum is fundamental in understanding the origin of the many images we use in this book. Similarly, the image model developed in Section 2.3.4 is used in the Chapter 4 as the basis for an image enhancement technique called *homomorphic filtering*.

The sampling and interpolation ideas introduced in Section 2.4 are the foundation for many of the digitizing phenomena you are likely to encounter in practice. We will return to the issue of sampling and many of its ramifications in Chapter 4, after you have mastered the Fourier transform and the frequency domain.

The concepts introduced in Section 2.5 are the basic building blocks for processing techniques based on pixel neighborhoods. For example, as we show in the following chapter, and in Chapter 5, neighborhood processing methods are at the core of many image enhancement and restoration procedures. In Chapter 9, we use neighborhood operations for image morphology; in Chapter 10, we use them for image segmentation; and in Chapter 11 for image description. When applicable, neighborhood processing is favored in commercial applications of image processing because of their operational speed and simplicity of implementation in hardware and/or firmware.

The material in Section 2.6 will serve you well in your journey through the book. Although the level of the discussion was strictly introductory, you are now in a position to conceptualize what it means to process a digital image. As we mentioned in that section, the tools introduced there are expanded as necessary in the following chapters. Rather than dedicate an entire chapter or appendix to develop a comprehensive treatment of mathematical concepts in one place, you will find it considerably more meaningful to learn the necessary extensions of the mathematical tools from Section 2.6 in later chapters, in the context of how they are applied to solve problems in image processing.

References and Further Reading

Additional reading for the material in Section 2.1 regarding the structure of the human eye may be found in Atchison and Smith [2000] and Oyster [1999]. For additional reading on visual perception, see Regan [2000] and Gordon [1997]. The book by Hubel [1988] and the classic book by Cornsweet [1970] also are of interest. Born and Wolf [1999] is a basic reference that discusses light in terms of electromagnetic theory. Electromagnetic energy propagation is covered in some detail by Felsen and Mar

The area of image sensing is quite broad and very fast moving. An excellent source of information on optical and other imaging sensors is the Society for Optical Engineering (SPIE). The following are representative publications by the SPIE in this area: Blouke et al. [2001], Hoover and Doty [1996], and Freeman [1987].

The image model presented in Section 2.3.4 is from Oppenheim, Schafer, and Stockham [1968]. A reference for the illumination and reflectance values used in that section is the *IESNA Lighting Handbook* [2000]. For additional reading on image sampling and some of its effects, such as aliasing, see Bracewell [1995]. We discuss this topic in more detail in Chapter 4. The early experiments mentioned in Section 2.4.3 on perceived image quality as a function of sampling and quatization were reported by Huang [1965]. The issue of reducing the number of samples and intensity levels in an image while minimizing the ensuing degradation is still of current interest, as exemplified by Papamarkos and Atsalakis [2000]. For further reading on image shrinking and zooming, see Sid-Ahmed [1995], Unser et al. [1995], Umbaugh [2005], and Lehmann et al. [1999]. For further reading on the topics covered in Section 2.5, see Rosenfeld and Kak [1982], Marchand-Maillet and Sharaiha [2000], and Ritter and Wilson [2001].

Additional reading on linear systems in the context of image processing (Section 2.6.2) may be found in Castleman [1996]. The method of noise reduction by image averaging (Section 2.6.3) was first proposed by Kohler and Howell [1963]. See Peebles [1993] regarding the expected value of the mean and variance of a sum of random variables. Image subtraction (Section 2.6.3) is a generic image processing tool used widely for change detection. For image subtraction to make sense, it is necessary that the images being subtracted be registered or, alternatively, that any artifacts due to motion be identified. Two papers by Meijering et al. [1999, 2001] are illustrative of the types of techniques used to achieve these objectives.

A basic reference for the material in Section 2.6.4 is Cameron [2005]. For more advanced reading on this topic, see Tourlakis [2003]. For an introduction to fuzzy sets, see Section 3.8 and the corresponding references in Chapter 3. For further details on single point and neighborhood processing (Section 2.6.5), see Sections 3.2 through 3.4 and the references on these topics in Chapter 3. For geometric spatial transformations, see Wolberg [1990].

Noble and Daniel [1988] is a basic reference for matrix and vector operations (Section 2.6.6). See Chapter 4 for a detailed discussion on the Fourier transform (Section 2.6.7), and Chapters 7, 8, and 11 for examples of other types of transforms used in digital image processing. Peebles [1993] is a basic introduction to probability and random variables (Section 2.6.8) and Papoulis [1991] is a more advanced treatment of this topic. For foundation material on the use of stochastic and random fields for image processing, see Rosenfeld and Kak [1982], Jähne [2002], and Won and Gray [2004].

For details of software implementation of many of the techniques illustrated in this chapter, see Gonzalez, Woods, and Eddins [2004].

Problems

Detailed solutions to th problems marked with star can be found in th book Web site. The si also contains suggeste projects based on the m terial in this chapter.

[illegible] Using the background information provided in Section 2.1, and thinking purely [illegible] geometric terms, estimate the diameter of the smallest printed dot that the [illegible] can discern if the page on which the dot is printed is 0.2 m away from the [illegible] Assume for simplicity that the visual system ceases to detect the dot when [illegible] image of the dot on the fovea becomes smaller than the diameter of one re[illegible] (cone) in that area of the retina. Assume further that the fovea can be modeled as a square array of dimensions 1.5 mm × 1.5 mm, and that the cones and spaces between the cones are distributed uniformly throughout this array.

2.2 When you enter a dark theater on a bright day, it takes an appreciable interval of time before you can see well enough to find an empty seat. Which of the visual processes explained in Section 2.1 is at play in this situation?

★2.3 Although it is not shown in Fig. 2.10, alternating current certainly is part of the electromagnetic spectrum. Commercial alternating current in the United States has a frequency of 60 Hz. What is the wavelength in kilometers of this component of the spectrum?

2.4 You are hired to design the front end of an imaging system for studying the boundary shapes of cells, bacteria, viruses, and protein. The front end consists, in this case, of the illumination source(s) and corresponding imaging camera(s). The diameters of circles required to enclose individual specimens in each of these categories are 50, 1, 0.1, and 0.01 μm, respectively.

(a) Can you solve the imaging aspects of this problem with a single sensor and camera? If your answer is yes, specify the illumination wavelength band and the type of camera needed. By "type," we mean the band of the electromagnetic spectrum to which the camera is most sensitive (e.g., infrared).

(b) If your answer in (a) is no, what type of illumination sources and corresponding imaging sensors would you recommend? Specify the light sources and cameras as requested in part (a). Use the *minimum* number of illumination sources and cameras needed to solve the problem.

By "solving the problem," we mean being able to detect circular details of diameter 50, 1, 0.1, and 0.01 μm, respectively.

2.5 A CCD camera chip of dimensions 7 × 7 mm, and having 1024 × 1024 elements, is focused on a square, flat area, located 0.5 m away. How many line pairs per mm will this camera be able to resolve? The camera is equipped with a 35-mm lens. (*Hint:* Model the imaging process as in Fig. 2.3, with the focal length of the camera lens substituting for the focal length of the eye.)

★2.6 An automobile manufacturer is automating the placement of certain components on the bumpers of a limited-edition line of sports cars. The components are color coordinated, so the robots need to know the color of each car in order to select the appropriate bumper component. Models come in only four colors: blue, green, red, and white. You are hired to propose a solution based on imaging. How would you solve the problem of automatically determining the color of each car, keeping in mind that *cost* is the most important consideration in your choice of components?

2.7 Suppose that a flat area with center at (x_0, y_0) is illuminated by a light source with intensity distribution

$$i(x, y) = Ke^{-[(x-x_0)^2+(y-y_0)^2]}$$

Assume for simplicity that the reflectance of the area is constant and equal to 1.0, and let $K = 255$. If the resulting image is digitized with k bits of intensity resolution, and the eye can detect an abrupt change of eight shades of intensity between adjacent pixels, what value of k will cause visible false contouring?

