A Qualitative Profile-based Approach to Edge Detection

by

Ting-jen Yen

A dissertation submitted in partial fulfillment of the requirements for the degree of Doctor of Philosophy

Department of Computer Science

New York University

September 2003

Dissertation Advisor

To my parents

Acknowledgment

I thank my research advisor, Professor Chee K. Yap, for his role in inspiring this project as well as for his ongoing guidance, and careful reading of several drafts of my dissertation.

I would also like to express my appreciation to Professor Davi Geiger and Professor Ernest Davis for reading my dissertation and making helpful comments that greatly improved the presentation and enhanced the clarity of this work. I would also like to thank Professor Demetri Terzopoulos and Professor Dennis Shasha for serving on my committee on a very short notice.

I also would like to thank all my friends during my graduate school years, although I can only mention a few here. Among them are Ee-Chien Chang, Hseu-Ming Chen, Li Chen, Huachie Lee, Horng Li. I am especially grateful to Hseu-Ming Chen, for helping me in more ways than he can probably imagine; in particular, for his advice and answers to many of my questions during my study.

And finally, I thank my parents Shih-Hung Yen and Lan-Je Hsieh, as well as my sister Ting-ju Yen, for their support throughout my PhD study. I could not finish the degree without their encouragement.

Abstract

Edge detection is a fundamental problem of computer vision and has been widely investigated. We propose a new framework for edge detection based on edge profiles.

Our model, based on one-dimensional qualitative edge profile fitting and edge consistency, will produce one continuous edge from an initial seed point. A "profile" is defined as a finite cross-section of a two-dimensional image along a line segment. "Edge consistency" means that all the profiles on the same edge should be consistent.

Appropriate evaluation functions are needed for different types of edge profiles, such as step edges, ramp edges, etc. An evaluation function must meet the requirement that it will produce local minima at the positions where edges of a given type occurs in the profile. Instead of subjective thresholding, image noise is measured statistically and used as a systematic way of filtering false edges. We describe our method as "qualitative edge profile fitting" because it is not based on arbitrary thresolding. Once an edge point is localized, it can be extended into an edge by matching compatible profiles. Two profiles are considered compatible as long as their average difference is within the noise measurement. Another feature of our approach is its subpixel accuracy. The utilization of profiles and noise-induced threshold selection make tasks such as joining broken edges more objective.

We develop the necessary algorithms and implement them. Different evaluation functions are constructed for different edge models and experimented on different

one-dimensional profiles. The edge detector, using these evaluation functions, is then examined using different images and under different noise conditions.

Contents

List of Figures

Chapter 1

Introduction

Edges in an image provide a representation of object boundaries within that image. Therefore, edge detection is an essential component in many computer image processing operations such as stereopsis, calibration, motion analysis and recognition. From a pixel level perspective, edges can be viewed as the regions of an image where the image values undergo a sharp variation. Normally, such regions form lines, curves and contours, which represent outlines of solid objects, marks on surfaces, and shadows. Moreover, line drawings are common and suggestive images for humans. Notice that image noise too causes intensity variations. Noise is among the most significant obstacles of edge detection. Another factor that complicates the edge detecting operation is digitization. A wide edge in a digital image would have the appearance of a staircase, and might be interpreted as multiple edges. In this thesis, we introduce a new approach of edge detection which employs minimization of the evaluation of profiles in edge localization, and consistency of an edge in edge linking process.

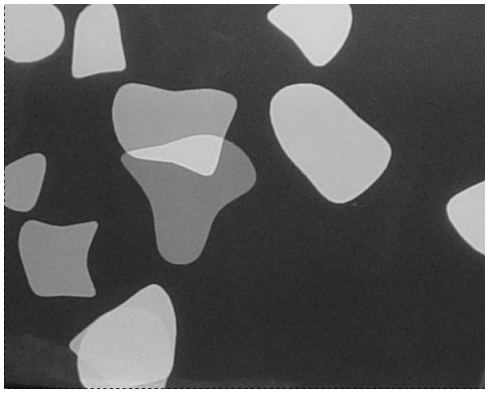
1.1 Problem Definition

In spite of our familiarity with the concept of an **edge**, there is no widely accepted rigorous definition for it. More precisely, the concept of edges we use is abstract, and can have different meanings in different contexts. Therefore, different edge detectors can produce edges in different forms of representation, while each detector can be accepted as a genuine edge detector. For example, in the edge detecting techniques mentioned in previous section, an edge could be considered as a pixel with local discontinuity in image intensity, a contour that connects such pixels and forms a outline of object, or a boundary that separates two or more objects. Although various definitions of an "edge" could be accepted in human perception, their complexity varies, and the way how edges are processed and represented in computers are all different.

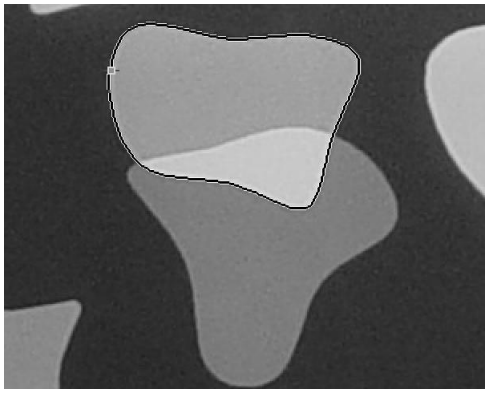
Most of the edge detectors only produce points at the positions of edges of images. The results generally consist of a quantitive value for each pixel, and an orientation. Normally, a threshold is applied to determine whether a pixel is on an edge or not, and results in a binary image indicating the position of the edges of the original image. Usually the results are examined by human eyes by comparing against the original image to determine if the outputs are acceptable or not. Nevertheless, the application of these detectors are limited, First, the points are not linked, therefore, we do not have any relation among the points. Such linking process can by performed, for example, Hough transform[19, 26]. However, except for straight edges, the linking process is far from perfect. More precisely, if the purpose of edge linking is to join edge points into groups, and eliminate isolated points which are not related to another points, in other words, a part of

noise reduction process, then the existing linking methods are adequate. However, if the purpose is to find an edge as complete as possible, or to distinguish different edges, then none of them is a good candidate, since few of the edge linking methods use any characteristics of the edge points other than their positions, orientations and strengths.

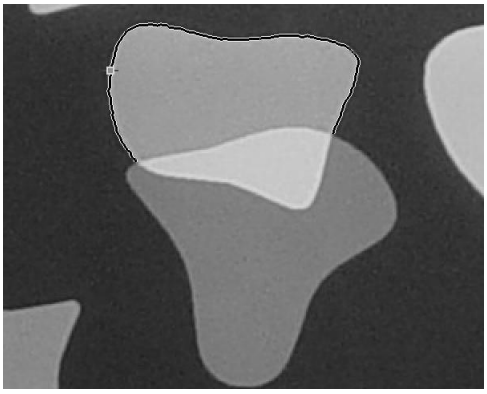
Even if all the points are linked into curves properly, these curves alone are still not sufficient for most of higher level image processing operations. These edge curves do not have such information since the curves are composed by edge points only, no color information is included. There is no guarantee that all the linked edge points really belong to the same edge curve without additional verification. Such verification is not trivial, especially with the presence of image noise, and alignment. Of the images in Figures 1.1 and 1.2, we demonstrate the reason why profile of edges should be considered during edge linking process. One of the images is the result of Canny's detector, with hysteresis threshold for linking. The other is obtained by using profile-based edge linking process we developed. For comparison, only one edge is detected starting from the same initial point. In Figure 1.1, it is arguable that the two results, while different, are both reasonable. In Figure 1.2, the result from the linker without profile-consistency is less desirable. While the Canny's detector does not focus on edge linking, since its hysteresis threshold is more about noise elimination than about edge linking, these figures manifest the merit of using profiles in edge linking process. Generally, for edge linking with any junctions, this kind of arbitrariness will always happen, unless the background is well defined. Someone might claim that the result of Canny's detector is more desirable than ours, since it does describe the outline of a floor tile. The problem



(a) An image of random blobs



(b) Canny's output of part of the image with blobs



(c) Profile-based detector's output of part of the image with blobs

Figure 1.1: Comparing the results of edge linkers with and without consideration of profile, using a synthetic image.

(a) An image of a box on floor

(b) Canny's output of part of the image

(c) Profile-based detector's output of part of the image

Figure 1.2: Comparing the results of edge linkers with and without consideration of profile, using a real image.

is that the linking process could just as reasonably have picked up the outline of the adjacent tile. Instead of defining a better linking mechanism at the junctions, we will simply enforce the consistency of the profiles and reply on a higher level linking mechanism to decide on forming edges with non-constant profiles.

Another problem we address in this thesis is thresholding. Edge detectors generally give a quantitive value for each pixel as the strength of edges. Some techniques such as **non-maximum suppression** or **zero-crossing** may reduce the necessity of threshold. Nevertheless, with the presence of noise, threshold is still required to distinguish real edges from noise. Normally, a threshold value is found using a trial-and-error process and then applied to all edges in the given image. However, since the threshold depends on the edge characteristics as well as the detector been used [12, 52, 54], it is not easy to find a single threshold value for an image, even when such value does exist, which is not always true. While the threshold value is detector-dependent, few detectors come with a threshold selecting method. Although robust threshold selection algorithms do exist, many of them can be applied to image segmentation using statistical characteristics of the image such as the histogram. Therefore it may not work well for the output of the edge detectors. Moreover, a single optimal threshold value may not exist, and different threshold values may be needed for different regions of the same image. In the end, unless there is some measurement of performance of an arbitrary threshold value, such that the best threshold value can be found by its optimal performance, the decision of the threshold value will have to depend on the judgment of human eyes. One of our goals is to eliminate the usage of thresholding, or at least provide some mechanism that would generate a reasonable threshold value from the information

gathered locally from the image.

1.2 Terminology

As mentioned, different edge detectors use different definitions of "edges". Therefore, a precise definition of how the term **edge** is used in this thesis is as follows.

(a) Ramp Profile (b) Ridge Profile

Figure 1.3: (a) Ramp Profile (b) Ridge Profile

For the purposes of this introduction, we will assume that an "edge" is, among other things, associated with a curve γ that represents a similar form of discontinuity on an image. We call γ an **edge curve**. Mathematically, we may assume a curve is a continuous, piecewise smooth function $\gamma : [0, 1] \to \mathbb{R}^2$. The minimal information that we expect in the output of a true edge detector is a collection of edge curves in an image.

One of our main motivations is to insist that an integral part of the concept of an edge is its "profile". For instance, at the simplest level, a curve should only be regarded as an "edge" when it possesses a consistent profile of some sort of one-dimensional edge. Intuitively, a profile is a one-dimensional image, and a curve determines a family of such profiles obtained as the local cross section of the image

along a normal to the curve. For instance, many edge detectors are tuned to detect the step profile, or more generally, a **ramp profile**. Some other detectors are specific for **line edges** which can be viewed as **ridge profiles**. See Figure 1.3 for these two common types of profiles.

For the present discussion, we use the following simple definition for a **profile**: a profile is a C^1 function $\phi : [-a, b] \to \mathbb{R}$, where a, b are positive reals that define the boundaries of the profile. Generally, a and b are set to an initial constant value T, though each could be adjusted to a more suitable boundary value later in the process. More parameters may be added to the profile. Although these additional parameters may provide better description of an edge, they are simply derived information from ϕ. Oftentimes, the term "profile" in this thesis referring the C^1 function unless stated otherwise. If a profile ϕ_0 is called an **edge profile**, that means $\phi_0(0)$ is the center of an edge. For a step edge, the center of the edge means the position with the discontinuity of color/grayscale values. Otherwise, a **transient region** is present between two regions with constant color/grayscale values. The two regions will be called **left stable region** and **right stable region** respectively. The center of this edge will be the middle of this transient region. The "left" and "right" of the stable regions are relative terms depending on the orientation of the profile sampled from the image, and can be interchanged as long as needed.

By using profiles, we could convert a two-dimensional edge detecting problem into a one-dimensional signal processing problem which has been studied longer and more thoroughly in the past. We can also use the profiles to classify different edges into categories. Also, with this definition of **edges**, if we can find all edges

in an image, we can usually use these edges to represent the whole image.

We will assume that the edge curve γ is piecewise smooth, so that for all except finitely many values of t, the **normal direction** $\mathbf{n}(t)$ at $\gamma(t)$ is well-defined. More precisely, if $\gamma(t) = (x(t), y(t))$ then $\mathbf{n}(t) = (-y'(t), x'(t))$, assuming that γ is parameterized by arc length. Let $\theta(t) = \arctan -x'(t)/y'(t) \in [0, \pi)$ be the orientation of $\mathbf{n}(t)$. The direction perpendicular to the normal direction is called **edge direction**, which is defined by the orientation $\arctan y'(t)/x'(t)$. For any $t_0 \in [0, 1]$, we call the pair $(\gamma(t_0), \theta(t_0))$ an **edge point**. In general, an **oriented point** is a pair (p, θ) where $p \in \mathbb{R}^2$ and $\theta \in [0, \pi)$. Thus, an edge point is simply an oriented point that is "derived" from an edge. Just as an edge has its profile, an oriented point can have a profile. The profile of an oriented point comprises of a collection of sampled pixel values from an image, at the position of the oriented point along its orientation. And if an oriented point has a profile that fits the criteria of an edge profile, the oriented point is an edge point. The profile of an edge is in fact the collective term of the profiles of all the edge points on that edge.

Many edge detectors actually only produce edge points for a given image, some of them without the orientation information. In order to distinguish our work from these detectors, we will name these detectors **edge point detectors**.

we need to define more terms to be used later. The **neighbor edge points** are the oriented points that are on both sides of the **edge direction** of a selected edge point at some specific distance d away. For example, for an edge point $\gamma_0(t_0)$ from the edge $\gamma_0(t)$, its neighbor edge points are $(x(t_0) \pm d \sin \mathbf{u}(t_0), y(t_0) \pm d \cos \mathbf{u}(t_0))$. They are used to locate the real **edge points** that are adjacent to the edge point. In our work, d is chosen between one to three pixels. If it is too small, the interference

between an edge point and its neighbor edge points would be too large to locate a new edge point accurately. On the other hand, if d is too large, we may miss some sharp corners.

1.3 Thesis overview

In this thesis, we describe a profile-based edge detecting technique to produce edges with sub-pixel accuracy. One of our goals is to produce enough abstract information of edges that could be used to represent an image. There are three key concepts in our approach: edge profile localization, edge profile verification, and edge continuity.

"Edge profile localization" is the process that the candidate edge point is selected based on the profiles from the neighborhood of an initial oriented point. An evaluation function is applied to these profiles, and the profile with a local minimum value will be chosen as the candidate edge point.

"Edge profile verification" means that for every candidate edge point, a verification process is performed to either validate it or reject it as noise. This verification process can be part of the evaluation function, or another independent function.

"Continuity of edges" means that once an edge point is labeled, we could "grow" the point into a curve by repeatedly extending the edge along the edge directions by finding new oriented points with profiles similar to the original edge point. The growth of the edge curve will continue until the profile of the new extended point does not match the original one.

The development of our edge detecting algorithm is a three-step process. The first step of our edge detection is to create edge point profile for a given oriented point (x, y, θ). Next we change its position and direction by $(t\cos\theta, t\sin\theta, \delta)$, where

$t \in [-T, T], \delta \in [-\Delta, \Delta]$; T and Δ are some small predefined values. Our goal is to get the profiles of these shifted oriented points, and find the locally optimized position and direction of the edge point with these profiles. The detail of this step is covered in Chapter 3.

Chapter 4 describes the second step which would grow an edge from an edge point derived from the first step. With the normal direction of the edge point, the position and normal direction of a nearby edge point can be estimated, and the result of the growing by the profiles of these edge points will then be fine-tuned and verified. This step is repeated until the new edge points on both ends of the edge are rejected. Additional processing can be performed after the growth stopped. For example, if two endpoints of the edge are very close, it is possible to close the edge as a loop. If an endpoint is close to an endpoint of another edge, there is possibility that these two edges can be joined if the profiles of the two edges match. Moreover, a smoothing process can be performed over the edge to damp the effect of digitalization.

The third step is to find all the edges from the image. This is done by traversing the image. For every unvisited point, we will try to find an edge point and grow an edge out of it. Precaution should be taken since our algorithm detects edge point positions at subpixel level, tracing the visited points is no longer trivial. In our test program, the combination of a sorted linked list and an accumulating array is used to emulate a hash table that keeps track of all the detected edge points.

After the mentioned three steps, certain post-processing can be applied. For example, the edges can be sorted by their edge magnitudes, and edges can be further categorized according to their edge types, or the similarity of the profiles.

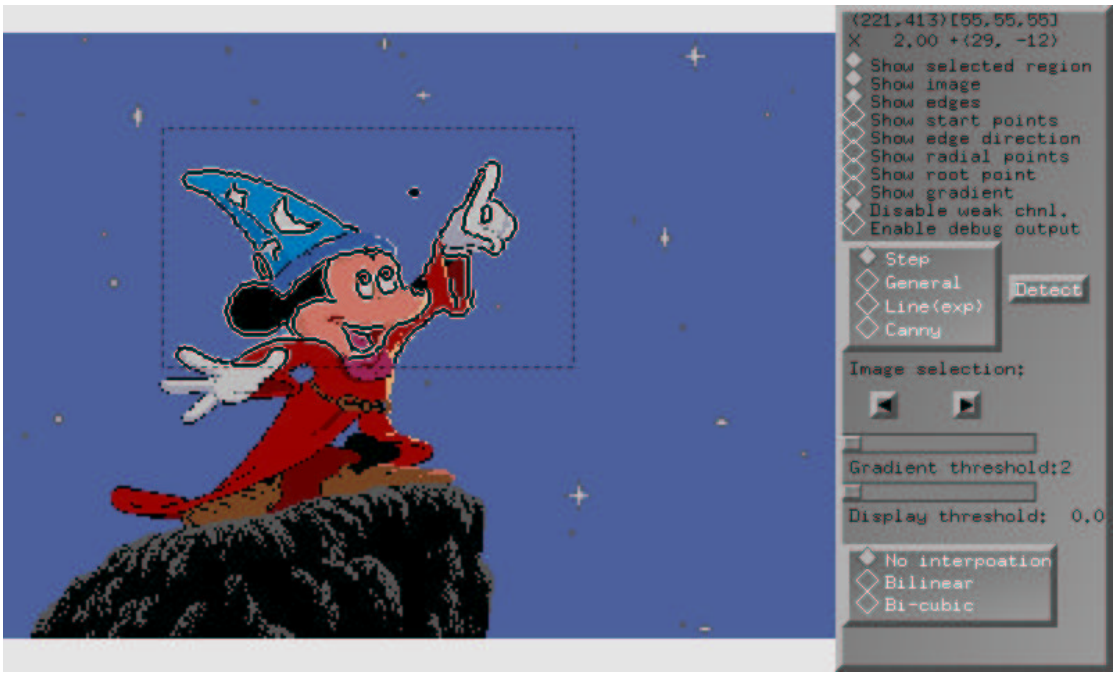


Figure 1.4: The interface of the software developed. A bounding box is used to limit the region for edge detecting process.

The post-processing tends to be application-dependent, so we will only discuss this part briefly.

1.4 Software and Data Availability

The software that is to achieve the results in this thesis, the experimental data, as well as the various tools developed to visualize and analyze them, are available on the web at

http://cs.nyu.edu/visual/yentj/dissertation/

This software is developed for visualizing images and edge detection results as well as capturing these results for printing. Zooming and panning are supported

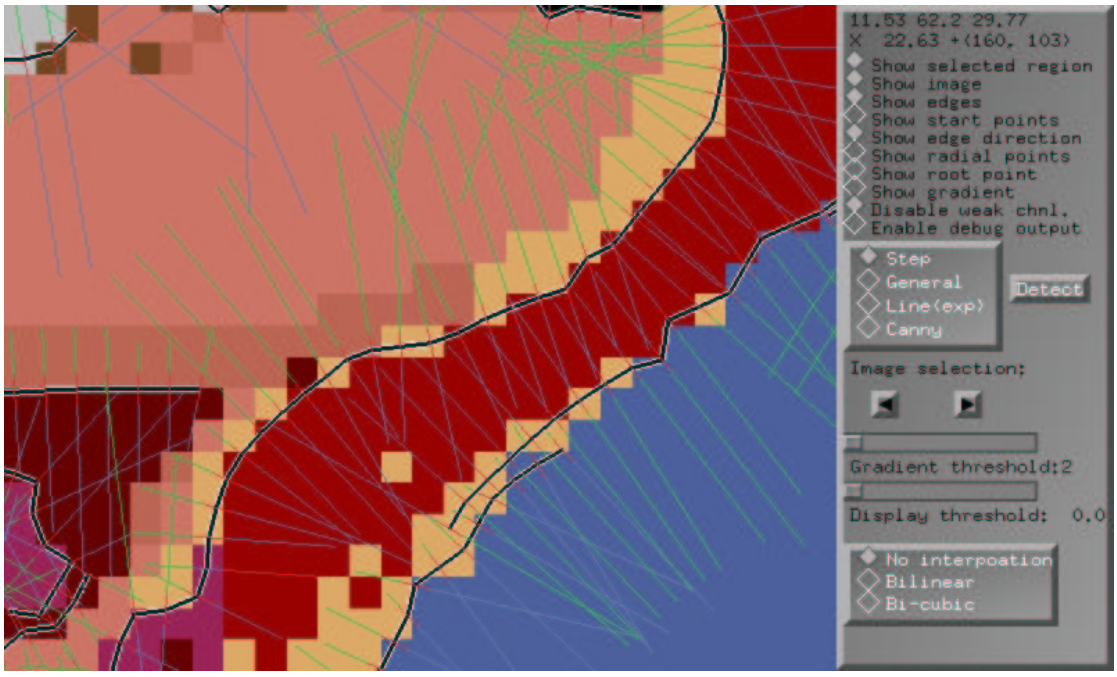


Figure 1.5: The interface of the software developed. The same image as in Figure 1.4 is used, with different resolution. Position and orientation of each edge point is shown as well.

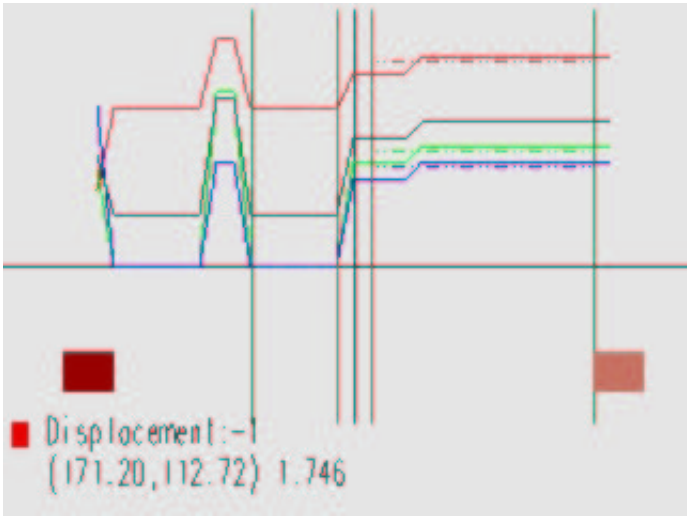


Figure 1.6: An example of the window showing the profile information.

so the detail in subpixel level can be observed. An additional window is used to display the profile information of a selected edge point. The main interface is shown in Figures 1.4 – 1.5, and the window displaying the profile information is shown in Figure 1.6.

