see how the statistical operations can reduce noise in the image, which is of benefit to the feature extraction techniques to be considered later. As such, these basic operations are usually for preprocessing for later feature extraction or to improve display quality.

3.2 Histograms

The intensity *histogram* shows how individual brightness levels are occupied in an image; the *image contrast* is measured by the range of brightness levels. The histogram plots the number of pixels with a particular brightness level against the brightness level. For 8 bit pixels, the brightness ranges from zero (black) to 255 (white). Figure 3.1 shows an image of an eye and its histogram. The histogram (Figure 3.1b) shows that not all the grey levels are used and the lowest and highest intensity levels are close together, reflecting moderate *contrast*. The histogram has a region between 100 and 120 brightness values which contains the dark portions of the image, such as the hair (including the eyebrow) and the eye's iris. The brighter points relate mainly to the skin. If the image was darker, overall, the histogram would be concentrated towards black. If the image was brighter, but with lower contrast, then the histogram would be thinner and concentrated near the whiter brightness levels.

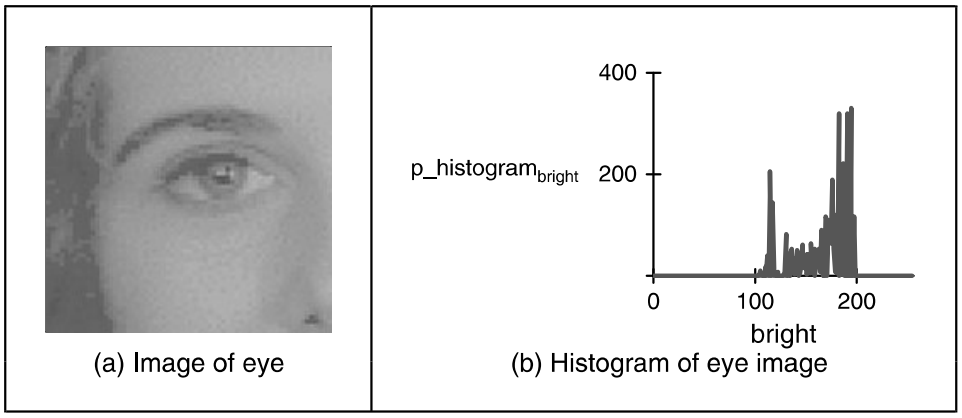


Figure 3.1 An image and its histogram

This histogram shows us that we have not used all available grey levels. Accordingly, we could stretch the image to use them all, and the image would become clearer. This is essentially cosmetic attention to make the image's appearance better. Making the appearance better, especially in view of later processing, is the focus of many basic image processing operations, as will be covered in this chapter. The histogram can also reveal whether there is much noise in the image, if the ideal histogram is known. We might want to remove this noise, not only to improve the appearance of the image, but also to ease the task of (and to present the target better for) later feature extraction techniques. This chapter concerns these basic operations which can improve the appearance and quality of images.

The histogram can be evaluated by the operator histogram, in Code 3.1. The operator first initializes the histogram to zero. Then, the operator works by counting up the number of image points that have an intensity at a particular value. These counts for the different values form the overall histogram. The counts are then returned as the two-dimensional (2D) histogram (a vector of the count values), which can be plotted as a graph (Figure 3.1b).

```
histogram(pic):= | for brighte∈0..255
                 |   pixels_at_level_bright←0
                 | for x∈0..cols(pic)-1
                 |   for y∈0..rows(pic)-1
                 |     level←pic_y,x
                 |     pixels_at_level_level←pixels_at_level_level+1
                 | pixels_at_level
```