2.8 Sketch the image in Problem 2.7 for $k = 2$.

★2.9 A common measure of transmission for digital data is
the number of bits transmitted per

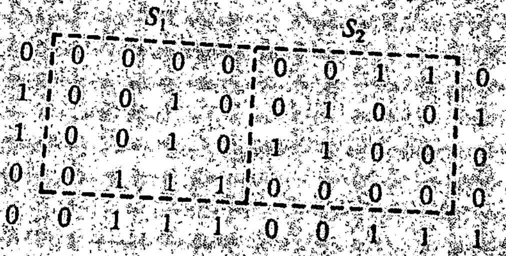
in packets consisting of a start bit, a byte (8 bits) of information, and a stop bit. Using these facts, answer the following:

(a) How many minutes would it take to transmit a 1024 × 1024 image with 256 intensity levels using a 56K baud modem?

(b) What would the time be at 3000K baud, a representative medium speed of a phone DSL (Digital Subscriber Line) connection?

2.10 High-definition television (HDTV) generates images with 1125 horizontal TV lines interlaced (where every other line is painted on the tube face in each of two fields, each field being 1/60th of a second in duration). The width-to-height aspect ratio of the images is 16:9. The fact that the number of horizontal lines is fixed determines the vertical resolution of the images. A company has designed an image capture system that generates digital images from HDTV images. The resolution of each TV (horizontal) line in their system is in proportion to vertical resolution, with the proportion being the width-to-height ratio of the images. Each pixel in the color image has 24 bits of intensity resolution, 8 bits each for a red, a green, and a blue image. These three "primary" images form a color image. How many bits would it take to store a 2-hour HDTV movie?

★2.11 Consider the two image subsets, S_1 and S_2, shown in the following figure. For $V = \{1\}$, determine whether these two subsets are (a) 4-adjacent, (b) 8-adjacent, or (c) m-adjacent.



★2.12 Develop an algorithm for converting a one-pixel-thick 8-path to a 4-path.

2.13 Develop an algorithm for converting a one-pixel-thick m-path to a 4-path.

2.14 Refer to the discussion at the end of Section 2.5.2, where we defined the background as $(R_u)^c$, the complement of the union of all the regions in an image. In some applications, it is advantageous to define the background as the subset of pixels $(R_u)^c$ that are not region hole pixels (informally, think of holes as sets of background pixels surrounded by region pixels). How would you modify the definition to exclude hole pixels from $(R_u)^c$? An answer such as "the background is the subset of pixels of $(R_u)^c$ that are not hole pixels" is not acceptable. (*Hint:* Use the concept of connectivity.)

2.15 Consider the image segment shown.

(a) Let $V = \{0, 1\}$ and compute the lengths of the shortest 4-, 8-, and m-path between p and q. If a particular path does not exist between these two points, explain why.

(b) Repeat for $V = \{1, 2\}$.

$$\begin{array}{cccc}
3 & 1 & 2 & 1\,(q) \\
2 & 2 & 0 & 2 \\
1 & 2 & 1 & 1 \\
(p)\,1 & 0 & 1 & 2
\end{array}$$

2.16 ☆(a) Give the condition(s) under which the D_4 distance between two points p and q is equal to the shortest 4-path between these points.

(b) Is this path unique?

2.17 Repeat Problem 2.16 for the D_8 distance.

☆2.18 In the next chapter, we will deal with operators whose function is to compute the sum of pixel values in a small subimage area, S. Show that these are linear operators.

2.19 The median, ζ, of a set of numbers is such that half the values in the set are below ζ and the other half are above it. For example, the median of the set of values $\{2, 3, 8, 20, 21, 25, 31\}$ is 20. Show that an operator that computes the median of a subimage area, S, is nonlinear.

☆2.20 Prove the validity of Eqs. (2.6-6) and (2.6-7). [*Hint:* Start with Eq. (2.6-4) and use the fact that the expected value of a sum is the sum of the expected values.]

2.21 Consider two 8-bit images whose intensity levels span the full range from 0 to 255.

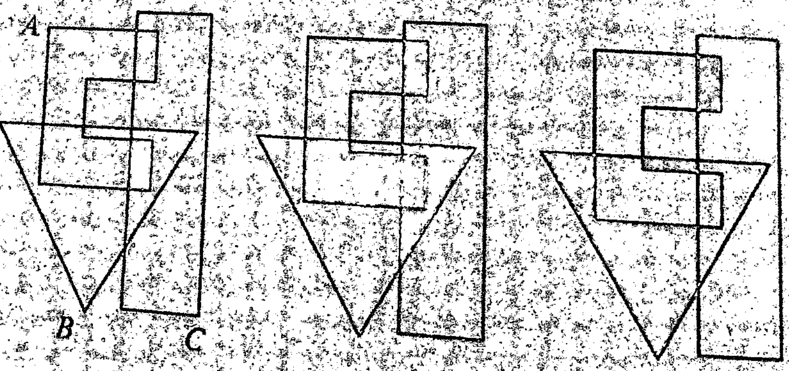
(a) Discuss the limiting effect of repeatedly subtracting image (2) from image (1). Assume that the result is represented also in eight bits.

(b) Would reversing the order of the images yield a different result?

★2.22 Image subtraction is used often in industrial applications for detecting missing components in product assembly. The approach is to store a "golden" image that corresponds to a correct assembly; this image is then subtracted from incoming images of the same product. Ideally, the differences would be zero if the new products are assembled correctly. Difference images for products with missing components would be nonzero in the area where they differ from the golden image. What conditions do you think have to be met in practice for this method to work?

2.23 ★(a) With reference to Fig. 2.31, sketch the set $(A \cap B) \cup (A \cup B)^c$.

(b) Give expressions for the sets shown shaded in the following figure in terms of sets A, B, and C. The shaded areas in each figure constitute one set, so give one expression for each of the three figures.



2.24 What would be the equations analogous to Eqs. (2.6-24) and (2.6-25) that would result from using triangular instead of quadrilateral regions?

2.25 Prove that the Fourier kernels in Eqs. (2.6-34) and (2.6-35) are separable and symmetric.

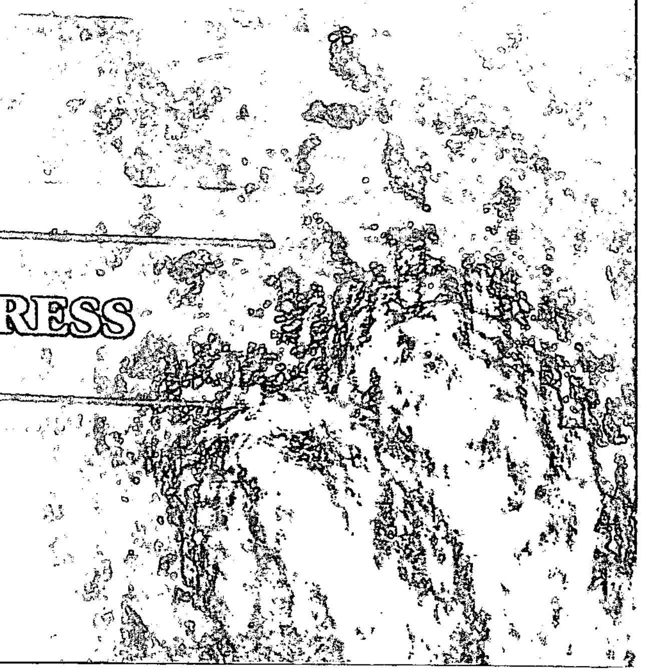
★2.26 Show that 2-D transforms with separable, symmetric kernels can be compute by (1) computing 1-D transforms along the individual rows (columns) of th input, followed by (2) computing 1-D transforms along the columns (rows) o the result from step (1).

