CHAPTER 4

SHARPENING

Godfried Toussaint

ABSTRACT

This chapter introduces the basic ideas behind image enhancement. Our goal here is in one sense the opposite of the one we had in *smoothing*. We are interested in suppressing areas of the picture where change is not present and exaggerating details and portions of the image where change is evident. Therefore extracting contours is a by-product of sharpening. Furthermore, digital operators for performing such transformations depend heavily on either neural network evidence from biology or the mathematical theory of spatial differentiation.

1. Introduction

Let $g(x)$ be a single-valued one dimensional function as in the previous chapter. We are interested in *sharpening* $g(x)$, i.e., accentuating those portions of $g(x)$ where there is change while suppressing those parts that are fairly constant. Clearly, the extraction of the contour of a binary image as was done with the square contour tracing algorithm is one form of sharpening. However, square tracing (and even Moore tracing) is limited. For one thing it is not immediately obvious how to make it work for grey-level pictures and for another it only extracts the outer boundary while ignoring the boundaries of "holes" in patterns. In an ideal situation if the original intensity function $g(x)$ looks something like the curve in Fig. 1.1 (a) we would like to obtain something like that illustrated in Fig. 1.1 (b). An excellent mathematical tool for detecting change is the operation of taking the derivative which works well in idealized situations. For example, the absolute value of the derivative of g(x) in Fig. 1.1 (a) yields precisely the curve in Fig. 1.1 (b). In practical situations where we must combat noise it may be necessary to combine differentiation with smoothing into one single operator. One possibility is to compute the following *derivative estimation operator*.

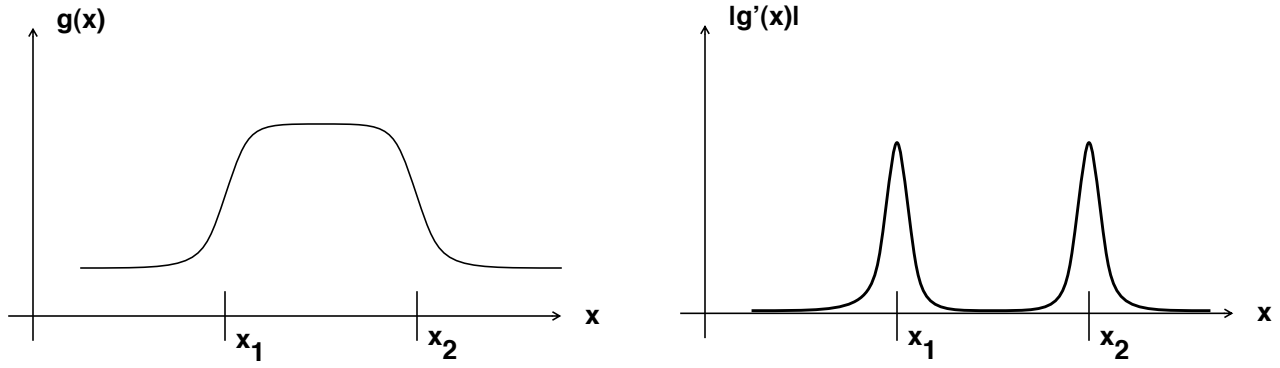
$$D_w(x) = (1/w) \mid [\int g(u)\, du - \int g(u)\, du] \mid$$

where the first integral is taken from x-w to x and the second from x to x+w.

Here, as in smoothing, the parameter w is a tuning parameter that must be set to a value dependent on the application at hand in order to obtain satisfactory results. Applying such an operator to the waveform of Fig. 1.2 (a) we obtain a curve such as that in Fig. 1.2 (b). If a binary output picture is desired we need only threshold the result with some suitable value t as illustrated in Fig. 1.2. If $D_w(x) > t$ we set $D_w(x)$ to *one*, otherwise we set $D_w(x)$ to *zero*.

