CHAPTER 3

EDGE DETECTION USING CLASICAL EDGE DETECTORS

Edge detection is one of the most important operations in image analysis. An edge is a set of connected pixels that lie on the boundary between two regions. The classification of edge detectors discussed in this chapter is based on the behavioral study of these edges with respect to the following operators:

- Gradient edge detectors
- Laplacian of Gaussian
- Gaussian edge detectors

3.1 GRADIENT EDGE DETECTORS

The first derivative assumes a local maximum at an edge. For a gradient image $f(x, y)$, at location (x, y), where x and y are the row and column coordinates respectively, we typically consider the two directional derivatives. The two functions that can be expressed in terms of the directional derivatives are the gradient magnitude and the gradient orientation.

The gradient magnitude is defined by

$$\left|\nabla f\right| = \left\|\begin{bmatrix} G_x \\ G_y \end{bmatrix}\right\| = \left\|\begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}\right\| = \left[G_x^2 + G_y^2\right]^{\frac{1}{2}} \qquad \textit{(3.1)}$$

This quantity give the maximum rate of increase of *f(x,y)* per unit distance in the gradient orientation of $\left|\nabla f\right|$. The gradient orientation is also an important quantity. The gradient orientation is given by

$$\angle \nabla f(x, y) = \tan^{-1}\left(\frac{G_y}{G_x}\right) \qquad \textit{(3.2)}$$

where the angle is measured with respect to the x- axis. The direction of the edge at (x, y) is perpendicular to the direction of the gradient vector at that point. The other method of calculating the gradient is given by estimating the finite difference.

$$\frac{\partial f}{\partial x} = \lim_{h \to 0} \frac{f(x+h, y) - f(x, y)}{h} \qquad \textit{(3.3)}$$

$$\frac{\partial f}{\partial y} = \lim_{h \to 0} \frac{f(x, y+h) - f(x, y)}{h}$$

therefore we can approximate this finite difference as

$$\frac{\partial f}{\partial x} = \frac{f(x+h, y) - f(x, y)}{h_x} = f(x+1, y) - f(x, y), (h_x = 1) \qquad \textit{(3.4)}$$

$$\frac{\partial f}{\partial y} = \frac{f(x+h, y) - f(x, y)}{h_y} = f(x, y+1) - f(x, y), (h_y = 1) \quad (3.5)$$

Using the pixel coordinate notation and considering that *j* corresponds to the direction of *x* and *i* corresponds to the *y* direction

$$\frac{\partial f}{\partial x} = f(i, y+1) - f(i, j) \quad (3.6)$$

$$\frac{\partial f}{\partial y} = f(i-1, y) - f(i, j) \text{ or } \frac{\partial f}{\partial y} = f(i, j) - f(i+1, j) \quad (3.7)$$

The most popular classical gradient-based edge detectors are Roberts cross gradient operator, Sobel operator and the Prewitt operator.

3.1.1 ROBERTS EDGE DETECTOR

The calculation of the gradient magnitude and gradient magnitude of an image is obtained by the partial derivatives $\frac{\partial f}{\partial x}$ and $\frac{\partial f}{\partial y}$ at every pixel location. The simplest way to implement the first order partial derivative is by using the Roberts cross gradient operator.

Therefore

$$\frac{\partial f}{\partial x} = f(i, j) - f(i+1, j+1) \quad (3.8)$$

$$\frac{\partial f}{\partial y} = f(i+1, j) - f(i, j+1) \quad (3.9)$$

The above partial derivatives can be implemented by approximating them to two 2x2 masks. The Roberts operator masks are

$$G_x = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

These filters have the shortest support, thus the position of the edges is more accurate, but the problem with the short support of the filters is its vulnerability to noise. It also produces very weak responses to genuine edges unless they are very sharp.