2.27 A plant produces a line of translucent miniature polymer squares. Stringent quality requirements dictate 100% visual inspection, and the plant manager finds the use of human inspectors increasingly expensive. Inspection is semiautomated. At each inspection station, a robotic mechanism places each polymer square over a light located under an optical system that produces a magnified image of the square. The image completely fills a viewing screen measuring 80 × 80 mm. Defects appear as dark circular blobs, and the inspector's job is to look at the screen and reject any sample that has one or more such dark blobs with a diameter of 0.8 mm or larger, as measured on the scale of the screen. The manager believes that if she can find a way to automate the process completely, she will increase profits by 50%. She also believes that success in this project will aid her climb up the corporate ladder. After much investigation, the manager decides that the way to solve the problem is to view each inspection screen with a CCD TV camera and feed the output of the camera into an image processing system capable of detecting the blobs, measuring their diameter, and activating the accept/reject buttons previously operated by an inspector. She is able to find a system that can do the job, as long as the smallest defect occupies an area of at least 2 × 2 pixels in the digital image. The manager hires you to help her specify the camera and lens system, but requires that you use off-the-shelf components. For the lenses, assume that this constraint means any integer multiple of 25 mm or 35 mm, up to 200 mm. For the cameras, it means resolutions of 512 × 512, 1024 × 1024, or 2048 × 2048 pixels. The *individual* imaging elements in these cameras are squares measuring 8 × 8 µm, and the spaces between imaging elements are 2 µm. For this application, the cameras cost much more than the lenses, so the problem should be solved with the lowest-resolution camera possible, based on the choice of lenses. As a consultant, you are to provide a written recommendation, showing in reasonable detail the analysis that led to your conclusion. Use the same imaging geometry suggested in Problem 2.5.

The IMAGE PROCESSING Handbook

Fourth Edition

John C. Russ



CRC PRESS

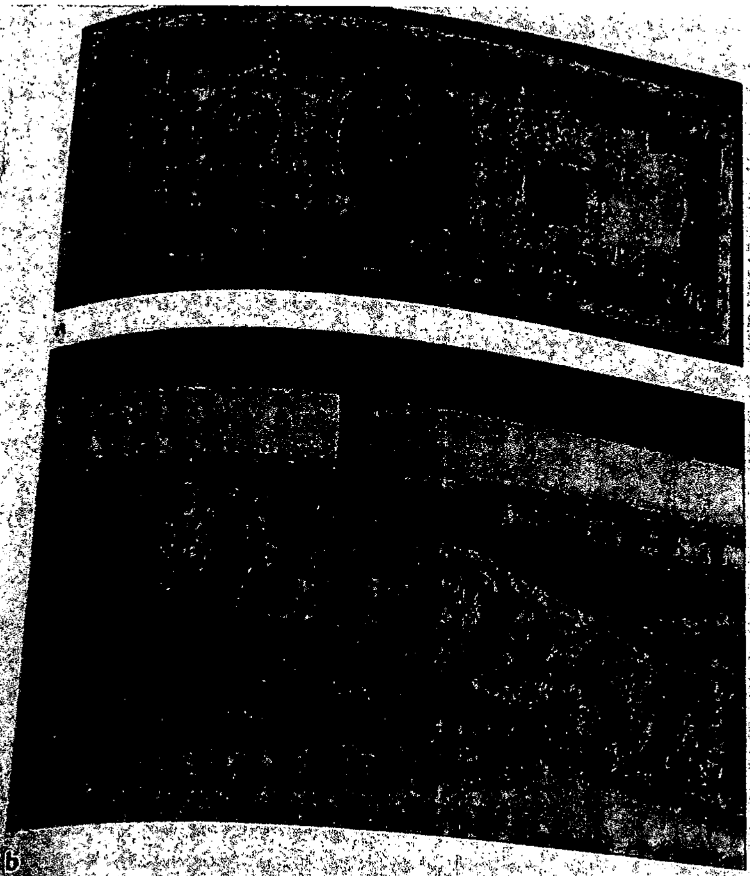


Figure 70. (a) Large mosaic image assembled from eight individual images, each 1600 × 1200 pixels from a digital camera, as discussed in the text. (b) detail of fit between two image tiles.

For scientific imaging purposes, the use of a microscope stage or other specimen positioning de-
ice that can shift the sample being imaged with reasonable precision while the camera remains
fixed would seem to offer the possibility of acquiring images of unlimited size. More constraints are
here to assist in the fitting together of the image. It is known, for example, that the images may be
ghtly rotated (the mechanism may have slight wobble or misalignment and the shifting cannot
counted on to provide exact edge-to-edge alignment), but they cannot be distorted (i.e., straight
s remain straight and angles are unchanged). Thus, the fitting together process can only shift
rotate the individual image tiles in their entirety.

erlap between tiles is between 10 and 20%, and the angular mismatch is no more than
es, matching each of the tiles together can indeed produce large high-resolution mo-
wn in **Figure 70**. The matching technique is based on cross-correlation (discussed
and an iterative procedure was used to match all of the tiles together for a best
ticularly effective for images acquired in the atomic force microscope, because the
these devices tends to be rather small, and with the very high spatial resolu-
ssible to design specimen shifting hardware that is absolutely precise (Condeco

aligned, it is possible to write the equations either in terms of the coordi-
as a function of the geometrically corrected one, or vice versa. In prac-
to use the grid of x,y coordinates in the corrected image to calculate for

