2.1

Basic Gray-Level Image Processing

Alan C. Bovik
The University of Texas at Austin

1 Introduction

This chapter, and the two that follow, describe the most commonly used and most basic tools for digital image processing. For many simple image analysis tasks, such as contrast enhancement, noise removal, object location, and frequency analysis, much of the necessary collection of instruments can be found in chapters 2.1–2.3. Moreover, these chapters supply the basic groundwork that is needed for the more extensive developments that are given in the subsequent chapters of the *Handbook*.

In the current chapter, we study basic gray-level digital image processing operations. The types of operations studied fall into three classes.

The first are *point operations*, or image processing operations that are applied to individual pixels only. Thus, interactions and dependencies between neighboring pixels are not considered, nor are operations that consider multiple pixels simultaneously to determine an output. Since spatial information, such as a pixel's location and the values of its neighbors are not considered, point operations are defined as functions of pixel intensity only. The basic tool for understanding, analyzing, and designing image point operations is the *image histogram*, which will be introduced below.

The second class includes *arithmetic operations* between images of the same spatial dimensions. These are also point operations in the sense that spatial information is not considered, although information is shared between images on a pointwise basis. Generally, these have special purposes, e.g., for noise reduction and change or motion detection.

The third class of operations are *geometric image operations*. These are complementary to point operations in the sense that they are not defined as functions of image intensity. Instead, they are functions of spatial position only. Operations of this type change the appearance of images by changing the coordinates of the intensities. This can be as simple as image translation or rotation, or may include more complex operations that distort or bend an image, or "morph" a video sequence. Since our goal, however, is to concentrate on digital image processing of real-world images, rather than the production of special effects, only the most basic geometric transformations will be considered. More complex

and time-varying geometric effects are more properly considered within the science of *computer graphics.*

2 Notation

Point operations, algebraic operations and geometric operations are easily defined on images of any dimensionality, including digital video data. For simplicity of presentation, we will restrict our discussion to two-dimensional images only. The extensions to three or higher dimensions are not difficult, especially in this case of point operations, which are independent of dimensionality. In fact, spatial/temporal information is not considered in their definition or application.

We will also only consider monochromatic images, since extensions to color or other multispectral images is either trivial, in that the same operations are applied identically to each band (e.g., R, G, B), or they are defined as more complex color space operations, which goes beyond what we want to cover in this basic chapter.

Suppose then that the single-valued image $f(\mathbf{n})$ to be considered is defined on a two-dimensional discrete-space coordinate system $\mathbf{n}=(n_1, n_2)$ or $\mathbf{n}=(m, n)$. The image is assumed to be of finite support, with image domain $[0, M-1] \times [0, N-1]$. Hence the non-zero image data can be contained in a matrix or array of dimensions $M \times N$ (rows, columns). This *discrete-space* image will have originated by sampling a continuous image $f(x, y)$ (see Chapter 7.1). Furthermore, the image $f(\mathbf{n})$ is assumed to be *quantized* to K levels $\{0, \ldots, K-1\}$, hence each pixel value takes one of these integer values (Chapter 1.1). For simplicity, we will refer to these values as *gray levels,* reflecting the way in which monochromatic images are usually displayed. Since $f(\mathbf{n})$ is both discrete-space and quantized, it is *digital.*

3 Image Histogram

The basic tool that is used in designing point operations on digital images (and many other operations as well) is the *image histogram.* The histogram H_f of the digital image f is a plot or graph of the *frequency of occurrence* of each gray level in f. Hence, H_f is a one-dimensional function with domain $\{0, \ldots, K-1\}$ and possible range extending from 0 to the number of pixels in the image, MN.

The histogram is given explicitly by

$$H_f(k) = J \tag{1}$$

if f contains *exactly* J occurrences of gray level k, for each $k=0, \ldots, K-1$. Thus, an algorithm to compute the image histogram involves a simple counting of gray levels, which can be accomplished even as the image is scanned. Every image processing development environment and software library contains basic histogram computation, manipulation, and display routines (Chapter 4.13).

Since the histogram represents a reduction of dimensionality relative to the original image f, information is lost — the image f cannot be deduced from the histogram H_f except in trivial cases (when the image is constant-valued). In fact, the number of images that share the same arbitrary histogram H_f is astronomical. Given an image f with a particular histogram H_f, every image that is a spatial shuffling of the gray-levels of f has the same histogram H_f.

The histogram H_f contains no spatial information about f — it describes the frequency of the gray-levels in f and nothing more. However, this information is still very rich, and many useful image processing operations can be derived from the image histogram. Indeed, a simple visual display of H_f reveals much about the image. By examining the appearance of a histogram, it is possible to ascertain whether the gray levels are distributed primarily at lower (darker) gray levels, or vice-versa. Although this can be ascertained to some degree by visual examination of the image itself, the human eye has a tremendous ability to adapt to overall changes in luminance, which may obscure shifts in the gray-level distribution. The histogram supplies an absolute method of determining an image's gray-level distribution.

For example, the *Average Optical Density,* or *AOD* is the basic measure of an image's overall average brightness or gray level. It can be computed directly from the image:

$$\mathrm{AOD}(f) = \frac{1}{NM} \sum_{n_1=0}^{N-1} \sum_{n_2=0}^{M-1} f(n_1, n_2) \tag{2}$$

or it can be computed from the image histogram:

$$\mathrm{AOD}(f) = \frac{1}{NM} \sum_{k=0}^{K-1} k H_f(k). \tag{3}$$

The AOD is a useful and simple meter for estimating the center of an image's gray-level distribution. A target value for the AOD might be specified when designing a point operation to change the overall gray-level distribution of an image.

Figure 1 depicts two hypothetical image histograms. The one on the left has a heavier distribution of gray levels close to zero (and a low AOD), while the one on the right is skewed towards the right (a high AOD). Since image gray levels are usually displayed with lower numbers indicating darker pixels, the image on the left corresponds to a predominantly dark image. This may occur if the image f was originally underexposed prior to digitization, or if it was taken under poor lighting levels, or perhaps the process of digitization was performed improperly. A skewed histogram often indicates a problem in gray-level allocation. The image on the right may have been overexposed or taken in very bright light.

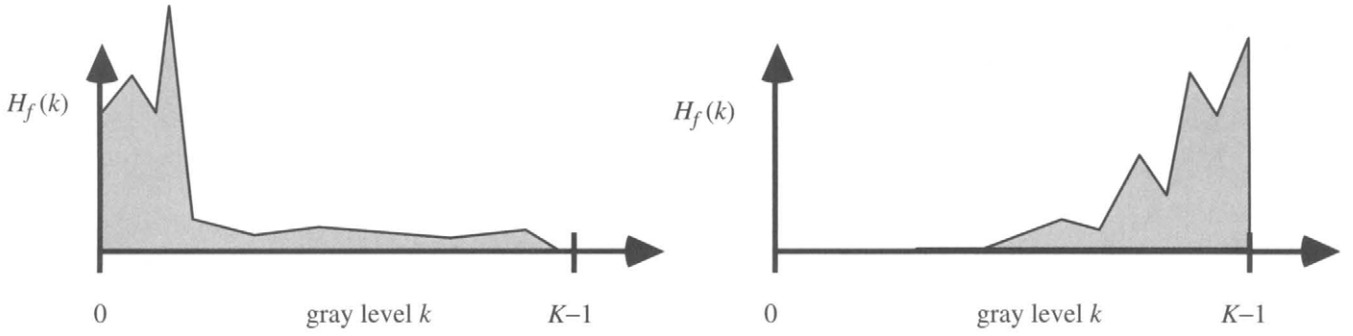


FIGURE 1 Histograms of images with gray-level distribution skewed towards darker (left) and brighter (right) gray levels. It is possible that these images are underexposed and overexposed, respectively.

Figure 2 depicts the 256×256 ($M = N = 256$) gray-level digital image "students" with gray-scale range $\{0, \ldots, 255\}$, and its computed histogram. Although the image contains a broad distribution of gray-levels, the histogram is heavily skewed towards the dark end, and the image appears to be poorly exposed. It is of interest to consider techniques that attempt to "equalize" this distribution of gray levels. One of the important applications of image point operations is to correct for poor exposures like the one in Fig. 2. Of course, there may be limitations on the effectiveness of any attempt to recover an image from poor exposure, since information may be lost. For example, in Fig. 2, the gray levels saturate at the low end of the scale, making it difficult or impossible to distinguish features at low brightness levels.

More generally, an image may have a histogram that reveals a poor usage of the available gray-scale range. An image with a compact histogram, as depicted in Fig. 3, will often have a poor visual contrast or a "washed-out" appearance. If the gray-scale range is filled out, also depicted in Fig. 3, then the image tends to have a higher contrast and a more distinctive appearance. As will be shown, there are specific point operations that effectively expand the gray-scale distribution of an image.

Figure 4 depicts the 256×256 gray-level image "books" and its histogram. The histogram clearly reveals that nearly all of the gray levels that occur in the image fall within a small range of gray-scales, and the image is of correspondingly poor contrast.