3.1.2 PREWITT EDGE DETECTOR

The Prewitt edge detector is a much better operator than the Roberts operator. This operator having a 3x3 masks deals better with the effect of noise.
An approach using the masks of size 3x3 is given by considering the below arrangement of pixels about the pixel *[i, j]*

$$\begin{matrix} a_o & a_1 & a_2 \\ a_7 & [i, j] & a_3 \\ a_6 & a_5 & a_4 \end{matrix}$$

The partial derivatives of the Prewitt operator are calculated as

$$G_x = (a_2 + ca_3 + a_4) - (a_0 + ca_7 + a_6) \qquad (3.10)$$

$$G_y = (a_6 + ca_5 + a_4) - (a_0 + ca_1 + a_2) \qquad (3.11)$$

The constant *c* implies the emphasis given to pixels closer to the center of the mask. G_x and G_y are the approximations at *[i, j]*.

Setting c = 1, the Prewitt operator is obtained. Therefore the Prewitt masks are as follows

$$G_x = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

These masks have longer support. They differentiate in one direction and average in the other direction, so the edge detector is less vulnerable to noise.

3.1.3 SOBEL EDGE DETECTOR

The Sobel edge detector is very much similar to the Prewitt edge detector. The difference between the both is that the weight of the center coefficient is 2 in the Sobel operator. The partial derivatives of the Sobel operator are calculated as

$$G_x = (a_2 + 2a_3 + a_4) - (a_0 + 2a_7 + a_6) \tag{3.12}$$

$$G_y = (a_6 + 2a_5 + a_4) - (a_0 + 2a_1 + a_2) \tag{3.13}$$

Therefore the Sobel masks are

$$G_x = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

Although the Prewitt masks are easier to implement than the Sobel masks, the later has better noise suppression characteristics.

3.1.4 FREI-CHEN EDGE DETECTOR

The Frei-Chen edge detector is also a first order operation like the previously discussed operators. Edge detection using the Frei-Chen masks is implemented by mapping the intensity vector using a linear transformation and then detecting edges based on the edges based on the angle between the intensity vector and its projection onto the edge subspace. Frei-Chen edge detection is realized with the normalized weights.

Frei-Chen masks are unique masks, which contain all of the basis vectors. This implies that a 3x3 image area is represented with the weighted sum of nine Frei-Chen masks. Primarily the image is convolved with each of the nine masks. Then an inner product of the convolution results of each mask is performed.

The Frei-Chen are

$$G_1 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & -\sqrt{2} & -1 \end{bmatrix} \quad G_2 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 1 & 0 & 1 \\ \sqrt{2} & 0 & \sqrt{2} \\ 1 & 0 & 1 \end{bmatrix} \quad G_3 = \frac{1}{2\sqrt{2}} \begin{bmatrix} 0 & -1 & \sqrt{2} \\ 1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix}$$

$$G_4 = \frac{1}{2\sqrt{2}}\begin{bmatrix} \sqrt{2} & -1 & 1 \\ -1 & 0 & 1 \\ 0 & 1 & -\sqrt{2} \end{bmatrix} \quad G_5 = \frac{1}{2}\begin{bmatrix} 0 & 1 & 0 \\ -1 & 0 & -1 \\ 0 & 1 & 0 \end{bmatrix} \quad G_6 = \frac{1}{2}\begin{bmatrix} -1 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & -1 \end{bmatrix}$$

$$G_7 = \frac{1}{6}\begin{bmatrix} -1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad G_8 = \frac{1}{6}\begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix} \quad G_9 = \frac{1}{3}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

The first four Frei-Chen masks above are used for edges and the next four are used for lines and the last mask is used to compute averages. For edge detection, appropriate masks are chosen and the image is projected onto it. The projection equations are given by

$$\cos e = \sqrt{\frac{M}{S}} \tag{3.14}$$

where $M = \sum_{k\in\{e\}} (f_s, f_k)^2$ and $S = \sum_{k=1}^{9} (f_s, f_k)^2$

3.2 LAPLACIAN OF GAUSSIAN (LOG)

The principle used in the Laplacian of Gaussian method is, the second derivative of a signal is zero when the magnitude of the derivative is maximum. The Laplacian of a 2-D function *f(x, y)* is defined as

$$(\nabla^2 f)(x, y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{3.15}$$

3.2.1 THE LOG OPERATOR

The two partial derivative approximations for the Laplacian for a 3x3 region are given as

$$\nabla^2 f = 4(a_8) - (a_1 + a_3 + a_5 + a_7) \qquad (3.16)$$

$$\nabla^2 f = 8(a_8) - (a_0 + a_1 + a_2 + a_3 + a_4 + a_5 + a_6 + a_7) \qquad (3.17)$$