Although the official version of this thesis is in black and white, the full color version of most of our images are available in the electronic copy of the thesis in PDF format, which is available at the following URL:

http://cs.nyu.edu/visual/yentj/dissertation/thesis.pdf

Chapter 2

Background

2.1 Properties of Edge Detectors

As mentioned in the previous chapter, edge detection is an essential component in many computer image processing operations. During the history of image processing, many edge detectors have been introduced for different purposes, using different approaches. Still, edge detection can be separated into three stages. These three stages are:

Smoothing. At this stage, image noise is removed as much as possible, without damaging the real edges too much. Generally, smoothing is achieved by filtering the image function with a low-pass filter. It could reduce the additive noise since noise is generally high frequency signal. On the other hand, the edges are also high frequency signals, thus will be removed as well. A smoothing process is a trade-off between information preservation and noise reduction. A parameter is usually associated with a smoothing operator to control the scale of the smoothing.

Edge enhancement. An edge enhancement filter is applied to the image. The requirement of the filter is that it should generate some specific response at the positions of edges, Edges are generally high frequency signals. Thus a high-pass filter is usually used to localize the edges. A differential operator is usually used as such filter.

Edge localization. Identify the edges, according to the type of the edge-enhancing filter applied. When a first-derivative operator is used, local maxima are expected. For a second-derivative operator, the zero-crossings are marked. This stage also determines which responses are caused by noise and most be removed. Thresholding techniques are generally used for such tasks.

2.2 Survey of Edge Detectors

Some of the earliest works of edge detection employ small convolution masks to approximate either the first derivative or the second derivative of an image; for example, Roberts filter, Sobel filter, Prewitt filter, and Laplacian filter. They focus on the "edge enhancement" part of edge detection, with none or very little "smoothing". A threshold is then applied to the output of these filters to identify the edge points. These filters, though easy to implement and generally with the advantage of speed over later edge detectors, provide very little control over smoothing and edge localization, by which noise is reduced. Therefore, these filters are very noise-sensitive.

Marr and Hildreth [34, 18] have proposed the use of zero-crossings of the Laplacian of Gaussian. Using the fact that a step edge corresponds to a sharp change

in the image, the first derivative of the image should have an maximum at the position corresponding to an edge in the image, and so the second derivative should be zero at the same position. Obvious, it is easier to find a zero-crossing than a local maximum in a two-dimensional function. On the other hand, the higher-order derivatives are also more sensitive to noise. In order to reduce noise, the image has to be smoothed. When choosing a smoothing filter, two criteria should be fulfilled [34]. First, the filter should be smooth and roughly band-limited in the frequency domain to reduce the number of frequencies at which image function changes can take place. Secondly, the constraint of spatial localization requires the response of a filter to be from nearby points in the image. These two criteria are conflicting, and the Gaussian filter

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{2.1}$$

is an compromise between spatial and frequential criteria. The edge detection operator is the second derivative of a smoothed two-dimensional image function $f(x, y)$, which is

$$\nabla^2(G(x, y; \sigma) * f(x, y)). \tag{2.2}$$

It is usually called **LoG** as an abbreviated of **Laplacian of Gaussian** where the Laplacian operator

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

is the second-derivative operator which can be represented numerically as the linear operator

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}.$$

The order of differentiation and Gaussian smoothing is interchangeable because of the *derivative property of convolution* [1]. Therefore, the operator in Equation (2.2) can be rewritten as

$$(\nabla^2 G(x, y; \sigma)) * f(x, y). \tag{2.3}$$

The filter now can be written as $\nabla^2 G$.

Poggio et al. [46] introduced a cubic spline filter. The image is smoothed by a cubic spline filter before differentiation. It was shown that the results of the cubic spline filter is very similar to Gaussian smoothing.

Canny described a widely used edge detecting algorithm [7] which is optimal to step edges corrupted by noise. Three criteria were defined for the optimality edge detection:

Good Detection. The detector must minimize the probability of false edges caused by noise, as well as missing real edges.

Good Localization. The edges detected must be as close as possible to the true edges.

Single Response. The detector must return one point only for each true edge point.

These criteria and their variations are also widely used in other research. First two criteria are then developed quantitatively into a set of functions with mini-

[1] **Derivative property of convolution**. If the signal $x(t)$ has an ordinary first derivative $\dot{x}(t)$, the convolution $x(t) * v(t)$ has an ordinary first derivative,

$$\frac{d}{dt}\left[x(t) * v(t)\right] = \dot{x}(t) * v(t) = x(t) * \dot{v}(t).$$

mal and maximal constraints. A closed form solution was found using variational calculus. Without the third criterion, the optimal detector for a step edge

$$G(x) = \begin{cases} 0 & x < 0 \\ A & x \geq 0 \end{cases}$$

is $f(x) = -G(x)$ in $[-W, W]$, assuming the filter width is $2W$. Therefore, the optimal, one-dimensional step edge detector is a truncated step. Unfortunately, this filter contains a very high bandwidth and tends to produce many maxima with noisy step edges. If the third criterion is added, the optimal solution may be found by numerical method. The result filter can be approximated with error less than 20% by the first derivative of a Gaussian smoothing filter. This is similar to Marr-Hildreth edge detector [34] which is based on the Laplacian of a Gaussian. The detector is then generalized to two-dimensions. **Non-maximum suppression** is then applied on the results of the filter to thin wide edges in order to produce 1-pixel wide edges. It is done by finding local maxima in the direction perpendicular to the edges. Finally, weak edges are removed using thresholding. The thresholding is applied with hysteresis. Edges contours are processed as units, and two threshold values are defined; each contour must have at least one point with gradient magnitude above the higher threshold, while all points in the contours must not go below the lower threshold. Canny also proposed **feature synthesis** which is a multiple-scale approach where the standard deviation σ of the Gaussian are used as the scaling factor. All significant edges from the operator with the smallest scale are marked first, and the edges for larger scale are synthesized from these marked edges and then compared to the actual detector output. Additional edges are marked only if they have a significantly stronger response than that predicted

by the synthetic response. Many researchers were inspired by Canny's work. For example, Deriche [13] extended Canny's initial filter to two-dimensions and implemented it using recursive filtering. Petrou and Kittler [45] derived another optimal detector for a blurred step edge model using criteria similar to those by Canny.

Canny's **feature synthesis** is not the first attempt to detect edges using different scale parameters. Marr and Hildreth [34] has suggested to obtain a description of an image at different scales by applying a feature detector at different scales and combining the edge information. If one creates a series of images $I_t(x, y, t)$ from original image $I_0(x, y)$ by convolving $I_0(x, y)$ with a Gaussian kernel $G(x, y; t)$ with variance t, as pointed out by Koenderink [29] and Hummel [23], this family of images can be viewed as the solution of the isotropic diffusion equation

$$\frac{dI}{dt} = \nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

with the initial condition $I(x, y, 0) = I_0(x, y)$, the original image. One disadvantage of linear isotropic diffusion is that the diffusion would blur features as well as smooth noise. An isotropic diffusion method was introduced by Perona and Malik [44]. A nonlinear diffusion equation is used:

$$\frac{dI}{dt} = \nabla \cdot (c(x, y, t)\nabla I) = c(x, y, t)\nabla^2 I + \nabla c \cdot \nabla I \tag{2.4}$$

where $c(x, y, t)$ is chosen as

$$c(x, y, t) = g(\nabla I(x, y, t)) \tag{2.5}$$

and g is a nonnegative monotonically decreasing function with $g(0) = 1$. A different function g would generate different scale-spaces, though they turn out to be perceptually similar. This way, the points with strong features, i.e., with high

$|\nabla I|$ values, would have less diffusion effect than other points, thus retain most of the features in the image. To make the diffusion anisotropic, the function $c(x, y, t)$ should be directional and is perpendicular to the orientation of the gradient of the image. The nonlinear diffusion methods process an image with different smoothing parameters such that it could blur noise and at the same time keep the edges sharp.

Another approach of edge detection is parametric fitting. This involves fitting the image with an edge model with parameters, then finding the parameters that minimize the fitting error. The Hueckel edge detection [22, 20, 21] is an example of the parametric fitting method. A two-dimensional step edge model in a circular window is fitted to each image pixel. The parameters of the edge model are the gray scale values, the edge orientation, and the distance from the center of the window to the edge. An edge is detected if the fit is accurate enough, while the accuracy of edge fitting is measured in terms of the mean square error. Hueckel introduced a polar Fourier expansion and used the first eight Fourier coefficients in the fitting process in order to reduce the computation. Experimental results indicated that Hueckel operator performs well in noisy and highly textured environments, though no analysis of the operator in presence of noise is documented.

Some other detectors that can be categorized as parametric fitting approach are proposed by O'Gorman [40], Hummel [23], Tabatabaui and Mitchell [51], Hartley [17], Nalwa and Binford [38], and Lyvers et al. [32]. Moreover, this approach is also used to detect other features. Hueckel [21] used the same model to detect line edges, and Rohr [49] and Parida et al. [31] used it to detect corners. Since this approach uses a rich description of the image structure, edge detectors using this approach could provide edge attributes such as position with subpixel accuracy, contrast, blur,

and width. In other words, this approach could provide more thorough description of the edges. One problem of this approach is that it is not easy to provide a two-dimensional edge model with an arbitrary-shaped curve. Most edge detectors use a straight edge model. For some complex features such as a corner, or more than one edges within the window, the detectors can still pick the accurate parameter set that minimizes the edge fitting error. Therefore the value of the edge fitting error will be increased, and the edge might be rejected once the error exceeds the default threshold.

Some edge detecting techniques convert the edge detecting problem into energy functions, then find the optimal edges by minimizing the energy functions. Two examples of such techniques are **deformable contours** [27], and **the Mumford-Shah theory** [37].

The deformable contour technique employs energy minimization to shift an initial line segment, which is called the **deformable contour**, the **active contour**, or the **snake**, into a curve that would adhere to an edge of the given image. For edge detecting purposes, two forms of energy are involved: an *internal energy* that generates a force keeping the curve continuous and smooth, and an *image energy* that move the curve towards edge points on the image. The Euler equation is used to minimize the energy function. When the energy is minimized, the two forms of energy are at balance. Therefore, the deformable contour is attached to an edge, and is smoothed so it is not zigzag due to digitization or noise. Deformable contours are usually used in motion analysis, and stereopsis, not in single stationary image. This is because this technique requires a proper initial line segment or closed curve to begin with. Without some external input, this technique does not work for a

single image. However, in a motion pictures, or in an image pair, outlines of objects can usually be detected by finding the difference of two images. Those outlines can be used as the initial line segments for deformable contours. There are different extensions of this model, include [42, 43, 1].

Mumford and Shah[37] proposed another form of energy minimizing to detect boundary of noisy images. For a given image $g : R \to \mathbb{R}$, the energy functional E for a boundary B and smoothed image f is

$$E(f,B) = \mu^2 \int_R \underbrace{(f-g)^2}_{\text{Data Fidelity}} + \int_{R-B} \underbrace{\|\nabla f\|^2}_{\text{Smoothness Constraint}} + \underbrace{\mu\nu|B|}_{\text{Edge Penalty}} \tag{2.6}$$

where R is the region of the image, $f : R \to \mathbb{R}$ is the smoothed ideal image, $B : R \to \{0,1\}$ is the binary edge process, μ and ν are scalar parameters. $|B|$ is the volume of B. For two-dimensional images, $|B|$ is the length of the edges and $R - B$ denotes the region R excluding the edges.

The Mumford-Shah energy $E(f,B)$ formalizes a tradeoff between noise removal and edge detection. However, this functional lacks a practical means to find the smoothed image f, and the binary edge process B that minimize the energy function $E(f,B)$. Many methods have been proposed to simplify the minimization of the functional. In one of them, [2], the binary edge process, B, of the energy function (2.6) is replaced by a continuous edge field. The energy functional would become

$$E(f,v) = \int_R \left\{ \mu \underbrace{(f-g)^2}_{\text{Data Fidelity}} + \underbrace{\alpha(1-v)^2\|\nabla f\|^2}_{\text{Smoothness Constraint}} + \underbrace{\frac{\beta}{2}\|\nabla v\|^2 + \frac{v^2}{2\beta}}_{\text{Edge Penalty}} \right\} \tag{2.7}$$

which is called the Ambrosio-Tororelli functional. The $g : R \to \mathbb{R}$ denotes the given image, $f : R \to \mathbb{R}$ the piecewise smoothed image, $v : R \to [0,1]$ the corresponding

continuous edge strength of the image, R the image domain, and α, β and μ are scalar parameters. This functional could be minimized using the Euler equation.

2.3 Edge Linking

Of all the mentioned edge detecting methods, most of them yield only the *points* at the positions of edges on an image, instead of *curves*. The only exceptions are from the parametric fitting methods, and active contour model. To link the points into curves, an additional stage should be added to the three-step process of edge detection: **edge linking**. The purpose of edge linking is to group the edge points into continuous curves, using certain criteria such as closeness or some specific models such as straight lines or circles.

The method of linking edge points by closeness is straightforward. Generally, all adjacent points are joined together into curves, or tree structures. There are some other edge linking methods that link edge points by closeness such as:

- Graph search or tree search
- Dynamic programming
- Contour finding

Another approach of edge linking is to group the edge points by certain constraint of similarity. This type of edge linkers is called global edge linkers. Hough transform [19] is an example of global edge linkers. This method transforms all edge points from spatial space into parameter space. For example, to look for straight edges, the form $y = ax + b$ is used. For an edge point (x_0, y_0), all possible sets of

(a, b) in parameter space that satisfy the equation $y = ax + b$ are marked. For any point (p, q) in the parameter space, the number of its occurrences is the number of points that are on the line $y = px + q$. Therefore, the higher the number, the more likely this straight line is an edge. Other objects used are circles and ellipses. Arbitrary shapes can be used in Hough transform as well[5], though this generalized Hough transform is used for matching predefined shapes rather than edge detection. In general, Hough transform is used in linking straight lines and circles only. The Hough transform has some advantages such as:

1. It can handle occlusion effectively.

2. It is relatively robust to noise. This is because the parameter space is discretized.

3. It detects multiple lines in a single pass.

Its disadvantages include:

1. If the dimension of the parameter space becomes larger, it would increase the complexity of the calculation rapidly. That's why it is used mostly to find lines and circles, since they at most three parameters.

2. Different edges will be treated as the same edge if they share the same parameter space even when they are separate.

3. It might mistake low-curvature curves as straight lines because of the discretization of the parameter space.

4. It cannot be applied to general, non-parametric curves.

The deformable contour models and the Mumford-Shah theory described in the previous section, also perform edge linking. Both of them employ the approach of energy minimization. As mentioned, the Mumford-Shah theory does not provide a practical algorithm to minimize the energy function. While some simplifications are provided to make the minimization applicable, such modification only perform the task of image segmentation. The minimization of the energy function in the deformable contour model are achieved by discrete Euler equation, which involves iterative computations.

Another approach of the deformable contour model, is to form this problem as a two-dimensional graph search problem, using dynamic programming(Dijkstra [14]), as in [3, 10], the "live-wire" in [15], and "intelligent scissors" in [35, 36]. This approach gives the user a larger control over the segmentation process. Once a start point is selected, an optimal path can be computed and drawn in real time between the start point and any given position. In the "intelligent scissors" model, the user is given some control during the curve extraction through "training" process.

Generally, the edge linking methods described in this section are more related to **image segmentation** than **edge detection**. While these two different processes have some common ground, they are not equivalent. While regions are bounded by edges, there are edges that does not bound any regions. Moreover, the deformable contour models do not provide a verification mechanism. While they may produce the most likely results, they could not verify the accuracy of results. Even when there is no edge in the image, this model will still produce a minimum-energy yet meaningless contour.

Chapter 3

Edge Point Profile

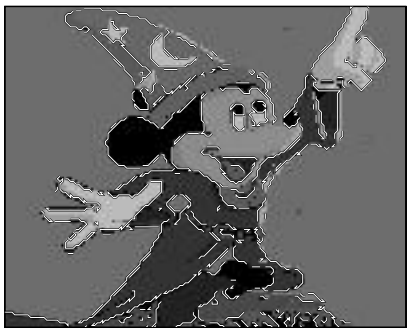
An edge point has two principle directions, the **edge normal direction**, and **edge direction**. These two directions are introduced in Section 1.2. The edge direction indicates the continuity of the edge, while the edge normal direction yields an edge profile. In this chapter, we will describe the process of localizing an edge point in the vicinity of a given point.

The detection of the first edge point is one of the most critical steps, since without any prior knowledge of the edge, choosing an optimal edge profile is difficult, and sometimes impossible. For example, given a point that forms a complicated edge profile, in other words, it is composed of multiple step/ramp profiles. Determining whether this point belongs to an edge with that complicated profile, or multiple edges with simple profiles is impossible without the global view of the edge curves. Sometimes multiple interpretations are equally valid. Sometimes, the consistency of the profiles along the edge curve can be used to identify the best interpretation among the different valid ones. If the edge curves with simple profiles are parallel, we may be able to use one edge curve with composite profile to replace all these

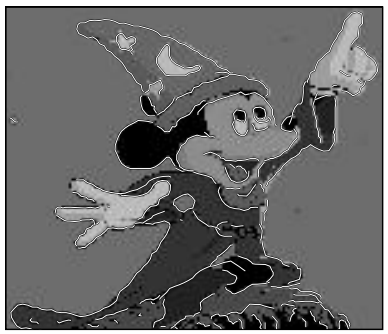
edges. On the other hand, sometimes the decision cannot be performed without prior knowledge, which is beyond the scope of our work. In this chapter, we will not address this issue at all, since it cannot be solved in this stage. We will come back to this problem in the next chapter.

One of the key structures of our edge detector is **profiles**. As mentioned in Chapter 1, a profile is an one-dimensional image extracted from a two-dimensional image along a line segment. The position of the profile is subpixel-level precision. Since the pixel values in profiles are sampled with subpixel-level precision, the choice of subpixel-level precision of the positions does not increase the complexity of the process of edge detection at all. Moreover, it can reduce the zigzag effect caused by the digitization of the images. Although such effect is not very significant, and the choice of subpixel-level precision will not remove it completely. The difference is still visible as shown in Figure 3.1. As it shows, the curves generated by a subpixel-level edge detector appear smoother, and more natural. While it is possible to perform curvature smoothing on the outputs of pixel-level edge detectors and achieve similar visual results as the subpixel-level detectors, precaution has to be taken in order to keep the smoothed curves remain attached to edges on the image. There is also the fact that smoothing tends to remove some fine detail, in this case, sharp corners might be removed accidentally.

To detect edges with our profile-based approach, some functions should be provided for different "classes" of edges. For example, there should be functions for step edges, ramp edges, and ridge edges. Such functions will take a profile as input, and yield a value to indicate the credulity that the input profile is an edge profile. They should also produce some abstract edge properties such as the edge

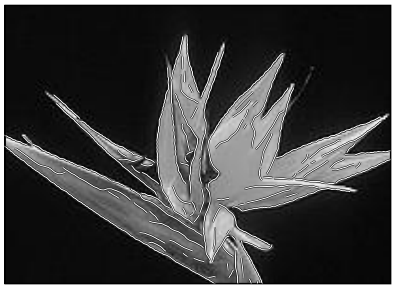


(a) Canny's output of the picture of Mickey Mouse

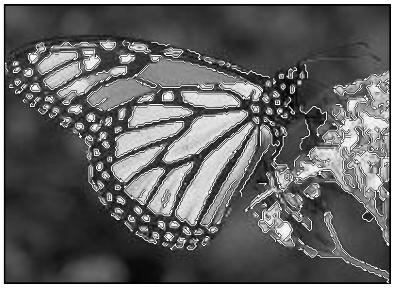


(b) Our profile-based detector's output of the picture of Mickey Mouse

(c) Canny's output of the picture of a flower



(d) Our profile-based detector's output of the picture of a flower



(e) Canny's output of the picture of a butterfly

(f) Our profile-based detector's output of the picture of a butterfly

Figure 3.1: Comparing the results of a pixel-level edge detector with a subpixel-level edge detector on 3 images. (The pixel-level edge detector used is Canny's detector, and the subpixel-level edge detector is our profile-based edge detector.)

width and colors on both sides of the edge. Such properties can then be used in edge linking process, because all the edge points on the same edge share many of these properties. A candidate edge point would be the oriented point whose profile yields a local minimum among its neighborhood according to a specific function. The minimization will reduce the necessity of a threshold, but cannot remove it completely. In the case where there is no edge point except random noise, a local minimum should not validate a false edge point. Therefore, a threshold is still required for the purpose of validation. As mentioned in Chapter 1, if thresholding cannot be avoided, we should at least make the selection of the threshold reasonable. Since the major purpose of thresholding is to distinguish real edges from edges caused by noise, the verification of the profile can be done by comparing the **noise magnitude** and the **edge magnitude**. The measurement of noise magnitude of an image is usually performed by calculating the standard deviation over the colors of quasi-constant regions. For a profile, this could be the **stable regions** we defined in Section 1.2. As of the edge magnitude, the standard deviation of the **transient region** of the profile should be adequate. While both magnitude values might not be very accurate since the data size is too small for statistic calculation, it does provide intuitive reason: if the **transient region** of a profile is no more fluctuating than both **stable regions** of the profile, then the edge centered at the **transient region** cannot be distinguished from noise. The ratio of the signal and the noise, as known as SNR, is widely used in signal process. Some edge detection methods, such as Canny's detector and other extended works [7, 13, 45], use SNR as one of the factors of edge detection. The SNR we measure would be more likely smaller than the real SNR, because of the additional effect of alignment. Yet, the

alignment effect itself could be considered as part of noise because our detector is subpixel-level.

The noise magnitude measured here not only contributes to SNR calculation. During the edge linking process, the noise magnitude is also used as the threshold value when profiles of two adjacent oriented points or two adjacent endpoints of different edges are compared. Moreover, in the post-processing stage, edges can be grouped as the same category of edges if their profiles are similar. Again, the noise magnitude is the main factor of determining the threshold value.

3.1 Edge Point Profiles and Evaluation Functions

The profile for an oriented point (x_0, y_0, θ_0) of a digital image $I(x, y)$ could be defined as $I'(x_0 + t\cos\theta, y_0 + t\sin\theta)$, where $t \in [-T, T]$, T is the default size of the profile, and I' is a interpolated image of I. We do not specify the method of interpolation. In practice, we choose "bi-linear interpolation" for its simplicity. On the other hand, it also removes some high frequency information, resulting in some degree of smoothing in the profile. To reduce such effect, other interpolation methods such as "bi-cubic" or some other spline-based interpolations might yield better interpolated results.

Some evaluating functions are required to perform the verification of a given edge point. The requirement of such a function f is that the function should be locally extremized if the edge point corresponds to an "actual edge" in the image. Since we do not give a formal notion of an "actual edge" for a profile, we will be contented to let the function f be the arbiter of "edge". However, f can be optimized to favor certain profiles such as step edges. Of course, humans can still judge the

appropriateness of any proposed f by comparing its performance against the human consensus. Here "extremized" means either maximized or minimized, depending on the function. For simplicity, we will assume a function should minimize with the input of an edge profile.

Many functions can satisfy such a criterion. For those detectors using first derivatives of images, local maxima are guaranteed at the edges. Most of these edge detectors are specific for step edges. One of the reasons that we do not use these detectors as the evaluation functions is that for these detectors, real edges and false edges caused by noise are separated by some arbitrary threshold whose value cannot be found in a systematic way.

Therefore, we will have to rely on some mechanism that allows us to distinguish the edges from noise. Some properties which could be used in such a function f to evaluate edges are listed below:

1. **Constancy of the stable regions of edge profiles**. The stable regions of the profiles of an edge do not really belong to the edge, but rather represent the regions this edge separates. On the other hand, these regions make the localization of an edge point possible. The pixel values on both regions should be "quasi-constant". It is "quasi-constant" since noise is to be considered. Once the transient region of an edge is bounded, we can use the standard deviations of each side to define the magnitude of local noise. There is the issue of interaction between edges here. When two edges get close, unless we reduce the sizes of the stable regions of the profiles, the edges would interfere with each other. Such interference will make the measurement of noise magnitude unreliable.

