Edge Detection Analysis

Tzu-Heng Henry Lee
Graduate Institute of Communication Engineering, National Taiwan University,
Taipei, Taiwan, ROC
E-mail: r96942133@ntu.edu.tw

Abstract

Edges of an image are considered a type of crucial information that can be extracted by applying detectors with different methodology. This research paper presents a brief study of the fundamental concepts of the edge detection operation, theories behind different edge detectors, and some simple self-written Matlab edge detection functions with the simulation results. Previous works on edge detection models are reviewed and simulated. The Matlab results coincide with the first and second order derivative edge detection models.

1. Introduction

Edge detection is a type of image segmentation techniques which determines the presence of an edge or line in an image and outlines them in an appropriate way [1]. The main purpose of edge detection is to simplify the image data in order to minimize the amount of data to be processed [2]. Generally, an edge is defined as the boundary pixels that connect two separate regions with changing image amplitude attributes such as different constant luminance and tristimulus values in an image [1], [3], [4]. The detection operation begins with the examination of the local discontinuity at each pixel element in an image. Amplitude, orientation, and location of a particular subarea in the image that is of interest are essentially important characteristics of possible edges [1]. Based on these characteristics, the detector has to decide whether each of the examined pixels is an edge or not. Frei and Chen [1] suggest that edge detection is best carried out by simple edge detector, followed by a morphological thinning and linking process to optimize the boundaries. This paper gives an overview of first and second order derivative edge detections, edge fitting detection model as well as the detector performance evaluation. Also, several Matlab functions that underlie the

principle of first and second order derivative edge detection techniques are written. The result of the simulations were analyzed and compared to the theoretical result of the edge detectors introduced in [4]. By writing simple edge detection Matlab functions, one can have a better understanding of the various edge detection algorithms developed in the past.

2. First Order Derivative Edge Detection

There are two methods for first order derivative edge detection. **1) One of the methods is evaluating the gradients generated along two orthogonal directions.** An edge is judged present if the gradient of the image exceeds our defined threshold value, t = T. The gradient can be computed as the derivatives along both orthogonal axes

$$G(x,y) = \frac{\partial F(x,y)}{\partial x}\cos\theta + \frac{\partial F(x,y)}{\partial y}\sin\theta \tag{1}$$

The gradient is estimated in a direction normal to the edge gradient. The spatial average gradient can be written as

$$G(j,k) = \sqrt{\left[G_R(j,k)\right]^2 + \left[G_C(j,k)\right]^2} \tag{2}$$

A simplest discrete row and column gradient is given by

$$G_R(j,k) = F(j,k) - F(j,k-1) \tag{3}$$
$$G_C(j,k) = F(j,k) - F(j+1,k) \tag{4}$$

Running the difference of the contiguous pixels in horizontal and vertical directions is found inefficient since the edges cannot be delineated and the detector is quite sensitive to small fluctuations. Diagonal edge gradients proposed by Roberts is shown as

$$G_1(j,k) = F(j,k) - F(j+1,k+1) \tag{5}$$
$$G_2(j,k) = F(j,k+1) - F(j+1,k) \tag{6}$$

Z1	Z2	Z3
Z4	Z5	Z6
Z7	Z8	Z9

Fig. 1 The convention for 3 by 3 edge detection operator

Robert's model is still susceptible to fluctuations in the image even though the edges can be properly positioned. Prewitt has developed another edge gradient detector which uses a different approach to approximate row and column edge gradients. The proposed gradients are defined as

$$G_R(j,k) = \frac{1}{K+2}[(z_3 + K \cdot z_6 + z_9) - (z_1 + K \cdot z_4 + z_7)] \quad (7)$$

$$G_C(j,k) = \frac{1}{K+2}[(z_1 + K \cdot z_2 + z_3) - (z_7 + K \cdot z_8 + z_9)] \quad (8)$$

The equations above follow the convention shown in Fig. 1. The K in the equations is equal to one, so that row and column gradient are normalized to provide unit gain positive weighted and unit gain negative weighted averages about a separated edge position. Sobel edge detector doubles the north, south, west, and east pixels of the Prewitt operator (i.e. K=2). This makes the Sobel edge detector more sensitive to diagonal edge than horizontal and vertical edges [4]. Frei and Chen [1] have adapted the Sobel's model and proposed a pair of isotropic operator which makes K equal to $\sqrt{2}$. This makes the gradient for horizontal, vertical, and diagonal edges the same at the edge center. The isotropic smoothed weighting operator proposed by Frei and Chen can easily pick up subtle edge detail and produce thinner edge lines, but it also increase the possibility of erroneously detect noise as real edge points. In [5], Ding analyzed the one-dimensional outputs of a general edge detector using differentiation method. The 1-D outputs reveal that differentiation method is quite susceptible to noise and unable to accurately detect step edges interfered by noise and ramp edges. Pratt [4] mentioned that properly extending the size of the neighborhoods over which the differential gradients are computed can alleviate the inability to detect the edges precisely in a high noise environment. The first order derivative edge detectors do provide solutions to edge detection process but none of the detectors can localize the edge to a single pixel.