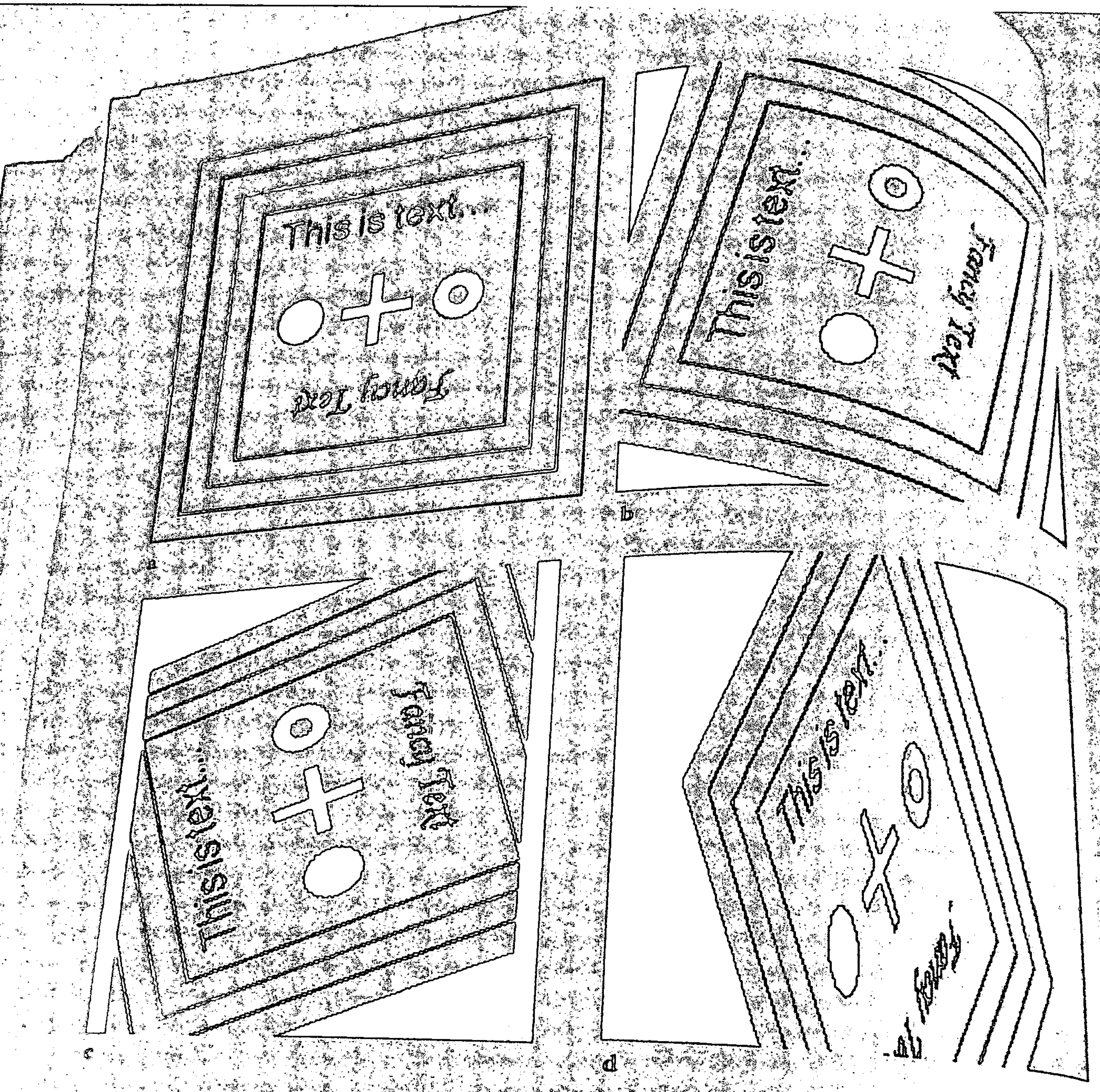


Figure 71. Rotation and stretching of a test image:
(a) original;
(b) rotation only, no change in scale;
(c) rotation and uniform stretching while maintaining angles;
(d) general rotation and stretching in which angles may vary (lines remain straight).

[illegible] in the original image, and to perform the calculation in terms of actual pixel

[illegible] coordinates for the original location will only rarely be integers
[illegible] lies "between" the pixels in the original image. Several methods are
[illegible]. The simplest is to truncate the calculated values so that the frac

ional part of the address is discarded and the pixel lying toward the origin of the coordinate system is used. Slightly better results are obtained by rounding the address values to select the nearest pixel, whose brightness is then copied to the transformed image array.

dier method introduces some error in location that can cause distortion of the transformed image. **Figure 71** shows examples using a test pattern in which the biasing of the lines and apparent variations in their width is evident.

When this distortion is unacceptable, another method may be used which requires more calculation. The brightness value for the transformed pixel may be calculated by interpolating between the four pixels surrounding the calculated address. This is called bilinear interpolation, and is calculated simply from the fractional part of the *X* and *Y* coordinates. First the interpolation is done in one direction, and then in the other, as indicated in **Figure 72.** For a location with coordinates *j* + *x*, *k* + *y* where *x* and *y* are the fractional part of the address, the equations for the first interpolation are

$$B_{j+x,k} = (1-x)\cdot B_{j,k} + x\cdot B_{j+1,k}$$
$$B_{j+x,k+1} = (1-x)\cdot B_{j,k+1} + x\cdot B_{j+1,k+1} \tag{11}$$

then the second interpolation, in the y direction, gives the final value

$$B_{j+x,k+y} = (1-y)\cdot B_{j+x,k} + y\cdot B_{j+x,k+1} \tag{12}$$

ghted interpolations over larger regions are also used in some cases. One of the most popular cubic fitting. Whereas bilinear interpolation uses a 2 × 2 array of neighboring pixel values to ulate the interpolated value, the cubic method uses a 4 × 4 array. Using the same notation as bilinear interpolation in **Equations 11** and **12**, the summations now go from *k*–1 to *k*+2 and *j*–1 to *k*+2. The intermediate values from the horizontal interpolation are:

$$B_j+_{x,k} = (1/6)\ (B_{j-1,k}\cdot R_1 + B_{j,k}\cdot R_2 + B_j+1,k\cdot R_3 + Bj+_2,k\cdot R_4) \tag{13}$$

the interpolation in the vertical direction is

$$B_{j+x,k+y} = (1/6)\ (B_j+_{x,k-1}\cdot R_1 + B_{j+x,k}\cdot R_2 + B_{j+x,k+1}\cdot R_3 + B_{j+x,k+2}\cdot R_4) \tag{14}$$

of pixel interpolation. The brightness values are first interpolated horizontally to ness values at the locations outlined in two values are interpolated vertically to at the target pixel outlined in blue, of the pixel addresses.

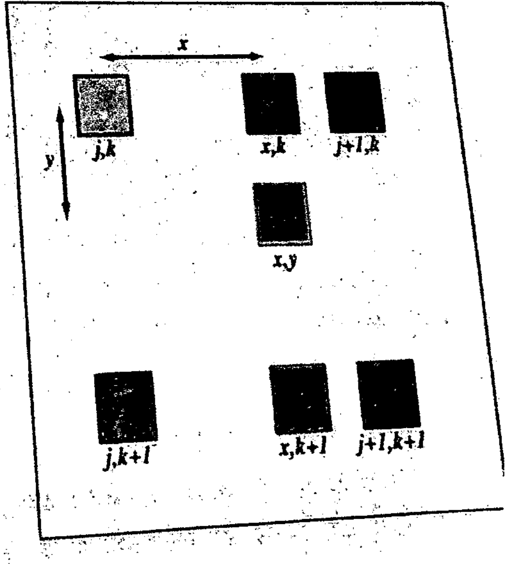
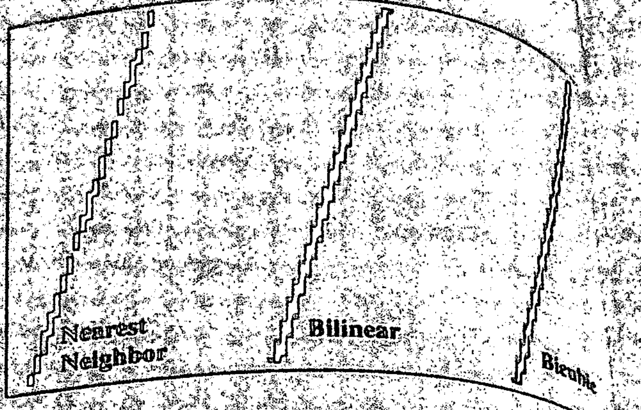


Figure 73. Effect of rotating a 1 pixel wide black line using nearest neighbor, bilinear, and bicubic interpolation.



where the weighting factors R_i are calculated from the real part (x or y respectively) of the add as