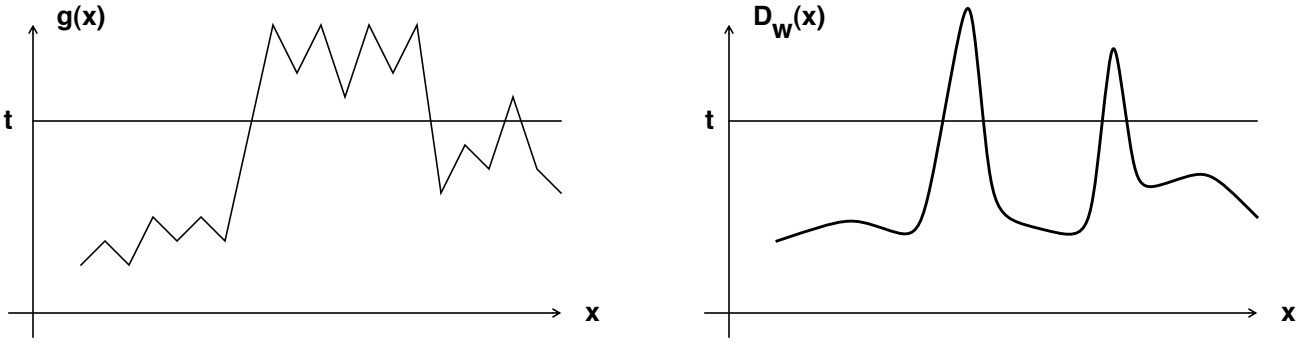
2. Spatial Differentiation

Let $g(x,y)$ be a single-valued two dimensional function denoting the light intensity across the field of vision. An excellent mathematical tool for detecting change in a two-dimensional func-



(a) Original waveform (b) Enhanced waveform

Fig. 1.1 Idealized one dimensional contour extraction.



(a) Original waveform (b) Enhanced waveform

Fig. 1.2 Contour extraction from a noisy waveform combining smoothing.

tion is the *gradient* defined as:

$$\nabla(x,y) = [\partial\ g(x,y)/\partial x]\ \boldsymbol{i} + [\partial\ g(x,y)/\partial y]\ \boldsymbol{j}.$$

This function being a vector has both a magnitude and a direction. Let $\Delta(x) = \partial\ g(x,y)/\partial x$ and $\Delta(y) = \partial\ g(x,y)/\partial y$. Then the direction of $\nabla(x,y)$ is given by $\theta = tan^{-1}(\Delta(x)/\Delta(y)$. While the direction θ is useful in image processing it is more often the magnitude of $\nabla(x,y)$ that is employed. Such is the case here. Several *norms* are used to measure the magnitude of $\nabla(x,y)$ but the L_1 and L_2 norms are the most commonly used. Given the function $\nabla(x,y)$, its magnitude measured by the

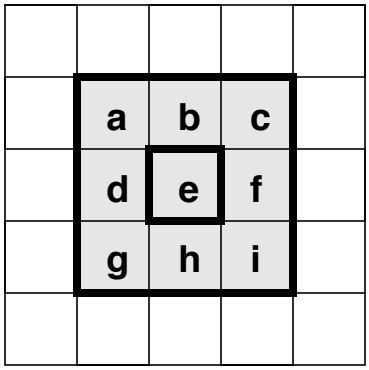


Fig. 2.1 A 3x3 window operator used in *spatial differentiation*.

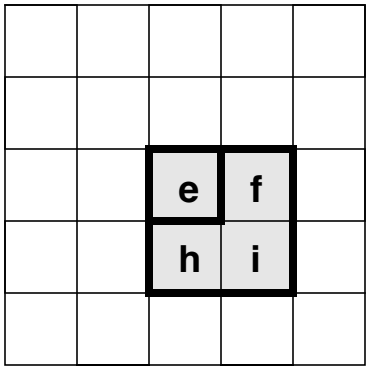


Fig. 2.2 A 2x2 window operator used in the *Roberts Cross Operator*.

p-norm or *p-Minkowski metric* of its components, denoted by, $|\nabla(x,y)|$, is defined as

$$|\nabla_p(x,y)| = \{|\Delta(x)|^p + |\Delta(y)|^p\}^{1/p}.$$

Thus for the L_2 norm the magnitude of the gradient is given by,

$$|\nabla_2(x,y)| = \{[\Delta(x)]^2 + [\Delta(y)]^2\}^{1/2},$$

and for the L_1 norm the magnitude of the gradient is given by,

$$|\nabla_1(x,y)| = |\Delta(x)| + |\Delta(y)|.$$

Let us turn now to the case of two-dimensional digital pictures. As with smoothing we will use a 3x3 window operator with the standard notation for the labels of the pixels as illustrated in Fig. 2.1. If we are interested in measuring the change in the x direction at pixel e, then it is reasonable to take the difference between the values of f and d. On the other hand this function ignores the information that the pixels in the first and third rows contain. Therefore it seems reasonable to incorporate $c - a$ as well as $i - g$. Let S_x and S_y denote the digital approximation of $\Delta(x)$ and $\Delta(y)$, respectively. Then we define:

$$S_x = (c + f + i) - (a + d + g)$$

$$S_y = (a + b + c) - (g + h + i).$$

The magnitude of the gradient approximation under the L_2 norm then becomes

$$S = [S_x^2 + S_y^2]^{1/2}.$$

Although the above expressions use the first and third rows for estimating the change at pixel e it is sometimes felt that more weight should be given to the central row and column. Accordingly the above expressions are generalized as follows:

$$S_x = (w_{13}c + w_{22}f + w_{33}i) - (w_{11}a + w_{21}d + w_{31}g)$$

$$S_y = (w_{11}a + w_{12}b + w_{13}c) - (w_{31}g + w_{32}h + w_{33}i).$$

where $0 \le w_{ii} \le 1$, i=1,2,3. It is customary to make the weights of pixels d, b, h and f equal to two and the rest equal to one. However in practice one notices little difference if all the weights

are set to one.

One may wonder if it is possible to differentiate a digital image with only a 2 by 2 operator and indeed the answer is in the affirmative. The so-called *Roberts Cross Operator* [Ro68] is one such procedure. In this operator our two chosen directions for measuring change are orthogonal to each other but not parallel to the x and *y* axes. The directions are instead, *x*+*y* and *y*-*x*. This choice is more natural for a 2 by 2 window and simplifies the algorithm further. Thus using the 2 by 2 window illustrated in Fig.2.2 the *Roberts Cross Operator* $R_p(e)$ under the *p*-norm, for *p*=2 and 1 respectively is given by

$$R_2(e) = [(e-i)^2 + (f-h)^2]^{1/2} \text{ and}$$

$$R_1(e) = |e-i| + |f-h|.$$

3. Why Not Second Derivatives?

If we can take the first derivative of a picture why not the second? In this section we explore this idea but first we take a short detour through the field of *neural networks*. Simple and clear introductions to the physiological aspects of neural networks can be found in Chapter 2 (*Neural Information Processing*) of [LN72] and in Chapter 1 of [Ar72]. An introduction to the geometrical aspects of neural networks with emphasis on which properties of a digital image can and cannot be computed with a given structural model of a neural network, can be found in [MP69]. A readable introductory exposition of the fundamental mathematical aspects of learning with neural networks is given in [Ni90]. An excellent account of the theory of neural computation from the statistical mechanics point of view can be found in [HKP91].

3.1 Neural Networks

A neural network is a network of *receptor* and *neural cells*. Some useful models of neural cells are illustrated in Figs. 3.1 and 3.2. Fig. 3.1 depicts a model of a *receptor cell*. Such a cell receives one input stimulus in the form of light or any other sensory stimulation, represented as a number *x*, and delivers one output of neural response (either in terms of a voltage, a current, or a rate of firing of impulse waveforms) represented by the same number *x*. Fig. 3.2 illustrates the model for a neural cell, in some sense the main component of neural networks. In general such a cell receives *d* inputs (typically neural responses from either receptor cells or outputs of other neural cells) represented by the numbers $X = (x_1, x_2,..., x_d)$ and delivers one output denoted by the number *y*. The neural response *y* of a neural cell is a weighted linear combination of the inputs X. With each element x_i of the vector X is associated a weight w_i which is a real number. Then, $y = w_1x_1 + w_2x_2 +... + w_dx_d$. The weight w is also called the gain and the connection is called either *inhibitory* or *excitatory* depending on whether the sign of w is either negative or positive, respectively. By connecting receptor cells and neural cells in a network and setting the weights to appropriate values we can obtain artificial neural networks that can simulate a variety of behaviors observed in biological neural networks.

Much of the knowledge of how visual receptors work in biological systems come from studies done on *Limulus*, the horseshoe crab, by Hartline and Ratliff [HR57]. They showed that the response of a cell can be diminished by exciting a neighboring cell. This simple idea, known as *lateral inhibition*, has become one of the most important mechanisms in sensory information processing and provides all the necessary computing power to perform image enhacement and data

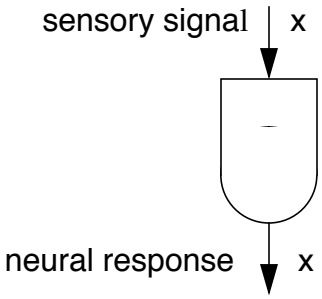


Fig. 3.1 A receptor cell.