2) The second approach of first order derivative edge detection is utilizing a set of discrete edge templates [1] with different orientations. This method is to convolute an image, $F(j, k)$ with a set of template gradient impulse response arrays, $H_m(j, k)$. The general form of the edge template gradient is

$$G(j,k) = MAX\left[\left|G_1(j,k)\right|,.....,\left|G_m(j,k)\right|,.....\left|G_M(j,k)\right|\right] \tag{9}$$

where

$$G_m(j,k) = F(j,k) \otimes H_m(j,k) \tag{10}$$

The edge angle is determined by the direction of the largest gradient. The direction in that particular template is not the exact orientation of the edge. In fact, the direction is only an approximation. The exact orientation is $\pm\pi/4$ of the orientation that gives the maximum gradient. The following equation shows a directional gradient proposed by Kirsch:

$$G(j,k) = \underset{i=0}{\overset{7}{Max}}\left[\left|5S_i - 3T_i\right|\right] \tag{11}$$

where

$$\begin{aligned} S_i &= A_i + A_{i+1} + A_{i+2} \\ S_i &= A_{i+3} + A_{i+4} + A_{i+5} + A_{i+6} + A_{i+7} \end{aligned} \tag{12}$$

Please note that the subscripts A_i are the parameters in the compass gradient matrix. 3-level and 5-level impulse response arrays are the other two 3 by 3 templates proposed by Robinson. Nevatia and Babu have developed the gain-normalized 5 by 5 masks that can be used to detect edges in various degree increments. The larger template size will result in finer quantization of the edge orientation angle, and less noise. The tradeoff is that more computational power will be required [4].

2.1 Threshold Selection

The edge is detected by comparing the edge gradient to a defined threshold value. This threshold represents the sensitivity of the edge detector. When dealing with noisy edges, one could miss valid edges while creating noise-induced false edges. Edge detection can be represented by the following conditional probability densities

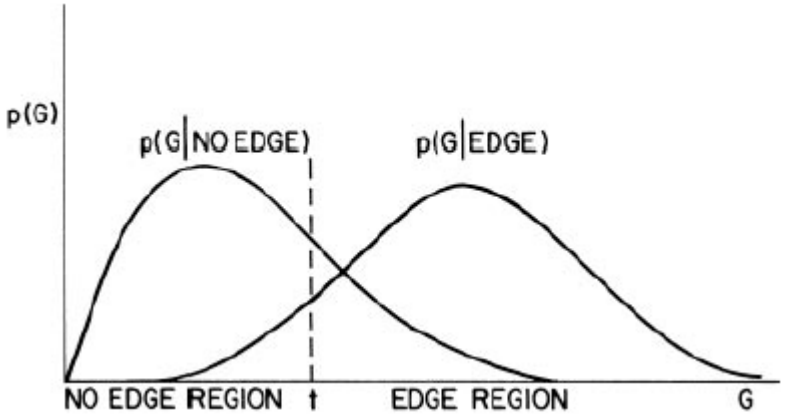

Fig. 2 Conditional probability densities of edge gradients [6].

of $G(j, k)$:

$$P_D = p(G \geq t \mid edge) = \int_t^{\infty} p(G \mid edge)dG$$
$$P_F = p(G \geq t \mid no-edge) = \int_t^{\infty} p(G \mid no-edge)dG \tag{13}$$

where P_D and P_F represent the probability of correct detection and the probability of false edge detection respectively. Also, the t is denoted as the detection threshold. Fig. 2 exhibits the conditional probability densities of edge gradient that vary in edge and non-edge regions. The probability of misclassification can be represented as

$$P_E = [1 - P_D]P(edge) + [P_F]P(no-edge) \tag{14}$$

According to Heyman-Pearson test, a threshold t is chosen to minimize P_F for a fixed P_D. An ideal threshold must produce minimum error P_E. This condition can be achieved if the following maximum likelihood ratio test associated with the Bayes minimum error decision rule of classical decision theory is satisfied [6]:

$$\frac{P(G \mid edge)}{P(G \mid no-edge)} \geq \frac{P(no-edge)}{P(edge)} \tag{15}$$

The conditional densities for 2 by 2 and 3 by 3 edge detection operators were derived by Abdou. The densities apply when the width of a ramp edge is one (w=1) and additive Gaussian noise is present. However, Reliability of the stochastic edge

Table 1. The relation of $G(x, y)$ with $F(x, y)$ [4]

	$F(x, y)$ is constant	$F(x, y)$ is changing linearly	*Rate of change of* $F(x, y)$ is increasing
Behavior of $G(x, y)$	Zero	Zero	Sign changes at the point of reflection of $F(x, y)$. (Indicates the presence of an edge.)

model and analytic difficulties in deriving the edge gradient conditional densities are two difficulties when we are determining the optimal threshold for our edge detector. Abdou and Pratt have developed an approach based on pattern recognition techniques. Their design produced a table which lists the optimal threshold value for several 2 by 2 and 3 by 3 edge detectors and the probability of correct and false edge detection. When edges and non-edges are equally probable, P_F equals $1 - P_D$. The edge detection threshold should be inversely proportional to SNR (Signal-to-Noise Ratio) [4].