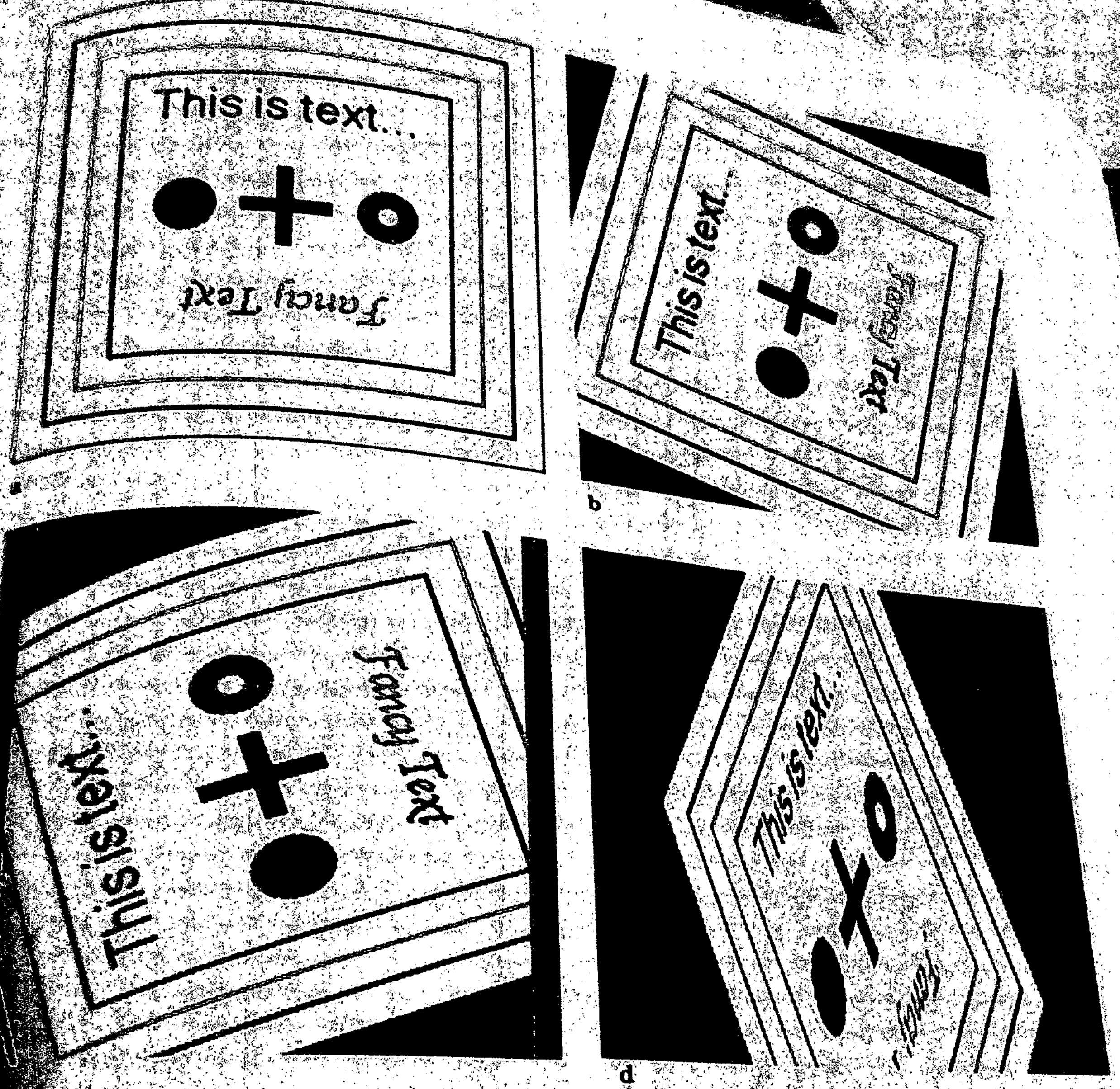
$$
\begin{aligned}
R_1 &= (3+x)^3 - 4\cdot(2+x)^3 + 6\cdot(1+x)^3 - 4\cdot x^3 \\
R_2 &= (2+x)^3 - 4\cdot(1+x)^3 + 6\cdot x^3 \\
R_3 &= (1+x)^3 - 4\cdot x^3 \\
R_4 &= x^3
\end{aligned}
$$

The bicubic fit is more isotropic than the bilinear method. Interpolation always has the effect of smoothing the image and removing some high frequency information, but minimizes aliasing or "stair-stepping" along lines and edges. Figure 73 shows the results of rotating a line (originally a black vertical line one pixel wide) by 17° with no interpolation (selecting the nearest neighbor pixel value), bilinear, and bicubic interpolation. The aliasing with the nearest neighbor method is evident. Bilinear interpolation reduces the line contrast more than bicubic, and both assign grey values to adjacent pixels to smooth the appearance of the line.

The advantage of interpolation is that dimensions are altered as little as possible in the transformation, and in particular boundaries and other lines are not biased or distorted. Figure 7 shows the same examples as Figure 71, with bilinear interpolation used. Careful examination of the figure shows that the lines appear straight and not "aliased" or stair-stepped, because some of the pixels along the sides of the lines have intermediate grey values resulting from the interpolation. In fact, computer graphics sometimes uses this same method to draw lines on CRT displays so that the stair-stepping inherent in drawing lines on a discrete pixel array is avoided. The technique is called anti-aliasing and produces lines whose pixels have grey values according to how close they lie to the mathematical location of the line. This fools the viewer into perceiving a smooth line.

Fitting of higher-order polynomials, or adaptive spline fits to the pixel intensity values, can also be used. This can be particularly useful when enlarging images in order to reduce the perceived fuzziness that results when sharp edges are spread out by conventional interpolation. Figure 7 shows an example (a fragment of the "flowers" image) in which a 4× enlargement has been performed using no interpolation, bilinear interpolation, adaptive spline fitting and fractal interpolation. The latter inserts false "detail" into the image, while spline fitting maintains the sharpness of edges best.

For image warping or rubber-sheeting, interpolation has the advantage that dimensions are preserved, although brightness values are not. With the nearest-pixel method achieved by rounding



[illegible] 74. *Same generalized rotation and stretching as in* ***Figure 71****, but with bilinear interpolation. Note* [illegible] *oothing of the lines and boundaries.*

[illegible] dresses, the dimensions are distorted but the brightness values are preserved. Choos- [illegible] ethod is appropriate to a particular imaging task depends primarily on which kind of [illegible] ore important, and secondarily on the additional computational effort required [illegible] me.

[illegible] the effect of adding higher order terms to the warping equations. With qua- [illegible] oidal distortion of a short focal length lens or SEM can be corrected. It is also [illegible] distortion of a spherical surface closely over modest distances. With higher [illegible] on is possible, but this is rarely useful in an image processing situation [illegible] determine such a distortion are not likely to be available.