2. **Edge magnitude**. The magnitude of an edge in a profile is intuitively clear in the standard examples of a ramp or a ridge. This property is important because the noise magnitude should be smaller than edge magnitude. Otherwise, the edge cannot be distinguished from image noise. Generally, the larger the magnitude of the profile is, the more significant the edge is. On the other hand, if the magnitude of an edge profile is very close to the scale of noise magnitude in the vicinity of that edge point, this edge will be considered to be induced by the image noise. For a ramp profile, the edge magnitude is the the difference of the two stable regions of the profile. For a ridge profile, it is the maximum rise of the two sides of the profile. In general, we can use the standard deviation of the transient region of the edge as the edge magnitude. This is not as intuitive as how we measure the magnitude for a ramp edge or a ridge edge. Moreover, standard deviation measures only the statistic information of the input, not momentum information. For example, the two series $1, 2, 3, 4, 5$ and $1, 4, 2, 3, 5$ have the same standard deviation value, but if they are the values from transient regions of two edges, the latter would be more visually significant since it contains higher frequency data. However, since standard deviation is also used in measuring the noise magnitude, the comparison between the noise magnitude and the edge magnitude becomes comparing the standard deviations between the stable region and the transient region.

3. **The sharpness of the edge**. The sharpness is defined by the width of the transient region of the profile. The narrower this transient region is, the better defined the edge point will be. This property serves two purposes. First, it defines the boundaries between the transient region and the stable regions.

Second, while profiles sampled from different orientations at the position of an edge point will all show proper edge profiles. Yet, the one with the edge normal direction will have thinnest transient region. On the other hand, the difference of the widths of these profiles is minimal and hard to measure. For example, for a ramp edge with a width of 2 pixels, if we take a profile P_0 centered at a point on the edge along the normal direction of the edge point, the sharpness of P_0 is 2 pixels. If another profile P_1 is taken at the same point, but with an orientation 5° apart from P_1. the sharpness of P_1 will be about 2.008 pixels. Even with sub-pixel precision, 0.008 pixel cannot be measured. Even if we increase the difference in orientation to 30°, the difference will still be only 0.31 pixel, which is barely noticeable. Moreover, when the same edge profile is measured in different orientations, the width of the transient region will be different because of digitization. Therefore, even though the width of the edge will be minimum when the orientation of the oriented point matches the edge normal direction theoretically, it is not practical to detect the edge normal direction by minimizing the widths of the transient regions of the profiles. Here we seem to downplay the importance of this property. However, for a profile, increasing the width of transient region into the regions that should belong to stable regions might increase the standard deviation of the transient region. Therefore, it will be important to keep the transient region bounded as tight as possible.

4. **Edge continuity**. Edge continuity refers to the "continuity" of edge points that form an edge. When one looks at an edge, a single edge point is not very significant. What is important is the persistence of the profiles along the edge

curve. The length of an edge should also be taken into consideration. Isolated edge points exist, but they are best ignored or treated as noise since they offer very little information. Once an edge point is located, each of its **neighbor edge points** should be very near a decent edge point and could be the proper location to initiate another search for the next edge point. Furthermore, the profiles of the edge points on the same edge should have many shared properties. The colors on left and right stable regions should be consistent. An issue is texture, which is usually anisotropic and might be considered repetitive only when sampled with the same orientation and specific interval, while our edge profiles are sampled from different directions, with a location of sub-pixel precision. Two profiles from the same edge could be completely different in a texture-rich image. Therefore, unless some texture-specific filters are used to pre-process the image, our profile-based technique would not work. More issues of "edge continuity" will be discussed in Chapter 4.

Note that the parameters of the signal-to-noise ratio (SNR), namely the edge magnitude and noise magnitude, can be used to represent the first and the second edge properties respectively. The continuity of the edge is arguably more important than the rest of the other properties, yet it only works when there exists one valid profile to compare with. Therefore, it seems that with the first two edge properties would form a good measurement of the edge. However, the measurement of the edge magnitude and the noise magnitude depends on the accuracy of the boundaries between the transient region and the stable regions. Moreover, the continuity of an edge cannot be well preserved unless these boundaries can be estimated correctly. While the localization of these boundaries seems straightforward to human eyes,

it requires more than just the maximization of SNR. Take a perfect step edge for example, as long as the discontinuity is included within the transient region, the SNR is maximized. It does not matter whether the discontinuity is at the center of the transient region or not. This will make the localization of the edge impossible. If noise is taken into consideration, then the criterion of maximization of SNR might cause the transient region to be expanded and the stable regions to shrink to minimize the measured noise. Therefore, the minimization of the edge width is to bound the transient region such that the noise will not be taken as part of edge, and to measure the center of the edge profile correctly.

Note that in the description of **edge magnitude**, a method of identifying true edges is presented. While the noise magnitude measured from the constant part of a profile may not be very accurate with the limited size of data, the fact that if the deviation of noise is in the vicinity of the measured edge magnitude, the edge cannot be distinguished and should be discarded. In other words, the ratio of the magnitude of the noise and the magnitude of the edge is measured, and only when the ratio is smaller than one, the profile is accepted as an edge profile. Another reason to measure the noise magnitude is that since Gaussian noise is assumed, it is isotropic. This noise magnitude can then be used not only within the profile, but also between two adjacent profiles. If difference between two edge point profiles is much higher than the noise magnitude, these two profiles can be considered to be from different edges. This can be used to ensure the continuity of an edge.

For this purpose, we divide a profile into three parts: **left stable region**, **transient region**, and **right stable region** as mentioned in Section 1.1. Left stable region and right stable region are supposed to be associated with some constant

values, with additive noise, and the transient region, at the center of the profile, supposed to be the region with discontinuity, defines the magnitude of the edge. The magnitude of noise can be determined statistically by the two stable regions, and the edge magnitude can be retrieved from the transient region. The problem becomes how to determine the range of the three regions of a given profile. The outcome depends on the type of the edge to be detected.

If the type of the edges is known in advance, edge detection would be easier because there are more specific features available. We will first give some examples of specific edge types, followed by an example of a more generalized profile evaluation function.

3.2 Constructing An Evaluation Function for Step Edge Profile

A step edge profile is the simplest kind. It is very easy to describe: flat on each side, the pixel values of the two sides are different, and there is no intermediary region between the two sides. A one dimensional step edge can be described as the function

$$P_{step}(x) = \begin{cases} h & \text{if } x_0 - \frac{w}{2} < x < x_0 \\ h + k & \text{if } x_0 \leq x < x_0 + \frac{w}{2}, \end{cases}$$

where h, k, x_0, w are constants. Of these constants parameters, h is the background intensity, k the edge magnitude, x_0 the edge position, and w the window size of the profile. The window size w can be seen as the scale parameter in the multiple-scale approach.

To incorporate noise, the edge magnitude k should be greater than the mag-

nitude of the noise. Without prior knowledge of noise, Gaussian noise is usually assumed. Gaussian noise is characterized by a zero mean and its standard deviation. Assuming the edge position is at $x = 0$, one possible evaluation function for step edge could be described as:

$$f_{step}(P) = \frac{\max(\operatorname{std} P(-w/2, 0), \operatorname{std} P(0, w/2)) + \sqrt{w/2}}{|P_{avg}(-w/2, 0) - P_{avg}(0, w/2)| + 1} \tag{3.1}$$

where P is the profile, w the window size of P, $P_{avg}(a, b)$ the average of $P(x)$ between a and b, which is defined as

$$\frac{\int_a^b P(x)dx}{b-a},$$

and $\operatorname{std} P(a, b)$ the standard deviation of $P(x)$ between a and b:

$$\sqrt{\frac{\int_a^b \left(P(x) - P_{avg}(a, b)\right)^2}{b-a}},$$

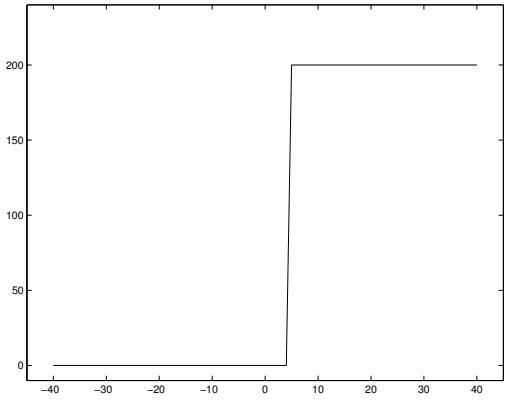
which can be written as

$$\sqrt{\frac{\int_a^b \left(P(x)\right)^2}{b-a} - \left(P_{avg}(a, b)\right)^2}.$$

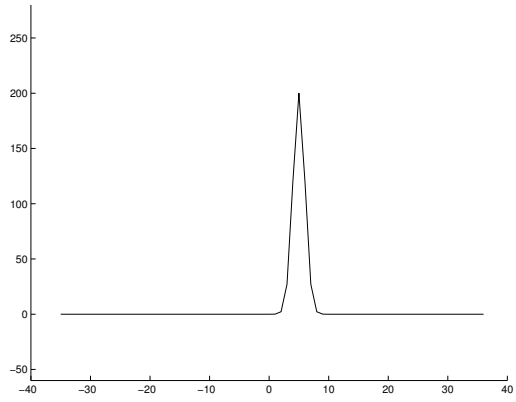
As we can see from (3.1), the denominator of f_{step} could be described as the edge magnitude of the step edge, while the numerator could be seen as the noise magnitude. Intuitively this function $f_{step}(P)$ is inversely proportional to SNR of P. Since SNR will be maxima at edge points, $f_{step}(P)$ will be minimum when P is the profile of an edge. The 1 added to the denominator is to ensure the denominator is not zero. This might not be necessary since if the denominator is zero, it would mean that there is no discontinuity at the center of the profile, which leads to the fact that the profile does not have an edge at the center. The $\sqrt{w/2}$ added to the numerator, on the other hand, is a boundary condition to make sure this function

continuous for a step edge without noise. In fact, it is not necessary either. The only purpose it serves is to improve the output.

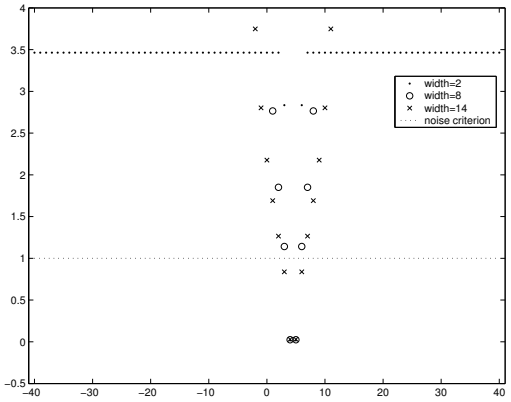
The results of f_{step} applied to various synthetic edges are demonstrated in Figure 3.2. A step edge, some ramp edges with different parameters, a ridge edge, and two step edges with large magnitude of noise are experimented with. Two samples with pure noise are also included for comparison. For purpose of visual comparison, the results of the Canny detector, that is, the first derivative of Gaussian, are also presented. The magnitude of the edges are the same, which is two hundred. These results show that this function f_{step} will minimize at the center of step edge. For a ramp edge, the minimum exists, though a window size smaller than the transient region might cause some problem to pinpoint a single local minimum. In the example of a ridge edge in Figures 3.2(m)-(o) on page 44, which could be considered as two close step edges, smaller window size works better when two step edges are close to each other. On the other hand, larger window size is generally less sensitive to noise, and could work with ramp edges better. Otherwise, the results from different window sizes are generally consistent. For a noisy step edge, the response from this evaluation function is pretty chaotic. Still, most of such confusion can be cleaned up after applying the criterion that the magnitude of noise should be smaller than the magnitude of the step which is represented as the regions under the dotted lines in the figures. Note that for a step edge, the function generally yields consistent local minimum with different window sizes. Therefore, if we take average of the results from different window sizes, such local minimum should still be consistent. The false edges produced by the noise-sensitive smaller window size will be filtered away. One of the drawbacks of the averaging method, is



(a) A step edge

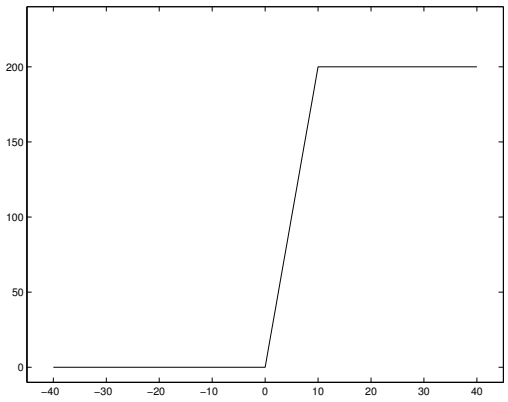


(b) First derivative of Gaussian applied to the edge

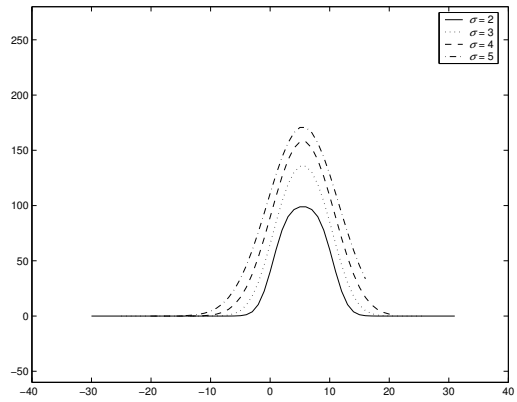


(c) Step edge evaluation function f_{step} applied to the edge

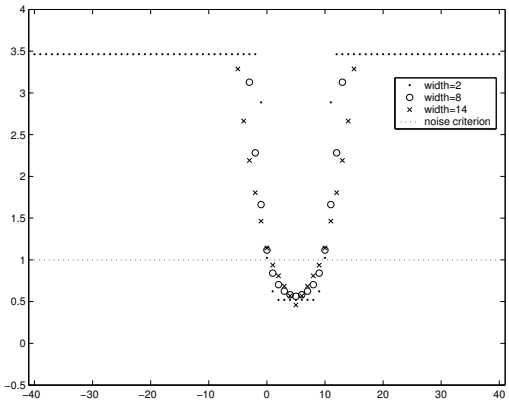
Figure 3.2: A step edge, and the results obtained when first derivative of Gaussian and f_{step} are applied to this edge.



(d) A ramp edge

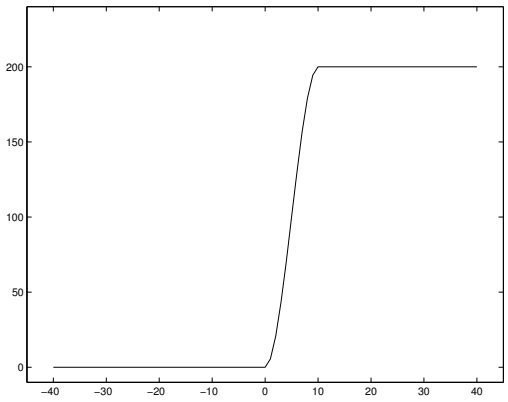


(e) First derivative of Gaussian applied to the edge

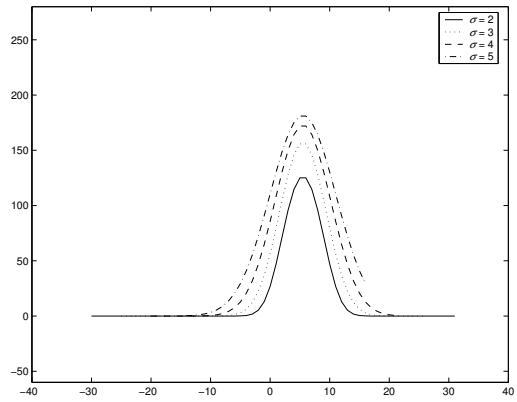


(f) Step edge evaluation function f_{step} applied to the edge

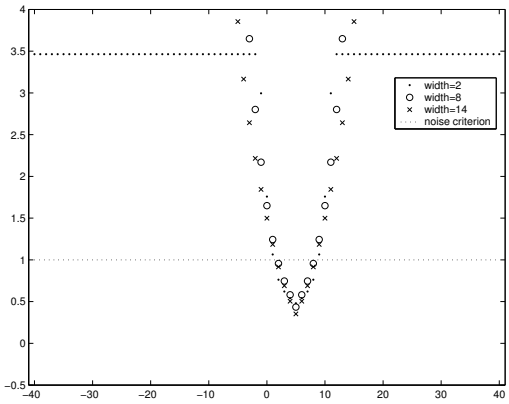
Figure 3.2: (cont.) A ramp edge, and the results obtained when first derivative of Gaussian and f_{step} are applied to this edge.



(g) A smoothed ramp edge

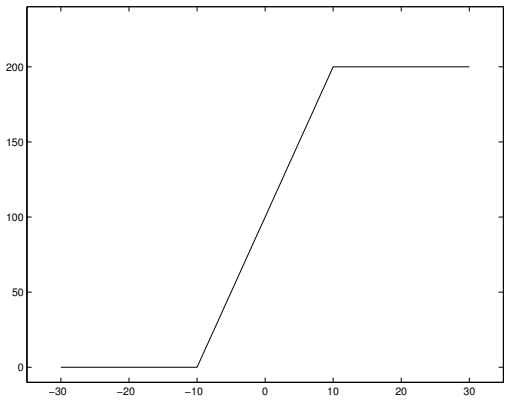


(h) First derivative of Gaussian applied to the edge

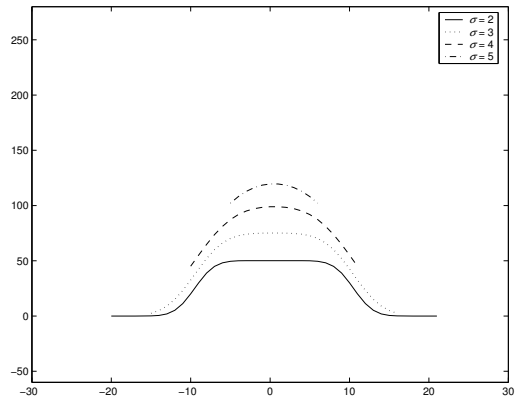


(i) Step edge evaluation function f_{step} applied to the edge

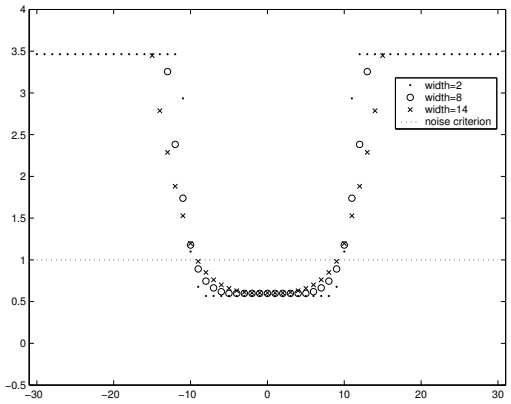
Figure 3.2: (cont.) A smoothed ramp edge, and the results obtained when first derivative of Gaussian and f_{step} are applied to this edge.



(j) A wide ramp edge

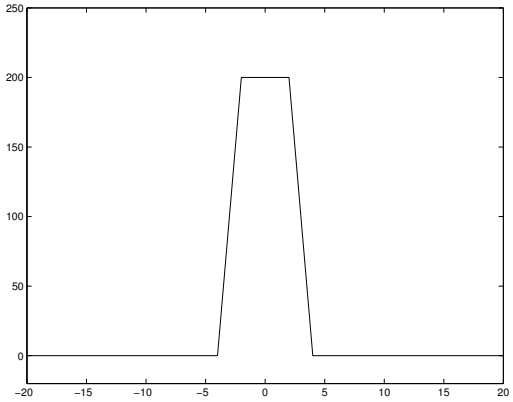


(k) First derivative of Gaussian applied to the edge

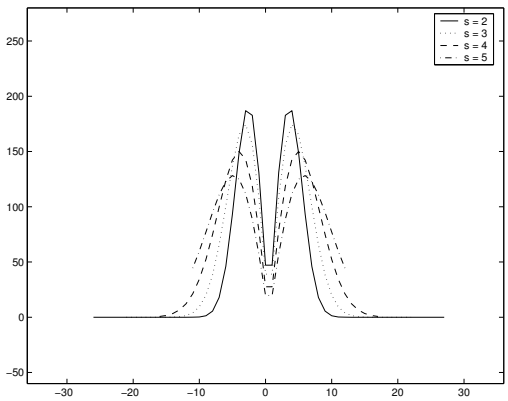


(l) Step edge evaluation function f_{step} applied to the edge

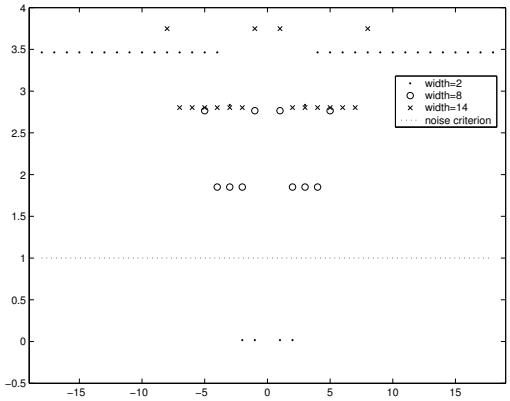
Figure 3.2: (cont.) A wide ramp edge, and the results obtained when first derivative of Gaussian and f_{step} are applied to this edge.



(m) A ridge edge, which consists of two step edges

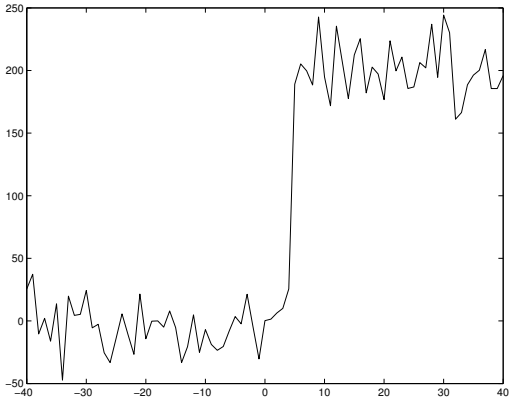


(n) First derivative of Gaussian applied to the ridge edge

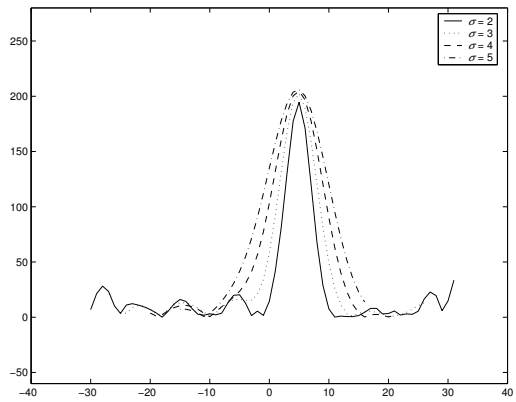


(o) Step edge evaluation function f_{step} applied to the ridge edge

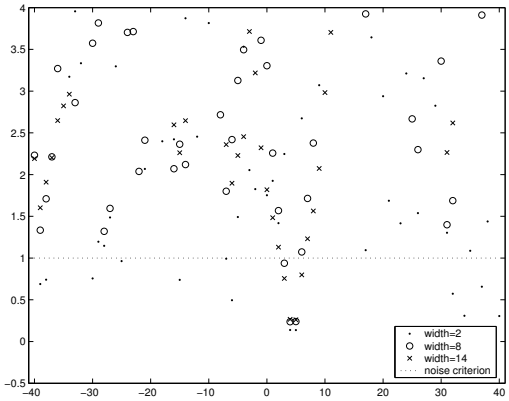
Figure 3.2: (cont.) A ridge edge, and the results obtained when first derivative of Gaussian and f_{step} are applied to this edge.