3 Second Order Derivative Edge Detection

If there is a significant spatial change in the second derivative, an edge is detected [4]. The following sub-sections introduce different approaches using second order derivative on edge detection:

A. Laplacian Generation in Continuous and Discrete Domain

Since the Laplacian is

$$\nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} \tag{16}$$

the edge Laplacian of an image $F(x, y)$ in the continuous domain can be written as

$$G(x, y) = -\nabla^2 \{F(x, y)\} \tag{17}$$

The negative sign gives the zero crossing of $G(x, y)$ a positive slope for an edge detected. Table 1 shows the behavior of $G(x, y)$ relative to $F(x, y)$. Computing the difference of slopes along each axis, as shown in the equation below, is the simplest

way to approximate the continuous Laplacian in discrete domain.

$$\begin{aligned} G(j,k) = &\left[F(j,k) - F(j,k-1)\right] - \left[F(j,k+1) - F(j,k)\right] \\ &+ \left[F(j,k) - F(j+1,k)\right] - \left[F(j-1,k) - F(j,k)\right] \end{aligned} \tag{18}$$

The convolution operation

$$G(j,k) = F(j,k) \otimes H(j,k) \tag{19}$$

with the two arrays

$$H = \begin{bmatrix} 0 & 0 & 0 \\ -1 & 2 & -1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & -1 & 0 \\ 0 & 2 & 0 \\ 0 & -1 & 0 \end{bmatrix} \tag{20}$$

or

$$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{21}$$

can generate this four–neighbor Laplacian. The gain normalized version of the previous impulse response is

$$H = \frac{1}{4}\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{22}$$

The gain normalized eight-neighbor Laplacian impulse response array proposed by Prewitt is

$$H = \frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{23}$$

In a separable eight-neighbor Laplacian, the difference of slopes is averaged over three rows and three columns. This is given by

$$H = \begin{bmatrix} -1 & 2 & -1 \\ -1 & 2 & -1 \\ -1 & 2 & -1 \end{bmatrix} + \begin{bmatrix} -1 & -1 & -1 \\ 2 & 2 & 2 \\ -1 & -1 & -1 \end{bmatrix} \tag{24}$$

The gain-normalized version is

$$H = \frac{1}{8}\begin{bmatrix} -2 & 1 & -2 \\ 1 & 4 & 1 \\ -2 & 1 & -2 \end{bmatrix} \tag{25}$$

B. Laplacian of Gaussian (LoG) Edge Detection in Continuous and Discrete Domain

According to the Laplacian of Gaussian edge detector operator proposed by Marr and Hildrith, Gaussian-shaped smoothing is applied prior to the application of the Laplacian. The LoG gradient in continuous domain can be written as

$$G(x,y) = -\nabla^2 \left\{ F(x,y) \otimes H_s(x,y) \right\} \tag{26}$$

where

$$H_s(x,y) = g(x,s)g(y,s) \tag{27}$$

is the impulse response of the Gaussian smoothing function:

$$g(x,s) = \left[2\pi s^2 \right]^{-\frac{1}{2}} \exp\left\{ -\frac{1}{2}\left(\frac{x}{s} \right)^2 \right\} \tag{28}$$

Due to the linearity of the second derivative operation and of the linearity of convolution, we can express the gradient as

$$G(x,y) = F(x,y) \otimes H(x,y) \tag{29}$$

and the impulse response is

$$H(x,y) = -\nabla^2 \left\{ g(x,s)g(y,s) \right\} = \frac{1}{\pi s^4}\left[1 - \frac{x^2+y^2}{2s^2} \right] \exp\left\{ -\frac{x^2+y^2}{2s^2} \right\} \tag{30}$$

To obtain LoG operator in discrete domain, one can simply sample the impulse response $H(x, y)$ in the continuous domain over a W x W window. To avoid any negative truncation effects, W should be greater or equal to $3c$, where c is $2\sqrt{2}s$, the centre positive part of the LoG function [4].

C. Directed Second Order Derivative Generation

There are two approaches that involve second order derivative generation to detect edges. The ability to precisely detect the edge direction is the major advantage of the directed second order derivative. Equation (31) displays the directed second order derivative in continuous domain with an edge angle θ:

$$F''(x,y) = \frac{\partial^2 F(x,y)}{\partial x^2}\cos^2\theta + \frac{\partial^2 F(x,y)}{\partial x \partial y}\sin\theta\cos\theta + \frac{\partial^2 F(x,y)}{\partial y^2}\sin^2\theta \tag{31}$$

An easier approach to detect edge is to determine the edge direction using first order derivative method before taking the approximation to the equation (31) in discrete domain [4].