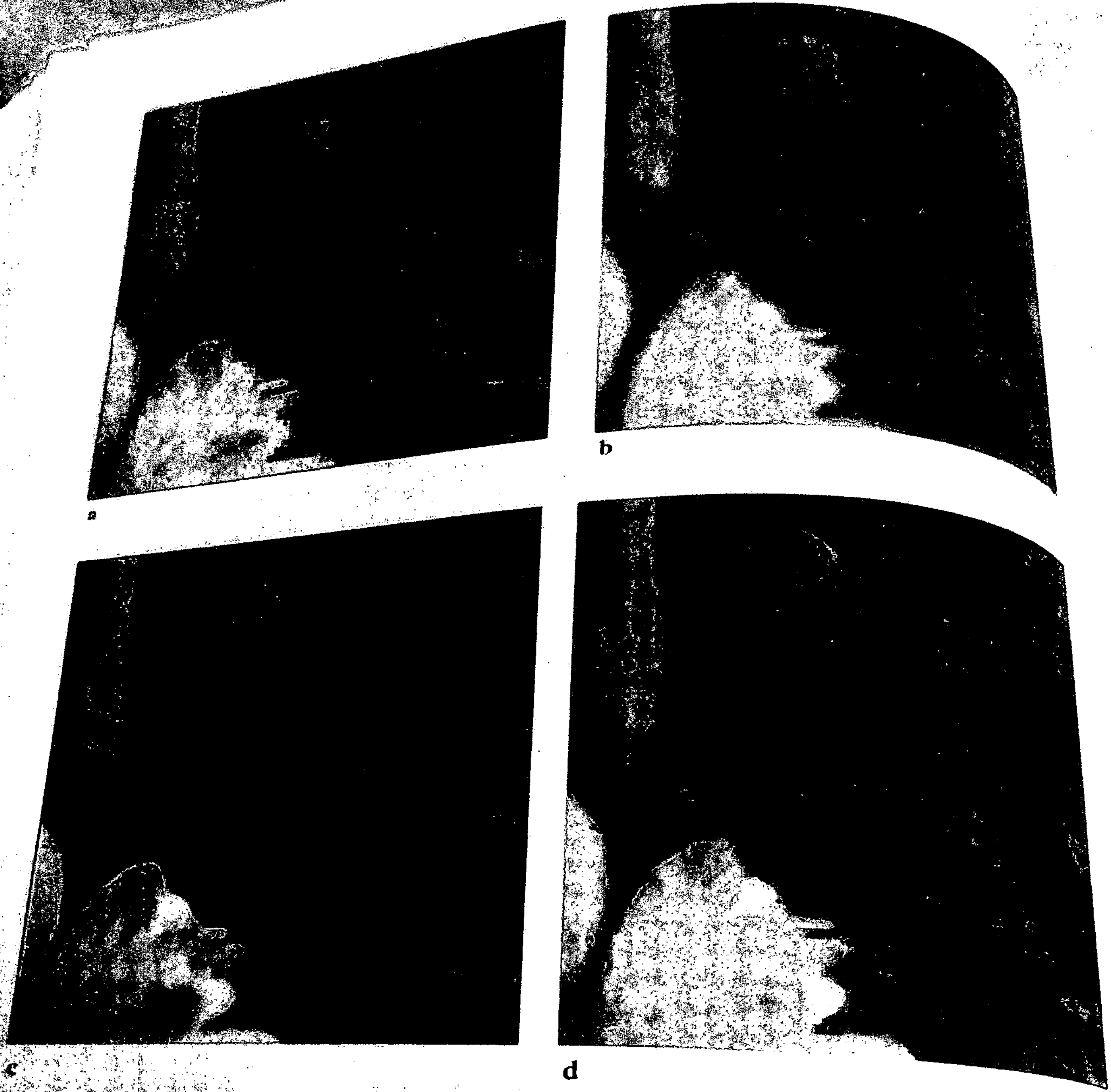
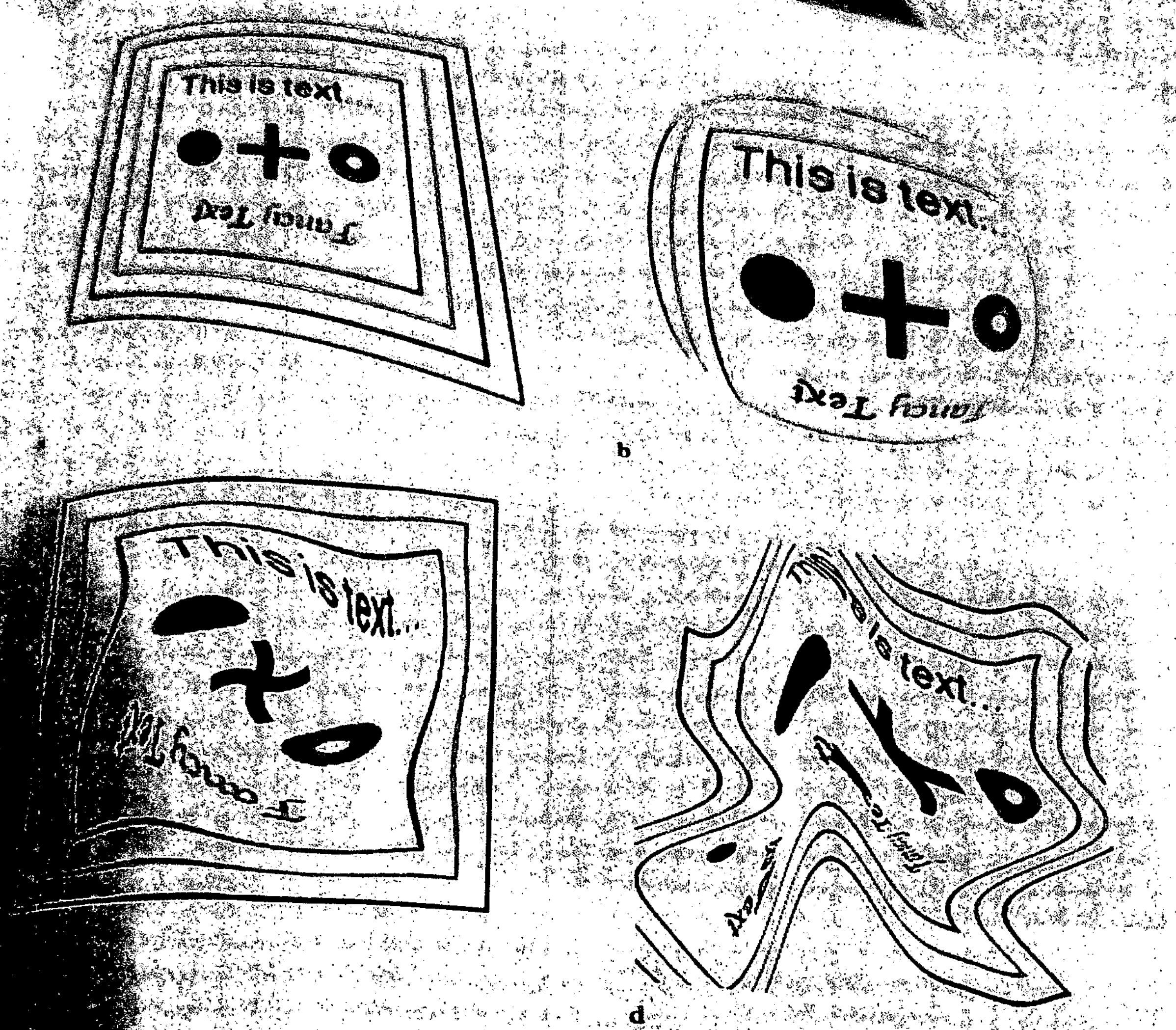


Figure 75. Enlargement of an image:
(a) no interpolation;
(b) bilinear interpolation;
(c) spline fitting;
(d) fractal interpolation.

can perform controlled warping according to mathematically defined relations
matrices of values from a set of identified fiducial or reference marks that
are generally rather specialized. But an entire class of consumer-level prog
for performing image morphing based on a net [illegible] er-defined c
generally placed at corresponding locations that a [illegible] nctive in th



Some additional examples of image warping using the same original image as Figure 71:
atic warping showing trapezoidal foreshortening;
warping in which lines are curved (approximation here is to a spherical surface);
the center of the field while holding the edges fixed (also cubic warping);
warping in which higher order and trigonometric terms are required.

ing two faces, for example, points at the tips of the eyes, corners of the mouth,
ine and chinline, and so forth are used as shown in the illustration in **Figure 77**.

the points to form a tesselation of the first image into triangles (technically, the
use as corners for the triangles is defined by a procedure called a Voronoi
uniformly stretched to fit the location of the corner points in the sec-
angles are uniformly stretched, so the points along the edges of ad-
although lines may be bent where they cross the boundaries of

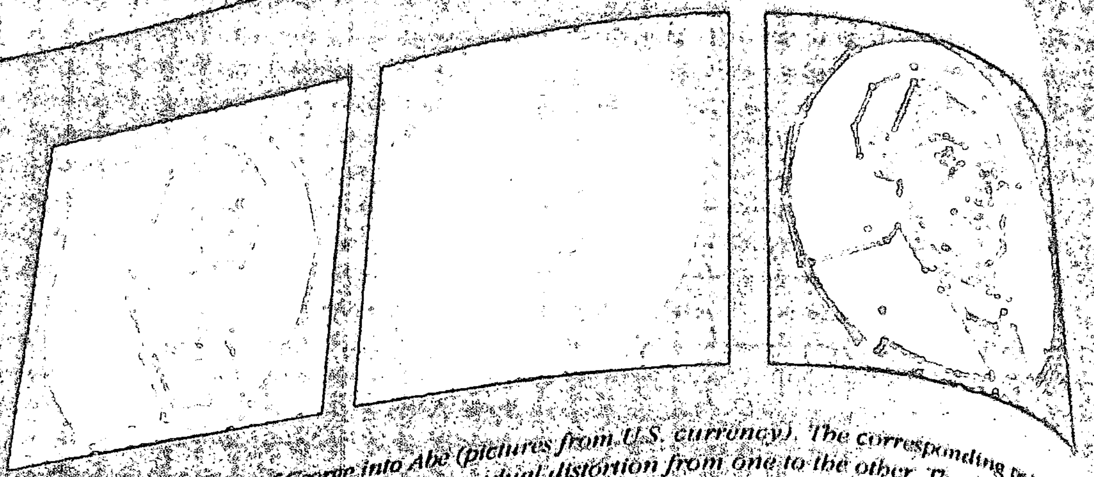


Figure 77. Transformation of George into Abe (pictures from U.S. currency). The corresponding points marked on each original image control the gradual distortion from one to the other. The midpoint fra is plausible as a person and shares feature characteristics with both endpoints.