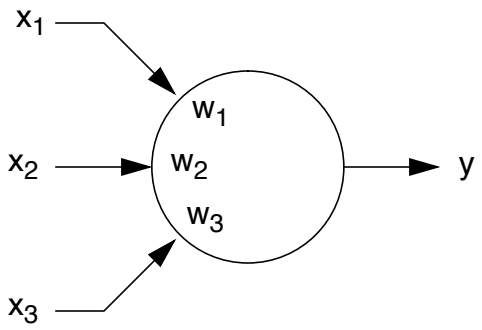


Fig.3.2 A neural cell.

reduction. For example, by setting the weights in the neural cell of Fig. 3.2 as follows: $w_1 = w_3 = -0.2$ and $w_2 = 1.0$ and combining many such cells together in parallel, in one-to-one correspondence with an array of receptor cells, we can simulate the lateral inhibition behaviour of the eye of the limulus crab by connecting the output of the receptor cells not only to their corresponding neural cell but also to the neural cell's neighbors as illustrated in Fig. 3.3. Of course we are free to chose whatever values of the weights we want in order to obtain useful results in image enhancement. However with the values shown we obtain very close agreement with the results of the physiological experiments performed by Hartline and Ratliff [HR57].

3.2 The Laplacian

Let us return briefly to the continuous light intensity model of spatial vision and let $g(x,y)$ be as before a single-valued two dimensional function denoting the light intensity across the field of vision. Another excellent mathematical tool for detecting some form of change in a two-dimensional function is the two-dimensional second derivative operator or *Laplacian* defined as:

$$L(x,y) = \partial^2 g(x,y)/\partial x + \partial^2 g(x,y)/\partial y.$$

A question that immediately arizes after a discussion of the gradient operator is: what exactly are the effects of performing such an operation on a light intensity function $g(x,y)$. To see this in a clear way it is more convenient to examine the Laplacian in its digital approximation form. In the discrete case let us denote $L(x,y)$ by $L(e)$, the approximate discrete Laplacian evaluated at pixel e. Accordingly let us see if we can first approximate the term $\partial^2 g(x,y)/\partial x$ using only a 3x3 window operator as used in spatial differentiation in Fig. 2.1. If the derivative along the row of pixels *def* measures the rate of change then the Laplacian must measure the rate-of-the-rate of change. Now in the gradient operator the rate of change along the row of pixels *def* at pixel e was measured approximately by the term $f-d$. Note that a 3x3 window actually allows us to measure the rate of change twice and at two neighboring locations on each side of pixel e. One such measure is $e-d$ and the other is $f-e$. Therefore we can measure the term $\partial^2 g(x,y)/\partial x$ using $(f-e)-(e-d)$ which is equivalent to $f+d-2e$. We could average such a function over the three rows in the 3x3 window but more clarity will be obtained in this case if we leave it as it is. Similarly, we may approximate $\partial^2 g(x,y)/\partial y$ by $(b-e)-(e-h)$. The digital Laplacian then becomes:

$$L(e) = (b+d+f+h) - 4e.$$

Note that the function $L(e)$ adds the pixel values of the *4-neighbors* of e and subtracts 4