It is possible that an image may be taken under correct lighting and exposure conditions, but that there is still a skewing of the gray-level distribution towards one end of the gray-scale or that the histogram is unusually compressed. An example would be an image of the night sky, which is dark nearly everywhere. In such a case, the appearance of the image may be normal but the histogram will be very skewed. In some situations, it may still be of interest to attempt to enhance

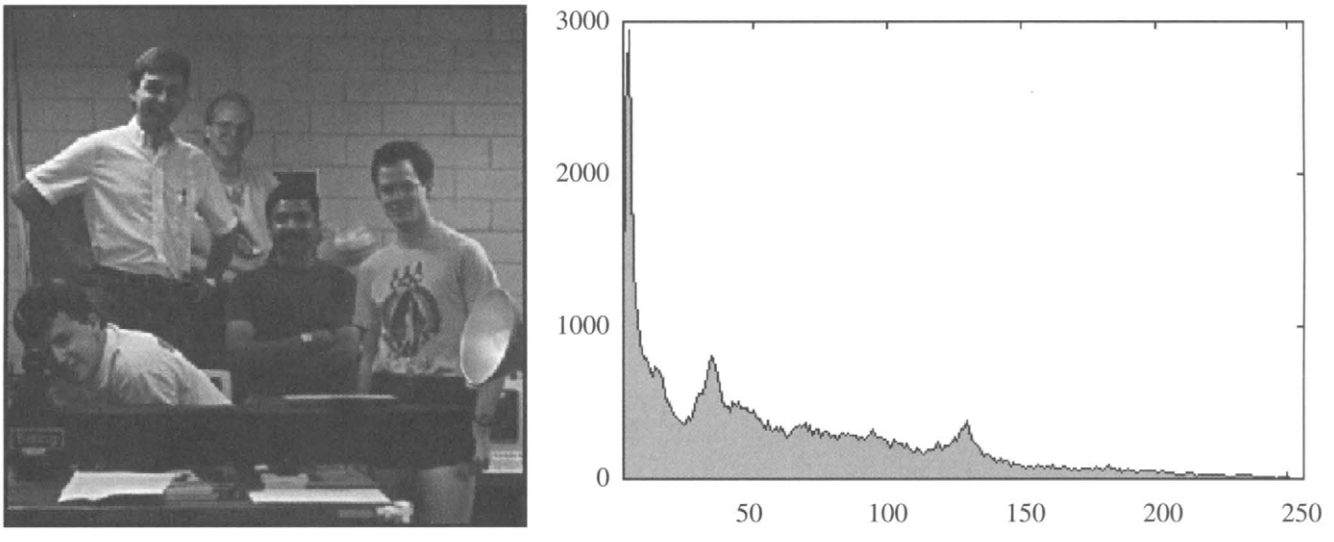


FIGURE 2 The digital image "students" (left) and its histogram (right). The gray levels of this image are skewed towards the left, and the image appears slightly underexposed.

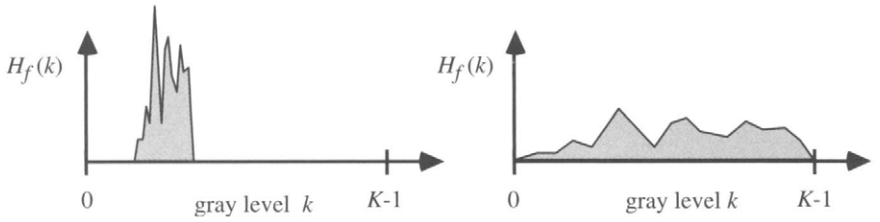


FIGURE 3 Histograms of images that make poor (left) and good (right) use of the available gray-scale range. A compressed histogram often indicates an image with a poor visual contrast. A well-distributed histogram often has a higher contrast and better visibility of detail.

or reveal otherwise difficult-to-see details in the image by application of an appropriate point operation.

4 Linear Point Operations on Images

A *point operation* on a digital image $f(\mathbf{n})$ is a function h of a single variable applied to identically to every pixel in the image, thus creating a new, modified image $g(\mathbf{n})$. Hence at each coordinate $\mathbf{n}$,

$$g(\mathbf{n}) = h[f(\mathbf{n})]. \tag{4}$$

The form of the function h is determined by the task at hand. However, since each output $g(\mathbf{n})$ is a function of a single pixel value only, the effects that can be obtained by a point operation are somewhat limited. Specifically, no spatial information is utilized in (4), and there is no change made in the spatial relationships between pixels in the transformed image. Thus, point operations do not effect the spatial positions of objects in an image, nor their shapes. Instead, each pixel value or gray level is increased or decreased (or unchanged) according to the relation in (4). Therefore, a point operation h does change the gray-level distribution or histogram of an image, and hence the overall appearance of the image.

Of course, there is an unlimited variety of possible effects that can be produced by selection of the function h that defines the point operation (4). Of these, the simplest are the *linear point operations*, where h is taken to be a simple linear function of gray level:

$$g(\mathbf{n}) = Pf(\mathbf{n}) + L. \tag{5}$$

Linear point operations can be viewed as providing a gray-level additive offset L and a gray-level multiplicative scaling P of the image f. Offset and scaling provide different effects, and so we will consider them separately before examining the overall linear point operation (5).

The *saturation* conditions $|g(\mathbf{n})| < 0$ and $|g(\mathbf{n})| > K-1$ are to be avoided if possible, since the gray levels are then not properly defined, which can lead to severe errors in processing or display of the result. The designer needs to be aware of this so steps can be taken to ensure that the image is not distorted

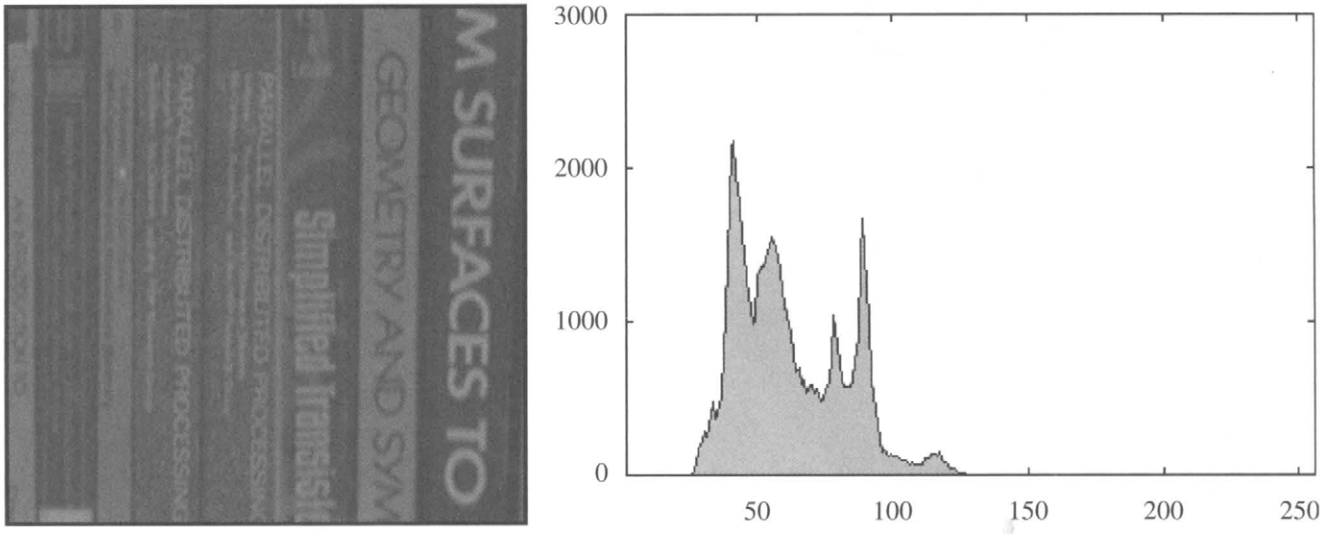


FIGURE 4 Digital image "books" (left) and its histogram (right). The image makes poor use of the available gray-scale range.

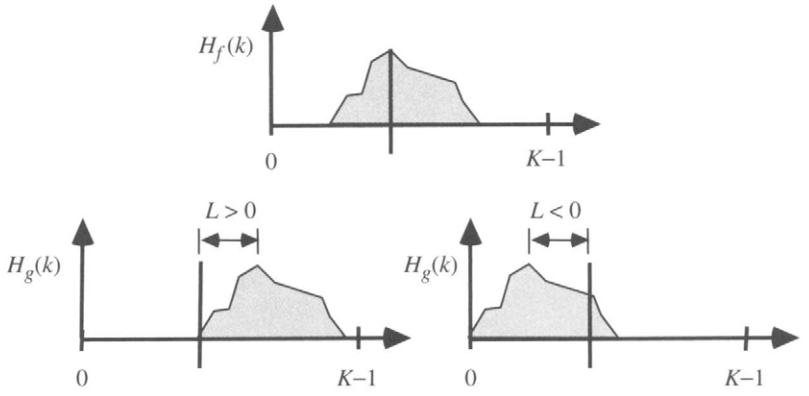


FIGURE 5 Effect of additive offset on the image histogram. Top: original image histogram; bottom: positive (left) and negative (right) offsets shift the histogram to the right and to the left, respectively.

by values falling outside the range. If a specific wordlength has been allocated to represent the gray level, then saturation may result in an overflow or underflow condition, leading to very large errors. A simple way to handle this is to simply clip those values falling outside of the allowable gray-scale range to the endpoint values. Hence, if $|g(\mathbf{n}_0)| < 0$ at some coordinate $\mathbf{n}_0$, then set $|g(\mathbf{n}_0)| = 0$ instead. Likewise, if $|g(\mathbf{n}_0)| > K-1$, then fix $|g(\mathbf{n}_0)| = K-1$. Of course, the result is no longer strictly a linear point operation. Care must be taken, since information is lost in the clipping operation, and the image may appear artificially flat in some areas if whole regions become clipped.

Additive Image Offset

Suppose $P=1$ and L is an integer satisfying $|L| \leq K-1$. An *additive image offset* has the form

$$g(\mathbf{n}) = f(\mathbf{n}) + L. \tag{6}$$

Here we have prescribed a range of values that L can take. We have taken L to be an integer, since we are assuming that images are quantized into integers in the range $\{0, \ldots, K-1\}$. We have also assumed that $|L|$ falls in this range, since otherwise, all of the values of $g(\mathbf{n})$ will fall outside the allowable gray-scale range.

