Chapter 7

Filtering and Enhancement

Signal filtering and enhancement include a broad range of operations that are performed on signals. It's a very important subject as some type of filtering is used in most systems that process information, and some type of enhancement is used in most applications, often as a final step to prepare an output for the end user.

Some goals of filtering and enhancement include detecting, extracting, or separating signals, reducing noise, or accentuating certain features of a signal. Filtering and enhancement techniques can be conveniently divided into the following groups

- Point/histogram operations
- Time/spatial domain operations
- Frequency domain operations
- Geometric operations

Before we proceed, we make some comments about terminology and our focus in this chapter. Often in signal processing, filtering refers to passing a signal through an LTI system. As we've seen, such operations can be viewed in the time domain as a convolution, or in frequency domain in terms of the frequency response of the system. These are the second and third groups of methods above, respectively. In fact, it is the frequency domain perspective that gives rise to the term "filtering" since this can be viewed as allowing certain frequencies of the original signal to pass through, while filtering out the unwanted frequencies. In filtering, there is often a well-defined objective, and part of this objective may be to carry out the desired operation on the signals while meeting some design constraints. The focus is often on designing special filters with desired characteristics that make them suitable for hardware implementation.

[*]©2002 by Sanjeev R. Kulkarni. All rights reserved.

[†]Lecture Notes for ELE201 Introduction to Electrical Signals and Systems.

[‡]Thanks to Sean McCormick for producing the figures.

In contrast, the goals of enhancement may be rather subjective, like trying to make an audio signal more pleasing to listen to, or an image more visually attractive. Despite the lack of a well-defined objective, enhancement is a very important subject since for many systems a human is the final "user" of the signals, and even simple enhancement techniques can greatly improve the experience of the user. For example, recall our discussion of halftoning from Chapter XX. In a broad sense, we can think of this as a type of enhancement to make a printed image more visually pleasing.

Thus, for our purposes, we take a rather broad view of filtering and enhancement rather than much more detailed discussions that would be appropriate for more advanced courses. Here, we focus on some general goals and techniques, rather than on details of filter design.

7.1 Point/Histogram Methods

In point/histogram enhancement techniques, the amplitude of the signal is modified in a way that does not depend on the location of the sample or the amplitudes of neighboring samples. For example, in an image, the gray levels would be modified pixel by pixel with the same transformation applied to each pixel separately.

7.1.1 Point Transformations

If x denotes the amplitude of a sample of the original signal, then the amplitude in the enhanced signal using a point transformation is given by

$$y = f(x)$$

where $f(\cdot)$ is the particular amplitude transformation. For convenience, we will assume that the amplitudes of both the input and output signals are confined to the range $[0, L]$. The key property that makes these point transformations, is that $f(\cdot)$ is independent of the location of the sample and of the amplitudes of neighboring samples.

A number of useful point operations arise from different choices for the transformation $f(\cdot)$. Some specific transformations are discussed below, where we focus on images since these allow us to illustrate a number examples in a visual way.

Contrast Stretching Suppose the original image doesn't occupy a full range of gray levels. Every gray level x of the original image lies in some range $[a, b]$ which is a subset of $[0, L]$.

In this case, a natural operation to perform is to "stretch" the the gray levels in the original image so as to take advantage of the full dynamic range available. This can be performed by the simple linear operation on the gray levels

$$f(x) = \frac{L}{b-a}(x-a) \qquad (7.1)$$

Even if the gray levels in the original image occupy whole range $[0, L]$, a piecewise linear transformation of the type

$$f(x) = \begin{cases} \alpha x & 0 \leq x < a \\ \beta(x-a) + v_a & a \leq x < b \\ \gamma(x-b) + v_b & b \leq x \leq L \end{cases} \tag{7.2}$$

can often still be useful.

In each linear region, a slope greater than 1 results in stretching the domain, while a slope less than 1 results in compressing the domain. Usually the interval $[a, b]$ is chosen where most of the gray levels occur or where certain features of interest lie. The transformation is then chosen to stretch this region and compress the rest. Often, the parameters of the transformation are chosen such that $\gamma(L-b) + v_b = L$ so that the whole range for v is used. Note that in the special case that $\alpha = \gamma = 0$, we have *clipping*, which is the transformation in (7.1).

Range Compression Sometimes a nonlinear gray level transformation such as

$$y = c \log(1+x) \tag{7.3}$$

can be useful in enhancing images which have a large dynamic range in the gray levels and have features of interest throughout the gray level range. Such a transformation, often referred to as *range compression* enhances small magnitude pixels compared with large magnitude.