Haralick proposed an approach called facet modeling which approximates continuous $F(x, y)$ in discrete domain using a 2-D polynomial shown in equation (32):

$$\hat{F}(r,c) = k_1 + k_2 r + k_3 c + k_4 r^2 + k_5 rc + k_6 c^2 + k_7 rc^2 + k_8 r^2 c + k_9 r^2 c^2 \tag{32}$$

The estimated edge angle can be represented as

$$\theta = \tan^{-1}\left[\frac{k_2}{k_3}\right] \tag{33}$$

Through this approach, the directed second order derivative can be computed analytically. In [4], Pratt suggested that "in principle any polynomial expansion can be used in the approximation"; therefore, the quadratic expansion form of equation (32) is presented as

$$\hat{F}(r,c) = \sum_{n=1}^{N} a_n P_n(r,c) \tag{34}$$

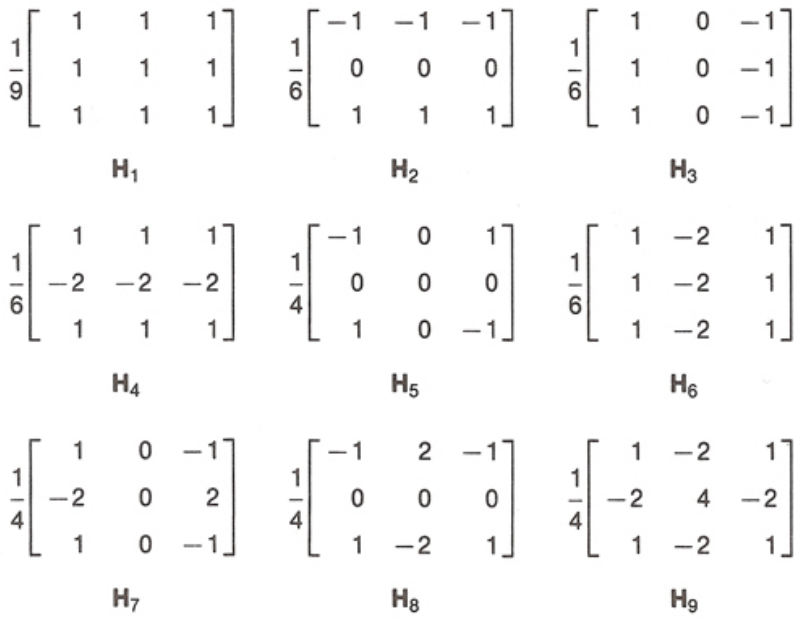

Fig. 3 Nine 3 by 3 impulse response arrays based on Chebyshev polynomial [4].

As a result of the linear property of the approximated weighting coefficient a_n, the weighting coefficient A_n (j, k) at each point of the image $F(j, k)$ can be found by convolution

$$A_n(j,k) = F(j,k) \otimes H_n(j,k) \tag{35}$$

The following figure shows the nine Chebyshev polynomial 3 by 3 impulse response arrays.

4. Edge Detection Using Edge Fitting Method

The image data of a real edge could be similar to the ideal edge model in either one-dimensional or two-dimensional aspects. Edge fitting detection, however, requires more computation in comparison with derivative edge detection techniques. In the one-dimensional edge fitting model as shown in Fig. 4, the actual image $f(x)$ is fitted to an ideal step function. The 1-D ideal step function is defined as

$$s(x) = \begin{cases} a & x < x_o \\ a + h & x \geq x_o \end{cases} \tag{36}$$

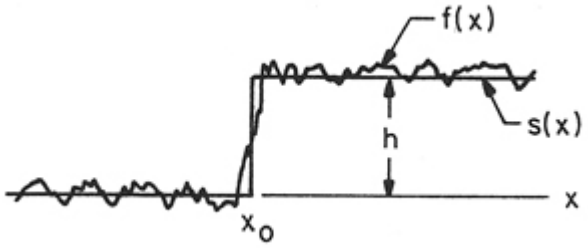

Fig. 4 One-dimensional edge fitting model [4].

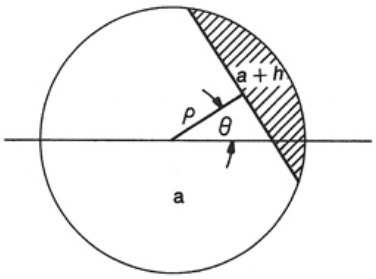

Fig. 5 Two-dimensional edge fitting model [4].

The two-dimensional ideal step function is

$$S(x,y)=\begin{cases} a & (x\cos\theta + y\sin\theta) < \rho \\ a+h & (x\cos\theta + y\sin\theta) \geq \rho \end{cases} \tag{37}$$

Fig. 5 demonstrates the two-dimensional edge fitting model [4].

5. Edge Detector Performance Evaluation

It is quite difficult to develop standard performance criteria and methods to evaluate the effectiveness of each edge detector. Locating a real edge pixel becomes extremely crucial. Edge slope angle and its spatial orientation are also important criteria in the evaluation. A good edge detector must have a good edge decision which the closeness of fit between the actual and the detected image is optimized [4].

5. 1 Edge Detection Probability

When we determine the performance of an edge detector, the probability of correct

detection P_D and the probability of false edge detection P_F play a key role. Both probabilities are displayed in equation (38) and (39).

$$P_D = \int_t^\infty p(G \mid edge) dG \tag{38}$$

$$P_F = \int_t^\infty p(G \mid no - edge) dG \tag{39}$$

Pratt [4] plotted the probability of correct edge detection against probability of false detection and made a comprehensive comparison of several edge detectors. Based on his plots, he found that Sobel and Prewitt 3-by-3 operators are superior to the Roberts 2-by-2 operator and the performances of Sobel and Prewitt differential operators are slightly better than the Robinson 3-level and 5-level operators.