In (6), if $L > 0$, then $g(\mathbf{n})$ will be a brightened version of the image $f(\mathbf{n})$. Since spatial relationships between pixels are unaffected, the appearance of the image will otherwise be essentially the same. Likewise, if $L < 0$, then $g(\mathbf{n})$ will be a dimmed version of the $f(\mathbf{n})$. The histograms of the two images have a simple relationship:

$$H_g(k) = H_f(k - L). \tag{7}$$

Thus, an offset L corresponds to a shift of the histogram by amount L to the left or to the right, as depicted in Fig. 5.

Figures 6 and 7 show the result of applying an additive offset to the images "students" and "books" in Figs. 2 and 4,

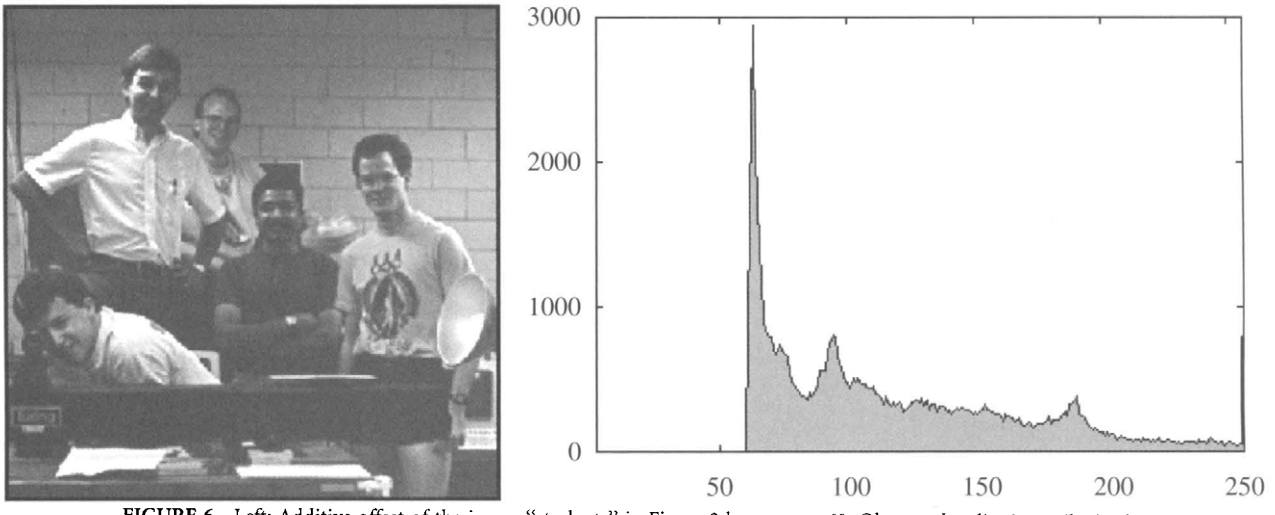


FIGURE 6 Left: Additive offset of the image "students" in Figure 2 by amount 60. Observe the clipping spike in the histogram to the right at gray level 255.

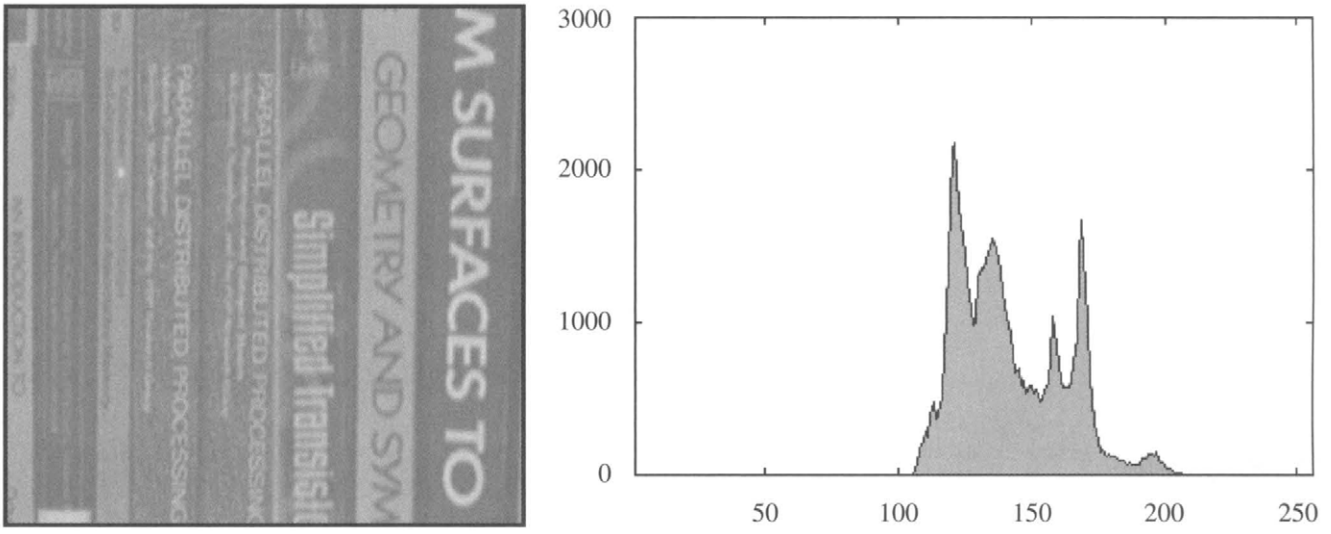


FIGURE 7 Left: Additive offset of the image "books" in Figure 4 by amount 80.

respectively. In both cases, the overall visibility of the images has been somewhat increased, but there has not been an improvement in the contrast. Hence, while each image as a whole is easier to see, the details in the image are no more visible than they were in the original. Figure 6 is a good example of saturation; a large number of gray levels were clipped at the high end (gray level 255). In this case, clipping did not result in much loss of information.

Additive image offsets can be used to calibrate images to a given average brightness level. For example, suppose we desire to compare multiple images $f_1, f_2, \ldots, f_n$ of the same scene, taken at different times. These might be surveillance images taken of a secure area that experiences changes in overall ambient illumination. These variations could occur because the area is exposed to daylight.

A simple approach to counteract these effects is to equalize the AOD's of the images. A reasonable AOD is the gray-scale center $K/2$, although other values may be used depending on the application. Letting $L_m = \text{AOD}(f_m)$, for $m = 1, \ldots, n$, the "AOD-equalized" images $g_1, g_2, \ldots, g_n$ are given by

$$g_m(\mathbf{n}) = f_m(\mathbf{n}) - L_m + K/2. \tag{8}$$

The resulting images then have identical AOD $K/2$.

Multiplicative Image Scaling

Next we consider the scaling aspect of linear point operations. Suppose that $L = 0$ and $P > 0$. Then, a *multiplicative image scaling* by factor P is given by

$$g(\mathbf{n}) = Pf(\mathbf{n}). \tag{9}$$

Here, P is assumed positive since $g(\mathbf{n})$ must be positive. Note that we have not constrained P to be an integer, since this would usually leave few useful values of P; for example, even taking $P = 2$ will severely saturate most images. If an integer result is required, then a practical definition for the output is to *round* the result in (9):

$$g(\mathbf{n}) = \text{INT}\big[Pf(\mathbf{n}) + 0.5\big] \tag{10}$$

where INT[R] denotes the nearest integer that is less than or equal to R.

The effect that multiplicative scaling has on an image depends larger on whether P is larger or smaller than one. If $P > 1$, then the gray levels of g will cover a broader range than those of f. Conversely, if $P < 1$, then g will have a narrower gray-level distribution than f. In terms of the image histogram,

$$H_g\big\{\text{INT}[Pk + 0.5]\big\} = H_f(k). \tag{11}$$

Hence, multiplicative scaling by a factor P either stretches or compresses the image histogram. Note that for quantized images, it is not proper to assume that (11) implies $H_g(k) = H_f(k/P)$, since the argument of $H_f(k/P)$ may not be integer.

Figure 8 depicts the effect of multiplicative scaling on a hypothetical histogram. For $P > 1$, the histogram is expanded (and hence, saturation is quite possible), while for $P < 1$, the histogram is contracted. If the histogram is contracted, then multiple gray levels in f may map to single gray levels in g, since the number of gray levels is finite. This implies a possible loss of information. If the histogram is expanded, then spaces may appear between the histogram bins where gray levels are not being mapped. This, however, does not represent a loss of information and usually will not lead to visual information loss.

As a rule of thumb, histogram expansion often leads to a more distinctive image that makes better use of the gray-scale

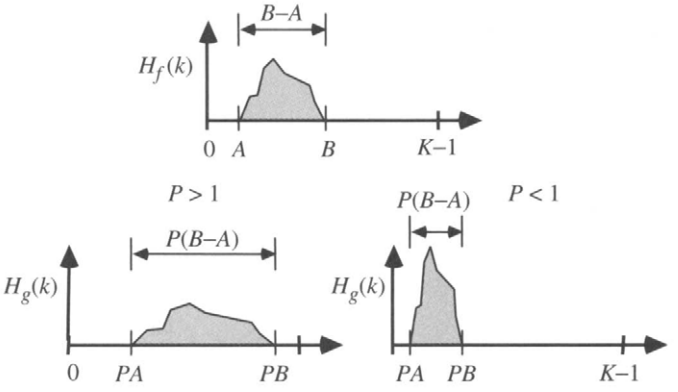


FIGURE 8 Effects of multiplicative image scaling on the histogram. If $P > 1$, the histogram is expanded, leading to more complete use of the gray-scale range. If $P < 1$, the histogram is contracted, leading to possible information loss and (usually) a less striking image.

range, provided that saturation effects are not visually noticeable. Histogram contraction usually leads to the opposite: an image with reduced visibility of detail that is less striking. However, these are only rules of thumb, and there are exceptions. An image may have a gray-scale spread that is too extensive, and may benefit from scaling with $P < 1$.