Thresholding *Thresholding* refers to the case in which the output gray level is either 0 or L depending on whether the input gray level is respectively below or above some constant T. That is,

$$f(x) = \begin{cases} 0 & x \leq T \\ L & x > T \end{cases}$$

This is a special case of the general piecewise linear transformation (7.2) in which $\alpha = \gamma = 0$ and $a = b$.

Thresholding results in a binary image and can be useful in trying to distinguish the foreground in an image from the background. Thresholding can also be useful if the original image is some binary image corrupted with (non-binary) noise. The operation of thresholding is actually a very common operation in a number of other image processing algorithms as well. For example, thresholding is generally used at some stage whenever a binary decision is made based on non-binary data (as in deciding whether or not an edge or some other feature is present).

A slightly more general operation than thresholding is *intensity level slicing* in which

$$f(x) = \begin{cases} L & a \leq x \leq b \\ 0 & \text{otherwise} \end{cases} \tag{7.4}$$

or

$$f(x) = \begin{cases} L & a \leq x \leq b \\ x & \text{otherwise} \end{cases} \tag{7.5}$$

This allows the selection of features in a specific gray level interval of interest.

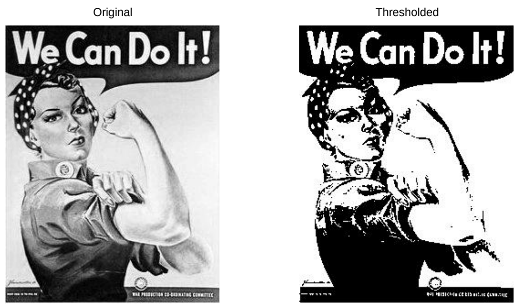

Figure 7.1: Digital Thresholding

Digital Negative Taking

$$f(x) = L - x$$

results in a *digital negative* of the original image which can be useful in bringing out certain features.

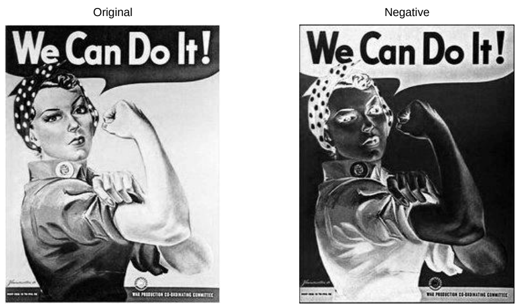

Figure 7.2: Digital Negative Example

Bit Extraction *Bit extraction* refers to generating an output image defined by

$$y = \begin{cases} L & \text{if } k\text{-th bit of } x = 1 \\ 0 & \text{otherwise} \end{cases} \tag{7.6}$$

As the name implies, this corresponds to forming an image by extracting the k-th bit of the input image. Bit extraction can be useful in seeing how many bits are visually significant in the image, together with any spatial dependence on the number of visually significant bits. See Figure 7.3.

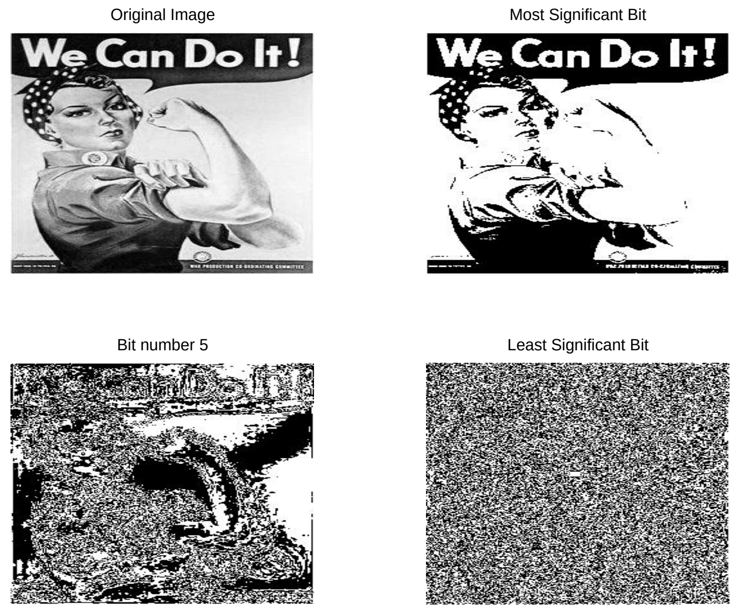

Figure 7.3: Bit Extraction Example

7.1.2 Histogram Modification

The enhancement techniques discussed above are implemented by selecting a specific gray level transformation. Of course, such transformations have an effect on the histogram (frequency of occurrence of gray levels) of an image, but the effect on the histogram is not directly specified.