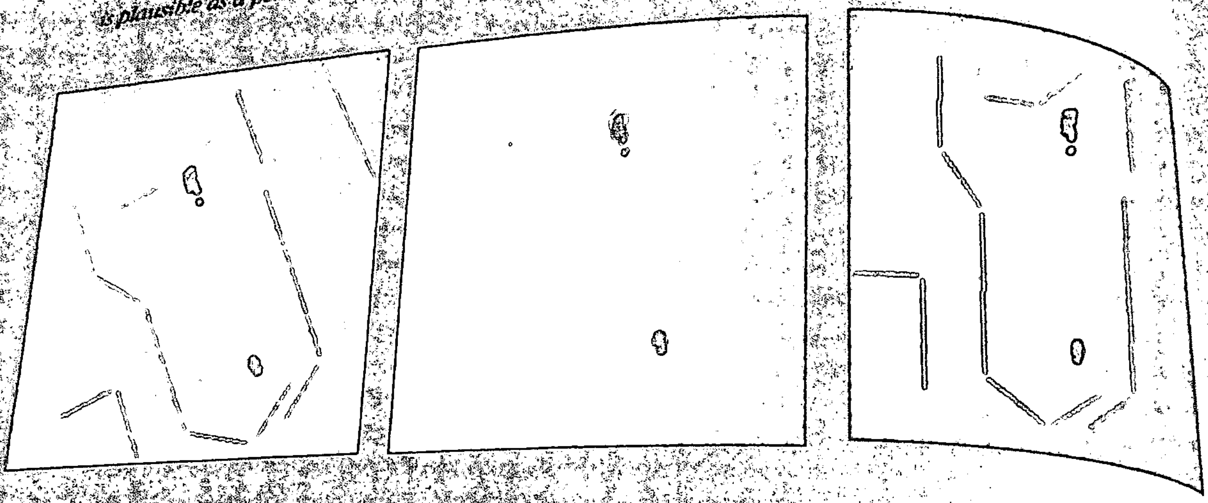


Figure 78. *Alignment of two images of a clock. The points shown on the two images control the distortion of one image to fit the second. Within the network of triangles, no abrupt distortions are apparent. Around the edges of the image and where pixels on one do not correspond to locations on the other, however, straight lines show sharp bends.*

triangles. **Figure 78** shows an example of this effect when this procedure is used to rotate one [illegible] to match another.

[illegible] triangular mesh and linear stretching of each triangle, lines crossing the boundaries of the [illegible] would be continuous, but sharply bent. Using spline or cubic equations to control the [illegible] appearance by making the curves smooth, but often at the expense of pre[illegible] measurements can be made on such images.

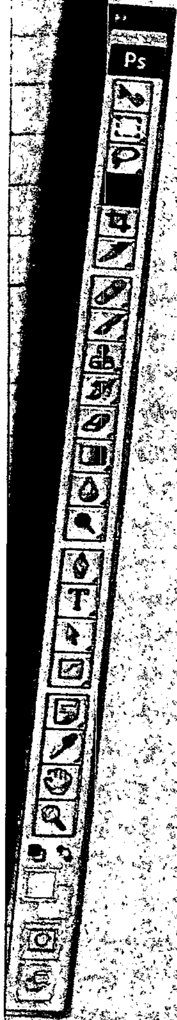
[illegible] morphing programs lies primarily in using enough control points, and the [illegible] look quite realistic, as shown in the examples. This is especially true [illegible] created with progressive motion of the control points from the to final locations. These morphing "movies" show one image transforming gradually into a second. These effects are used routinely in creating television advertisements. It is not clear there are any technical applications requiring image measurement that can be satisfactorily accomplished with these programs, considering the somewhat arbitrary placement of points [illegible] of dimensions and directions.

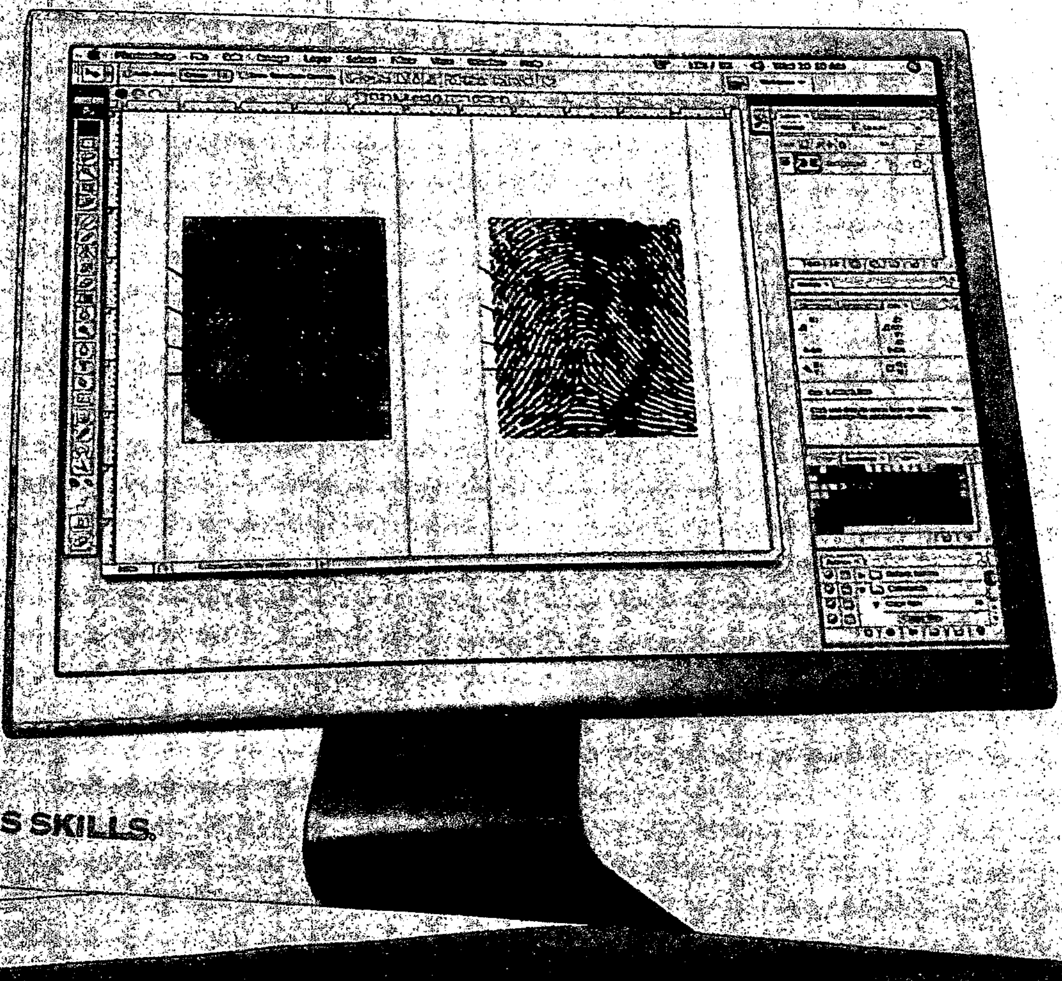
[illegible] to use morphing to align images of different objects and produce visually convincing [illegible] can be a powerful tool for communicating results to others. This procedure can be useful [illegible] the similarity between two objects, but it is extraordinarily susceptible to misuse, producing apparent matching between different images that are really not the same.