Code 3.1 Evaluating the histogram

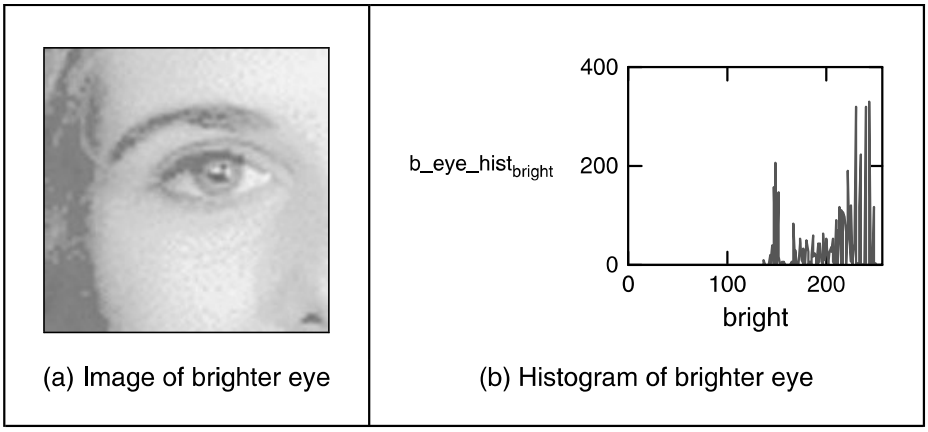
3.3 Point operators

3.3.1 Basic point operations

The most basic operations in image processing are point operations where each pixel value is replaced with a new value obtained from the old one. If we want to increase the brightness to stretch the contrast we can simply multiply all pixel values by a scalar, say by 2 to double the range. Conversely, to reduce the contrast (although this is not usual) we can divide all point values by a scalar. If the overall brightness is controlled by a *level, l*, (e.g. the brightness of global light) and the range is controlled by a *gain, k*, the brightness of the points in a new picture, **N**, can be related to the brightness in old picture, **O**, by:

$$\mathbf{N}_{x,y} = k \times \mathbf{O}_{x,y} + l \qquad \forall x, y \in 1, N \tag{3.1}$$

This is a point operator that replaces the brightness at points in the picture according to a linear brightness relation. The level controls overall brightness and is the minimum value of the output picture. The gain controls the contrast, or range, and if the gain is greater than unity, the output range will be increased. This process is illustrated in Figure 3.2. So the image of the eye, processed by $k = 1.2$ and $l = 10$, will become brighter (Figure 3.2a) and with better contrast, although in this case the brighter points are mostly set near to white (255). These factors can be seen in its histogram (Figure 3.2b).



(a) Image of brighter eye

(b) Histogram of brighter eye

Figure 3.2 Brightening an image

The basis of the implementation of point operators was given earlier, for addition in Code 1.3. The stretching process can be displayed as a mapping between the input and output ranges, according to the specified relationship, as in Figure 3.3. Figure 3.3(a) is a mapping where the output is a direct copy of the input (this relationship is the dotted line in Figure 3.3c and d); Figure 3.3(b) is the mapping for brightness *inversion* where dark parts in an image become bright and vice versa. Figure 3.3(c) is the mapping for *addition* and Figure 3.3(d) is the mapping for *multiplication* (or *division*, if the slope was less than that of the input). In these mappings, if the mapping produces values that are smaller than the expected minimum (say negative when zero represents black), or larger than a specified maximum, then a *clipping* process can be used to set the output values to a chosen level. For example, if the relationship between input and output aims to produce output points with intensity value greater than 255, as used for white, the output value can be set to white for these points, as it is in Figure 3.3(c).