Figure 9 shows the image "students" following a multiplicative scaling with $P = 0.75$, resulting in compression of the histogram. The resulting image is darker and less contrasted. Figure 10 shows the image "books" following scaling with $P = 2$. In this case, the resulting image is much brighter and has a better visual resolution of gray levels. Note that most of the high end of the gray-scale range is now used, although the low end is not.

Image Negative

The first example of a linear point operation that uses both scaling and offset is the *image negative*, which is given by $P = -1$ and $L = K-1$. Hence

$$g(\mathbf{n}) = -f(\mathbf{n}) + (K - 1) \tag{12}$$

and

$$H_g(k) = H_f(K - 1 - k). \tag{13}$$

Scaling by $P = -1$ reverses (flips) the histogram; the additive offset $L = K-1$ is required so that all values of the result are positive and fall in the allowable gray-scale range. This operation creates a *digital negative image*, unless the image is already a negative, in which case a positive is created. It should be mentioned that unless the digital negative (12) is being computed, $P > 0$ in nearly every application of linear point operations.

An important application of (12) occurs when a negative is scanned (digitized), and it is desired to view the positive image. Figure 11 depicts the negative image associated with "students." Sometimes, the negative image is viewed intentionally, when the positive image itself is very dark. A common example of this is for the examination of telescopic images of star fields and faint galaxies. In the negative image, faint bright objects appear as dark objects against a bright background, which can be easier to see.

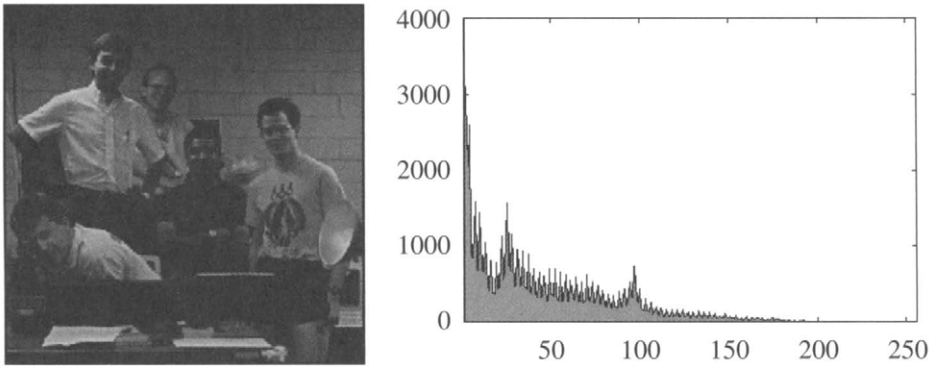


FIGURE 9 Histogram compression by multiplicative image scaling with $P = 0.75$. The resulting image is less distinctive. Note also the regularly-spaced tall spikes in the histogram; these are gray levels that are being "stacked," resulting in a loss of information, since they can no longer be distinguished.

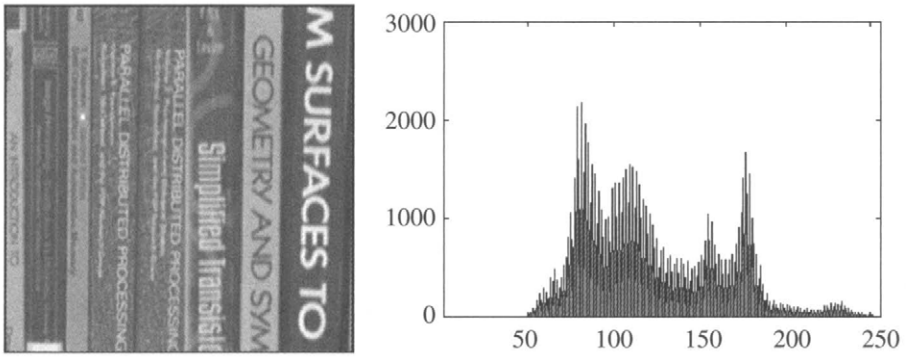


FIGURE 10 Histogram expansion by multiplicative image scaling with $P = 2.0$. The resulting image is much more visually appealing. Note the regularly-spaced gaps in the histogram that appear when the discrete histogram values are spread out. This does not imply a loss of information or visual fidelity.

Full-Scale Histogram Stretch

We have already mentioned that an image that has a broadly distributed histogram tends to be more visually distinctive. The *full-scale histogram stretch*, which is also often called a *contrast stretch*, is a simple linear point operation that expands the image histogram to fill the entire available gray-scale range. This is such a desirable operation that the full-scale histogram stretch is easily the most common linear point operation. Every image processing programming environment and library contains it as a basic tool. Many image display routines incorporate it as a basic feature. Indeed, commercially-available digital video cameras for home and professional use generally apply a full-scale histogram stretch to the acquired image before being stored in camera memory. It is called automatic gain control (AGC) on these devices.

The definition of the multiplicative scaling and additive offset factors in the full-scale histogram stretch depend on the image f. Suppose that f has a compressed histogram with maximum gray level value B and minimum value A, as shown in Fig. 8 (top):

$$A = \min_{\mathbf{n}}\{f(\mathbf{n})\} \quad \text{and} \quad B = \max_{\mathbf{n}}\{f(\mathbf{n})\} \tag{14}$$

The goal is to find a linear point operation of the form (5) that maps gray levels A and B in the original image to gray levels 0 and $K-1$ in the transformed image. This can be expressed in two linear equations:

$$PA + L = 0 \tag{15}$$

and

$$PB + L = K - 1 \tag{16}$$

in the two unknowns (P, L), with solutions

$$P = \left(\frac{K-1}{B-A}\right) \tag{17}$$

and

$$L = -A\left(\frac{K-1}{B-A}\right). \tag{18}$$

Hence, the overall full-scale histogram stretch is given by

$$g(\mathbf{n}) = \mathrm{FSHS}(f) = \left(\frac{K-1}{B-A}\right)[f(\mathbf{n}) - A]. \tag{19}$$

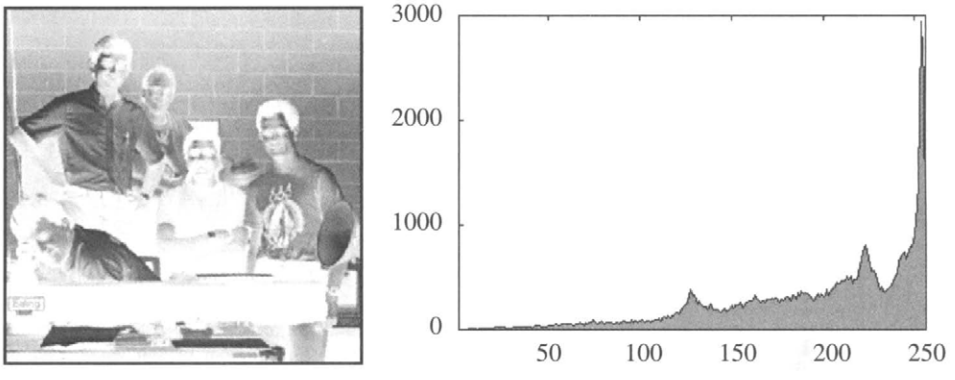


FIGURE 11 Example of image negative with resulting reversed histogram.

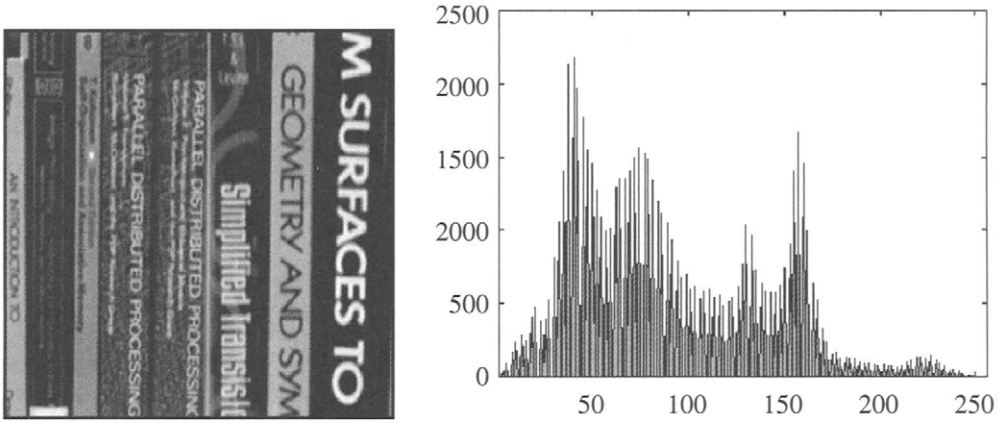


FIGURE 12 Full-scale histogram stretch of image "books."

We make the shorthand notation FSHS, since (19) will prove to be commonly useful as an addendum to other algorithms. The operation in (19) can produce dramatic improvements in the visual quality of an image suffering from a poor (narrow) gray-scale distribution. Figure 12 shows the result of applying the full-scale histogram stretch to the images "books." The contrast and visibility of the image was, as expected, greatly improved. The accompanying histogram, which now fills the available range, also shows the characteristics gaps of an expanded discrete histogram.