(p) A step edge with Gaussian noise, $\sigma = 20$

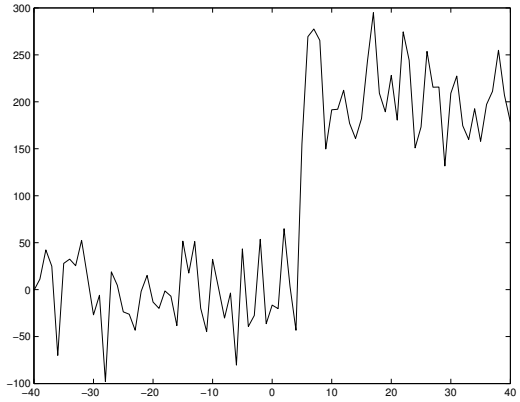


(q) First derivative of Gaussian applied to the edge

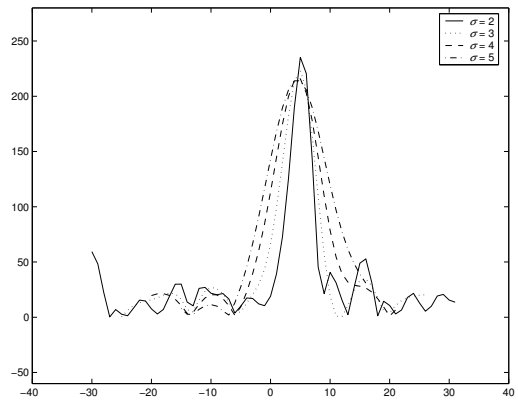


(r) Step edge evaluation function f_{step} applied to the edge

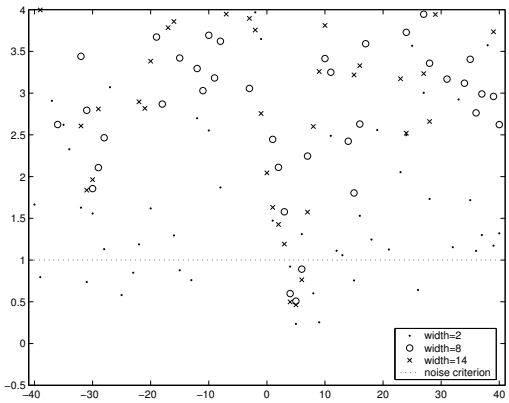
Figure 3.2: (cont.) A noisy edge, and the results obtained when first derivative of Gaussian and f_{step} are applied to this edge.



(s) A step edge with Gaussian noise, $\sigma = 40$

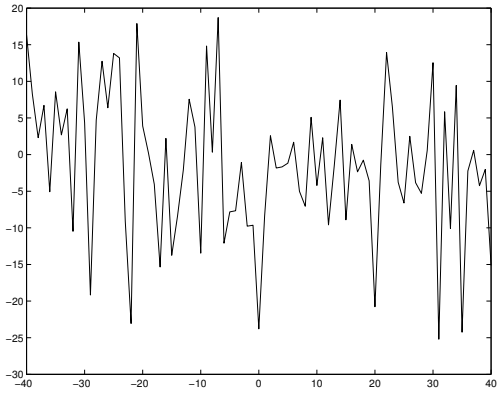


(t) First derivative of Gaussian applied to the edge

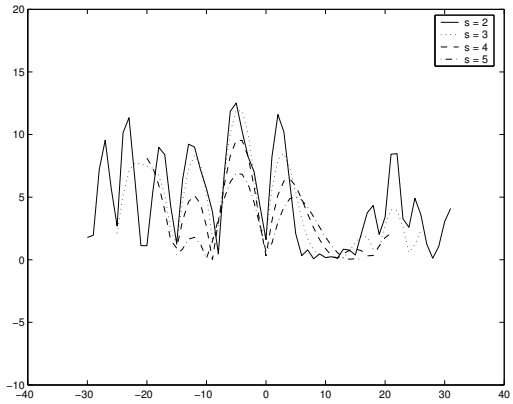


(u) Step edge evaluation function f_{step} applied to the edge

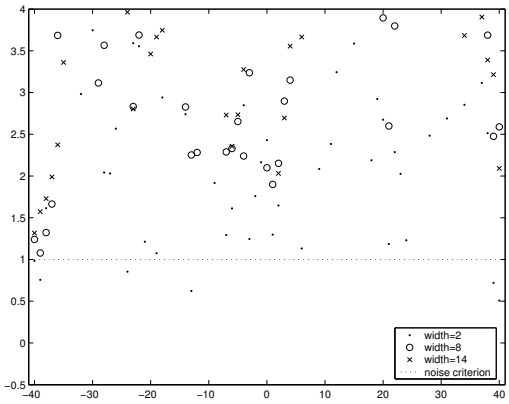
Figure 3.2: (cont.) Another noisy edge, and the results obtained when first derivative of Gaussian and f_{step} are applied to this edge.



(v) A profile with Gaussian noise, $\sigma = 10$



(w) First derivative of Gaussian applied to this profile



(x) Step edge evaluation function f_{step} applied to this profile

Figure 3.2: (cont.) The results when first derivative of Gaussian and f_{step} are applied to a given profile with Gaussian noise.

that for some edges, which can only be detected with specific window size, will not be detected. An example of such an edge is the ridge edge in Figure 3.2(o). From empirical data, we conclude that if geometric mean is used instead of arithmetic mean, the accuracy is improved slightly in some cases. Such outcome is expected since our evaluation function is a ratio, and geometric mean generally works better for mean of ratios.

In comparison, the Canny detector, although showing better visual results, has trouble distinguishing between weak edges and strong noise. While the response from the edge is obvious much greater than those responses from noise, it is also obvious the magnitude of the correct response depends on the slope of the edge, as well as the smoothing parameter σ. Thus, the decision will rely on thresholding without a reliable threshold value that can be obtained from σ and the input signal. Our detector, on the other hand, while appearing chaotic in the presence of noise, provides very good localization of the edge generally. With the simple criterion of noise magnitude, most of the false edges induced by noise can be removed.

3.3 Constructing An Evaluating Function for Ramp Edges

We will use a ramp edge as another example for evaluation function construction(see Figure 1.3(a) for a ramp edge). As shown in Figure 3.2, the evaluation function constructed for step edge works pretty well for a ramp edge. However, for a wide ramp edge, larger window size w is required in order to get a local minimum. Moreover, such local minima usually vary very little within its neighborhood. Also, in order to get better estimate of the noise magnitude, which is useful in later stage, the transient region should be taken into account. Therefore, an evaluation

function can be constructed specifically for a ramp edge.

Note that a ramp edge can be described as a smoothed step edge or an interpolated one. Detectors designed for step edges usually work for ramp edges as well. For a natural image, step edges do not exist since the pixel value of each pixel is actually the value for the region of the pixel instead of the pixel value for one single point when an image is captured by an digitization device. Therefore, the pixel values at edges should be some interpolation between two sides of the edges. The difference between a ramp edge and a step edge is that the former has a transient region between two sides. This transient region is normally either monotonic increasing or monotonic decreasing. If we consider the ramp edge to be a blurred step edge, we could also assume that the intermediary region will be anti-symmetric since the blurring operators are usually isotropic. Therefore, the anti-symmetricity of step edges will be preserved in ramp edges.

We will use the aforementioned features to construct our evaluation function by first defining the feature of anti-symmetry. The basic definition of anti-symmetry is that if $f(x+c) = -f(c-x)$, then f is anti-symmetric about $x = c$. Since the pixel values are always positive for a digital image, we will change the definition by shifting the function values so that if $g(c+x) - g(c) = g(c) - g(c-x)$, g is anti-symmetric at $x = c$. Therefore, the equation

$$\int_0^{x_{max}} k(x)|g(c+x) + g(c-x) - 2g(c)|dx$$

will be a reasonable way to measure the anti-symmetry given a function g and a center c. The parameter $k(x)$ are the weights of the sum and x_{max} is the size of the **window** of a profile. For a perfect ramp edge profile, the parameter $k(x)$ does not matter as long as it is a non-zero function, because we will get 0 at the

center of the ramp edges. Even when the edge profile is noisy, we will still get minimum near the center. However, if there are other edges present in the profile, the result will be unpredictable. In order to prevent that, the weights near the center should be increased. Therefore, the only criteria we set for $k(x)$ is that it should be a monotonic decreasing positive function, and the normalizing condition $\int_0^{x_{max}} k(x) = 1$. Since we get the profiles by sampling the image, these profiles are discrete functions instead of continuous ones. Therefore, we rewrite our evaluation function of anti-symmetry as

$$f_{anti-sym}(g) = \sum_{i=0}^{x_{max}} k_i |g(c-x) + g(c+x) - 2g(c)|.$$

The function above should measure anti-symmetry pretty well. However, anti-symmetry alone would not make a ramp edge. One can easily construct a function that has the anti-symmetric property, but not a ramp edge. For example, any constant function $f(x) = a$ would fit the anti-symmetry well even though there is no edge. In order to identify ramp edges more precisely, we should incorporate other properties such as the monotonic property and the constance on both sides of the edges.

Even though it is easy to understand the monotonic property, measuring it quantitatively is not trivial. We will have to measure the "quantity" of the monotonicity of the edge profile. Due to the presence of noise and other close edges, it would make the confirmation of the monotonic property impractical. A simple measurement of monotonicity of a function $g(x)$ could be defined as

$$f_{monotonicity}(g) = \frac{\min(\int_{-w_c}^{w_c} g'(x)_{g'(x)>0} dx, -\int_{-w_c}^{w_c} g'(x)_{g'(x)<0} dx)}{\int_{-w_c}^{w_c} |g'(x)|\, dx}$$

which measures the occurrences of the part that $g'(x)$ is positive and the part that

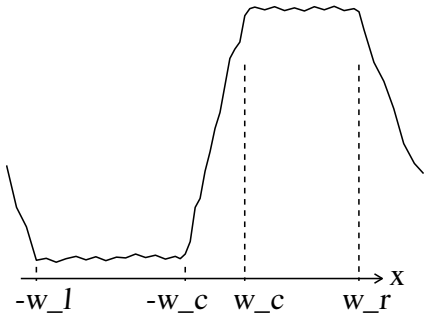


Figure 3.3: Example of an ramp edge profile with center width w_c and side boundaries w_l and w_r.

$g'(x)$ is negative. Here w_c is a boundary we will mention later. The positive part means where $g(x)$ is increasing, while the negative part means where $g(x)$ is decreasing. Therefore, if the given function $g(x)$ is monotonic, then $f_{monotonicity}(g) = 0$ since the smaller part is always 0.

The measurement of the constancy on both sides of the profiles is pretty straightforward. We could simply modify the evaluation function for step edges. The modification is to define the range of each side. If we use the whole profile, the transient region as well as some close-by edges would cause some interference. This approach introduces three unknown parameters into the image and further complicate the computation. For a profile P, we will need the center width w_c, left boundary w_l, right boundary w_r, as seen in Figure 3.3. Then the way to measure the constancy on both sides would be

$$f_{smooth}^{w_c,w_l,w_r}(P) = \max \frac{(\operatorname{std} P(-w_l, -w_c), \operatorname{std} P(w_c, w_r)) + \sqrt{w}}{|P_{avg}(-w_c, 0) - P_{avg}(0, w_c)| + 1} \tag{3.2}$$

where w is half of the window size and the maximum for w_l and w_r, $\operatorname{std} P(i, j)$ and $P_{avg}(i, j)$ have the same definitions as in (3.1).

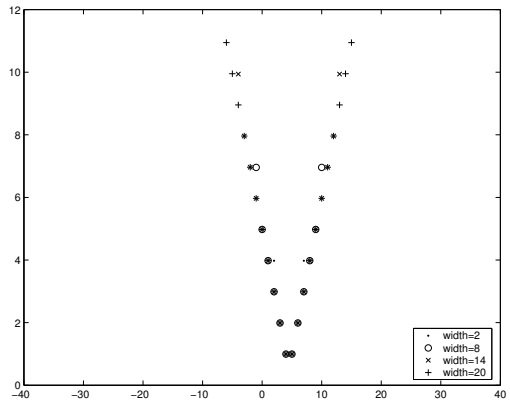


Figure 3.4: The evaluation function f_{ramp} applied to the step edge in Figure 3.2(a).

In order to get the measurement of the constancy of each side of profile P without pre-defined values of w_c, w_l, and w_r, a straightforward method is to first calculate $f_{smooth}^{w_c,w_l,w_r}(P)$ for all possible combinations of w_c, w_l, and w_r, then find the minimum. However, this approach will not work without other constraints. For example, when $w_c = w_l = w_r$, we can always produce a minimum for $f_{smooth}(P)$. Therefore, we will have to find a local minimum with some constraints instead. First constraint is that w_c should be as small as possible. Because once we get the real width of the edge $w_{c,0}$, any $w_c > w_{c,0}$ will have a similar $f_{smooth}^{w_c,w_l,w_r}$(P) value. The values of both w_l and w_r should be as large as possible, for the similar reason of the constraint on w_c. In other words, we want to minimize w_c and $f_{smooth}^{w_c,w_l,w_r}(P)$ while maximizing w_l and w_r at the same time.

A linear combination of these three evaluation functions should capture a ramp edge profile pretty well. For simplicity, we can replace w_l and w_r by a single parameter w_b. Generally, as we mentioned, a ramp edge can usually be detected by a step edge detector. Therefore, because of its additional complexity, this detector

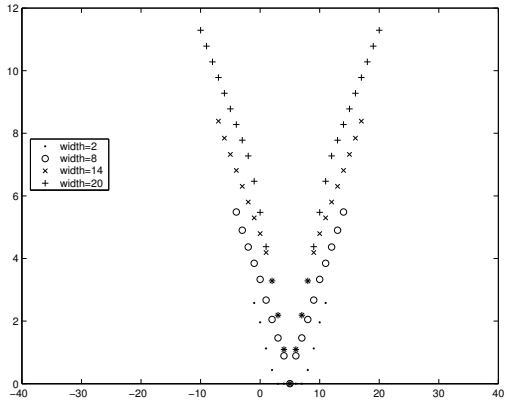


Figure 3.5: The evaluation function f_{ramp} applied to the ramp edge in Figure 3.2(d).

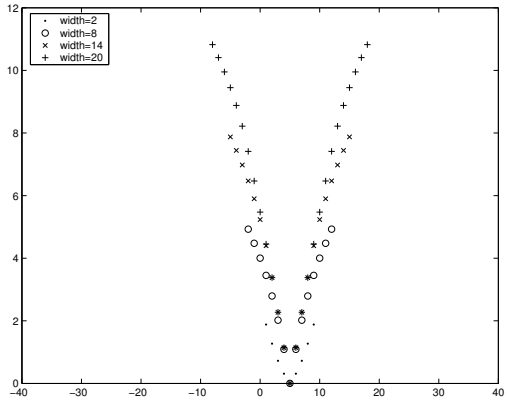


Figure 3.6: The evaluation function f_{ramp} applied to the smoothed ramp edge in Figure 3.2(g).

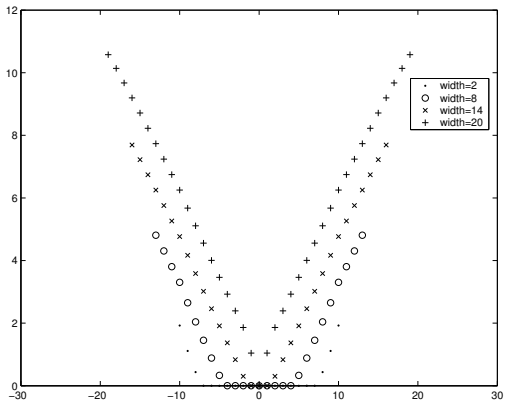


Figure 3.7: The evaluation function f_{ramp} applied to the wide ramp edge in Figure 3.2(j).

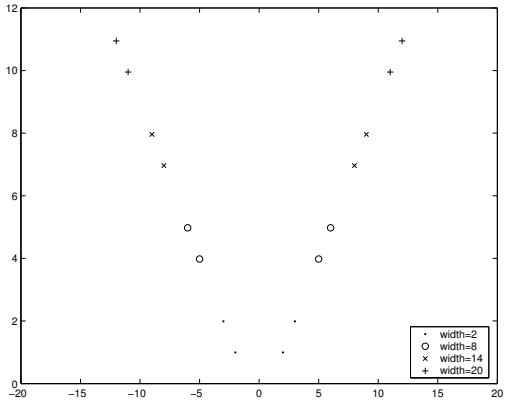


Figure 3.8: The evaluation function f_{ramp} applied to the ridge edge in Figure 3.2(m).

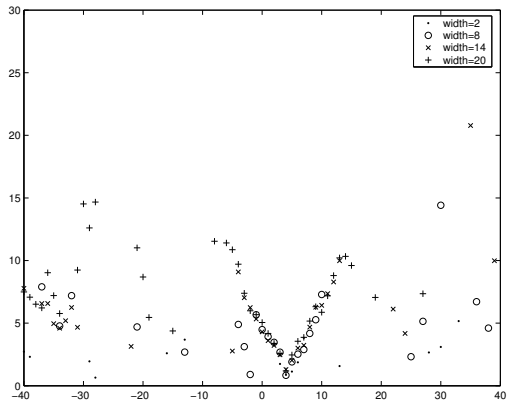


Figure 3.9: The evaluation function f_{ramp} applied to the noisy step edge in Figure 3.2(p).

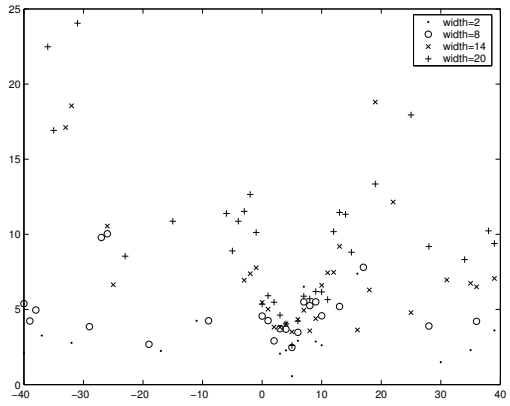


Figure 3.10: The evaluation function f_{ramp} applied to the noisy step edge in Figure 3.2(s).

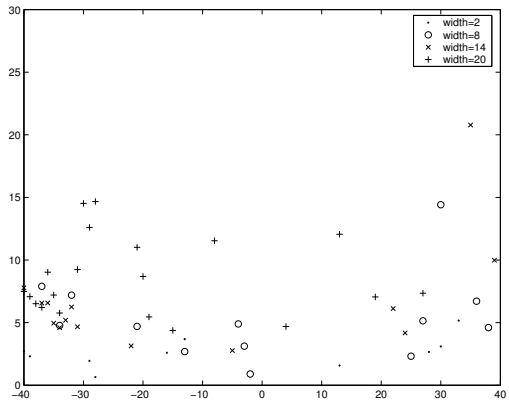


Figure 3.11: The evaluation function f_{ramp} applied to the random noise in Figure 3.2(v).

is not as useful as the simpler detector for step edges. Moreover, with three unrelated terms, it would be difficult to make a proper trade-off among the minima of these terms. Our approach is to find the local minima of the combination of the anti-symmetry and the monotonicity first, then verify these minima using f_{smooth}. Nevertheless, the proper combination parameters for the anti-symmetric term and the monotonic term are still required. Because these two terms are not related, the proper parameters for such combination cannot be determined without the trial-and-error process which is the exact reason we want to avoid in thresholding. To remove such guesswork, the simplest solution is to use only one of the terms and ignore the other. The monotonicity requirement is arguably the more important one. On the other hand, it cannot identify center of the edge given a perfect step edge since the whole input profile is monotonic. Therefore we keep anti-symmetric term for our evaluation function of ramp edges f_{ramp}, it is a two-step operation:

$$f_{ramp} = \begin{cases} f_{anti-sym} & \text{if } f_{smooth} < 1 \\ \infty & \text{otherwise.} \end{cases}$$

The results of this f_{ramp} are shown in Figures 3.4– 3.11. Although the results are already filtered by the f_{smooth}, which is very similar to f_{step}, the ability of f_{ramp} to locate edges seems less satisfactory when compared to f_{step}. This is because we simplify the function in order to visualize the results. The parameters w_c is the **width** in the figures, while w_l and w_r are combined into a hidden boundary parameter w_b. Each value in the figures actually comes from the minimal value with different w_b ranged from $w_c + 1$ to $w_c + 5$. Recall the definition of this evaluation function, we have to minimize both f_{ramp} and w_c. The separation of different w_c values in these figures are to demonstrate the different behaviors of this evaluation function with different w_c. It is more obvious in f_{ramp} than in f_{step} that smaller window sizes will perform better in localizing edges, but also more vulnerable to noise.

Similar to f_{step}, different window sizes for f_{ramp} produce consistent results for perfect edges. Therefore, it will be easier and simpler taking average of different w_c results, rather than trying to minimize both w_c and f_{ramp}.

3.4 Constructing A Generalized Evaluation Function for Edges

We have described a example on how to construct evaluation functions for a step edge and a ramp edge. Now we will demonstrate on how to construct an evaluation function that could be used to detect as many types of edges as possible. For a more generalized edge, we cannot longer detect more specific features other than the four properties described earlier in this chapter. See Figure 3.12 for one example

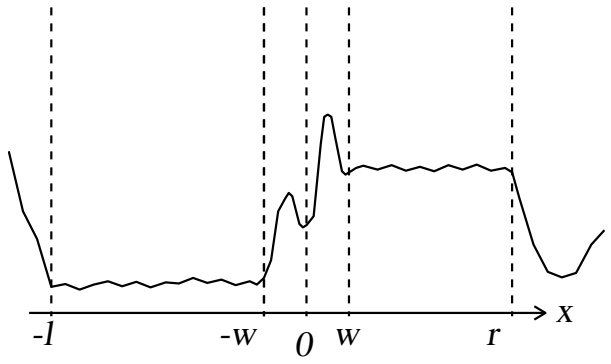


Figure 3.12: Example of an generalized edge profile with center w_c width and side boundaries w_l and w_r.

of an edge that cannot be categorized into some specific type of edges. While this edge can still be described as combination of multiple "sub-edges", which can be step edges or ramp edges. There are no stable regions for most of these sub-edges, therefore these sub-edges cannot be detected by using our evaluation function for step edges. The simple evaluation function for step edges is described in Equation (3.2), wherein the principle of **signal-to-noise ratio** is applied. However, unlike step edges or ramp edges, the edge magnitude of a generalized edge is not necessarily defined by the difference of the two sides of the edge. For example, the magnitude of a ridge edge should instead be defined by the difference between the center of the edge and the stable regions of the edge. Therefore, the denominator of that function need to be modified to reflect the change of the definition of the edge magnitude.

A preferable approach to measure the edge magnitude is to use the discontinuity

of the edge. In other words, the transient region at the center of the edge. The discontinuity can be measured by calculating the standard deviation of the center of the profile. It is obvious the standard deviation of the center of an edge should be larger than the standard deviation of either side of the edge. Therefore, if we define

$$f_{general}^{w_c,w_l,w_r}(P) = \frac{\max(\operatorname{std} P(-w_l, -w_c), \operatorname{std} P(w_c, w_r))}{\operatorname{std} P(-w_c, w_c)}, \tag{3.3}$$

The same constraint $f_{general}(P) < 1$ is still hold. As long as $f_{general}(P_0)$ is local minimum, at the same time smaller than 1, then P_0 can be accepted as an edge profile.