5. 2 Edge Detection Orientation and Localization

The sensitivity to edge orientation and the ability to localize an edge are both important properties of an edge detector. Pratt [4] plotted the edge gradient as a function of actual edge orientation and concluded that square root combination of orthogonal gradients is superior to the magnitude combination of the orthogonal gradients. He also sampled continuous ramp edge and examined the edge displacement from the center of the first order derivative operator. Fig. 6 contains the analysis performed to determine the edge detector's ability of edge localization. Pratt's analysis also reveals that all the edge detectors except Kirsch operator have the same edge displacement property which the edge displacement increases as the edge gradient amplitude decreases. Similar properties are possessed by variable size boxcar operators and several orthogonal gradient operators. By setting the threshold to half or higher of the edge height, edge location can be properly localized. Setting a high threshold could cause the detector to miss the real edge with low amplitude [4].

5. 3 Edge Detector Performance Characterization

Failure to detect real edges, misclassification of noise-induced points as edge points, and inability to localize edge points are the three major errors that could be made by edge detectors. The probability of the true edge detection can be found by comparing the detected image with the edge maps resulted from an ideal edge

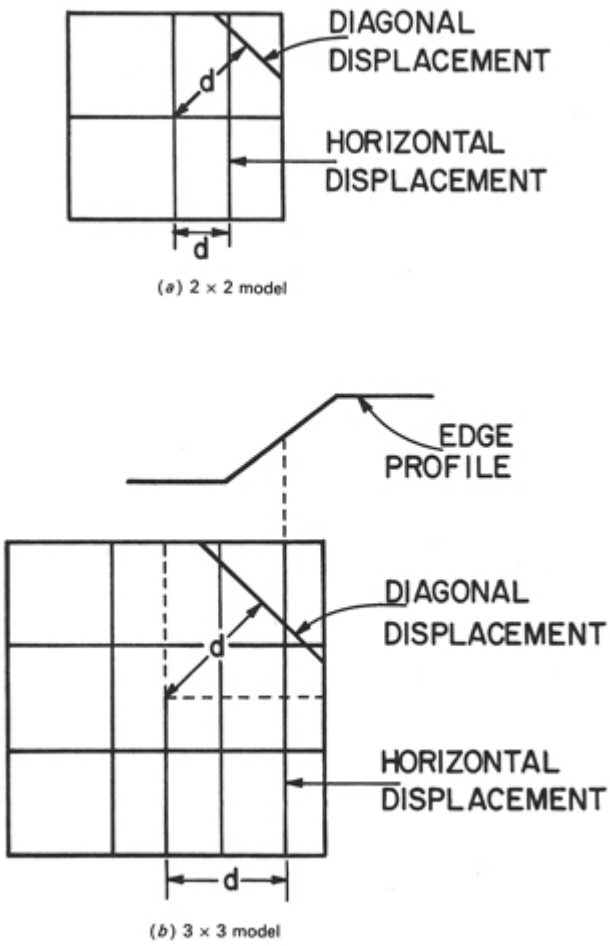

Fig. 6 Edge localization analysis for (a) 2-by-2 model and (b) 3-by-3 model [4].

detector. A figure of merit introduced by Pratt [6] is written as

$$R = \frac{1}{I_N} \sum_{i=1}^{I_A} \frac{1}{1 + ad^2} \tag{40}$$

In equation (40), $I_N = MAX(I_I, I_A)$ and I_I and I_A represent the number of ideal and actual edge map points. "a" is a scaling constant and d is the separation distance of an actual edge point normal to a line of ideal edge points. A rating factor, R that equals to one means the edge is precisely detected. The value of scaling factor, a, reflects the performance on edge localization. A smeared edge is easier to be fixed than an offset edge since a smeared edge can be thinned by morphological post processing operation.

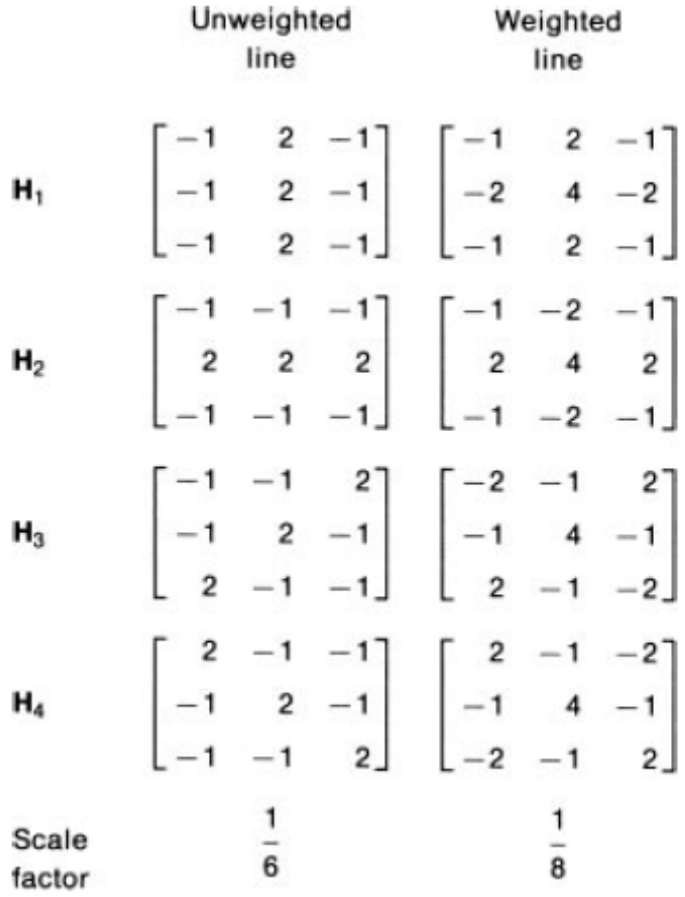

Fig. 7 The 3 by 3 Line Detector Impulse Response [4].