Another approach to gray level transformation techniques is to modify gray levels in order to shape the histogram of the image to certain specifications. The most widely used of such methods is *histogram equalization*, in which the goal is to produce an output image that has flat (uniform) histogram. The rationale is that if the histogram is very peaked in some areas and very low (or zero) in other areas, then some gray levels are over utilized and other gray levels are under utilized (or not used at all). We should be able to exhibit more features in the signals if we use all the gray levels roughly equally. This is similar to the rationale behind contrast stretching, but takes it even further. In many cases, histogram equalization does in fact enhance the image, bringing out detail that wasn't visible in the original. Figure 7.4 shows an example of this.

If the gray levels take on a continuous range of values, then with a suitable transformation, the histogram can be exactly equalized. However, in the usual case where the gray levels have been quantized, we may not be able to exactly equalize the histogram because certain gray level may occur too often. No matter what value these gray levels are mapped to, the new value will also occur too often. However, we can try to flatten the histogram as much as

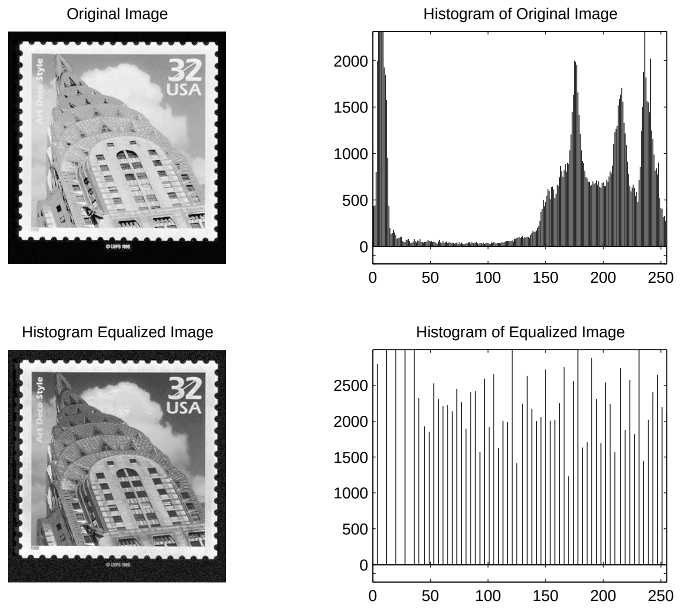

Figure 7.4: Example of Histogram Equalization

possible.

If a strictly flat histogram is desired, the gray levels of the equalized image can be randomly redistributed to different quantization bands. However, this results in a loss of spatial detail, and the strictly flat histogram is seldom necessary.

Histogram equalization is a special case of *histogram specification* in which one performs a transformation to make the histogram of the output image satisfy particular specifications.

7.2 Time/Spatial Domain Methods

Two common goals in time/spatial and frequency domain filtering and enhancement are to reduce noise or enhance edges in the image. Edge enhancement (also called sharpening) is accomplished by emphasizing high frequencies (or de-emphasizing low frequencies) Noise reduction is accomplished by emphasizing low frequencies (or de-emphasizing high frequencies). Both of these operations can be performed in either the time/spatial domain or in frequency domain. Time/spatial domain operators are discussed in this section and frequency domain methods are discussed in the next section.

In the time/spatial domain, the operations are performed by a convolution. Usually a small mask is chosen, which corresponds to the impulse response of the filter. This original signal is convolved with this mask. The choice of the mask determines the type of operation done to the signal.

7.2.1 Noise Reduction

Noise reduction is accomplished by averaging, which corresponds to low pass (or low emphasis) filtering. This can be accomplished by convolving the image with a spatial mask that has all positive coefficients.

Some common common low pass filters are

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

and

$$\frac{1}{5}\begin{bmatrix} & 1 & \\ 1 & 1 & 1 \\ & 1 & \end{bmatrix}$$

Entries that are not shown are assumed to be zero. The gray level of a pixel in the filtered image is the weighted average of the gray levels in a neighborhood from the original image. The center pixel of the mask is assumed to be the center of the impulse response for the convolution.

Usually the filters are normalized so that the sum of the entries is one. This avoids amplitude bias so that average gray levels are kept the same. For example, in an area of the image that has constant gray levels, the filtered image will also have this same constant gray level. If the sum were not one, then the gray level in a constant region would get multiplied by the sum of the pixels in the mask.