Shown in Figures 3.13 – 3.20 are the results obtained from the same sample profiles used in the previous sections. For sample profiles without noise, the results are as expected. Especially for the result of ridge edge, as depicted in Figure 3.17. For smaller windows, the ramp edge is treated as two step edges. For larger windows, it is then treated as a single edge. However, the results get erratic as noise is introduced. A local minimum can appear near the center of the edge. However, we have a lot of local minima induced by noise as well. Some of the minima caused by noise are even more significant than the minimum for the real edge when noise gets stronger. Though if the same trick in previous two sections can be applied by taking average of the results from different w_c parameters, the results will still be acceptable.

3.5 Edge Point Detection

In the previous section, a few evaluation functions have been introduced. The functionalities of these evaluation functions are to evaluate profiles based on some

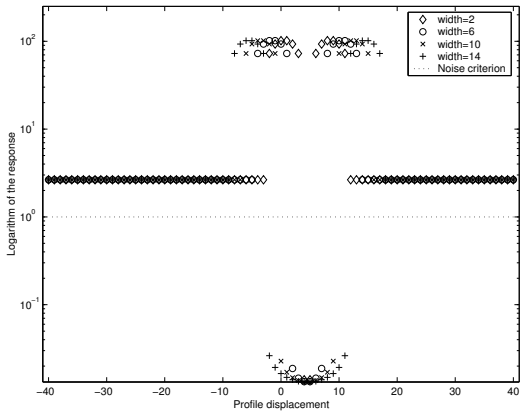


Figure 3.13: The evaluation function $f_{general}$ applied to the step edge in Figure 3.2(a).

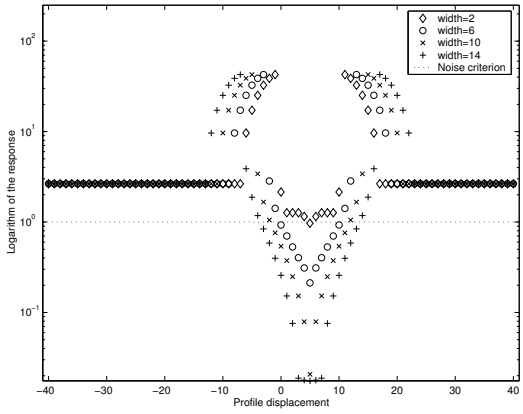


Figure 3.14: The evaluation function $f_{general}$ applied to the ramp edge in Figure 3.2(d).

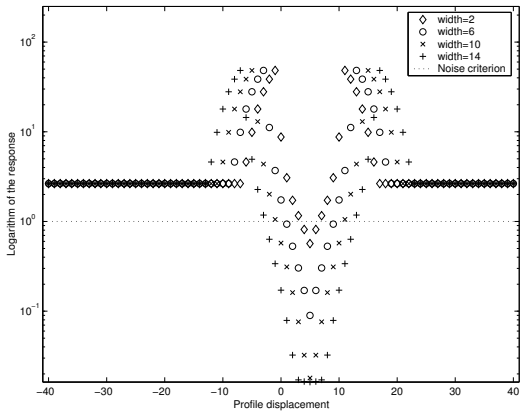


Figure 3.15: The evaluation function $f_{general}$ applied to the smoothed ramp edge in Figure 3.2(g).

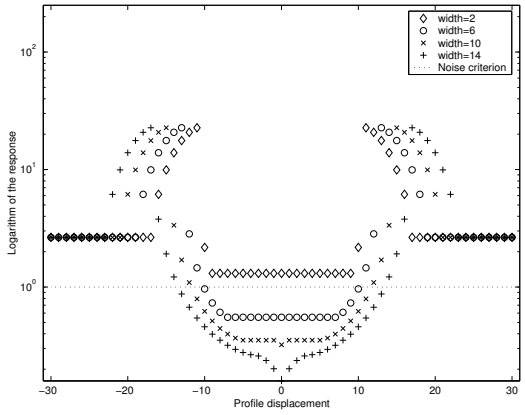


Figure 3.16: The evaluation function $f_{general}$ applied to the wide ramp edge in Figure 3.2(j).

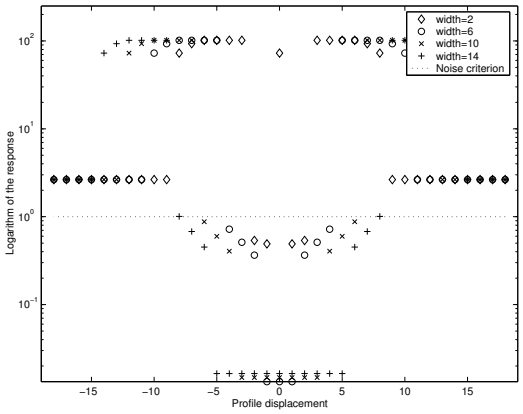


Figure 3.17: The evaluation function $f_{general}$ applied to the ridge edge in Figure 3.2(m).

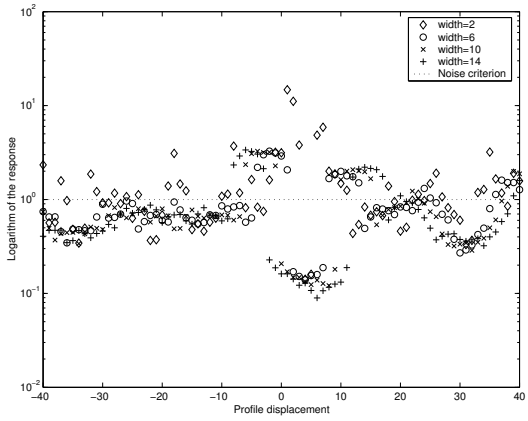


Figure 3.18: The evaluation function $f_{general}$ applied to the noisy step edge in Figure 3.2(p).

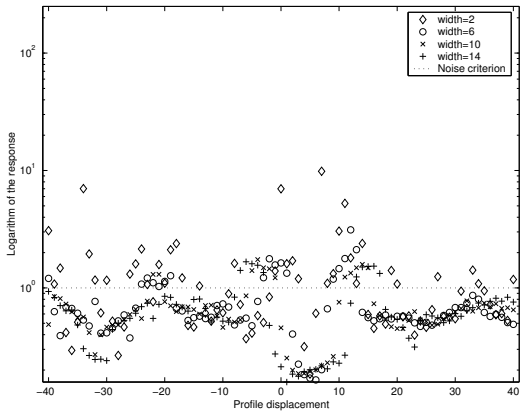


Figure 3.19: The evaluation function $f_{general}$ applied to the noisy step edge in Figure 3.2(s).

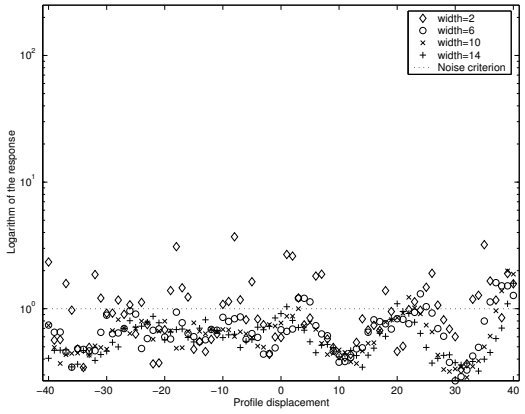


Figure 3.20: The evaluation function $f_{general}$ applied to the random noise in Figure 3.2(v).

properties of edges. In addition to providing the estimates of the boundaries of the profiles and the noise magnitude, an evaluation function is expected to yield a value which will be a local minimum if the profile is an edge profile centered at $x = 0$. The real detection requires evaluating the *local area* of the initial oriented point to find an oriented point that gives minimal value.

The *local area* of an oriented point is more complicated than the local area of a point. Since an oriented point is defined by a position and an orientation, the range of the search for local minimum is three-dimensional. Moreover, if we choose the local area of an oriented point (x_0, y_0, θ_0) to be $(x_0 + x_d, y_0 + y_d, \theta_0 + \theta_d)$, where $x_d \in [-T, T]$, $y_d \in [-T, T]$, $\theta_d \in [-\phi, \phi]$, and T, ϕ are small values defining the size of the local area, there would be more than one edge points of the same edge within the region no matter how small the local area is. A local minimum will not work properly since many edge points from the same edge exist within this area. In order to get all the edge points within such definition of local area, thresholding has to be used, without any guideline of the selection of the threshold value. Since we are against the use of arbitrary thresholding by other edge detectors, we will try to avoid such problem here. Therefore, the *local area* has to be carefully redefined.

Consider the given oriented point (x_0, y_0, θ_0), if the image is sampled one-dimensionally along the orientation θ_0 and centered at (x_0, y_0), while there may be more than one edges in the sampled one-dimensional signal, one obtains a single edge by adjusting the boundaries of the sampling. The local minimum can be used to identify the position of an edge point. Here we assume that the profile sampled from a two-dimensional edge have an edge profile no matter what the sampling orientation is. This assumption is generally true except for the **edge**

direction. Therefore, the *local area* concept is divided into two-part operation, the first part to locate the location of edge point, and the second part to find the orientation of edge point.

The first part of edge point detection is to find local minimum of *local area* $(x_0 + t\cos\theta_0, y_0 + t\sin\theta_0, \theta_0)$, where $t \in [-T, T]$. Once the oriented point (x_1, y_1, θ_0) with local minimum is found, the next part of edge point localization would be finding local minimum of in another *local area* $(x_1, y_1, \theta_0 + \theta_d)$ where $\theta_d \in [-\phi, \phi]$. The second part is more problematic. As stated previously, once the position of an edge point is located, profile taken from any orientation would produce an edge profile.

As previously mentioned, a single edge detector would not be able to detect all possible edges. Our detector is actually a framework of multiple edge detectors where each is served by an evaluation function. If an evaluation function cannot detect an edge point from the vicinity of an initial point, other evaluation functions may be applied one by one until some specific evaluation function confirms a certain edge point. The evaluation function for step edges is the natural candidate as the first evaluation function since almost all types of edges can be derived from step edges. With the requirement of the stable regions, some of the degenerated step edges that form more complex edges will not be detected and require some more specific evaluation function such as the generalized evaluation function. This provides more flexibility to our detector since newer designs of evaluation functions can be added to provide better detection of edges that cannot be detected before.

3.6 Initial Edge Points And Follow-up Edge Points

There is difference between the detection of the initial edge points, and of the follow-up edge points. First of all, there is no prior knowledge for the initial edge points. Therefore, it is necessary to search through all possible combinations of variant boundaries for the transient region and the stable regions to get the optimal profile settings. It is more straightforward for the follow-up edge points, since almost all the profile settings can be inherited from the profile of the edge point it follows. The use of profile evaluation functions for each case will be discussed in the following.

3.6.1 Using Evaluation Functions for Initial Edge Points

Without prior knowledge about the profile boundaries, an exhaustive search is almost guaranteed. As noted, all evaluation functions have some parameters for boundaries of the transient region and the stable regions. Yet, so far we have not provided any guidance for choosing the parameters. In fact, there are some simple guidelines for the selection of the boundary parameters, not only for initial edge points, but all edge points:

- The transient region should be always at the center of the profile.
- The transient region should be as thin as possible. This does not apply to the evaluation function for step edges, since step edges do not have transient regions. More precisely, their transient regions have always one-pixel width containing the discontinuity.
- The stable regions should be as wide as possible. After all, the stable regions are used to measure noise statistically. If the stable regions are too small, the

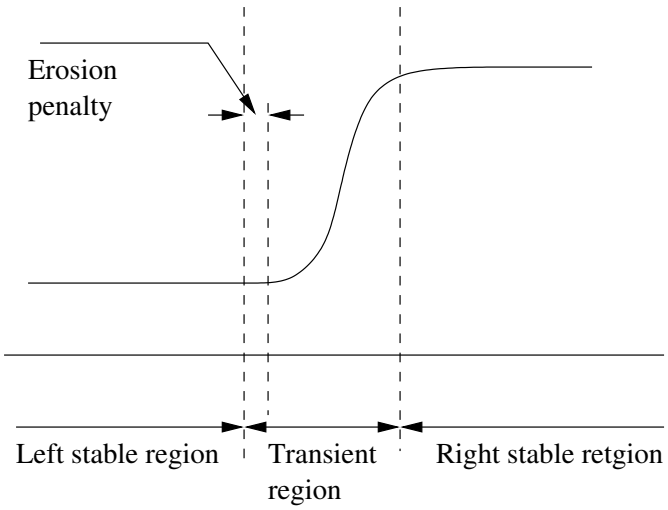


Figure 3.21: A profile whose center is away from the position of an edge.

noise measurement might be unreliable, so are the other measurement derived from the noise measurement, including the SNR.

Therefore, for the detection of an initial edge point, the evaluation of each candidate profile (recall that the localization of an edge point requires the evaluation of the whole local area then finds the minimum) would have to search exhaustively for all different boundaries. For those boundaries that satisfy the SNR requirement, additional adjustments are performed before the minimization:

1. The width of the transient region is added as part of a penalty function. The larger the transient region, the larger the penalty.

2. The widths of both stable regions are also taken into consideration for the adjustment. The smaller the width, the larger the penalty.

3. Another penalty is called **erosion penalty**. This is the penalty to prevent the transient region from expansion which converts part of the stable regions

into the transient region. Starting from the boundaries between the transient region and the stable regions, the color of each point within the transient region is compared with the color of the stable region it is adjacent to. If the difference is smaller than the noise magnitude, it would mean the boundary is not correctly marked, and a penalty is used. Moreover, if such penalties on both sides are not equal, as shown in Figure 3.21, it would mean the center of the profile is misplaced, and should be shifted.

The forming of the penalty function is empirical. While the reasons behind the penalty are sound, we are unable to arrive at an analytic conclusion on the optimal penalty function. In our experiment, the penalty function in the form of

$$f_{penalty} = \sqrt{\frac{w_c}{w_l + w_r}} + (e_l + e_r)/2$$

does produce acceptable results. Here w_c, w_l, and w_r are the values defining boundaries as described in the sections of evaluation functions, and e_l and e_r are the values of erosion penalty on each side.

The result of the evaluation function will be determined by the boundary settings whose NSR (noise-to-signal ratio, the inverse of SNR) multiplied by $f_{penalty}$ yields minimal value. The computation for the noise magnitude and edge magnitude for all possible combination of transient regions and stable regions is very time-consuming.

3.6.2 Using Evaluation Functions for Follow-up Edge Points

While all the profiles can be evaluated the same way as described in the previous section, it is time-consuming. Once an initial edge point is found, the rest of the edge points on the same edge can actually take advantage of the existing profile

information. Among the derived profile information: the average colors on both stable regions, noise magnitude and edge magnitude are all very useful information. While the boundaries are not so reliable due to alignment, they are still useful. The transient region is not as useful due to its higher noise magnitude.

We start out by setting the boundaries between the transient region and the stable regions the same as those of the template profile. Then the colors of the pixels in left and right stable regions of current profile are compared with the average colors of left and right stable regions of the template profile. Using the noise magnitude as the threshold, we can find the outer boundaries of both stable regions. The next step is to fine-tune the boundaries of the transient region. It is done by comparing the color on the boundaries with the colors of the stable regions. If each side has similar color with the stable region, the boundaries are shifted inward, and new comparison on the boundaries is repeated. If only one side has such similarity, then the center of the profile is not on an edge, and the "erosion penalty" is applied instead. Once all the boundaries are properly formed, the other derived information of the profile can be calculated easily.

3.7 Detection of the Orientation of An Edge Point

In the previous sections, we have described several evaluation functions and how to use these function to localize an edge point. However, these functions are not very useful at finding a correct orientation. Once the center of an edge point is located, any orientation would give a suitable edge profile. The width of the center of the edge would be the minimum for the optimal orientation. However, the difference is very small, and could not be measured precisely. In other words, the profile

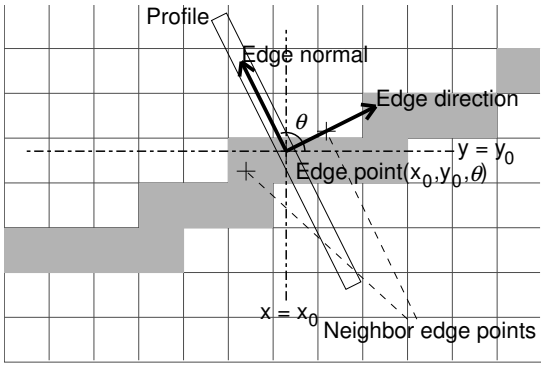


Figure 3.22: Example of some notations used for edge points. Note that the gray area is the digitized form of a straight edge.

of an edge point is not sufficient to determine the orientation of the edge point. In fact, in our introduction, we have already mentioned that for an edge curve $\gamma(t) = (x(t), y(t))$, the edge normal $\theta(t)$ is $\arctan -x'(t)/y'(t)$. Therefore, the edge normal of an edge point P_0 is decided by the position of the adjacent edge points of P_0.

In order to get the orientation, we have to consider the local area around the point. The most widely used method in finding edge point orientation is to take the direction of the gradient of the point, which involves its four neighboring pixels.

Taken the edge in Figure 3.22 as example, gradients could only give approximate orientations. While smoothing filters can increase the accuracy of the gradients, there are limits what a smoothing filter can do before destroying the edge itself. In order to get better estimate of edge orientations, a larger neighboring region is required.

Consider a straight edge such as the one in Figure 3.22 first. If $e(x_0, y_0, \theta_0)$ is an

accurate edge point, then its **neighbor edge points** $(x_0 \mp d \sin\theta_0, y_0 \pm d \cos\theta_0, \theta_0)$ will also be correct edge points, too. Therefore, one could find the correct orientation by the following approach: with a given initial orientation θ_1, applying an evaluation function on both neighbor edge points with some variation of θ_1, summing them up, then finding the orientation that produces the local minimum. It is more involved to generalize this approach to curved edges. Note that the direction of an edge point is the slope of the edge at that point, and the slope is determined by its two adjacent edge points. Therefore, we could experiment with various orientations to find the optimal one. With each orientation the chosen evaluation function will be applied to both neighbor edge points, and local minimum on each side can be found. These two local minima will determine the positions of the adjoint edge points, therefore the orientation of current edge point. In other words, after the positions of the two adjacent edge points are determined, the edge normal direction can be found by the positions of these points.

Chapter 4

Edge Tracking

In the previous chapter, we demonstrate how to localize an edge point from an arbitrary oriented point with a specific evaluation function. Some evaluation functions are constructed and tested upon a set of synthetic edge profiles. The results from these tests show that most of them could perform well even with very noisy profiles. In most cases, with the criterion regarding the magnitude of noise, most of noise-induced false edge points can be removed, and edge points can be localize without using any arbitrary thresholding.

However, a profile alone is not sufficient. As previously mentioned, a main characteristic of a two-dimensional edge is its continuity. A single edge point, or even all the edge points in an given image, will not be able to present this property. In order to describe this characteristics, the edge points need to be linked into edge curves. Moreover, any evaluation function can at best remove "almost all" noise-induced response, not all of them. The continuity of the edges can be viewed as another approach remove noise response, since random noise will not be able to form any continuous consistent profiles. However the continuity of an edge cannot be

measured by profile evaluation functions. A long edge is definitely more significant than a shorter one, and an edge point with a profile of strong contrast is generally of no importance if no other point with similar profile can be found in the vicinity of this edge point.

Most of the earlier works in edge linking only involve the continuity of the positions and orientations of the edges, no other property is employed. Except for the parametric fitting method and the image segmentation approach, edge detectors do not retrieve most of the important edge attributes such as colors values, and widths of edges.

Therefore, we incorporate the edge linking into our edge detection approach by extending an initial detected edge point into a group of edge points that form an edge. Since the profile of each edge point carries most of the edge attributes, if the edge points are linked only if they have similar edge attributes, continuity derived is much better than the results using only positions and orientations.

There are three stages in our edge tracking process:

1. Initialization. The initialization stage ensures the existence of an edge in the vicinity of an initial point. A certain number of adjacent edge points with a consistent edge profile is required. Otherwise, the initial edge point is considered as an isolated edge point and is thus ignored.

2. Extension. Once the presence of an edge is confirmed by its initial length, both ends of the edge will continue to extend. This stage ends when no further edge point with compatible profile can be found, or the new point is near some other existing edge point with similar edge profile. Prediction of new edge points according to the existing edge points can speed up the edge

growing process, and reduce the effect of noise and occlusion. A simplest and most obvious example of such prediction is for the case of straight edges.

3. Refining. Once the edge stops growing, some refining processes can be performed. If the two endpoints of the edge are very close, these two points can be joined to form a loop. If one endpoint is very close to an edge point of another edge, and these two edges have similar profile, these two edges might be joined into one single edge.

Edge growing process is required in the first two stages. The only difference is that in the first stage, the growth will stop once the requirement of the length is satisfied. If the requirement is not satisfied, the edge is discarded. We will first describe this process before going into details of these three stages.

4.1 Edge Growing Process

The edge growing process refers to the way we extend the edge point by point, beginning from a selected and then refined initial edge point. Two edge points will grow from this initial edge point. Two more new points will derived from these two points, as edge growing process grows outwards. This process repeats until certain situation occurs that would stop the growth.

There are two cases for edge growing process:

- The given edge point is an initial edge point and is not attached to any point yet. This point will be the first point of the edge. Two edge points are expected to grow from this point.

- The given edge point is grown from some other edge point. Only a new edge point is expected to grow from this point, in the opposite direction from where the point grown from.

Both cases are treated similarly. We will discuss the first case first. With the initial edge point, (x_0, y_0, θ_0), as well as the positions of two neighbor edge points $(x_1, y_1), (x_2, y_2)$, where $x_1 = x_0 - k\sin\theta_0$, $y_1 = y_0 + k\cos\theta_0$, $x_2 = x_0 + k\sin\theta_0$, $y_2 = y_0 - k\cos\theta_0$, and k a pre-defined distance between the two edge points, the oriented points (x_1, y_1, θ_0) and (x_2, y_2, θ_0) can be used to search for two more edge points (for explanation of neighbor edge points, see Section 3.7). The process is to locate the new edge points according to the attributes of the initial edge point. A prototype edge profile can be constructed from information such as the colors on both sides and the noise magnitude to localize the new edge points,

New edge points can then be localized by searching the vicinity of the neighbor edge points of the initial edge point. Because a template profile is available, the process described in Section 3.6.2 can be used to construct the prototype edge profile.

The new edge point for each side is chosen as the point with a maximal SNR among those points in the vicinity of the neighbor point which pass the validating process. The vicinity of the neighbor point (x_1, y_1, θ_0) is defined as $(x_0 - k\sin(\theta_0 + \rho), y_0 + k\cos(\theta_0 + \rho), \theta_0 + \rho)$, where $\rho \in [-\rho_0, \rho_0]$, and ρ_0 a constant. In other words, it is an arc centered at the initial point (x_0, y_0). The orientation of this point is then refined as described in Section 3.7.

Once the maximal SNR is smaller than one, no eligible edge point can be found in the vicinity. Therefore, the edge growing process stops. Another condition

that will stop the edge growing process is when a new edge point is close to another existing edge point with a similar profile. This condition is set to prevent redundant growing. If the existing edge point is from the same edge, the edge can be closed into a loop. If that point belongs to a different edge, it is possible that these two edges can be joined together to form a single edge. The closing and joining process will be discuss later.

For an edge point P which has already been attached to another edge point P^-, the new point P^+ grown from P will be on the other side of P^-. The angle between $\overrightarrow{P^-P}$ and $\overrightarrow{PP^+}$ should be sharp. Therefore, only one of the two neighbor edge points will be used to grow an edge point. The purpose of this constraint of the angle is to disallow sharp change of curvature direction during edge growing process. This prevents sharp corners in the edge detection. While this might cause some inconvenience since sharp corners, which do not occur very frequently, are not very rare either. Such inconvenience can be handled by allowing joining two separate edges together later. With the exception that only one edge point is grown, the growing process is exactly the same as the process from the initial edge points.