If the image f already has a broad gray-level range, then the histogram stretch may produce little or no effect. For example, the image "students" (Fig. 2) has gray scales covering the entire available range, as seen in the histogram accompanying the image. Therefore, (19) has no effect on "students." This is unfortunate, since we have already commented that "students" might benefit from a histogram manipulation that would re-distribute the gray level densities. Such a transformation would need to nonlinearly re-allocate the images gray level values. Such nonlinear point operations are described next.

5 Nonlinear Point Operations on Images

We now consider *nonlinear point* operations of the form

$$g(\mathbf{n}) = h[f(\mathbf{n})] \tag{20}$$

where the function h is nonlinear. Obviously, this encompasses a wide range of possibilities. However, there are only a few functions h that are used with any great degree of regularity. Some of these are functional tools that are used as part of larger, multi-step algorithms, such as absolute value, square, and square-root functions. One such simple nonlinear function that is very commonly used is the logarithmic point operation, which we describe in detail.

Logarithmic Point Operations

Assuming that the image $f(\mathbf{n})$ is positive-valued, the *logarithmic point operation* is defined by a composition of two operations: a point logarithmic operation, followed by a full-scale histogram stretch:

$$g(\mathbf{n}) = \mathrm{FSHS}\{\log[1 + f(\mathbf{n})]\}. \tag{21}$$

Adding unity to the image avoids the possibility of taking the logarithm of zero. The logarithm itself acts to nonlinearly compress the gray level range. All of the gray level are compressed to the range $[0, \log(K)]$. However, larger (brighter) gray levels are compressed much more severely than are smaller gray levels. The subsequent FSHS operation then acts to linearly expand the log-compressed gray levels to fill the gray-scale range. In the transformed image, dim objects in the original are now allocated a much larger percentage of the gray-scale range, hence improving their visibility.

The logarithmic point operation is an excellent choice for improving the appearance of the image "students," as shown in Fig. 13. The original image (Fig. 2) was not a candidate for FSHS because of its broad histogram. The appearance of the original suffers because many of the important features of the image are obscured by darkness. The histogram is significantly spread at these low brightness levels, as can be seen by comparing to Fig. 2, and also by the gaps that appear in the low end of the histogram. This does not occur at brighter gray levels.

Certain applications quite commonly use logarithmic point operations. For example, in astronomical imaging, a relatively few bright pixels (stars and bright galaxies, etc.) tend to dominate the visual perception of the image, while much of the interesting information lies at low bright levels (e.g., large,

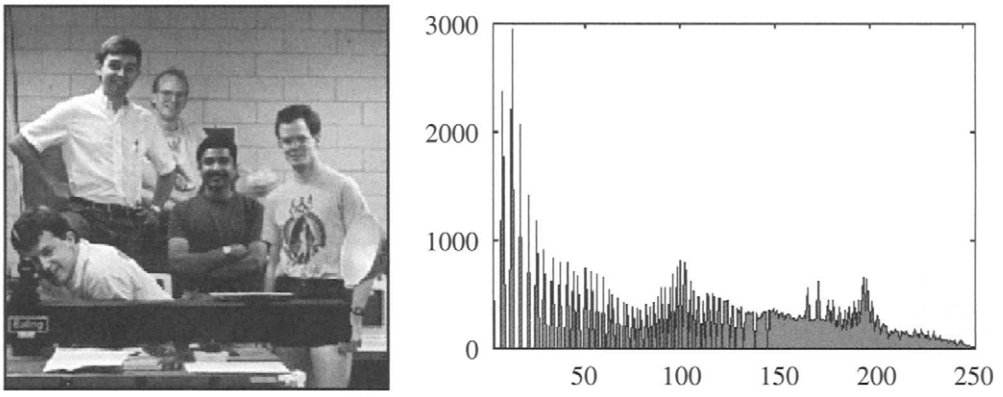


FIGURE 13 Logarithmic gray-scale range compression followed by FSHS applied to image "students."

faint nebulae). By compressing the bright intensities much more heavily, then applying FSHS, the faint, interesting details visually emerge.

Later, in Chapter 2.3, the Fourier Transforms of images will be studied. The Fourier transform magnitudes, which are of the same dimensionalities as images, will be displayed as intensity arrays for visual consumption. However, the Fourier transforms of most images are dominated visually by the Fourier coefficients of a relatively few low frequencies, so the coefficients of important high frequencies are usually difficult or impossible to see. However, a point logarithmic operation usually suffices to ameliorate this problem, and so image Fourier transforms are usually displayed following application of (21), both in this *Handbook* and elsewhere.

Histogram Equalization

One of the most important nonlinear point operations is *histogram equalization,* also called *histogram flattening.* The idea behind it extends that of FSHS: not only should an image fill the available gray-scale range, but it should be uniformly distributed over that range. Hence, an idealized goal is a flat histogram. Although care must be taken in applying a powerful nonlinear transformation that actually changes the shape of the image histogram, rather than just stretching it, there are good mathematic reasons for regarding a flat histogram as a desirable goal. In a certain sense,[1] an image with a perfectly flat histogram contains the largest possible amount of *information* or complexity.

In order to explain histogram equalization, it will be necessary to make some refined definitions of the image histogram. For an image containing MN pixels, the *normalized image histogram* is given by

$$p_f(k) = \frac{1}{MN} H_f(k) \tag{22}$$

[1]In the sense of maximum entropy, see Chapter 5.1.

for $k = 0, \ldots, K-1$. This function has the property that

$$\sum_{k=0}^{K-1} p_f(k) = 1. \tag{23}$$

The normalized histogram $p_f(k)$ has a valid interpretation as the empirical probability density (mass function) of the gray level values of image f. In other words, if a pixel coordinate $\mathbf{n}$ is chosen at random, then $p_f(k)$ is the probability that $f(\mathbf{n}) = k$: $p_f(k) = \Pr\{f(\mathbf{n}) = k\}$.

We also define the *cumulative normalized image histogram* to be

$$P_f(r) = \sum_{k=0}^{r} p_f(k)\ ;\ r = 0,..., K-1. \tag{24}$$

The function $P_f(r)$ is an empirical probability distribution function, hence it is a nondecreasing function, and also $P_f(K-1) = 1$. It has the probabilistic interpretation that for a randomly selected image coordinate $\mathbf{n}$, $P_f(r) = \Pr\{f(\mathbf{n}) \leq r\}$. From (24) it is also true that:

$$p_f(k) = P_f(k) - P_f(k-1)\ ;\ k = 0,..., K-1 \tag{25}$$

so $P_f(k)$ and $p_f(k)$ can be obtained from each other. Both are complete descriptions of the gray level distribution of the image f.

In order to understand the process of *digital histogram equalization,* we first explain the process supposing that the normalized and cumulative histograms are functions of continuous variables. We will then formulate the digital case of an approximation of the continuous process. Hence, suppose that $p_f(x)$ and $P_f(x)$ are functions of a continuous variable x. They may be regarded as image probability density function (pdf) and cumulative distribution function (cdf), with relationship $p_f(x) = dP_f(x)/dx$. We will also assume that P_f^{-1} exists. Since P_f is non-decreasing, this is either true or P_f^{-1} can be defined by a convention. In this hypothetical

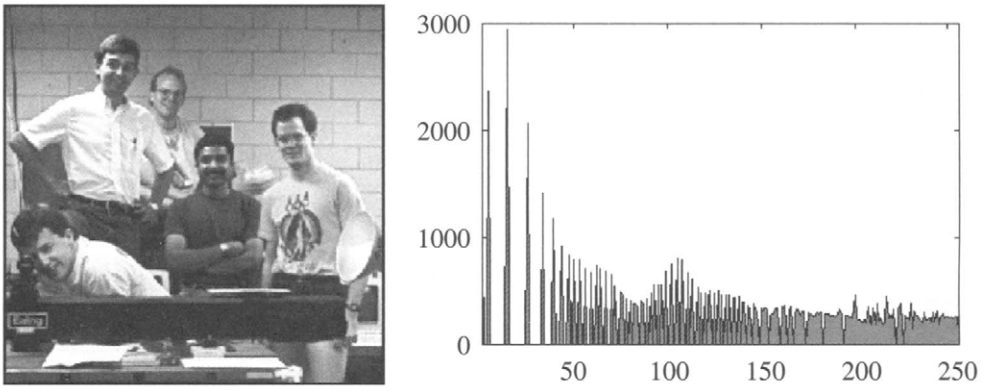


FIGURE 14 Histogram equalization applied to the image "students."

continuous case, we claim that the image

$$\mathrm{FSHS}(g) \tag{26}$$

where

$$g = P_f(f) \tag{27}$$

has a uniform (flat) histogram. In (26), $P_f(f)$ denotes that P_f is applied on a pixelwise basis to f:

$$g(\mathbf{n}) = P_f[f(\mathbf{n})] \tag{28}$$

for all $\mathbf{n}$. Since P_f is a continuous function, (26)–(28) represents a smooth mapping of the histogram of image f to an image with a smooth histogram. At first, (27) may seem confusing since the function P_f that is computed from f is then applied to f. To see that a flat histogram is obtained, we use the probabilistic interpretation of the histogram. The cumulative histogram of the resulting image g is:

$$\begin{aligned} P_g(x) &= \Pr\{g \le x\} = \Pr\{P_f(f) \le x\} \\ &= \Pr\left\{f \le P_f^{-1}(x)\right\} = P_f\left\{P_f^{-1}(x)\right\} = x \end{aligned} \tag{29}$$

for $0 \le x \le 1$. Finally, the normalized histogram of g is

$$p_g(x) = dP_g(x)/dx = 1 \tag{30}$$

for $0 \le x \le 1$. Since $p_g(x)$ is defined only for $0 \le x \le 1$, FSHS in (26) is required to stretch the flattened histogram to fill the gray-scale range.