6. Color Edge Detection

Three tristimulus values T1, T2, and T3 can be used to quantify the amount of RGB colors at each pixel of a color image. Several different definitions of color edge detection have been proposed. A definition states that the detection depends on the vector sum gradient of the three tristimulus values:

$$G(j,k) = \left\{ \left[G_1(j,k) \right]^2 + \left[G_2(j,k) \right]^2 + \left[G_3(j,k) \right]^2 \right\}^{\frac{1}{2}} \qquad (41)$$

Each gradient in (41) represent the three tristimulus component values. A color edge is detected if the gradient exceeds the defined threshold.

7. Line and Spot Detection

The approach introduced by Pratt [4] for the unit width line and spot detection can be achieved by finding a line gradient

$$G(j,k) = \underset{m=1}{\overset{4}{MAX}} \left\{ \left| F(j,k) \otimes H_m(j,k) \right| \right\} \tag{42}$$

The $H_m(j,k)$ could be one of the weighted or unweighted 3 by 3 line detector impulse response shown in Fig. 7. For spot detection, a spot gradient displayed in equation (43) is used to detect unit width step spots.

$$G(j,k) = F(j,k) \otimes H(j,k) \tag{43}$$

There several ways to implement the impulse response operator in the above equation. One of the approaches is using the Laplacian operators that are display in the equation (22), (23), and (25). These operators are thresholded for spot detection, but they could end up with detecting the false spots in a noisy image. Prewitt has developed another operator

$$\mathrm{H} = \frac{1}{8}\begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \tag{44}$$

which only detects diagonally oriented edges. By using the operator proposed by Prewitt, the noise-induced false spot detections

8. Improved Algorithms on Edge Detection

There are several algorithms proposed to resolve the noise problem and enhance the detector ability to detect ramp edges:

A. Split Gaussian Function

An edge detection algorithm that uses a split Gaussian function suggested by Argyle [7] involves a convolution of the image to be processed and a split Gaussian

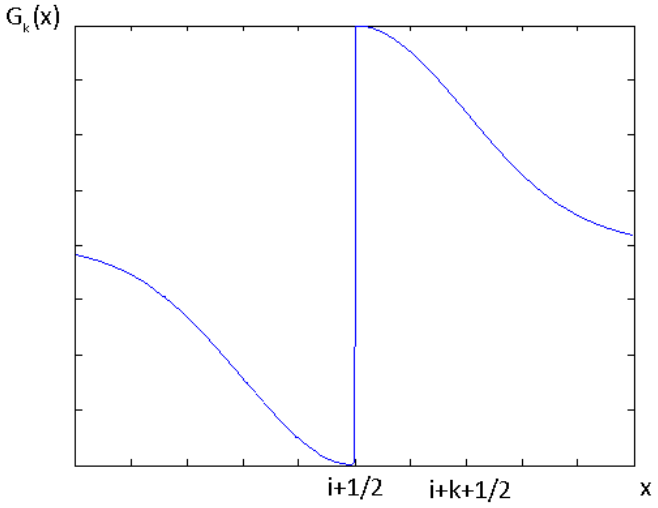

Fig. 8 Split Gaussian Function[7].

$$d_k(i) = a_k f(x) \otimes G_k(x) \tag{45}$$

where a_k is a normalizing factor. The split Gaussian function $G_k(x)$ is defined by

$$G_k(x) = \frac{\operatorname{sgn}(x - i - \frac{1}{2})}{\sqrt{2\pi}k} \exp\left(\frac{-\frac{1}{2}(x - i - \frac{1}{2})^2}{k^2}\right) \tag{46}$$

An example of a split Gaussian function in Fig. 8 displays that only one discontinuity exists in the function and according to Argyle, the function would minimize the generation of noise since the function is an odd decaying function that would not produce any unnecessary noise at both decaying ends of the function. The only concern about this method is that the gradual decaying part of the function would still produce some noise after convolution.

B. Hilbert Transform Method

In [5], Hilbert Transform (HLT) is mentioned as a way to detect edges while reducing the effect of noise. The HLT is defined as

$$g_H(\tau) = h(x) \otimes g(x) \xrightarrow{FT} G_H(f) = H(f)G(f) \tag{47}$$

where $h(x)$ is $1/(\pi x)$. The Fourier transform of $h(x)$ turns out to be a sign function

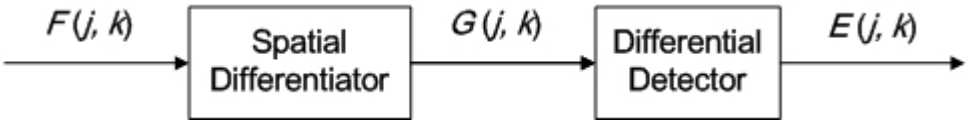

Fig. 9 Differential Detection Process [4].

$$H(f) = -j\,\mathrm{sgn}(f) \tag{48}$$

The long impulse response of HLT greatly improved its ability to detect ramp edges and reduce the undesired noise.