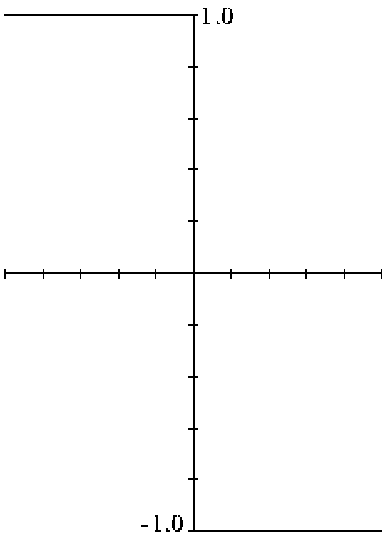
The masks for implementing these two equations are as follows

$$G_x = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \qquad G_y = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

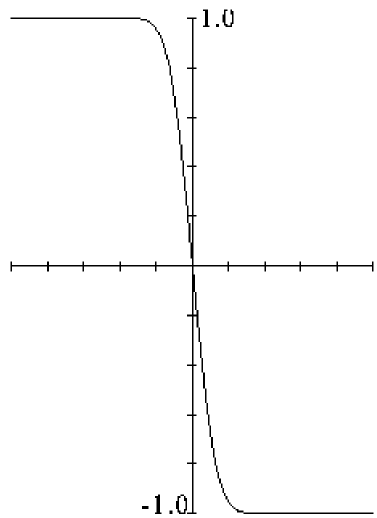
The above partial derivative equations are isotropic for rotation increments of 90^0 and 45^0, respectively. Edge detection is done by convolving an image with the Laplacian at a given scale and then mark the points where the result have zero value, which is called the zero-crossings. These points should be checked to ensure that the gradient magnitude is large. Marr and Hildreth develop this method.

Marr and Hildreth method

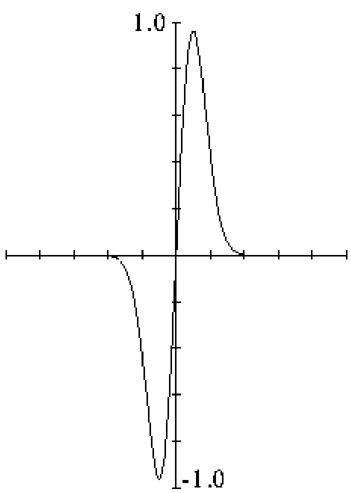
The edge pixels in an image are determined by a single convolution operation. The basic principle of this method is to find the position in an image where the second derivatives become zero. These positions correspond to edge positions. The Gaussian function firstly smoothens or blurs any edge as shown in the figure 3.1. Blurring is advantageous here because Laplacian would be *infinity* at unsmoothed edge and therefore edge position is still preserved. LOG operator is still susceptible to noise, but by ignoring zero-crossings produced by small changes in image intensity can reduce the effects of noise.



(a) *f(x)*



(b) Gaussian smoothing function of *f(x)*



(c) Laplacian of Gaussian of *f(x)*

Figure 3.1: The smoothing of a signal with a Gaussian function

LOG operator gives edge direction information as well as edge points, determined from the direction of the zero-crossing. Hence the purpose of the Gaussian function in the LOG formulation is to smooth the image and the purpose of the Laplacian operator is to provide an image with zero crossings used to establish the location of edges. Some of the disadvantages of LOG are, the LOG being a second derivative operator the influence of noise is considerable. It always generates closed contours, which is not realistic. The Marr-Hildreth operator will mark edges at some locations that are not edges.