George Reis

PHOTOSHOP® CS3 FOR
Forensics Professionals

A Complete Digital Imaging Course for Investigators





SYBEX SERIOUS SKILLS.

issues for pri... with
images—and other images of a genera
will be covered separately in Chapter 1
frames that need pixel aspect correction

printing, and controlling the color, is that there
Photoshop and the separate print driver. The
manufacturer, and each manufacturer uses its own
because of this, the step-by-step methods for printing
than the step-by-step methods for printing to an
will be the same. When reading this chapter, keep the
concepts rather than the specific steps—especially regarding

printer drivers is that because they are not part of Photoshop
by Photoshop. However, most print driver settings are sticky
Photoshop or restart the computer). This means that once you
remaining prints should not require rechoosing each parameter
driver. Additionally, many print drivers allow for saving presets
these common settings simply and quickly.

preparing an image to print is to determine what size the print will
to set for the image.
you will generally print all images full frame; that is, without any
the image. Depending on the format of the original negative, slide, CCD
may make prints that are square, panoramic, or most likely some
between the two. This means that what you commonly call an 8x10
8×10 inches. It may be closer to 6.5×10 inches for prints from 35mm
many digital cameras or 8×8 inches for prints from many medium-format

that you set will be in pixels per inch (ppi) and will determine the
print and the speed at which your image is printed. The important
pixels and dots are frequently not the same thing, and therefore
frequently not the same thing. If you have a printer that can print
the best resolution for printing will not be 1440ppi for several

reasons, but the most important reason is that in ink-jet printers and laser printers, it takes many dots to print a single pixel.

In my own tests, and in tests that I have read of, most ink-jet and laser printers will print very good quality prints at 200 to 300ppi and excellent quality prints at 300 to 400ppi. A good rule of thumb is to print at a resolution that is an even factor of the printer's dpi for efficiency and in the range listed here for quality. That is, if your printer will be set to 1440dpi, a ppi of 240 for very good quality prints and 360 for excellent quality prints should work well. For a printer capable of 600dpi, settings of 200 and 300 would be substituted.

Access the Image Size dialog box by choosing Image > Image Size. In Photoshop CS2 and CS3, the keyboard shortcut is Ctrl+Alt+I/Cmd+Option+I.

The Image Size dialog box (Figure 8.1) is divided into three sections: Pixel Dimensions, Document Size, and Interpolation Options.

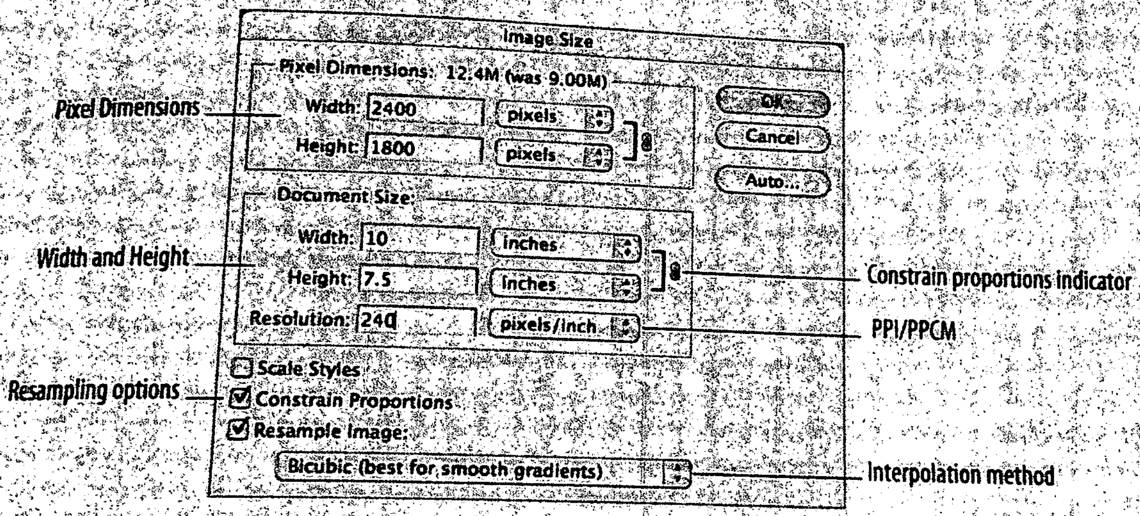


Image Size

Pixel Dimensions: 12.4M (was 9.00M)

Width: 2400 pixels

Height: 1800 pixels

Document Size:

Width: 10 inches

Height: 7.5 inches

Resolution: 240 pixels/inch

☐ Scale Styles

☑ Constrain Proportions

☑ Resample Image:

Bicubic (best for smooth gradients)

Figure 8.1 The Image Size dialog box

- The Pixel Dimensions area shows the number of pixels in your image. The pop-up menus can be changed from pixels to percent. The values for these can only be changed if the Resample Image box is checked.
- The Document Size area allows you to change the width, height, and/or pixel resolution of the image. The Height and Width settings can be made in percent, inches, centimeters, millimeters, point, picas, and columns. The Constrain Proportions indicator is locked on the width and height if the Constrain Proportions box is checked. Resolution is also constrained if Resample Image is unchecked.
- In the resampling options section, you can choose whether to resample the image (change the number of pixels in the image) and, if so, what method to use and whether to constrain proportions. Resampling an image will usually cause some degradation to the image, but it is generally not noticeable when sizing a photographic image of modest resolution for printing.

To print an image as an 8×10, I have made several choices. First, I decided to resample the image and checked the Resample Image check box. If I didn't resample the image, it may be too pixilated or I may not be able to print it at my preferred printing resolution. It is accepted practice in forensics to resample an image for printing. Not doing so could lead to images that don't match your monitor in image quality or apparent resolution. Constrain Proportions is also checked. This will generally be the case; if this box is not checked, the image will be stretched or squashed when it's resampled (the primary exception will be when printing images that are at an incorrect pixel aspect ratio, such as many video images; see Chapter 23). I set Width to 10 inches, which automatically set Height to 7.5 inches because Constrain Proportions is checked. I set the resolution to 240ppi because I will be printing to an Epson 1280 printer. I left the interpolation method set to Bicubic, which is a good general interpolation method. If I were making a substantial change in image size, I would choose Bicubic Sharper to make the image smaller or Bicubic Smoother to make the image larger.

The Print Dialog

Clicking OK prepares you for the next step of setting the print orientation and printer profiles. In Photoshop CS2, there are five menu items associated with printing: Page Setup, Print With Preview, Print, Print One Copy, and Print Online. Photoshop CS3 has brought this down to three menu items. In CS3, select Print from the File menu, and in CS2, select Print With Preview—in either case, this gives you a dialog box that includes a preview window and several options. You can access the page orientation from this window, and you can control color management issues. The key advantage of this dialog box is the ability to color manage your printing.

If a printer is giving you incorrect colors, such as a magenta or green shift, chances are that one of three things is happening: the color is not being managed in Photoshop or the print driver, the color is being managed in both Photoshop and the print driver, or the wrong paper/ink combination is being used. By first managing color in Photoshop, then making some choices in the printer driver, you can avoid color shifts and have consistently good quality prints—assuming that everything is in good working order.

The Print dialog box in CS3 (Figure 8.2) is slightly different than the Print With Preview dialog box of CS2, but most features are similar, although their placement in the window may be different. The left side displays a large preview window with icons for changing print orientation below it. Clicking the Landscape or Portrait icon will change the size and shape of the preview window.

Choose the printer you will be printing to in the top center of the window. If your printer of choice isn't listed, download the most recent drivers from the printer manufacturer's website and reinstall.