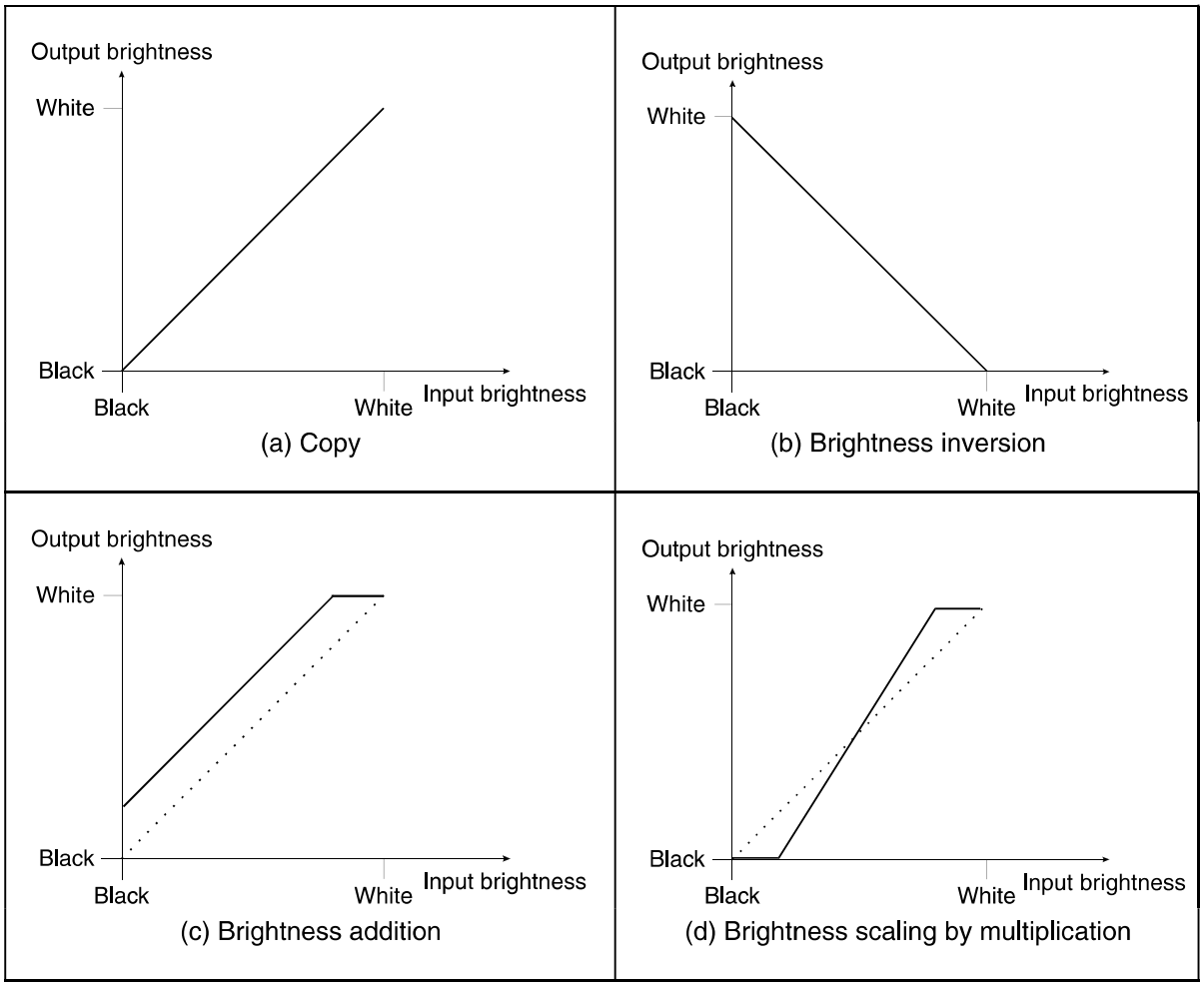


Figure 3.3 Intensity mappings

The *sawtooth* operator is an alternative form of the linear operator and uses a repeated form of the linear operator for chosen intervals in the brightness range. The sawtooth operator is used to emphasize local contrast change (as in images where regions of interest can be light or dark). This is illustrated in Figure 3.4, where the range of brightness levels is mapped into four linear regions by the sawtooth operator (Figure 3.4b). This remaps the intensity in the eye image to

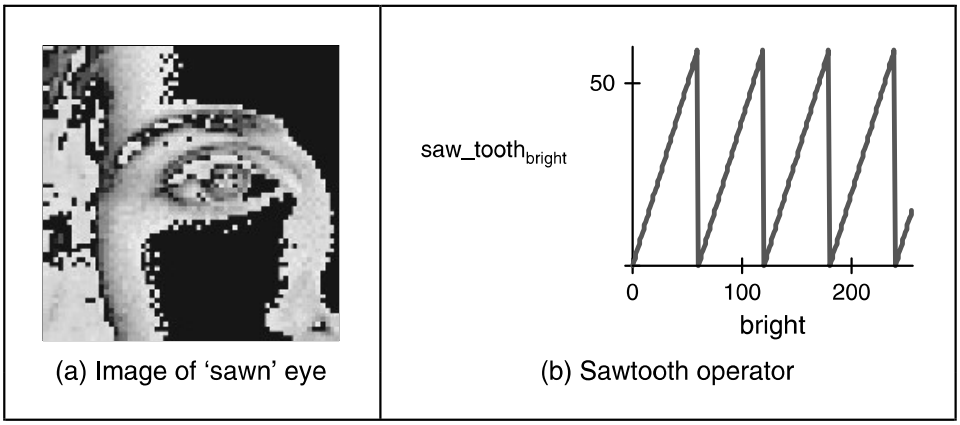


Figure 3.4 Applying the sawtooth operator

highlight local intensity variation, as opposed to global variation, in Figure 3.4(a). The image is now presented in regions, where the region selection is controlled by the intensity of its pixels.