To flatten the histogram of a digital image f, first compute the discrete cumulative normalized histogram $P_f(k)$, apply (28) at each $\mathbf{n}$, then (26) to the result. However, while an image with a perfectly flat histogram is the result in the ideal continuous case outlined above, in the digital case the output histogram is only approximately flat, or more accurately, more flat than the input histogram. This follows since (26)–(28) collectively is a point operation on the image f, so every occurrence of gray level k maps to $P_f(k)$ in g. Hence, histogram bins are never reduced in amplitude by (26)–(28), although they may increase if multiple gray-levels map to the same value (thus destroying information). Hence, the histogram cannot be truly equalized by this procedure.

Figures 14 and 15 show histogram equalization applied to our ongoing example images "students" and "books," respectively. Both images are much more striking and viewable than the original. As can be seen, the resulting histograms

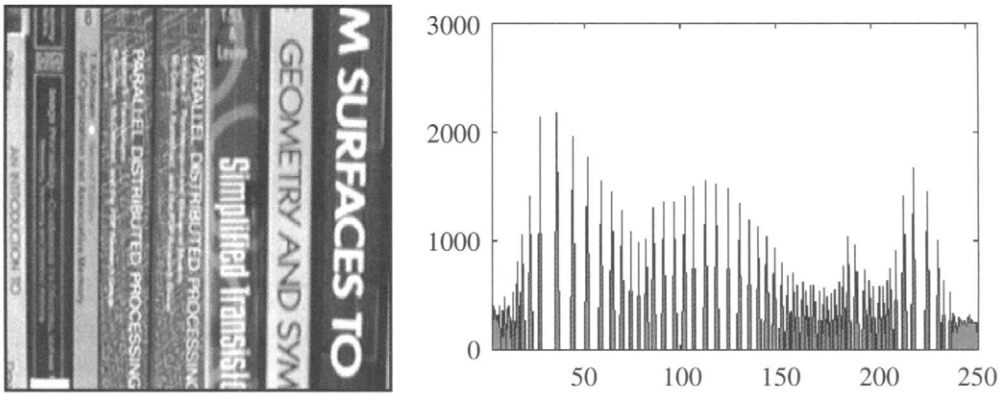


FIGURE 15 Histogram equalization applied to the image "books."

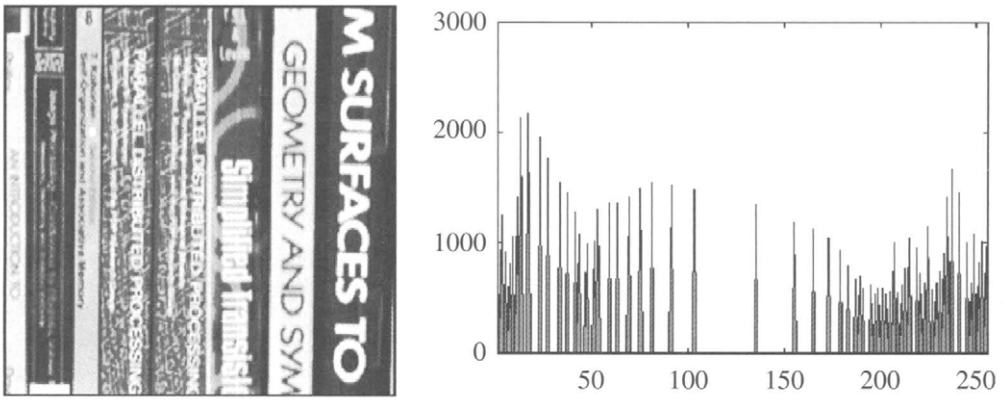


FIGURE 16 Histogram of the image "books" shaped to match a "V."

are not really flat; it is "flatter" in the sense that the histograms are spread as much as possible. However, the heights of peaks are *not* reduced. As is often the case with expansive point operations, gaps or spaces appear in the output histogram. These are not a problem unless the gaps become large and some of the histogram bins become isolated. This amounts to an excess of quantization in that range of gray levels, which may result in false contouring (Chapter 1.1).

Histogram Shaping

In some applications, it is desired to transform the image into one that has a histogram of a specific shape. The process of histogram shaping generalizes histogram equalization, which is the special case where the target shape is flat. Histogram shaping can be applied when multiple images of the same scene, but taken under mildly different lighting conditions, are to be compared. This extends the idea of AOD-equalization described earlier in this chapter. By shaping the histograms to match, the comparison may exclude minor lighting effects. Alternately, it may be that the histogram of one image is shaped to match that of another, again usually for the purpose of comparison. Or it might simply be that a certain histogram shape, such as a Gaussian, produces visually agreeable results for a certain class of images.

Histogram shaping is also accomplished by a nonlinear point operation defined in terms of the empirical image probabilities or histogram functions. Again, exact results are obtained in the hypothetical continuous-scale case. Suppose that the target (continuous) cumulative histogram function is $Q(x)$, and that Q^{-1} exists. Then let

$$g = Q^{-1}\big[P_f(f)\big] \tag{31}$$

where both functions in the composition are applied on a pixelwise basis. The cumulative histogram of g is then:

$$\begin{aligned} P_g(x) &= \Pr\{g \le x\} = \Pr\{Q^{-1}[P_f(f)] \le x\} \\ &= \Pr\{P_f(f) \le Q(x)\} = \Pr\{f \le P_f^{-1}[Q(x)]\} \\ &= P_f\{P_f^{-1}[Q(x)]\} = Q(x), \end{aligned} \tag{32}$$

as desired. Note that FSHS is not required in this instance. Of course, (32) can only be approximated when the image f is digital. In such cases, the specified target cumulative histogram function $Q(k)$ is discrete, and some convention for defining Q^{-1} should be adopted, particularly if Q is computed from a target image and is unknown in advance. One common convention is to define

$$Q^{-1}(k) = \min_S\{S : Q(S) \ge k\}. \tag{33}$$

As an example, Fig. 16 depicts the result of shaping the histogram of "books" to match the shape of an inverted "V" centered at the middle gray level and extending across the entire gray scale. Again, a perfect "V" is not produced, although an image of very high contrast is still produced. Instead, the histogram shape that results is a crude approximation to the target.

6 Arithmetic Operations Between Images

We now consider *arithmetic operations* defined on multiple images. The basic operations are pointwise image addition/subtraction and pointwise image multiplication/division. Since digital images are defined as arrays of numbers, these operations need to be defined carefully.

Suppose we have n $N \times M$ images $f_1, f_2, \ldots, f_n$. It is important that they be of the same dimensions since we will be defining operations between corresponding array elements (having the same indices).

The *sum* of n images is given by

$$f_1 + f_2 + \ldots + f_n = \sum_{m=1}^{n} f_m \tag{34}$$

while for any two images f_r, f_S the *image difference* is

$$f_r - f_S. \tag{35}$$

The *pointwise product* of the n $N \times M$ images $f_1, \ldots, f_n$ is denoted by

$$f_1 \otimes f_2 \otimes \ldots \otimes f_n = \prod_{m=1}^{n} f_m \tag{36}$$

where in (36) we do *not* infer that the matrix product is being taken. Instead, the product is defined on a pointwise basis. Hence $g = f_1 \otimes f_2 \otimes \ldots \otimes f_n$ if and only if

$$g(\mathbf{n}) = f_1(\mathbf{n}) f_2(\mathbf{n}) \ldots f_n(\mathbf{n}) \tag{37}$$

for every $\mathbf{n}$. In order to clarify the distinction between matrix product and pointwise array product, we introduce the special notation "$\otimes$" to denote the pointwise product. Given two images f_r, f_S the *pointwise image quotient* is denoted

$$g = f_r \Delta f_S \tag{38}$$

if for every $\mathbf{n}$ it is true that $f_S(\mathbf{n}) \neq 0$ and

$$g(\mathbf{n}) = f_r(\mathbf{n}) / f_S(\mathbf{n}). \tag{39}$$

The pointwise matrix product and quotient are mainly useful when manipulating Fourier transforms of images, as will be seen in Chapter 2.3. However, the pointwise image sum and difference, despite their simplicity, have important applications that we will examine next.

Image-Averaging for Noise Reduction

Images that occur in practical applications invariably suffer from random degradations that are collectively referred to as *noise.* These degradations arise from numerous sources, including radiation scatter from the surface before the image is sensed; electrical noise in the sensor or camera; channel noise as the image is transmitted over a communication channel; bit errors after the image is digitized, and so on. A good review of various image noise models is given in Chapter 4.4 of this *Handbook.*

The most common generic noise model is *additive noise,* where a noisy observed image is taken to be the sum of an original, uncorrupted image g and a noise image q:

$$f = g + q, \tag{40}$$

where q is an two-dimensional $N \times M$ random matrix, with elements $q(\mathbf{n})$ that are random variables. Chapter 4.4 develops the requisite mathematics for understanding random quantities and provides the basis for noise filtering. In this basic chapter we will not require this more advanced development. Instead, we make the simple assumption that the noise is *zero mean.* If the noise is zero mean, then the average (or sample mean) of n independently occurring noise matrices $q_1, q_2, \ldots, q_n$ tends towards zero as n grows large:[2]

$$\left(\frac{1}{n}\right) \sum_{m=1}^{n} q_m \approx \mathbf{0} \tag{41}$$

where $\mathbf{0}$ denotes the $N \times M$ matrix of zeros.