4.2 Edge Initialization

The initialization of an edge is to produce a short line segment as the initial edge curve. This procedure could be considered as part of the noise eliminating process. Only the edge point that can be extended to an edge curve of specific length will be accepted. To a certain degree, this procedure is similar to Canny's hysteresis threshold process [7] as they both try to eliminate weak edges caused by image noise. Instead of filtering edges by their maximum strength, our process focuses on

the length. This required minimal length of the edge curve can be a pre-determined constant, which can be presented as some function of the edge profile. For example, an edge profile with lower contrast may require a longer length to be verified. Moreover, a profile with complex transient region may require more consistent profile matching to ensure this is really a single composite edge instead of multiple step edges. On the other hand, the choice of the length is pretty subjective, and varies in different situations. In certain cases, such as OCR(optical character recognition), any constraint of edge length might result in losing some important features. For the time being, this threshold of minimal edge length is set to a small constant value whose value is about six to ten pixels.

We can use the hysteresis method to improve the selection of the threshold of minimal edge length. A high threshold and a low threshold for the edge length are defined in advance. Presently, the high threshold is five pixels and the low threshold is three pixels. Our process of edge initialization can be described as follows:

1. In order to find the position and orientation which yields minimum with the given evaluation function, one can shift and rotate the given initial oriented point in its neighborhood. The evaluation function will produce abstract profile information such as the color information and noise measurement of both sides of the profile of the initial edge point. If no valid edge point can be found, the edge initialization process fails.

2. Using the abstract profile information of the initial edge point, grow the edge on either side of the initial edge point according to the edge growing process described in the previous section.

3. If the length of the edge could not even reach the low threshold, the edge initialization process fails. A special case is when both sides stop because they reach the vicinity of some other edge points with similar profile. In this case, this edge initialization process is considered partially successful, and will move directly to the edge refining process so it can be joined with another edge.

4. If the length reach the high threshold, the edge initialization process is successful.

5. If the length is between two threshold values, the initial edge point is replaced by one endpoint, and the high threshold is reduced by one. If it is smaller than the low threshold, the edge initialization process fails, otherwise it is start over from step 2.

The last step requires some explanation. As our edge detection is subpixel-level based, a slight change of the initial oriented point may yield completely different set of edge points. Most of time, the two edges look exactly the same, just with edge points of different positions. However, sometimes one edge might be shorter than the other because of noise or digitalization. In other words, a change of initial condition might have some unpredictable result. Therefore, it is expected that the change of the initial edge point will end up in either step 3 or step 4.

The result of this initialization process either succeeds or fails. If it succeeds, the initial edge curve is passed to the edge extension process. If it fails, no edge is marked.

4.3 Edge Extension

As the name of this process suggests, it will extend the initial edge curve of both ends. This process is quite straightforward. Both sides will be extended using the edge growing process mentioned in Section 4.1, where the stop conditions are also defined.

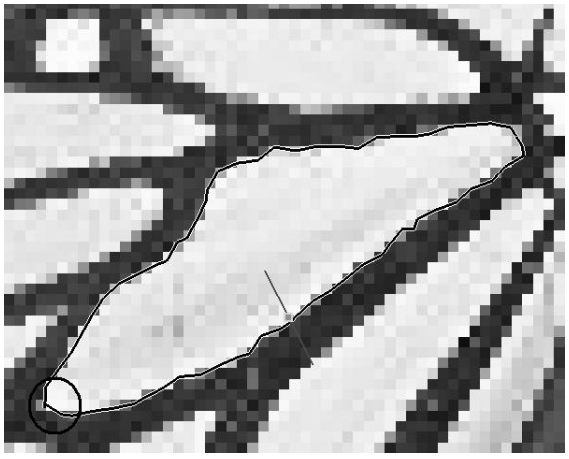
4.4 Edge Refining

This is the most involved process. In fact, there are countless processes that can be considered as part of the edge refining process. We will only discuss three of the most fundamental processes: edge closing, edge joining, and edge smoothing.

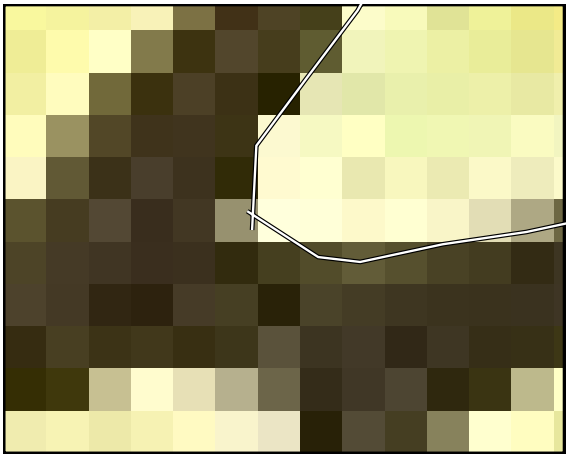
4.4.1 Edge Closing

When two end points of an edge are close enough, it might mean that this edge form a closed loop. To verify the possibility of a closed edge, the middle point of the two edge points becomes the position of a new oriented point, while the orientation of this new oriented point is set to be perpendicular to the line segment formed by the two points. The profile of this new oriented point is then compared with the profile of the edge. If they are similar enough, then this oriented point is added to both ends of the edge, thus the edge is closed to form a loop.

Sometimes, due to image noise or digitalization, a few end points might need to be removed before the loop-closing is performed. Figure 4.1 shows one of the situations that requires further processing before the closing. The points to be removed are determined by the smoothness of the curvature around the closing.

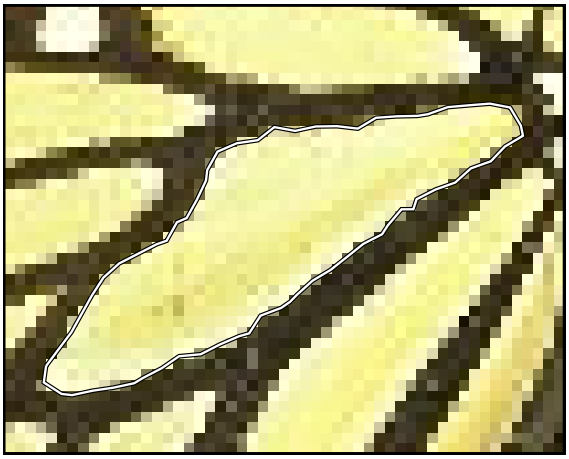


(a) part of Figure 3.1(f).

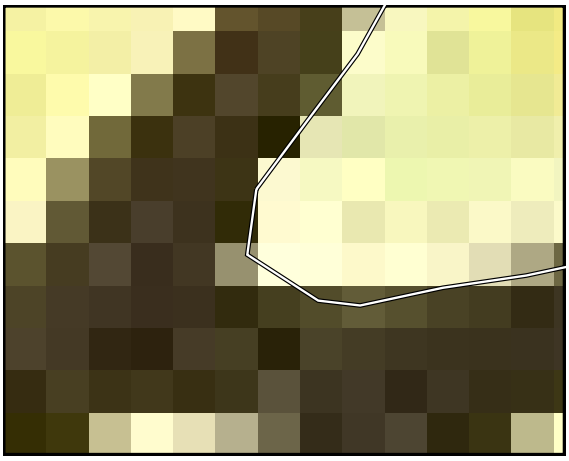


(b) Magnified part of Figure 4.1(a) at the bottom-left corner.

Figure 4.1: An edge loop that requires removal of some edge points at the endpoints before closing.



(a) Closed loop from the same edge in Figure 4.1(a).



(b) Magnified part of Figure 4.2(a).

Figure 4.2: The closed edge loop, after removal of one edge point.

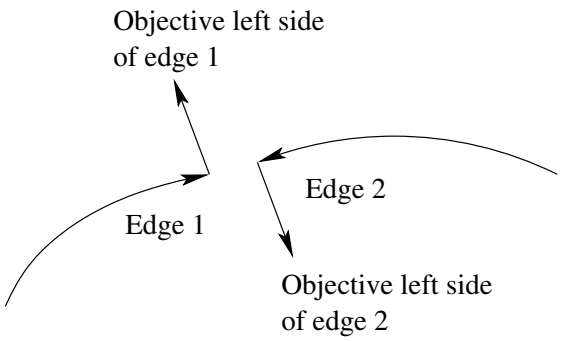


Figure 4.3: Definition of **objective left side** of an edge endpoint during an edge joining operation.

For the edge points P_i's, $i = 1 \ldots n$, of the edge e, we will remove $n - m$ points at the end, to make sure that the the angles between vectors $\overrightarrow{P_2P_1}$ and $\overrightarrow{P_1P_m}$ and between vectors $\overrightarrow{P_1P_m}$ and $\overrightarrow{P_mP_{m-1}}$ are small.

4.4.2 Edge Joining

Due to noise or occlusion, an edge might be broken into separate edges. While it is possible to enforce the edge continuity, there are too many cases which prevent the edge continuity to be established. A reasonable approach is to join broken edges back. Without the edge profile information, this is very challenging. Two edge curves with close endpoints may or may not belong to the same edge, depending on their profiles. The most important information of the edge profile in this process are

- Colors on both stable regions of the edge. They are the most significant information of an edge. If these colors on the two edges are similar, then the two edges might be joinable.

- Noise magnitude of the edge. This is the parameter used to determine if the colors of two edges are similar or not. If their difference is below the noise magnitude, they are similar.

- Edge normal directions and edge directions of the endpoints. The edge direction of an endpoint is defined by the direction from the edge point adjacent to this endpoint to this endpoint. The edge normal direction is defined as the orientation of each edge point. In profiles, we have left color and right color. However, left and right are defined relative to the orientation of the edge point. The same profile will be obtained if the orientation of the edge point is rotated by 180°, however, the definition of the left side and right side are now reversed. In order to establish some common ground for two edges, we will define a more objective term of edge normal directions from the edge direction. As the edge directions of the two would-be-joined endpoints should face each other, the **objective left side** of one edge is compared against the **objective right side** of the other edge, and vise versa. Here the **objective left side** should be the direction that is counterclockwise perpendicular to the edge direction, and the **objective right side** clockwise perpendicular. See Figure 4.3 for the definition of the objective left side of an edge endpoint. The normal direction marks the direction from the left side to right side of the profile. If the left side and the **objective left side** contradict each other, the left side and the right side of the profile should be inverted before the comparison.

In most cases, a closed edge will not join with another edge, since technically a closed loop does not have end points. Therefore, an closing process is performed

on an edge first. If it is a closed edge, no joining is required.

The edge joining process consists of the following steps:

1. Find all the edge points near one endpoint p_1 of the given edge.

2. For each edge point p_2 found, if it belongs to the given edge, or a closed edge, it is ignored.

3. If p_2 is an endpoint of edge e_2,

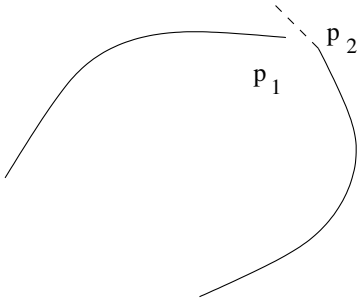
   (a) find its edge direction, then its objective left side and objective right side;

   (b) compare the color on the objective left side of p_2 to the objective right side of p_1, then the objective right side of p_2 to the objective left side of p_2. If the difference is within the range of the noise magnitude in both comparison, p_1 and p_2 might be eligible to be joined. Otherwise, the joining fails.

   (c) If the distance between p_1 and p_2 is smaller than one, let p_2^+ be the edge point adjacent to p_2. Otherwise p_2^+ is the same as p_2.

   (d) Get the point p_c at the center between p_1 and p_2^+. Let the orientation be clockwise perpendicular to the vector for p_1 to p_2^+. Compare the profile of p_c and the edge profile of the given edge. If they are similar, the joining process can be performed.

4. If p_2 is not an endpoint, but only an edge point of e_2, then points on one side of p_2 might need to be removed before joining the two edges (see Figure 4.4 for an example).



(a) Two separated edges can be joined, though some points should be removed to make a smoothed joining.

(b) Two joinable edges

(c) Magnified of part of Figure 4.4(b)

Figure 4.4: One of the problems in edge joining process.

(a) The edge direction of p_2 has to be defined. First compare the left color of p_2 with the objective right color of p_1, then right color of p_2 with the objective left color of p_1. If the difference is within the range of the noise magnitude, the left side is defined as the objective left side. Otherwise, check if the right side can be defined as the objective left side. If the objective left side cannot be defined, the joining process fails.

(b) With objective left side defined, edge direction can be derived as the direction clockwise perpendicular to the objective left side. Now p_2 is considered as an endpoint of e_2, and all edge points beyond p_2 are ignored for the joining purpose. Perform continuity verification similar to the steps 3c to 3d.

5. If an edge point p_2^+ is considered joinable with p_1, add all edge points of e_2 up to p_2^+. Then the edge e_2 is removed, and the joining process is successful. Otherwise, try to join the next near-by edge point.

4.4.3 Edge Smoothing

Unless the edge points of an edge are obtained using some specific parametric forms such as straight lines, circles, or ellipses, the effect of digitization of an image would make the edge zigzag. Even subpixel-level edge detectors would only damp the degree of zigzag, not remove it completely. A straightforward approach to smooth a curve is to straight the curve locally. That is, for each edge point $P(i) = (x((i), y(i))$, $i = 1 \ldots n$, move its position toward the middle of its two neighbors points, which is $(P(i-1)+P(i+1))/2$. This process should be performed repeatedly, since the oscillation of the curves are not necessary one pixel. The process can be

written as

$$P_{t+1}(i) = (1 - \alpha(i))P_t(i) + \alpha(i)\frac{P_t(i-1) + P_t(i+1)}{2}, \tag{4.1}$$

where $\alpha(i) \in (0, 1)$ is the parameter that control the degree of smoothing, and t is the number of iterations. For this iterating model, a few problems should be noted.

- This smoothing process smoothes not only zigzag, not also real corners.
- This process by itself would converge into a straight line eventually.

The "active contour" model [27] would be more appropriate in the aspect. While this model requires some proper initial edge curve and makes it not as widely used in edge detection of single image as in motion picture. Such limit is no longer a problem since we have already obtained the edge curve as the initial condition. On the other hand, assuming that the edge points are of subpixel-level accuracy, and only require minimal adjustment, then this model is not suitable since the **image energy** is not effective in subpixel-level, only **internal energy** remains effective. Therefore the degree of smoothing only depends on the curvature of each edge points.

Equation 4.1 can be modified, so it is applicable for the purpose of edge smoothing. To keep the real corners, there are two possible approaches. This first approach would be setting $\alpha(i)$ as a decreasing function of the distance of $P(i)$ and $(P(i-1) + P(i+1))/2$. In other words, the larger the curvature, the less the smoothing is performed. The other approach is to re-calculate the evaluation function every time an adjustment of edge point is made. Once the evaluation function determines the new adjustment is invalid, the smoothing stops. In fact, there is no conflict in these both approaches so they can be used at the same time.

The energy minimization concept of "active contour" model is also useful. It can be applied by separating each iteration of smoothing into two stages: smoothing and measuring. The smoothing stage generates the new edge points, while the measuring stage validates the new set of edge points. In the smoothing stage, the new edge points are calculated according to Equation 4.1. In the measuring stage, two sets of energy are calculated: profile energy and internal energy. The internal energy is similar to the internal energy in active contour model, and the profile energy is the sum of the evaluated values of all the edge points according to the evaluation function. Since the profile energy is not differentiable, Euler equation used in active contour model is not applicable. Therefore, in the measuring stage, we simply compare the total energy before smoothing with the total energy after smoothing. The smoothing process will continue as long as the total energy decreases.

A disadvantage of this smoothing model in comparison with the active contour model is that the former does not have the ability to adjust edge points locally. Unlike the active contour model which adjusts each point according to the feedback of the energy function, our smoothing process only uses the sum of the energy along the edge to decide whether to make more adjustment or not. It is possible that some part of the edge will not be smoothed properly because further smoothing will move another part of the edge curve away from the edge in the image.

In order to address such problems, we further modify our smoothing approach to smooth edges piecewisely. A smoothing process now will only process a small piece of an edge curve each time, and move on to the next piece once the smoothing of the current piece is complete. Note that there should be some overlap between two

pieces of edge segments to ensure the smoothness between pieces. To summarize the smoothing process, the steps are as follows:

1. The process begins from the first point as the initial smoothing point.

2. Select the next n edge points starting from the initial smoothing point and label them as $P_1, P_2, \ldots, P_n$. In our experiment, $n = 8$.

3. Calculate the internal energy

$$E_{internal} = \sum_{i=1}^{n-1} |P_{i+1} - P_i| + \sum_{i=1}^{n-2} |P_{i+2} - 2P_{i+1} + P_i|$$

   and the profile energy

$$E_{profile} = \sum_{i=1}^{n} F_{eval}(P_i),$$

   where F_{eval} is the chosen evaluation function.

4. Combine $E_{internal}$ and $E_{profile}$ into total energy

$$E_{total} = aE_{internal} + bE_{profile},$$

   where a and b are the control parameters. Because $E_{profile}$ tends to have larger magnitude than $E_{internal}$, a is assigned a larger value to balance the effect of the two energy functions.

5. For the point P_i, $i = 2 \ldots n-1$, calculate the new edge points

$$P_i^+ = (1 - \alpha)P_i + \alpha \frac{P_t(i-1) + P_t(i+1)}{2},$$

   where

$$\alpha = e^{-d/2}/2,$$

(a) An edge before smoothing. (b) The same edge after smoothing.

Figure 4.5: The outcome of an edge smoothing process

and the distance between old edge point and new edge point

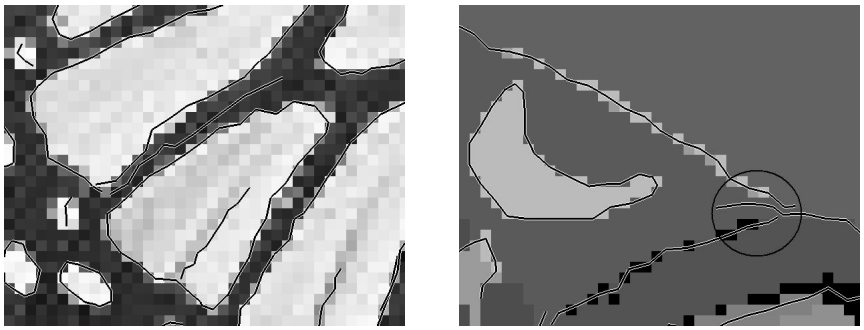
$$d = \left\| P_i - \frac{P_t(i-1) + P_t(i+1)}{2} \right\|.$$

Make $P_1^+ = P_1$, and $P_n^+ = P_n$.

6. Calculate the evaluated values for all P_i^+'s, if any of the values is greater than the threshold value 0, reduce the magnitude of adjustment by replacing α by $\alpha/2$ and repeat the previous step, unless a valid edge point is found, or α is too insignificant. If no valid edge point can be found for P_i^+, $P_i^+ = P_i$.

7. Get the total energy E_{total}^+ for the newly generated edge points. If $E_{total}^+ \leq E_{total}$, replace P_i with P_i^+, E_{total} with E_{total}^+, then go back to step 5. Otherwise, the smoothing of this piece of edge curve is finished. Move the initial smoothing point $n/2$ points forward, then go back to step 2 to smooth the next piece.

Note that this modified smoothing process would only move each edge point toward the center of its two neighbor edge points. In other words, its only purpose



(a) Magnified part of Figure 3.1(f). (b) Magnified part of Figure 3.1(b).

Figure 4.6: Two figures with **phantom edges**. The one in (a) is completely stand-alone, while the one in (b) is extended from a real edge.

is smoothing. If the original curve is incorrect, this process might not move toward the real edge, as in the case of the active contour model. An example of the outcome of smoothing is shown in Figure 4.5.

4.5 Problems of Edge Growing

In most cases, the outcome of the edge growing process depends on the evaluation functions, the edge growing process itself seems more straightforward. However, certain details need to be defined:

1. When to stop growing? Even though in the previous section, the conditions to stop the edge growing have been defined, no all possible cases are covered. For example, one of the conditions is that the colors on both sides of the edge do not match, with tolerant value twice the noise magnitude. For a color image, we notice that if the tolerant value is applied to the difference of three color channels separately, the edge will tend to be fragile. However, if we take

the average of the difference of the three color channels instead, the result would be much better since single channel noise would be smoothed by the other two channels. Another example is the tolerant value for the width of the transient region. In our experiment, we limit it to vary within one pixel width. However, such limit is merely from empirical result. Greater tolerance tends to make the detected edges move away from the center of the edges, while smaller tolerance will likely break the edges.

2. Another problem plagues this edge growing process is the **phantom edges**. This usually happens when the generalized evaluation function (see Section 3.4) is applied, or when the results of more than one evaluation functions are combined. Two unrelated ramp edges might be treated as a single ridge edge if they are close to each other. As long as the two ramp edges do not suddenly depart away each other, this ridge edge will be extended. Two examples of such **phantom edges** are shown in Figure 4.6. In Figure 4.6(a), an edge is grown between the two parallel edges. The other example is shown in the circled region in Figure 4.6(b), a valid edge is grown into a region between two unrelated edges, and the edge transforms from a ramp edge category into a ridge edge category. Since the transient region is ignored due to its noisy situation, such transform is unnoticed by the tracker. To reduce the number of occurrences of the phantom edges of the type shown in Figure 4.6(b), one can enforce consistency within the transient regions of all the profiles on the same edge. However, this will cause problems for the dotted outlines appearing in the same image since the outlines are not very consistent when we view it closely, not to mention the effect of digitization. Another solution is to

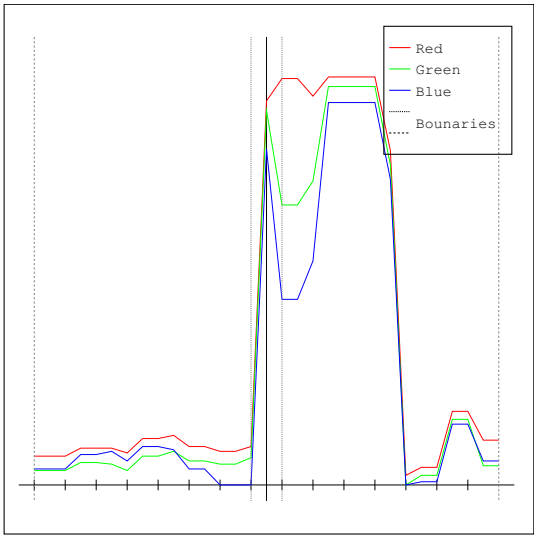
Figure 4.7: An edge detected from an image

remove every ridge edge that is parallel to some other edge with overlapped boundary. This will remove the case for the phantom edges of the type as depicted in Figure 4.6(a). If we remove part of all ridge edges that is parallel to some other edges with overlapped boundary, most of the phantom edges can be removed.

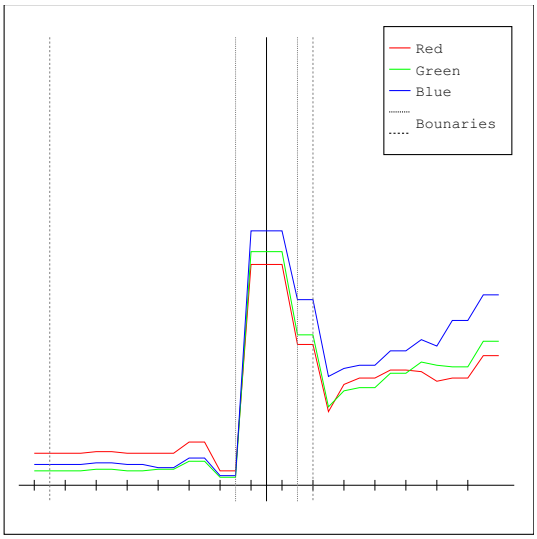
4.6 Edge Profiles

As the edges are collections of edge points, an edge profile can be considered as the collective representation of the profiles of the edge points on the edge. An edge profile provides a simplified description of the edge. Such description can be used as the base for low-level image processing tasks such as joining edges separated by image noise, or higher-level tasks such as categorizing edges on an image, or even multiple images.