Finally, rather than simple multiplication we can use arithmetic functions such as logarithm to reduce the range or exponent to increase it. This can be used, say, to equalize the response of a camera, or to compress the range of displayed brightness levels. If the camera has a known exponential performance, and outputs a value for brightness which is proportional to the exponential of the brightness of the corresponding point in the scene of view, the application of a *logarithmic point operator* will restore the original range of brightness levels. The effect of replacing brightness by a scaled version of its natural logarithm (implemented as $\mathbf{N}_{x,y} = 20\ln(100\mathbf{O}_{x,y})$) is shown in Figure 3.5(a); the effect of a scaled version of the exponent (implemented as $\mathbf{N}_{x,y} = 20\exp(\mathbf{O}_{x,y}/100)$) is shown in Figure 3.5(b). The scaling factors were chosen to ensure that the resulting image can be displayed since the logarithm or exponent greatly reduces or magnifies pixel values, respectively. This can be seen in the results: Figure 3.5(a) is dark with a small range of brightness levels, whereas Figure 3.5(b) is much brighter, with greater contrast. Naturally, application of the logarithmic point operator will change any *multiplicative* changes in brightness to become *additive*. As such, the logarithmic operator can find application in reducing the effects of multiplicative intensity change. The logarithm operator is often used to

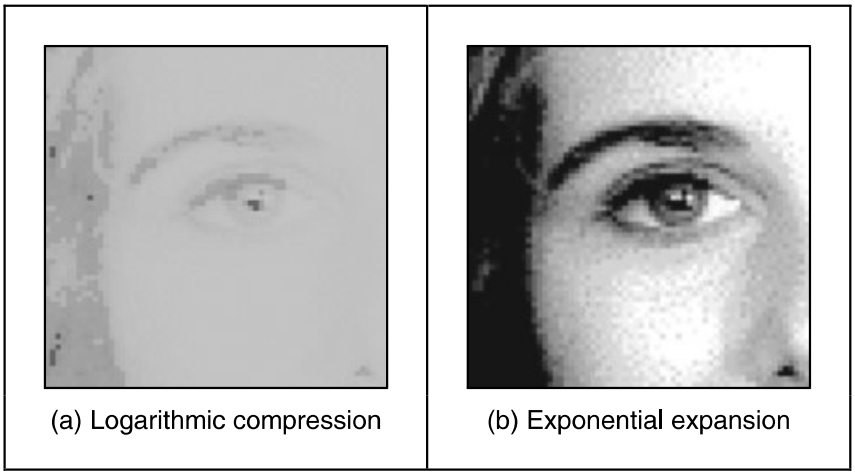


Figure 3.5 Applying exponential and logarithmic point operators

compress Fourier transforms, for display purposes. This is because the d.c. component can be very large with contrast, too large to allow the other points to be seen.

In hardware, point operators can be implemented using *look-up tables* (LUTs), which exist in some framegrabber units. LUTs give an output that is programmed, and stored, in a table entry that corresponds to a particular input value. If the brightness response of the camera is known, it is possible to preprogram a LUT to make the camera response equivalent to a uniform or flat response across the range of brightness levels.

3.3.2 Histogram normalization

Popular techniques to stretch the range of intensities include *histogram* (*intensity*) *normalization*. Here, the original histogram is stretched, and shifted, to cover all the 256 available levels. If the original histogram of old picture $\mathbf{O}$ starts at $\mathbf{O}_{\min}$ and extends up to $\mathbf{O}_{\max}$ brightness levels, then we can scale up the image so that the pixels in the new picture $\mathbf{N}$ lie between a minimum output level $\mathbf{N}_{\min}$ and a maximum level $\mathbf{N}_{\max}$, simply by scaling up the input intensity levels according to:

$$\mathbf{N}_{x,y} = \frac{\mathbf{N}_{\max} - \mathbf{N}_{\min}}{\mathbf{O}_{\max} - \mathbf{O}_{\min}} \times \left(\mathbf{O}_{x,y} - \mathbf{O}_{\min}\right) + \mathbf{N}_{\min} \qquad \forall x, y \in 1, N \tag{3.2}$$

A Matlab implementation of intensity normalization, appearing to mimic Matlab's `imagesc` function, the `normalize` function in Code 3.2, uses an output ranging from $\mathbf{N}_{\min} = 0$ to $\mathbf{N}_{\max} = 255$. This is scaled by the input range that is determined by applying the `max` and the `min` operators to the input picture. Note that in Matlab, a 2D array needs double application of the `max` and `min` operators, whereas in Mathcad `max(image)` delivers the maximum. Each point

```
function normalized=normalize(image)
%Histogram normalization to stretch from black to white

%Usage: [new image]=normalize(image)
%Parameters: image-array of integers
%Author: Mark S. Nixon

%get dimensions
[rows,cols]=size(image);

%set minimum
minim=min(min(image));

%work out range of input levels
range=max(max(image))-minim;

%normalize the image
for x=1:cols %address all columns
  for y=1:rows %address all rows
    normalized(y,x)=floor((image(y,x)-minim)*255/range);
  end
end
```

Code 3.2 Intensity normalization