Filters can also be used which give more weight to certain pixels. Usually the weights near the center of the mask are larger since this gives more weight to those pixels in the original image close to the corresponding location in the filtered image. Some examples are

$$\frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

$$\frac{1}{6}\begin{bmatrix} & 1 & \\ 1 & 2 & 1 \\ & 1 & \end{bmatrix}$$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

Of course, we are not restricted to using only 3×3 masks. We could use smaller or larger masks, e.g., 2×2, 5×5, or 7×7. However, there is a fundamental trade-off that drives the choice of the mask size.

These low pass filters do reduce noise, particularly i.i.d noise at each pixel, but they also blur edges. The larger the mask, the more averaging, and hence the more noise reduction we can obtain. On the other hand, larger masks increase the blurring of edges and result in a filtered image of lower resolution.

There are some techniques that try to better preserve the edges. One such technique is directional smoothing in which one can try to estimate an edge and then perform averaging along the direction of the edge, but not across it. These techniques can sometimes greatly reduce the blurring due to low emphasis filtering.

Another popular and often very effective technique is *median filtering*. In this method, the input gray level is replaced by the median of the gray levels of pixels in some neighborhood. The median of a set of numbers is middle value (or the average of the two middle values if the number of values is even).

Typically the neighborhood is taken to be a 3×3, 5×5, 7×7 window, or the pixel with its four nearest neighbors. The value of a pixel in the filtered image is given by the median of the values of the pixels in the corresponding neighborhood of the original image.

Median filtering is a nonlinear operation. It cannot be represented by a convolution. It works particularly well for a kind of noise that is often called salt/pepper noise (or impulsive noise, or binary noise). With this type of noise, most of the pixels are unchanged, but occasionally a pixel is very noisy – e.g., it might get replaced by a pure white or pure black pixel. With this type of noise, the corrupted image looks like the original but with a sprinkling of random white and black pixels (hence the term salt/pepper noise). When not too many pixels in the window are noisy, median filtering can have a dramatic effect. For example, if only one pixel is corrupted in a smooth area of the image, then the median value in a 3×3 neighborhood does not change much. On the other hand, the average of nine pixels in a 3×3 neighborhood can change by a large amount if just a single pixel is changed from its original gray level to 255. Median filtering also tends to preserve edges well.

However, if the noise is the usual type in which each pixel is corrupted by adding some random value, then median filtering won't work so well. In this case, averaging tends to work better. Also, median filtering can be time consuming if the window is large since a direct implementation requires a large number of comparisons of values. However, this can be somewhat reduced using more efficient implementations of median filtering.

7.2.2 Sharpening

The basic idea in image sharpening is to emphasize high frequencies (or relatively de-emphasize low frequencies). Often high emphasis filtering is performed by either subtracting a low pass filtered version of the original image or adding a high pass filtered version. Namely, the sharpened image y is given by

$$y(m,n) \;=\; x(m,n) - c x_{LP}(m,n) \qquad (7.7)$$

or

$$y(m,n) \ = \ x(m,n) + cx_{HP}(m,n) \tag{7.8}$$

As discussed above, the low pass filtered image x_{LP} is some kind of averaging filter. A high pass filtered image x_{HP} is generally an approximation to some type of spatial derivative operator such as a Laplacian operator. A common approximation for the Laplacian is

$$\begin{bmatrix} & -1 & \\ -1 & 4 & -1 \\ & -1 & \end{bmatrix}$$

As with the averaging operators for noise reduction, the sum of the coefficients in a sharpening operator is also commonly taken to be one to preserve amplitudes in regions of constant gray level.

Some examples are

$$\begin{bmatrix} & -1 & \\ -1 & 5 & -1 \\ & -1 & \end{bmatrix}$$

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

$$\begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$

7.3 Frequency Domain Methods

As with the time/spatial domain operators, two common goals of frequency domain filtering and enhancement are noise reduction (low emphasis filtering) or sharpening (high emphasis filtering). The only difference is that the low or high emphasis filtering is done in the frequency domain.

The general method is to apply a transform such as the DFT to the original image. Then in frequency domain, the appropriate frequencies are either emphasized (increased in magnitude) or de-emphasized (decreased in magnitude, possible even set to 0).

For sharpening the high frequencies are emphasized and/or the low frequencies are de-emphasized, while for noise reduction/smoothing the opposite is done.

In general these enhancement operations are performed by multiplying the frequency coefficients by some weights representing the frequency response of the filter. The enhanced image is then obtained by performing the inverse transform.