Conceptually, the edge profile is the average of the edge point profiles along the



(a) The profile of one endpoint of the edge in Figure 4.7.



(b) The profile of another endpoint of the edge in Figure 4.7.

Figure 4.8: Compare the profiles of two different edge points on the same edge.

edge. In a real image, it probably is over-simplified. The approximation of other edges, the alignment of digitization, the image noise, among other effects, will eventually reduce the accuracy of most of the characteristics of the edge instead of emphasizing them. For example, we take two profiles from the same edge as shown in Figure 4.7. The profiles on the two endpoints are shown in Figure Figure 4.8. In this example, only one side of stable regions remain consistent. Moreover, since the width of the transient region has also changed, by the edge point profiles themselves, these two profiles should not be together.

Here we will give the edge profiles a more abstract form than the profiles of edge points. Most of the raw data in the profiles are discarded. What remain in an edge profile are the processed data such as the color values on both sides, the width of transient region, the noise magnitudes on both sides, and the edge magnitude. These data are obtained by averaging the respective fields in the profiles of edge points on the edge. Such abstract information is usually enough for the purpose of comparing profiles. In some sense, our edge profile approach is very similar to those of parametric fitting edge methods. The major difference is that our approach is one-dimensional based. The rationale of the choice of one-dimensional profile is described in the previous chapter. For models using the two-dimensional profile, oftentimes, they cannot handle corners or even curvature edges very well, especially for in choosing of threshold of the fitting error. On the other hand, in most cases, one-dimensional model fails to determine the edge orientations precisely. Still, the edge orientations can be approximated by the image gradient, and adjusted from the positions of the adjacent edge points.

During the edge growing process, the edge profile is updated for every new grown

edge point by taking average of the parameters from the profiles of all edge points on the edge, and with the exception of the first few points, the edge profile is used to be compared with the profile of the candidate edge point, instead of the profile from the previous edge point. The reason that the edge profile replaces the profile of the previous edge point is obvious. Since the parameters profile comparison, such as left colors and right color, are calculated statistically from pixel values of the image, and the source of an edge profile comes from those of multiple profiles of edge points, the edge profile is more reliable statistically.

The edge profile also includes some additional parameters that are not part of an edge point profile. Among them is the length of the edge. As noted previously, a longer edge is generally more significant than a shorter one. Therefore, the length of an edge is always an important factor when judging the importance of the edge.

Another useful parameter is the list of adjacent edges at the end points, which can be used when joining edges, or defining borders of objects. Since this parameter is not complete until all the edges in the image are identified, we will discuss it in the chapter about post-processing of the edges.

4.7 Straight Edges and Prediction of New Edge Points

Straight edges are special cases for edge detection. Many edge detection methods focus on straight edges since once an edge is known to be straight, it can be measured very accurately even with the presence of image noise and alignment. Hough transform [19] is one of the well-known methods, even it is not designed specifically for edge detection.

To take advantage of straight edges in our profile-based edge detection model,

edge smoothing process should be utilized first. Fortunately, since our edge smoothing process can be done piecewisely, it can be performed during the edge extending process. If the piece of edge curve at the edge point becomes almost straight after smoothing, we can predict the new point by extrapolating the straight line. In the case that the extrapolated point has a compatible profile, the local minimization of vicinity profile evaluation can be omitted.

Since each step of the minimization of the profile evaluation requires the sampling and evaluation of multiple profiles, a successful extrapolation will greatly improve the performance of the edge detection by reducing the complexity of computation. Moreover, since the extrapolation is calculated from the smoothed edge points, the extrapolated point is smoothed by default, unaffected by any noise.

To incorporate this straight edge extrapolation process, which we call **edge point prediction**, the process of the edge growing will be modified. In our original model, the the three steps for edge growing are: initializing, extending, and refining. Smoothing is part of the refining step that is performed once the edge growing process has completed. The modification is to move the edge smoothing into the edge extending step. For every other new grown edge point, a piecewise smoothing is performed at the edge endpoint. Once the smoothing finishes, a simple check of straightness is performed. The edge piece is straight if their directions are consistent. If it is considered straight, an extrapolated point is generated along the straight line, then its profile is compared to the edge profile. If the profiles are compatible, the edge growing process is omitted and another extrapolated point is generated. The extrapolation stops when an extrapolated point is not acceptable. The edge growing process then resumes its minimization process.

Certain curvatures other than straight lines can be extrapolated as well. A circle, or part of it, for example, can be extrapolated easily, since as long as the distance between two adjacent edge points are constant, the angular difference of the edge directions between two adjacent edge points is consistent. While planar circles are not as common in the real world as other curves such as ellipses and parabola, small pieces of such curves can usually be matched to circles, with very little errors.

The edge point prediction can easily be extended to be applicable by other shapes. However, allowing too many such extension in the approach introduces severe performance degradation, which may compromise the advantage offered by the edge point prediction. Therefore, we will apply only linear and circular prediction models along with the edge growing process.

4.8 Edge Detection of the Whole Image

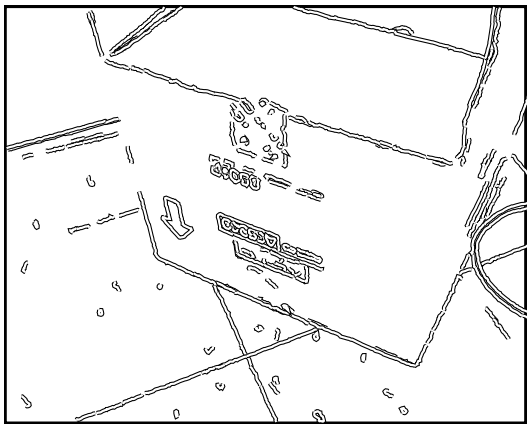
With all the edge growing processes mentioned in this chapter, we can localize an edge with a given initial oriented point. However, to detect edges in a given image using this approach without any user-input, one has to traverse the whole image, using every pixel as an initial point and try to grow an edge from it. An optional threshold may be applied to the image gradient. Only the pixels with gradient value greater than the threshold value will be used as the initial points. This improves the performance of the speed, at the cost of missing weak edges.

This process is very similar to normal edge linkers. However, instead of linking existing points as in most edge linkers, in our approach, the points "grown" from the initial points are calculated on the fly. This process is not by any mean original,

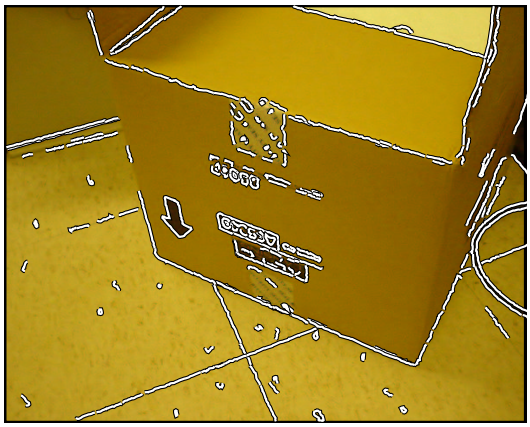
as an almost identical edge-tracking process has been described in [22]. However, since edge linking is seldom performed at the same stage as the edge localization, such method is not widely used in edge detection. Another major difference is that this growing process uses subpixel level precision. The reason behind the subpixel level precision is mentioned at the beginning of Chapter 3. The use of subpixel level precision increases the complexity of the image traversal. As mentioned briefly in Section 1.3, a hash table is used to keep track of all edge points added. When a pixel is traversed, a test of approximation is performed in the vicinity of that pixel. Only if no existing edge point is near that pixel, a new edge can be grown from that pixel. Moreover, this hash table allows us to find edges that close to a certain edge for edge joining, in addition to remove redundant edges.

Some results of such processing are already shown in Figures 3.1 (b), (d), and (f). More results are shown in Figures 4.9–4.15. All except Figure 4.9 are detected by the step edge detector. The only exception is for Figure 4.9, which is detected using the generalized edge detector. The reason we choose the step edge detector is for its simplicity. There are fewer arbitrary conditions that generate phantom edges. The detector performs well when texture is not a problem, and when there is no obvious illuminating effect such as shadow, such as in Figures 4.11, 4.12, 4.14. In Figure 4.11, the unfocused background makes the localization harder, and some of the edges are detected twice, though. For the Lenna image in Figure 4.10, because of illumination, a few weak and generally unnecessary edges, most of them on the face and the shoulder, are detected. In Figure 4.13, the refection on the smooth surface of a canbinet is located. The edge caused by shadow at the bottom-right corner is also detected. However, because this edge is wider than general step

(a) An image with a box on floor.



(b) Edges detected from Figure 4.9(a).



(c) Detected edges in Figure 4.9(b) super-imposed on the original image in Figure 4.9(a).

Figure 4.9: A result of the edge detection.

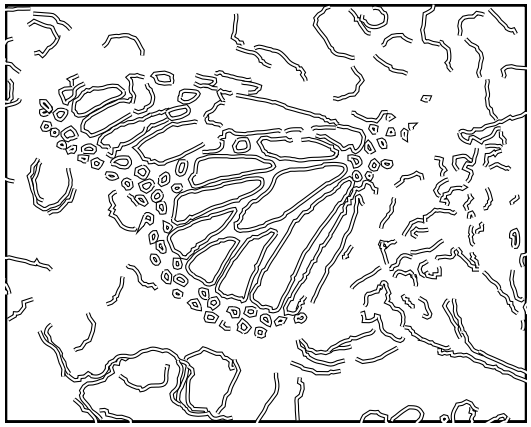
(a) The Lenna image.

(b) Edges detected from Figure 4.10(a).

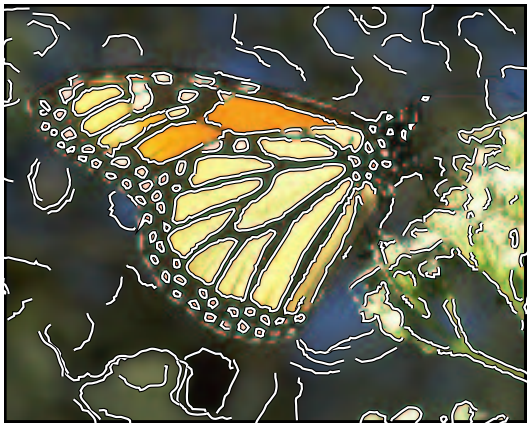
(c) Detected edges in Figure 4.10(b) superimposed on the original image in Figure 4.10(a).

Figure 4.10: A result of the edge detection.

(a) The image of a butterfly.



(b) Edges detected from Figure 4.11(a).



(c) Detected edges in Figure 4.11(b) superimposed on the original image in Figure 4.11(a).

Figure 4.11: A result of the edge detection.

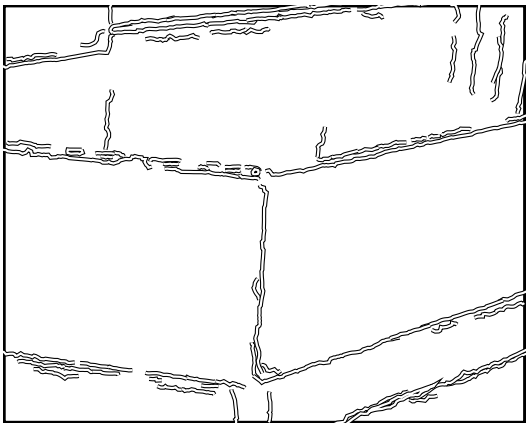
(a) The image of a flower.

(b) Edges detected from Figure 4.12(a).

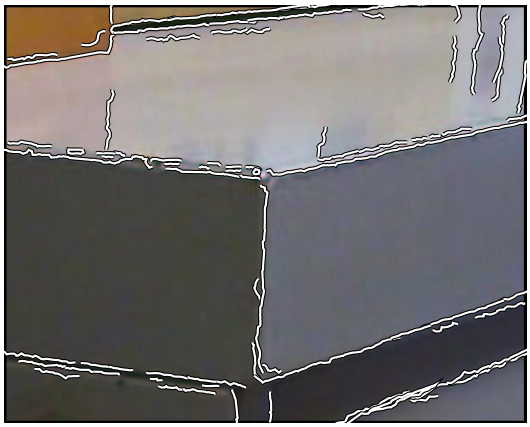
(c) Detected edges in Figure 4.12(b) superimposed on the original image in Figure 4.12(a).

Figure 4.12: A result of the edge detection.

(a) The image of a corner of a canbinet (note the reflection).

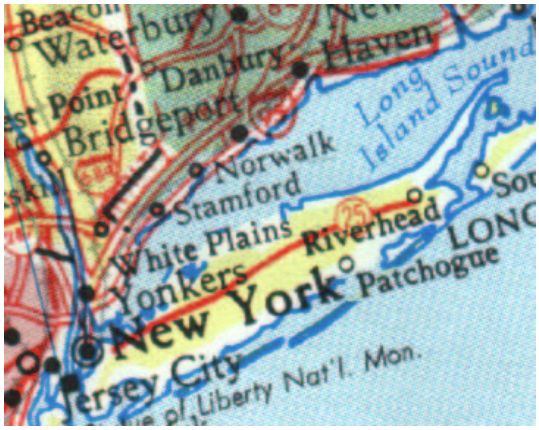


(b) Edges detected from Figure 4.13(a).



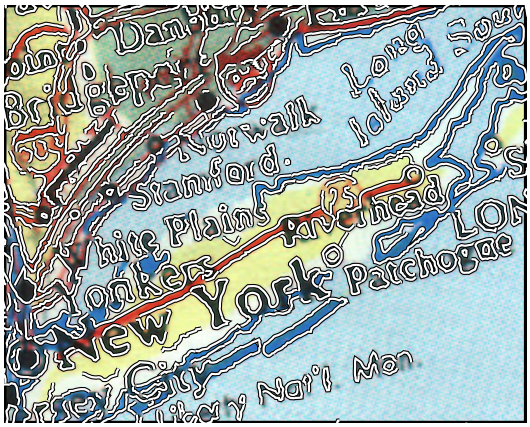
(c) Detected edges in Figure 4.13(b) superimposed on the original image in Figure 4.13(a).

Figure 4.13: A result of the edge detection.



(a) The image scanned from a map.

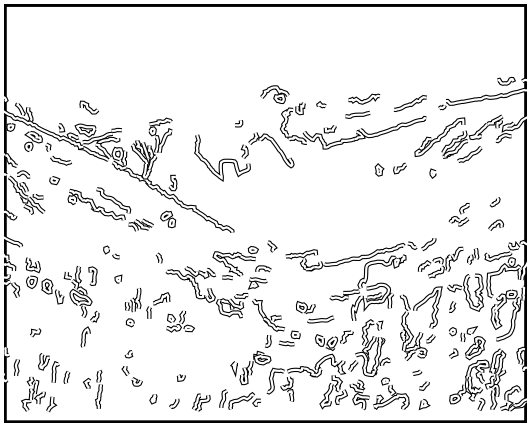
(b) Edges detected from Figure 4.14(a).



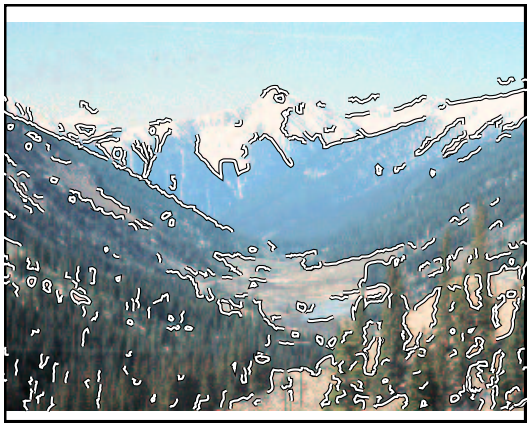
(c) Detected edges in Figure 4.14(b) superimposed on the original image in Figure 4.14(a).

Figure 4.14: A result of the edge detection.

(a) The image a picture with mountains and a valley.



(b) Edges detected from Figure 4.15(a).



(c) Detected edges in Figure 4.15(b) superimposed on the original image in Figure 4.15(a).

Figure 4.15: A result of the edge detection.

edges, the localization is not performed very well for our step edge detector. The mountain image in Figure 4.15, however, is the major challenge. It is a very texture-rich image. Almost all the trees are neglected, not that we expect a better result in this aspect. As all the trees have similar colors, an edge between two trees can be distinguished from both global texture pattern (as the almost parallel vertical lines), and the slightly different illumination. Our detector is not sophisticated enough to handle either.

Chapter 5

Further Analysis

In earlier chapters, the architecture of our edge detector is described. We now discuss some possible concerns regarding our edge detector.

- The first concern is the robustness of our detector under different noise conditions. We will compare our detector to Canny's in this regard.

- The second concern is the presence of implicit parameters in our edge detector. All edge detectors have such parameters. One of our goals is to minimize the dependence on such parameters, which makes the edge detection process as automatic as possible (otherwise users will have to "tune" our detector for each application). Our built-in noise measurement feature is a major contributor to this automation. The facade of the automaton is partly achieved by several implicit parameters which we occasionally mentioned. For instance, we have pre-defined values for the required minimal edge length and the gap between successive samples in a profile. We will investigate the behavior of our edge detector when these implicit parameters are varied. One such parameter is

the influence of various interpolation schemes used by our sampling process.

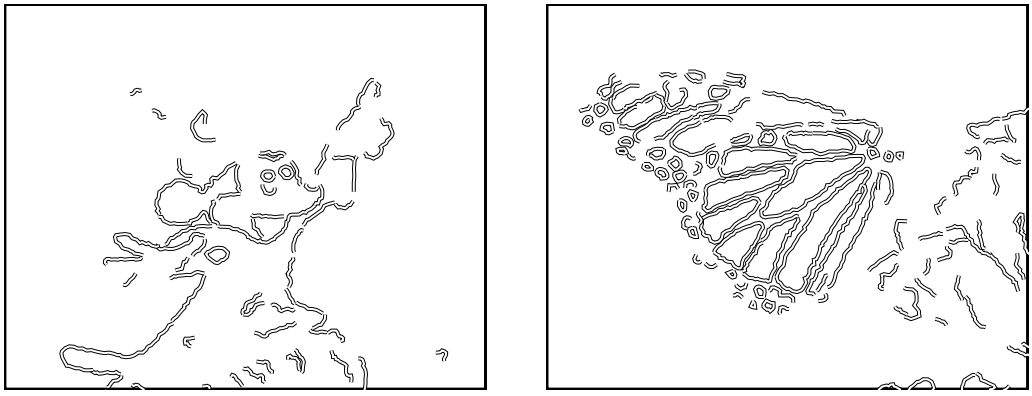
5.1 Image Noise

Image noise is always an important issue for edge detection. Considerable effort is usually required to reduce false edges induced by noise. More precisely, of the three stages of edge detection, namely, smoothing, enhancement and localization, the major task of both the smoothing and localization stage is noise removal.

(a) A generated noisy image from Figure 3.1(a) (b) A generated noisy image from Figure 3.1(e)

Figure 5.1: Two noisy images for test.

In our edge detector, there is no smoothing stage. Nevertheless, a considerable portion of our detector can be described as noise handling. Our detector estimates local noise, upon which all other operations depend. For instance, identifying edges and comparing two profiles for similarity are among such operations. Therefore our detector is expected to perform well with the presence of noise. While weak edges may still go undetected due to noise, the possibility of false edges caused by image noise is minimal. Note that it is not likely to derive continuous edge points with



(a) Edges detected from Figure 5.1(a)

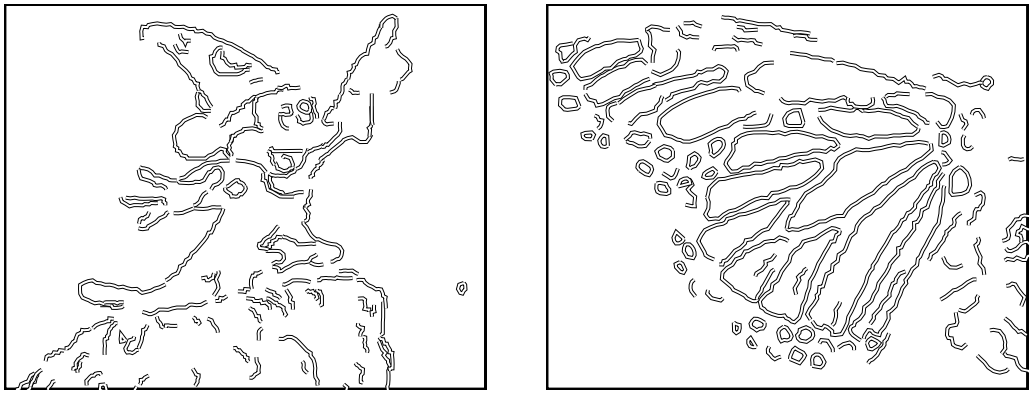
(b) Edges detected from Figure 5.1(b)

Figure 5.2: Edges from noisy images detected using our edge detector.

(a) Edges detected from Figure 5.1(a)

(b) Edges detected from Figure 5.1(b)

Figure 5.3: Edges in Figure 5.2 super-imposed on the noisy images.



(a) Edges detected from Figure 3.1(a)

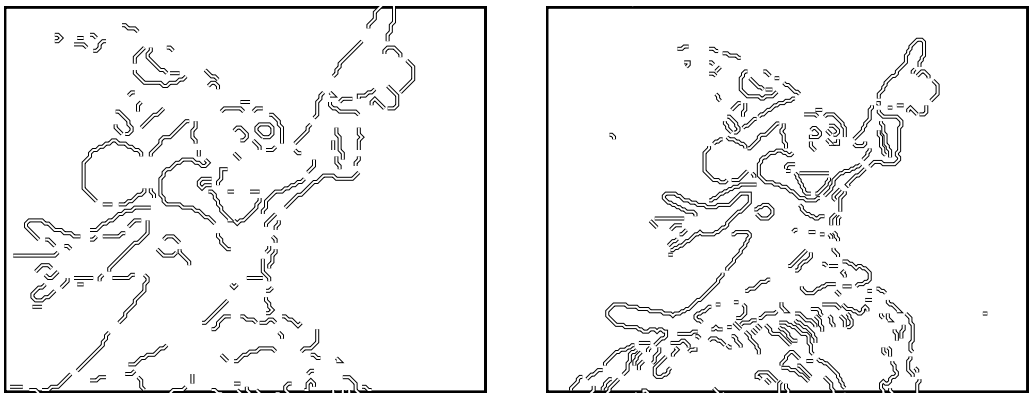
(b) Edges detected from Figure 3.1(e)

Figure 5.4: Edges from original images detected using our edge detector.

(a) Edges detected from Figure 3.1(a)

(b) Edges detected from Figure 3.1(e)

Figure 5.5: Edges in Figure 5.4 super-imposed on the original images.