C. Short Response Hilbert Transform Method

Ding and his colleagues [5] proposed a compromised edge detector between the differential and the HLT edge detection techniques and it is called SRHLT. This paper will not discuss this approach in detail. The detailed derivation and algorithm of the SRHLT and the resulting plots are analyzed in [5].

9. Matlab Simulations for Edge Detection

General differential edge detection process contains 2 fundamental elements: spatial differentiator and differential detector. A spatial differentiator take the original image $F(j, k)$ as an input and produce an output differential image $G(j, k)$. The differential image $G(j, k)$ is the spatial amplitude changes between the pixels in a defined direction. After the spatial differentiation process, a differential detection operation is performed to determine the pixel locations of the significant differentials [4]. A block diagram in Fig. 9 illustrates the process of differential edge detection.

9.1 Simple Edge Detectors

Edge1, a Matlab function for simple edge detection, was written based on the differential edge detection process. The edge1 function allows two inputs, f and t. The input f is the image to be processed which is shown in Fig. 10 and t is a defined edge detection threshold. The function first computes the row and column gradients as shown in equation (49) and (50):

$$G_C(j,k) = F(j+1,k) - F(j,k) \tag{49}$$
$$G_R(j,k) = F(j,k+1) - F(j,k) \tag{50}$$

Then the spatial gradient amplitude is calculated using the equation (2). Instead of using differential detector, edge1 function directly compare the spatial gradient to the defined threshold input by the function user. Through the comparison, a binary indicator map is generated indicating the position of edges detected within the original image. Fig. 11 displays an example binary map produced by this function and the binary image obtained is quite satisfactory considering the edge1 algorithm is only a simple approximation of row and column gradients. A similar function, edge2, calculates the spatial differential in both orthogonal directions. The Gradient equations in both orthogonal directions are defined as

$$G_1(j,k) = F(j+1,k+1) - F(j,k) \tag{51}$$
$$G_2(j,k) = F(j,k+1) - F(j+1,k) \tag{52}$$

Again, the spatial gradient amplitude is computed as the following square root form:

$$G(j,k) = \sqrt{\left[G_1(j,k)\right]^2 + \left[G_2(j,k)\right]^2} \tag{53}$$

An example binary map was generated in Fig. 12. This figure demonstrates the strength of orthogonal edge detector because by comparing Fig. 12 to Fig. 11, the resolution of the orthogonal edges in Fig. 12 is improved. In both Fig. 11 and Fig. 12, the undesired edge thickness reveals that neither of the outcomes of edge1 and edge2 functions can precisely position the edge within only a few pixels.

According to the Laplacian Generation in discrete domain, the simplest way to approximate the continuous Laplacian in discrete domain is displayed in equation (54):

$$\begin{aligned} G(j,k) = &\left[F(j,k) - F(j,k-1)\right] - \left[F(j,k+1) - F(j,k)\right] \\ &+ \left[F(j,k) - F(j+1,k)\right] - \left[F(j-1,k) - F(j,k)\right] \end{aligned} \tag{54}$$

A Matlab program edge3 was written to simulate the Laplacian approximation. Fig. 13 shows an example binary map generated by edge3 function. The figure demonstrates the advantageous edge locating ability of second order derivative edge

Fig. 10 The original input image.

Fig. 11 The binary indicator map generated by the approximation to row and column gradients with t = 0.05.

Fig. 12 The binary indicator map generated by the approximation to orthogonal gradients with t = 0.05.

Fig. 13 The binary image generated by the Laplacian approximation with t = 0.03.

detector; the strong edges are detected as thinner lines. However, the image also shows the common drawbacks of the second order derivative edge detectors such as the high sensitivity to noise and the inability to detect edge directions.

9.2 Two-Dimensional Edge Fitting Model Simulation

Another self-written Matlab function, edgefit, based on the two-dimensional step function described in (37) is written to generate a 2-D step edge fitting matrix, *s(x, y)*. By convolution of the image and the edge fitting matrix, one can obtain a binary edge image and examine the notable characteristics of this model. The function algorithm starts with the generation of two 5-by-5 simple step function matrix which are defined as

$$x = \begin{bmatrix} -2 & -2 & -2 & -2 & -2 \\ -1 & -1 & -1 & -1 & -1 \\ 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 \end{bmatrix} \tag{55}$$

and

$$y = \begin{bmatrix} -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \\ -2 & -1 & 0 & 1 & 2 \end{bmatrix} \tag{56}$$

The boundary definition of the 2-D model

$$x\cos\theta + y\sin\theta < 0 \tag{57}$$

then utilizes the two matrix to generate a 5-by-5 2-D edge fitting matrix. The θ defined in (57) determines the edge direction that is the same with the orientation of the polar distance ρ shown in Fig. 5. An example edge fitting matrix with $\theta=\pi/4$ is