As in time/spatial domain, certain nonlinear frequency domain enhancement techniques have also been found to give good results. In root filtering the transform coefficients $v(k,l)$ are expressed in polar form

$$v(k,l) \;=\; |v(k,l)|e^{j\theta(k,l)} \tag{7.9}$$

Then enhancement is performed by replacing these coefficients with

$$v^{\bullet}(k,l) \;=\; |v(k,l)|^{\alpha}e^{j\theta(k,l)}. \tag{7.10}$$

and performing the inverse transform. For $0 < \alpha < 1$, the high frequency coefficients are generally emphasized (since the high frequency coefficients generally have smaller magnitude than the low frequency coefficients). Hence, this corresponds to a sharpening operation. Taking $\alpha > 1$ corresponds to a low emphasis operation.

Another nonlinear technique that can sometimes give dramatic results is homomorphic filtering. Here the transform coefficients are replaced by

$$v^{\bullet}(k,l) \;=\; [\log|v(k,l)|]e^{j\theta(k,l)} \tag{7.11}$$

or

$$v^{\bullet}(k,l) \;=\; [\log(|v(k,l)|+c)]e^{j\theta(k,l)} \tag{7.12}$$

where c is a positive constant to prevent the argument of $\log(\cdot)$ from being zero.

7.4 Geometric Transformations

A very different class of enhancement techniques from those discussed so far are *geometric operators*. These operators modify the geometric *positions* of pixels. That is, these techniques operate on the *domain* of the image rather than directly operating in the range (gray levels) of the image. Geometric transformations are useful in image registration and matching, reducing certain geometric degradations such as warping, and in various graphics applications.

The general geometric transformation maps the input image pixel coordinates i, j to some new coordinates i_o, j_o, which are the output image pixel coordinates. Then the transformed image is given by

$$y(i_o, j_o) = x(i,j) \tag{7.13}$$

That is, the gray level of the transformed image at i_o, j_o is the given by gray level of the original image at i, j. Of course, for digitized images the coordinates from the transformation often will not correspond exactly to the center of a pixel in which case (7.13) needs to be replaced by interpolation over several input pixels.

Two common methods for subpixel interpolation are nearest neighbor interpolation and bilinear interpolation. As its name implies, in nearest neighbor interpolation, the gray level at a particular point is taken to be the gray level of the pixel closest to the point. In bilinear interpolation, linear interpolation is performed along each now, then the result is linearly interpolated along each column. The net result is nonlinear. Bilinear interpolation can be much more effective than nearest neighbor interpolation in reducing artifacts. There are many other interpolation methods as well.

Below we discuss some of the standard geometric transforms. These are quite simple, but often can be very useful.

Translation For translating (shifting) the original image, the transform between the input and output pixels is

$$i = i_0 - t_i \tag{7.14}$$
$$j = j_0 - t_j \tag{7.15}$$

or equivalently

$$i_0 = i + t_i \tag{7.16}$$
$$j_0 = j + t_j \tag{7.17}$$

The output image is given by

$$y(i_0, j_0) = x(i_0 - t_i, j_0 - t_j) \tag{7.18}$$

If t_i, t_j are multiples of pixels, then the transformation is completely trivial. Otherwise some interpolation method must be used.

Rotation For rotating the original image, the transform is given by

$$i_0 = i\cos\theta - j\sin\theta \tag{7.19}$$
$$j_0 = i\sin\theta + j\cos\theta \tag{7.20}$$

or equivalently

$$i = i_0\cos\theta + j_0\sin\theta \tag{7.21}$$
$$j = -i_0\sin\theta + j_0\cos\theta \tag{7.22}$$

The rotated image is therefore

$$y(i_0, j_0) = x(i_0\cos\theta + j_0\sin\theta, -i_0\sin\theta + j_0\cos\theta) \tag{7.23}$$

Interpolation is necessary unless θ is an integer multiple of 90°.

Scaling For scaling the original image (i.e., for magnification or minification), the transform is given by

$$i_0 = S_i i, \quad j_0 = S_j j \tag{7.24}$$

or equivalently

$$i = \frac{1}{S_i} i_0, \quad j = \frac{1}{S_j} j_0 \tag{7.25}$$

$S_i, S_j > 1$ corresponds to magnification while $S_i, S_j < 1$ corresponds to minification. For minification, several pixels of input replaced by one pixel in the output image. This can be performed by subsampling (with interpolation if necessary) or averaging. For magnification, interpolation is necessary. The simplest method is simply to replicate the input image gray levels (which is basically nearest neighbor interpolation), but this can give a very blocky appearance in the output image. Bilinear interpolation can improve the results significantly.

There are other geometric modifications such as warping or camera transformations that are also quite useful in certain applications.