Now suppose that we are able to obtain n images $f_1, f_2, \ldots, f_n$ of the same scene. The images are assumed to be noisy versions of an original image g, where the noise is zero-mean and additive:

$$f_m = g + q_m \tag{42}$$

for $m = 1, \ldots, n$. Hence, the images are assumed either to be taken in rapid succession, so that there is no motion between frames, or under conditions where there is no motion in the scene. In this way only the noise contribution varies from image to image.

By averaging the multiple noisy images (42):

$$\begin{aligned} \left(\frac{1}{n}\right) \sum_{m=1}^{n} f_m = \left(\frac{1}{n}\right) \sum_{m=1}^{n} (g + q_m) &= \left(\frac{1}{n}\right) \sum_{m=1}^{n} g + \left(\frac{1}{n}\right) \sum_{m=1}^{n} q_m \\ &= g + \left(\frac{1}{n}\right) \sum_{m=1}^{n} q_m \\ &\approx g \end{aligned} \tag{43}$$

using (41). If a large enough number of frames are averaged together, then the resulting image should be nearly noise-free, and hence should approximate the original image. The amount of noise reduction can be quite significant; one can expect a reduction in the noise variance by a factor n. Of course, this is subject to inaccuracies in the model, e.g., if there is any change in the scene itself, or if there are any

[2]More accurately, the noise must be assumed *mean-ergodic,* which means that the sample mean approaches the statistical mean over large sample sizes. This assumption is usually quite reasonable. The statistical mean is defined in Chapter 4.5.

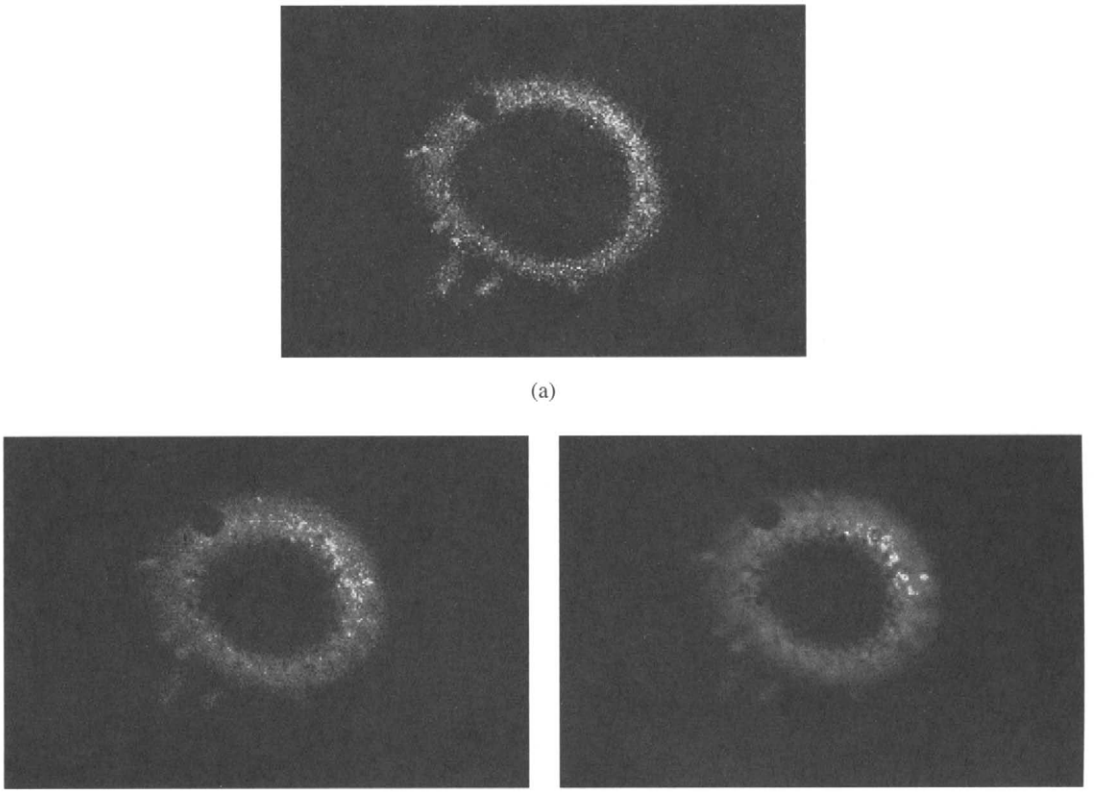


FIGURE 17 Example of image averaging for noise reduction. (a) Single noisy image; (b) average of four frames; (c) average of 16 frames (courtesy of Chris Neils).

dependencies between the noise images (e.g., in an extreme case, the noise images might be identical), then the reduction in the noise will be limited.

Figure 17 depicts the process of noise reduction by frame averaging in an actual example of confocal microscope imaging (Chapter 10.7). The image(s) are of Macroalga Valonia microphysa, imaged with a laser scanning confocal microscope (LSCM). The dark ring is chlorophyll fluorescing under Ar laser excitation. As can be seen, in this case the process of image averaging is quite effective in reducing the apparent noise content, and in improving the visual resolution of the object being imaged.

Image Differencing for Change Detection

Often it is of interest to detect changes that occur in images taken of the same scene but at different times. If the time instants are closely placed, e.g., adjacent frames in a video sequence, then the goal of change detection amounts to image motion detection (Chapter 3.8). There are many applications of motion detection and analysis. For example, in video compression algorithms, compression performance is improved by exploiting redundancies that are tracked along the motion trajectories of image objects that are in motion. Detected motion is also useful for tracking targets, for recognizing objects by their motion, and for computing three-dimensional scene information from two-dimensional motion.

If the time separation between frames is not small, then change detection can involve the discovery of gross scene changes. This can be useful for security or surveillance cameras, or in automated visual inspection systems, for example. In either case, the basic technique for change detection is the *image difference*. Suppose that f_1 and f_2 are images to be compared. Then the absolute difference image

$$g = |f_1 - f_2| \tag{44}$$

will embody those changes or differences that have occurred between the images. At coordinates $\mathbf{n}$ where there has been little change, $g(\mathbf{n})$ will be small. Where change has occurred, $g(\mathbf{n})$ can be quite large. Figure 18 depicts image differencing. In the difference image, large changes are displayed as brighter intensity values. Since significant change has occurred, there are many bright intensity values. This difference image could be processed by an automatic change detection algorithm. A simple series of steps that might be taken would be to binarize the difference image, thus separating change from non-change, using a threshold (Chapter 2.2), counting the number

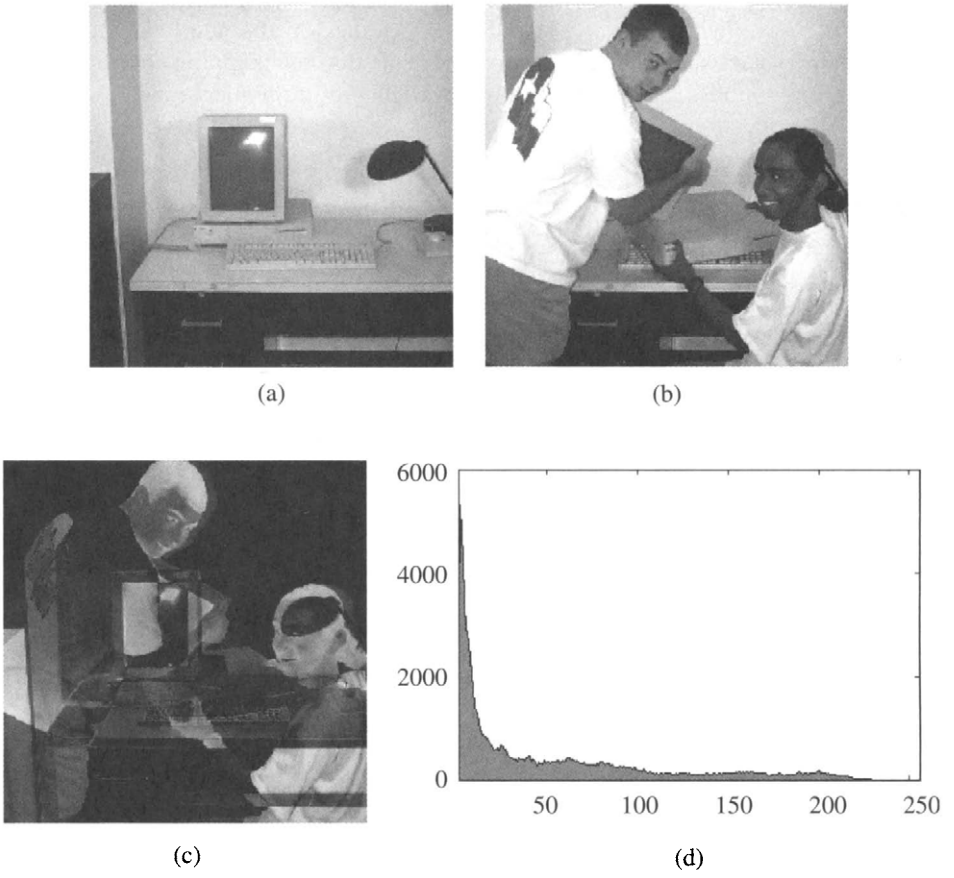


FIGURE 18 Image differencing example. (a) Original placid scene; (b) a theft is occurring! (c) the difference image with brighter points signifying larger changes; (d) the histogram.

of high-change pixels, and finally, deciding whether the change is significant enough to take some action. Sophisticated variations of this theme are currently in practical use. The histogram in Fig. 18(*d*) is instructive, since it is characteristic of differenced images; many zero or small gray-level changes occur, with the incidence of larger changes falling off rapidly.