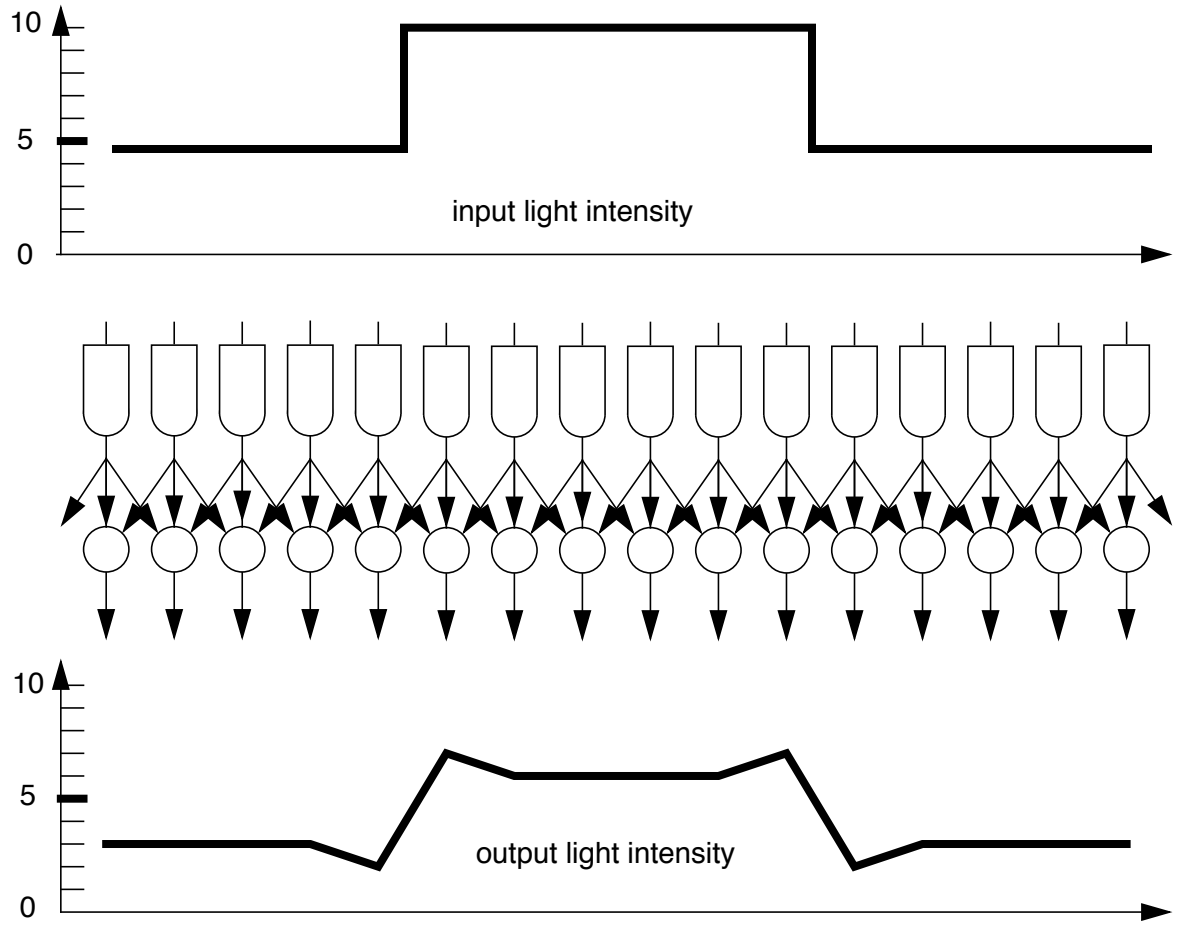


Fig. 3.3 A lateral inhibition network with parameters set to simulate the output of the eye of *Limulus*.

times the value of the center pixel e. Instead of using the 4-neighbors we can just as well use all the 8-neighbors and define the Laplacian as follows:

$$L(e) = (a+b+c+d+f+g+h+i) - 8e.$$

Furthermore, this expression is not changed if we add e inside the parenthesis and subtract e outside the parenthesis to obtain:

$$L(e) = (a+b+c+d+e+f+g+h+i) - 9e.$$

Finally we remark that in almost all image processing algorithms the absolute values of pixel values are not important. What counts are the relative values because no matter what the absolute values are the picture intensity is scaled by a gain control to suit the viewer. Therefore we can multiply or divide all the pixel values in a picture at will without changing the inherent structure com-

puted by an operator. Therefore we can divide, if we like, the above Laplacian by 9 to obtain:

$$L(e) = (a+b+c+d+e+f+g+h+i)/9 - e.$$

Here we recognize that the first term is the value of the pixel e when it is smoothed with a 3x3 window. The second term is just the negative of the pixel e. If we let $S(e)$ denote the smoothed picture at e and $N(e)$ the *negative* of a picture at e, then we see that:

$$L(e) = S(e) + N(e).$$

In other words, the Laplacian can be viewed as the superimposition of a *blurred positive* with a *sharp negative*. Indeed, photographers were using these types of techniques (which they called *unsharp-masking*) in their laboratories to experiment with special effects years before computer scientists used them for image processing applications and before neural physiologists discovered neural networks in biological organisms that computed such functions. Let us now consider some other functions or interpretations of the Laplacian and some of its variations.