(a) (b)

(c)

Fig. 14 The edge map produced with (a) $\theta = \pi/4$ (b) $\theta = -\pi/4$ (c) a combination of (a) and (b).

$$s(x,y) = \begin{bmatrix} 1 & 1 & 1 & 1 & -1 \\ 1 & 1 & 1 & -1 & -1 \\ 1 & 1 & -1 & -1 & -1 \\ 1 & -1 & -1 & -1 & -1 \\ -1 & -1 & -1 & -1 & -1 \end{bmatrix} \tag{58}$$

Fig. 14 illustrates the edge map by using various values of θ. Fig. 14a shows that the edge fitting model with $\theta = \pi/4$ is only sensitive to the edges with the same

(a) (b)

Fig. 15 (a) The edge map before morphological post processing. (b) The edge map image after morphological post processing. The single isolated edge pixels are deleted as indicated.

orientation. The same case works in Fig. 14b with $\theta = -\pi/4$. The edge fitting model with $\theta = -\pi/4$ missed a quite large amount of edge details in the opposite orientation. A combined edge map from Fig. 14a and Fig. 14b are displayed in Fig. 14c. The missing edge details in Fig. 14a and Fig. 14b can be clearly identified by a comparison of the edge maps in Fig. 14.

9.3 Morphological Image Processing Function

In [4], Pratt mentioned that by using the morphological majority black operation, one can remove the noise-induced image. A simple Matlab program was written to remove the single isolated edge pixels in an edge map.

Fig. 15b shown below is produced by performing Matlab function “morph1”. Fig. 15b shows that the morphological processing applied significantly reduced the noise-induced dots. The “morph1” function employs a 3 by 3 mask filter as shown in Fig. 16. The purpose of applying this mask is that the isolated noise dots (i.e. edge detected pixels which the values of 8 surrounding pixels are 0) can be eliminated.

0	0	0
0	1	0
0	0	0

Fig. 16 The 3 by 3 mask filter deployed in “morph1” function.

10. Conclusions

Various edge detection algorithms and detector design methods have been described and discussed in this paper. The binary edge maps produced by the Matlab programs that simulate the approximated version of first order derivative edge detectors revealed the detectors’ inability to localize the edges within only a few pixels. Expanding the size of impulse response operators can mitigate but cannot eliminate the noise effects. The resulting edge maps produced by the approximation to the second order derivative edge detection indicate that this model does accurately position the real edges. However, the problem of this approach is the high sensitivity to the image noise. Although in section 9.3, a morphological post processing program is written to delete the isolated noise pixels, more post-processing improvements could be done. Further research on morphological processing is beyond the scope of this research paper.

The edge detector performance criterion and methods of evaluation provides us a good understanding on possible ways of finding out the effectiveness of each developed detection model. Meanwhile, the improved algorithms pointed out in section 8 are proved to be partially effective in precise ramp edge detection and reduction of noise-induced edges. The major research directions that can be followed and improvements to be made in the future edge detection techniques are categorized in the following categories:

1) Image noise reduction.
2) Precise edge detections with a minimum error detection possibility.
3) Accurate edge localization that can detect edges within a single pixel.
4) More sophisticated algorithms and models on morphological image process

11. References

[1] W. Frei and C. Chen, "Fast Boundary Detection: A Generalization and New Algorithm," *IEEE Trans. Computers*, vol. C-26, no. 10, pp. 988-998, Oct. 1977.

[2] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, Vol. 8, No. 6, pp. 679-698, Nov. 1986.

[3] R. C. Gonzalez and R. E. Woods, *Digital Image Processing*. Upper Saddle River, NJ: Prentice-Hall, 2001, pp. 572-585.

[4] W. K. Pratt, *Digital Image Processing*. New York, NY: Wiley-Interscience, 1991, pp. 491-556.

[5] J. J. Ding, S. C. Pei, J. D. Huang, G. C. Guo, Y. C. Lin, N. C. Shen, and Y. S. Zhang, "Short response Hilbert transform for edge detection," *CVGIP*, 2007.

[6] I. E. Abdou and W. K. Pratt, "Quantitative design and evaluation of enhancement/thresholding edge detectors," *Proceedings of the IEEE*, vol.67, no.5, pp. 753-763, May 1979

[7] E. Argyle; A. Rosenfeld, "Techniques for edge detection," *Proceedings of the IEEE*, vol.59, no.2, pp. 285-287, Feb. 1971

[8][1] Harris and M. Stephens, "A Combined Corner and Edge Detector," *Proceedings of The Fourth Alvey Vision Conference*, Manchester, pp. 147-151, 1988

[9][2] S. C. Pei and J. J. Ding, "Improved Harris' algorithm for corner and edge detections," accepted by *ICIP 2007*.

[10][3]J. Canny, "Finding Edges and Lines in Images," Massachusetts Institute of Technology, Cambridge, Massachusetts, USA, Tech. Rep. no. 720, 1983.

[11][4]Marr and E. Hildrith, "Theory of Edge Detection," *Proc. Royal Society of London*, B207, pp.187–217, 1980.

[1,2,3,4] These journal articles are not cited within this article. These are my recommended references for future research.