7 Geometric Image Operations

We conclude this chapter with a brief discussion of *geometric image operations*. Geometric image operations are, in a sense, the opposite of point operations: they modify the spatial positions and spatial relationships of pixels, but they do not modify gray level values. Generally, these operations can be quite complex and computationally intensive, especially when applied to video sequences. However, the more complex geometric operations are not much used in engineering image processing, although they are heavily used in the computer graphics field. The reason for this is that image processing is primarily concerned with correcting or improving images of the real world, hence complex geometric operations, which distort images, are less frequently used. Computer graphics, however, is primarily concerned with creating images of an unreal world, or at least a visually modified reality, and subsequently geometric distortions are commonly used in that discipline.

A geometric image operation generally requires two steps: First, a spatial mapping of the coordinates of an original image f to define a new image g:

$$g(\mathbf{n}) = f(\mathbf{n}') = f[\mathbf{a}(\mathbf{n})]. \tag{45}$$

Thus, geometric image operations are defined as functions of position rather than intensity. The two-dimensional, two-valued mapping function $\mathbf{a}(\mathbf{n}) = [a_1(n_1, n_2),\ a_2(n_1, n_2)]$ is usually defined to be continuous and smoothly-changing, but the coordinates $\mathbf{a}(\mathbf{n})$ that are delivered are not generally integers. For example, if $\mathbf{a}(\mathbf{n}) = (n_1/3, n_2/4)$, then $g(\mathbf{n}) = f(n_1/3, n_2/4)$, which is not defined for most values of (n_1, n_2). The question then is, which value(s) of f are used to define $g(\mathbf{n})$, when the mapping does not fall on the standard discrete lattice?

Thus implies the need for the second operation: *interpolation* of non-integer coordinates $a_1(n_1, n_2)$ and $a_2(n_1, n_2)$ to integer values, so that g can be expressed in a standard row-column format. There are many possible approaches

for accomplishing interpolation; we will look at two of the simplest: *nearest neighbor interpolation*, and *bilinear interpolation*. The first of these is too simplistic for many tasks, while the second is effective for most.

Nearest Neighbor Interpolation

Here, the geometrically transformed coordinates are mapped to the nearest integer coordinates of f:

$$g(\mathbf{n}) = f\{\text{INT}[a_1(n_1, n_2) + 0.5], \text{INT}[a_2(n_1, n_2) + 0.5]\} \quad (46)$$

where INT$[R]$ denotes the nearest integer that is less than or equal to R. Hence, the coordinates are *rounded* prior to assigning them to g. This certainly solves the problem of finding integer coordinates of the input image, but it is quite simplistic, and, in practice, it may deliver less than impressive results. For example, several coordinates to be mapped may round to the same values, creating a block of pixels in the output image of the same value. This may give an impression of "blocking," or of structure that is not physically meaningful. The effect is particularly noticeable along sudden changes in intensity, or "edges," which may appear jagged following nearest neighbor interpolation.

Bilinear Interpolation

Bilinear interpolation produces a smoother interpolation than does the nearest neighbor approach. Given four neighboring image coordinates $f(n_{10}, n_{20})$, $f(n_{11}, n_{21})$, $f(n_{12}, n_{22})$, and $f(n_{13}, n_{23})$ (these can be the four nearest neighbors of $f[\mathbf{a}(\mathbf{n})]$), then the geometrically transformed image $g(n_1, n_2)$ is computed as

$$g(n_1, n_2) = \text{A}_0 + \text{A}_1 n_1 + \text{A}_2 n_2 + \text{A}_3 n_1 n_2 \quad (47)$$

which is a bilinear function in the coordinates (n_1, n_2). The bilinear weights $\text{A}_0, \text{A}_1, \text{A}_2$, and A_3 are found by solving

$$\begin{bmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \end{bmatrix} = \begin{bmatrix} 1 & n_{10} & n_{20} & n_{10}n_{20} \\ 1 & n_{11} & n_{21} & n_{11}n_{21} \\ 1 & n_{12} & n_{22} & n_{12}n_{22} \\ 1 & n_{13} & n_{23} & n_{13}n_{23} \end{bmatrix}^{-1} \begin{bmatrix} f(n_{10}, n_{20}) \\ f(n_{11}, n_{21}) \\ f(n_{12}, n_{22}) \\ f(n_{13}, n_{23}) \end{bmatrix} \quad (48)$$

Thus, $g(n_1, n_2)$ is defined to be a linear combination of the gray levels of its four nearest neighbors. The linear combination defined by (48) is in fact the value assigned to $g(n_1, n_2)$ when the best (least-squares) planar fit is made to these four neighbors. This process of optimal averaging produces a visually smoother result.

Regardless of the interpolation approach that is used, it is possible that the mapping coordinates $a_1(n_1, n_2)$, $a_2(n_1, n_2)$ do not fall within the pixel ranges

$$0 \le a_1(n_1, n_2) \le M - 1$$
$$\text{and/or} \quad (49)$$
$$0 \le a_2(n_1, n_2) \le N - 1,$$

in which case it is not possible to define the geometrically transformed image at these coordinates. Usually a nominal value is assigned, such as $g(\mathbf{n}) = 0$, at these locations.

Image Translation

The most basic geometric transformation is the *image translation*, where

$$a_1(n_1, n_2) = n_1 - b_1 \text{ and } a_2(n_1, n_2) = n_2 - b_2 \quad (50)$$

where (b_1, b_2) are integer constants. In this case $g(n_1, n_2) = f(n_1 - b_1, n_2 - b_2)$, which is a simple shift or translation of g by an amount b_1 in the vertical (row) direction and an amount b_2 in the horizontal direction. This operation is used in image display systems, when it is desired to move an image about, and it is also used in algorithms, such as image convolution (Chapter 2.3), where images are shifted relative to a reference. Since integer shifts can be defined in either direction, there is no need for the interpolation step.

Image Rotation

Rotation of the image g by an angle θ relative to the horizontal (n_1) axis is accomplished by the following transformations:

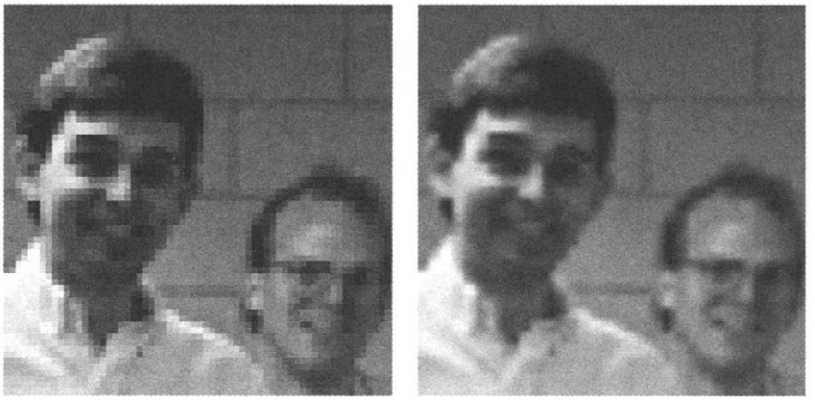
$$a_1(n_1, n_2) = n_1 \cos\theta - n_2 \sin\theta$$
$$\text{and} \quad (51)$$
$$a_2(n_1, n_2) = n_1 \sin\theta + n_2 \cos\theta$$

The simplest cases are: $\theta = 90°$, where $[a_1(n_1, n_2), a_2(n_1, n_2)] = (-n_2, n_1)$; $\theta = 180°$, where $[a_1(n_1, n_2), a_2(n_1, n_2)] = (-n_1, -n_2)$; and $\theta = -90°$, where $[a_1(n_1, n_2), a_2(n_1, n_2)] = (n_2, -n_1)$. Since the rotation point is not defined here as the center of the image, the arguments (51) may fall outside of the image domain. This may be ameliorated by applying an image translation either before or after the rotation to obtain coordinate values in the nominal range.

Image Zoom

The *image zoom* either magnifies or minifies the input image according to the mapping functions

$$a_1(n_1, n_2) = n_1/c \quad \text{and} \quad a_2(n_1, n_2) = n_2/d \quad (52)$$



(a) (b)

FIGURE 19 Example of (4x) image zoom followed by interpolation. (a) Nearest-neighbor interpolation; (b) bilinear interpolation.

where $c \geq 1$ and $d \geq 1$ to achieve magnification, and $c < 1$ and $d < 1$ to achieve minification. If applied to the entire image, then the image size is also changed by a factor c (d) along the vertical (horizontal) direction. If only a small part of an image is to be zoomed, then a translation may be made to the corner of that region, the zoom applied, and then the image cropped.

The image zoom is a good example of a geometric operation for which the type of interpolation is important, particularly at high magnifications. With nearest neighbor interpolation, many values in the zoomed image may be assigned the same gray scale, resulting in a severe "blotching" or "blocking" effect. The bilinear interpolation usually supplies a much more viable alternative.

Figure 19 depicts a 4x zoom operation applied to the image in Fig. 13 (logarithmically transformed "students"). The image was first zoomed, creating a much larger image (16 times as many pixels). The image was then translated to a point of interest (selected, e.g., by a mouse), then was cropped to size 256 × 256 pixels around this point. Both nearest-neighbor and bilinear interpolation were applied for the purpose of comparison. Both provide a nice "close-up" of the original, making the faces much more identifiable. However, the bilinear result is much smoother, and does not contain the blocking artifacts that can make recognition of the image difficult.

It is important to understand that image zoom followed by interpolation does not inject *any* new information into the image, although the magnified image may appear easier to see and interpret. The image zoom is only an interpolation of known information.

Acknowledgment

Many thanks to Professor Scott Acton for carefully reading and commenting upon the early versions of this chapter.