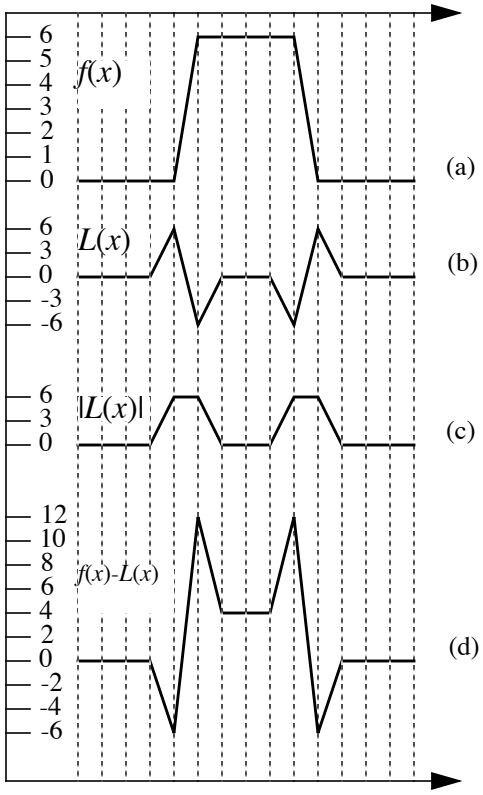


Fig. 3.4 Using the Laplacian to simulate lateral inhibition.

To simplify the explanation consider a one dimensional picture or a slice of a two-dimensional picture and let the light intensity function denoted by $f(x)$ be a simple step function as illustrated in Figure 3.4 (a). Furthermore let us discretize the function so that x only takes on values along the vertical dotted lines thus simulating pixels. The discrete Laplacian at a point x, denoted by $L(x)$, then is equal to the sum of the values of f at the neighboring point of x minus twice the value of f at x. The Laplacian of $f(x)$ is shown in Fig. 3.4 (b) and its absolute value, $|f(x)|$, is given in Fig. 3.4 (c). Note that if we threshold this function appropriately we can use the Laplacian to extract the boundary of the pattern. Fig. 3.4 (d) shows what happens when we subtract the Laplacian from the original function, i.e., $f(x)$-$|f(x)|$. Note that the result has the same form as the output of the lateral inhibition neural network of Fig. 3.3. Therefore the Laplacian can also be used in this manner to simulate lateral inhibition.

In general then the Laplacian is used for image enhancement as well as boundary extraction and edge detection. There are many variations of the basic Laplacian that researchers have investigated in their search for operators that do basically the same thing but with varying degrees of robustness to noise [SR78]. We mention here only one such variant which is called the *pseudo*-Laplacian. Consider again the 3X3 window operator of Fig. 2.1. If we denote the x-component of the Laplacian by $\Delta^2 x = (f+d)$-$2e$ *and the* y-component by $\Delta^2 x = (b+h)$-$2e$ then the pseudo-Laplacian is given by the maximum of the two rather than their sum as in the ordinary Laplacian. In other

words, the pseudo Laplacian is given by *max* $\{\Delta^2 x, \Delta^2 y\}$.

4. References

[Ar72] Arbib, M.A., *The Metaphorical Brain*, John Wiley, New York, 1972.

[BB82] Ballard, D. H. and Brown, C. M., *Computer Vision,* Prentice-Hall, Inc., 1982.

[DH73] Duda, R. O. and Hart, P. E., *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1973.

[HKP91] Hertz, J., Krogh, A. and Palmer, R. G., *Introduction to the Theory of Neural Computation*, Addison-Wesley Publishing Co., 1991.

[HR57] Hartline, H. K. and Ratliff, F., "Inhibitory interaction of receptor units in the eye of limulus," *Journal of General Physiology,* vol. 40, 1957, pp. 357-376.

[LN72] Lindsay, P. H. and Norman, D. A., *Human Information Processing*, Academic Press, New York, 1972.

[MP69] Minsky, M. and Papert, S., *Perceptrons: An Introduction to Computational Geometry*, MIT Press, Cambridge, MA., 1969.

[Ni90] Nilsson, N. J., *The Mathematical Foundations of Learning Machines*, Morgan Kaufmann Publishers, San Mateo, CA., 1990.

[Pa82] Pavlidis, T., *Algorithms for Graphics and Image Processing*, Computer Science Press, 1982.

[Ro68] Roberts, L. G., "Machine perception of three-dimensional solids," in *Optical and Electro-Optical Information Processing,* J. T. Tippet et al. (eds.), MIT Press, Cambridge, 1968, pp. 159-197.

[Se82] Serra, J., *Image Analysis and Mathematical Morphology*, Academic Press, 1982.

[SR78] Schacter, B. J. and Rosenfeld, A., "Some new methods of detecting step edges in digital pictures," *Communications of the ACM*, vol. 21, No. 2, February 1978, pp. 172-176.