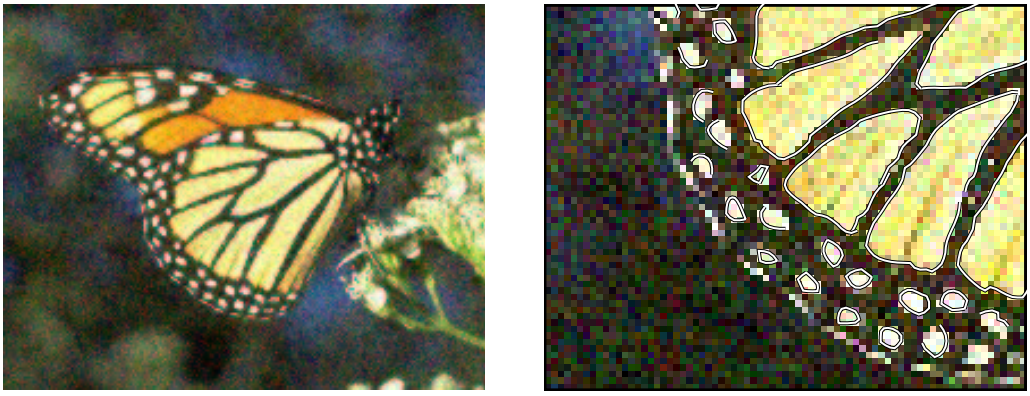
(a) Edges detected from Figure 5.1(a) using Canny's detector, $\sigma = 1$, $t_h = 36$, $t_l = 5$

(b) Edges detected from Figure 5.1(a) using Canny's detector, $\sigma = 2$, $t_h = 22$, $t_l = 5$

Figure 5.6: Results from Canny's detector for one of the noisy images.

consistent profiles from random noise. The following experiments will bear this out.

Images with heavy noise are generated for this demonstration. Starting from the images in Figures 3.1(a) and 3.1(e), we add a Gaussian noise with $\sigma = 40$ on all three color channels, producing the noisy images in Figures 5.1(a) and 5.1(b). The results of our edge detector on these noisy images are in Figure 5.2. The evaluation function for step edge is used. For comparison, the results for the original images are in Figure 5.4. Figures 5.3 and 5.5 display the same edges super-imposed on the images. Other than the low-contrast regions, such as the wizard hat in the Micky Mouse image, edges are well preserved even with heavy noise. We can also notice that there is no obvious false edges. There is still limit in the ability the edge detector can cope with noise. Once the added noise magnitude exceeds $\sigma = 50$, some of the significant edges begin to break up. Moreover, while our detector can filter noise, the process is time-consuming. The additional time spent comes from the evaluation and linking of false edges. For example, the original Micky Mouse



(a) A noisy image

(b) Edges detected from Figure 5.7(a)

Figure 5.7: Edges detected from a noisy image.

image takes the detector about 2 minutes, and the noisy one in Figure 5.1(a) takes more than 4 minutes under the same condition to complete the detection. For comparison, we also perform Canny's edge detector on the same noisy images. While it is still possible to detect edges as well as our detector, it requires proper settings for both the smoothing parameter σ and the hysteresis threshold values t_h and t_l. Figure 5.6 shows some of the results by Canny's detector.

Image smoothing is one of the widely used methods for noise removal. While our edge detector performs well with noisy images, we also experimented with smoothed images, using both Gaussian smoothing, and an anisotropic smoothing method which is loosely based on the nonlinear anisotropic diffusion introduced by Perona and Malik [44]. The purpose of this experiment is to observe the effect of image smoothing upon our edge detecting process. The noisy image and some edges detected are shown in Figure 5.7. And the smoothed images and their results are shown in Figure 5.8 and 5.9.

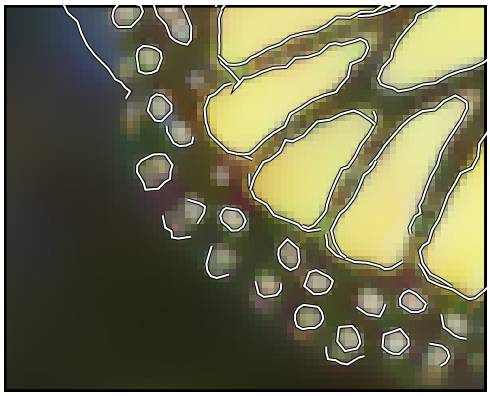
As expected, the edges detected from the smoothed images are smoother than

(a) A smoothed image of Figure 5.7(a) using Gaussian smoothing with $\sigma = 1$

(b) Edges detected from Figure 5.8(a)

Figure 5.8: Edges detected from a Gaussian smoothed image.

(a) A smoothed image of Figure 5.7(a) using anisotropic diffusion



(b) Edges detected from Figure 5.9(a)

Figure 5.9: Edges detected from an anisotropically smoothed image.

the noisy image, and the performance is better in general. However, there is also slight degeneration and displacement of the edges in the smoothed image. For example, proximate edges will interfere each other, and make the consistence of the profiles hard to maintain. This is shown in the results in Figures Figure 5.8(b) and Figure 5.9(b) as broken edge contours, and misconnected edges.

Therefore, while image smoothing is an efficient noise removal method, it also degenerates the profile information. The degeneration of the edge itself is not very serious, since the noise itself is also reduced, and the two effects cancel each other out. Image smoothing also makes the interference between two proximate edges more severe.

5.2 Interpolation

When profiles are sampled from images with oriented points with arbitrary positions and orientations, grid points are no longer guaranteed. To get pixel values with non-integer coordinates, interpolation is a common solution. In our experiment, three different approaches are used:

- No interpolation is used. Pixel values are taken from the nearest grid points.
- The linear interpolation is applied.
- The cubic B-spline interpolation is applied.

The properties of different interpolation approaches have already been well studied. We will only show the results of these different interpolation methods applied to different type of images. The images used in this experiment are the butterfly

(a) An image of a butterfly.

(b) Edges detected from Figure 5.10(a) without interpolation.

(c) Edges detected from Figure 5.10(a) with linear interpolation.

(d) Edges detected from Figure 5.10(a) with cubic interpolation.

Figure 5.10: A real image and the edges detected using different interpolation methods.

(a) An image of Micky Mouse.

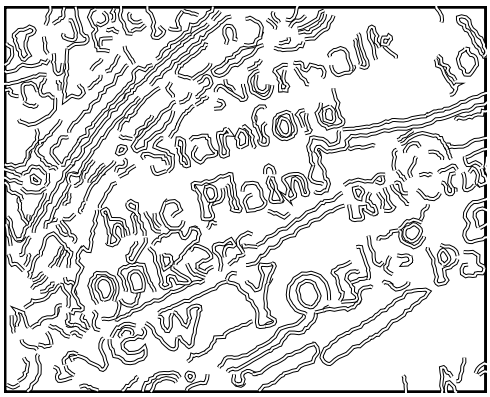
(b) Edges detected from Figure 5.11(a) without interpolation.

(c) Edges detected from Figure 5.11(a) with linear interpolation.

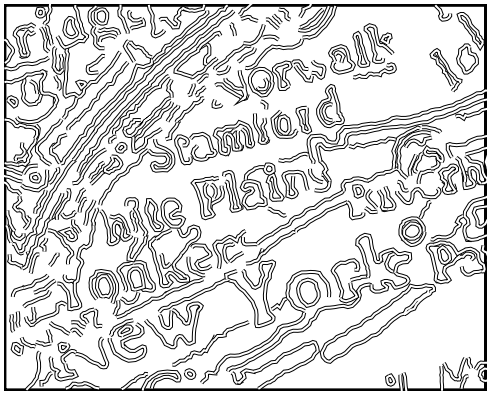
(d) Edges detected from Figure 5.11(a) with cubic interpolation.

Figure 5.11: A drawn image and the edges detected using different interpolation methods.

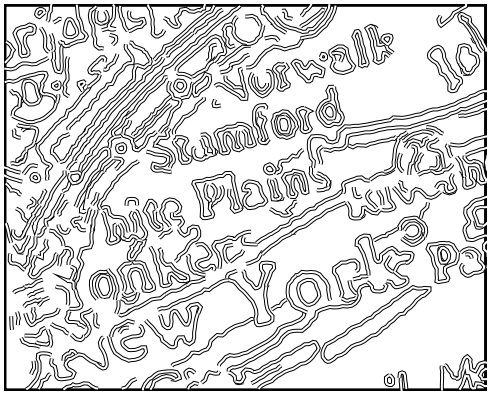
(a) A map image.



(b) Edges detected from Figure 5.12(a) without interpolation.



(c) Edges detected from Figure 5.12(a) with linear interpolation.



(d) Edges detected from Figure 5.12(a) with cubic interpolation.

Figure 5.12: A map image and the edges detected using different interpolation methods.

image, the Micky Mouse image, and a scanned map image. The first image represents a "real" image that is captured by optical device. The second one represents an artificial image, with no image noise, but with some thin black outlines. The third one represents another artificial image, with moderate amount of texture. In order to make the comparison clearer, the curvature smoothing of edge refinement in Section 4.4.3 is not performed. The results are shown in Figures 5.10–5.12. All of the edges are super-imposed on the images except the ones in Figure 5.12 because the context in the map image might make the edges hard to observe.

From these results, it seems that in general the methods of interpolation do not affect the accuracy of the edge detection. Even if no interpolation is used, the results are still quite accurate. Yet, interpolation does provide moderately improved results, which amounts to better connected and smoother edges. The cubic interpolation produces the best results of all three approaches in most cases. However, the speed of the detection is also an important factor to be considered. Linear interpolation takes more than twice the time required to perform the detection without interpolation. Cubic interpolation suffers more than 200% performance degradation when compared with linear interpolation. As mentioned in the previous section, a noisy image also takes additional time. Therefore, even though cubic interpolation usually yields the best results, interpolation is not used in most of the edge detection experiments in this thesis. Moreover, the edge smoothing process would significantly reduce the advantage of the interpolation.

Chapter 6

Conclusion and Future Work

The major purpose of this dissertation research is to propose a new approach to the edge detecting and edge linking process. We advocate associating profile information with an edge to make the edge detection more useful not only for higher level image processing, but for the accuracy as well.

We have demonstrated a framework for edge detection based on profiles. A strength of our framework is its flexibility. Various evaluation functions can be attached to the framework, as demonstrated in Chapter 3. Furthermore, this framework utilizes the measurement of local image noise as the parameter for thresholding in both edge labeling and edge refinement.

While our edge joining process is still far from perfect, it nevertheless demonstrates the advantage of integrating profiles with edges. Though closeness and compatible edge direction may suggest the possibility of joining of two edges, it would be difficult to verify such possibility without comparing their profiles. Moreover, the judgment on similarity between two profiles requires some threshold value. Without a robust threshold selection, it is difficult to determine objectively whether

two edges have similar profiles or not.

There are problems in our model. First of all, the measurement of the noise is crucial in our approach. The noise is used as a parameter in edge localization, as well as most of the edge refinement process. And the measurement of the noise is derived from the stable regions of profiles. Any outside turbulence in the stable region would cause problem for our noise measurement. The outside turbulence can come from the interference from proximate edges, or texture. While texture may be treated as part of image noise, their tendency to be directional makes the measurement anisotropic. Since our noise-induced threshold values are used for both edge normal direction and edge direction, the anisotropic noise measurement does not work well for our model. Furthermore, it might be impossible to find a stable region when texture is involved. This problem is beyond our scope since Gaussian noise is assumed in our model. While only statistic information such as averages and standard deviations of the regions with a profile are calculated, and no properties of Gaussian distribution is utilized. However, a implicit "double of the standard deviation" rule is used as a threshold value in more than one places, and it is based on the **Empirical Rule** of Gaussian distribution that 95% of the values fall within 2 standard deviations of the mean. While this detector may still work with the presence of noise with other distributions, unfortunately, the accuracy will decrease. As for problem about texture, it is a segmentation issue rather than an edge detection issue. For the treatment for textures, other measurement than noise magnitude can be used when constructing evaluation functions. Examples of such measurements are entropy, moments, and homogeneity. Efficiency of these measures highly depends on the types of texture.

Another potential issue is the **phantom edges** described in Section 4.5. So far, we cannot completely remove either kind of phantom edges without affecting the accuracy of the edge detection. Normally, each phantom edge is usually paired with two other edges that are parallel to it. This makes it a good candidate for phantom edge removal. However, by no means imply that every edge bounded by a pair of edges parallel to it is by default a phantom edge. The judgment of whether an edge is a phantom edge is very subjective. For example, in the image of Figure 4.6(a), we labeled the edge at the center as a phantom edge because it does not look like a real edge. Especially when compared with other edges in the same image. In fact, this perspective is true only if we consider the black color as the background. The edge would be a valid line edge if the black color is considered as the foreground.

Last but not least: the speed of the edge detection. For an image with size about 200×200, it takes a few second for normal edge point detectors. Edge linking takes even longer. With Canny's detector and the hysteresis threshold, it takes about 20 seconds to detect edges in the image in Figure 5.10(a), whose size is 230×173. For our detector, it takes 112 seconds to detect all the edges. The situation worsens if more noise is added or time-consuming interpolation method is used. In the worst case, it would take more than 15 minutes to complete the detection. Since our detector does produce additional information, 2+ minutes might be acceptable, but 10+ minutes for a single image might be considered as unbearable. The **edge point prediction** in Section 4.7 is one of the attempts to reduce the time needed. The performance improvement is substantial. Using same image in Figure 5.10(a), if no prediction is performed, the time required for the edge detection of the whole image would take 137 seconds. In other words, it

is about 20% improvement. The performance gain is not always that significant, good. On the average, it can achieve 10% performance enhancement. Though this prediction process is the best we can find to speed up the edge detection. While it is still possible to optimize the testing program for speed, the main issue lies in the framework. In order to localize a valid edge point, many profiles around that points have to be evaluated. This is the main factor that slows down our edge detectors. To be more specific, a lot of calculations are required in finding the correct edge normal direction. Since our profile is one-dimensional, a lot of profiles with different orientations have to be first sampled from the same point and then evaluated. A two-dimensional model of profiles may be more efficient. However, as mentioned in Chapter 1, it is non-trivial to describe an edge which is not straight in the two-dimensional model. For a curve edge, the set of parameters that minimize the fitting error can still be identified. The minimal fitting error, equivalent to our noise magnitude definition, becomes less reliable when a non-0straight edge is fit into a straight edge model. If this issue can be resolved, and parameters can be fit from the image pixel values, the performance can be greatly improved.

There is another issue. Since our detector produces edges with subpixel-level positions, the results are not very stable. With two initial points with slight different positions, their results may and may not look different. It is because some of local noise may affect one of the edge but not the other. In other words, some initial oriented point cannot produce the optimal result, and the edge growing process would stop before the edge really ends. Yet, since there are almost infinite possible initial edge points on an edge, it is impossible to use exhaustive search just for a single edge. In fact, one of the reason we construct the edge joining process is try

Figure 6.1: The detected edges of Figure 4.10(a) with edge magnitude greater than 16.

to minimize such effect. An edge might break due to noise, but we still can try to grow another edge on the other size, then join these two edges together.

6.1 Future Work

There are some possible extensions of this work. First, the edges can be sorted by their significance. The simplest measurement of the strength of an edge is generally judged by its edge magnitude or its length. However, other measurements such as noise magnitude could also be involved. A threshold value can be used to filter the edges out with strength less than the threshold value. While we consider thresholding on the output of edge detector a poor practice, there is merit showing only the edges with specific strength, especially we threshold the strength of edges instead of individual edge points. Moreover, no additional detection has to be performed if another threshold value is selected. For example, for the edges in Figure 4.10(b), if a threshold value 16 is applied to the edge magnitude, the result is shown in Figure 6.1. This result is more acceptable in certain aspect than the

result before thresholding. However, in this example, we do not take the edge length into consideration. In order to efficiently perform threshold with multiple control values, the interaction among these control values will have to be studied.

Another extension is to categorize edges by the colors of their stable regions. The colors on two sides of an edge are compared using different color space models other than RGB, such as YIQ, CMY and HSV, and the channels with major contribution to the edge may be identified. While this step may not always yield meaningful results, there are some special cases that can be identified. For example, the edges caused by shadow tend to differ in the illumination channel, but very little in the hue channel.

Because edges are associated with their profiles, it is possible to reconstruct an image from the edges. While this reconstructed image will be abstract, and may lose some detail, it might be easier to process this abstract image than the original one. Image segmentation process can perform the same task. However, our approach offers better control. For example, by utilizing the thresholding of the sorted edges, an arbitrary threshold can be set to remove weak edges. With our edge detecting approach, only one pass is required to produce different scales of abstract images.

A possible extension is to apply multiple edge detectors together. While this approach is already possible in our framework by utilizing different evaluation functions, it also creates some arbitrary situations that make the combination of the results of different detectors difficult. For example, edges from different detectors may all be valid, but only under different situations. Different sets of edges may conflict to each other if shown at the same time. An example is the phantom edge

Figure 6.2: A mistaken joined edge.

in Figure 4.6(a). While the phantom edge seems to be wrong, it does make sense when the resolution of the image decreases, and in that case, the "correct" edges might become insignificant.

Joining two edges broken by occlusion is another possible task. With our edge joining process, and the **edge point prediction** process described in Section 4.7, we should already be able to perform it. However, we still cannot produce a reliable verification mechanism yet. Even in current state of our edge joining process would make mistakes such as in Figure 6.2 where the gap is only one pixel, a larger gap would increase the possibility of such mistakes, unless a more reliable confirmation method can be performed.

Bibliography

[1] Blake A. and A. Yuille. *Active Contour.* MIT Press, Cambridge, Massachusettts, 1992.

[2] L. Ambrosio and V.M. Tororelli. Approximation of functionals depending on jumps by elliptic functional via γ-convergence. *Comm. Pure and Appl. Math*, 43(8), December 1990.

[3] A.A. Amini, T.E. Weymouth, and R.C. Jain. Using dynamic programming for solving variational problems in vision. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(9), Sept. 1990.

[4] S. Baker. *Design and Evaluation of Feature Detectors.* PhD thesis, Columbia University, 1998.

[5] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. *Pattern Recognition*, 13(2):111–122, 1981.

[6] F. Bergholm. Edge focusing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6), November 1987.

[7] John Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6), November 1986.

[8] G.-H. Cottet and L. Germain. Image processing through reaction combined with nonlinear diffusion. *Mathematics of Computation*, 61, 1993.

[9] R. Courant and D. Hilbert. *Methods of Mathematical Physics*. Wiley-Interscience, New York, 1953.

[10] Luiz A. Costa Davi Geiger, Alok Gupta and John Vlontzos. Dynamic programming for detecting, tracking and matching deformable contours. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(3), 1995.

[11] Larry S. Davis. A survey of edge detection techniques. *Computer Graphics and Image Processing*, 4, 1975.

[12] S. Dellepiane, D.D. Guisto, and G. Vernazza. Automatic parameter computation for edge detection by the zero-crossing method. In *12th Conference on Signal Processing and Images*, pages 617–620, 1989.

[13] Rachid Deriche. Using canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1987.

[14] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1, 1959.

[15] W.A. Barrett E.N. Mortensen, B.S. Morse and J.K. Udupa. Adaptive boundary detection using 'live-wire' two-dimensional dynamic programming. In *IEEE Proceedings of Computers in Cardiology*, Oct. 1992.

[16] R. C. Gonzales and R. E. Wood. *Digital Image Processing.* Addison-Wesley Pub. Co, Reading, Massachusettts, 1992.

[17] R. Hartley. A gaussian-weighted multiresolution edge detector. *Computer Vision, Graphics, and Image Processing*, 30(1), 1985.

[18] E. C. Hildreth. The detection of intensity changes by computer and biological vision system. *Computer Vision, Graphics, and Image Processing*, 22, 1983.

[19] Paul V. C. Hough. Methods and means for recognising complex patterns. United States Patent 3,069,654, Dec 1962.

[20] M. Hueckel. An operator which locates edges in digitized pictures. *Journal of the ACM*, 18:113–125, January 1971.

[21] M. Hueckel. A local visual operator which recognizes edges and lines. *Journal of the ACM*, 20:634–647, October 1973.

[22] Manfred H. Hueckel. An operator which locates edges in digitied pictures. Technical Report Memo AIM-105, Artif. Intell. Proj., Stanford University, October 1969.

[23] A. Hummel. Representations based on zero-crossings in scale-space. In *Proc. IEEE Computer Vision and Pattern Recognition Conf.*, pages 204–209, 1987.

[24] R.A. Hummel. Feature detection using basis functions. *Computer Graphics and Image Processing*, 9, 1979.

[25] P.D. Hyde and L.S. Davis. Subpixel edge estimation. *Pattern Recognition*, 16(4):413–420, 1983.

[26] J. Illingworth and J. Kittler. A survey of the hough transform. *Computer Vision, Graphics, and Image Processing*, 44, 1988.

[27] Michael Kass, Andrew Witkin, and Demetri Terzopoulos. Snakes, active contour models. *International Journal of Computer Vision*, 1988.

[28] M. Kisworo, S. Venkatesh, and G.A.W. West. Modeling edges at subpixel accuracy using the local energy approach. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(4):405–410, April 1994.

[29] J. Koenderink. The structure of images. *Biological Cybernetics*, 50, 1984.

[30] Andreas Koschan. A comparative study on color edge detection. In *Proc. 2nd Asian Conf. on Computer Vision ACCV'95*, volume III, pages 574–578, 1995.

[31] R. Hummel L. Parida, D. Geiger. Junctions:detection, classification, and reconstruction. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(7), July 1998.

[32] E.P. Lyvers, O.R. Michell, M.L. Akey, and A.P. Reeves. Subpixel measurement using a moment-based edge operator. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(12):1293–1309, December 1989.

[33] P.J. MacVicar-Whelan and T.O. Binford. Line finding with subpixel precision. In *SPIE*, volume 281, 1981.

[34] D. Marr and E. C. Hildreth. Theory of edge detection. In *Proceedings of the Royal Society of London*, volume B207, pages 187–217, 1980.

[35] E.N. Mortensen and W.A. Barrett. Intelligent scissors for image composition. In *ACM SIGGRAPH '95. International Conference on Compter Graphics and Interactive Techniques.*, Augest 1995.

[36] E.N. Mortensen and W.A. Barrett. Interactive segmentation with intelligent scissors. *Graphical Models and Image Processing*, 60(5), September 1998.

[37] D. Mumford and J. Shah. Boundary detection by minimizing functionals. In *IEEE Conf. Computer Vision and Pattern Recognition*, 1985.

[38] V.S. Nalwa and T.O. Binford. On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):699–714, November 1986.

[39] M. Nitzberg and T. Shiota. Nonlinear image filtering with edge and corner enhancement. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14, 1992.

[40] O'Gorman. Edge detection using walsh functions. *Artifical Intelligence*, 10:215–233, 1978.

[41] J. R. Parker. *Algorithms for Image Processing and Computer Vision.* John Wiley & Sons, Inc., New York, 1996.

[42] A.P. Pentland and B. Horowitz. Recovery of nonrigid motion and structure. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(7), July 1991.

[43] A.P. Pentland and J.R. Williams. Good vibrations: Modal dynamics for graphics and animation. In *Proc. ACM SIGGRAPH Conf.*, 1989.

[44] Pietro Perona and Jitendra Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7), July 1990.

[45] M. Petrou and J. Kittler. Optimal edge detector for ramp edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5), May 1991.

[46] T. Poggie, H. Voorhees, and A. Yuille. A regularized solution to edge detection. Technical Report AI Memo 833, Massachusette Institute of Technology, Artifical Intelligence Laboratory, May 1985.

[47] O.S. Rice. Mathematical analysis of random noise. *Bell System Technical Journal*, 23, 1944.

[48] Gerhard X. Ritter and Joseph N. Wilson. *Handbook of Computer Vision Algorithms in Image Algebra*. CRC Press LLC, Boca Raton, Florida, 2000.

[49] K. Rohr. Recognizing corners by fitting parametric models. *International Journal of Computer Vision*, 9(3):213–230, 1992.

[50] Stephen M. Smoth and J. Michael Brady. SUSAN – a new approach to low level image processing. *International Journal of Computer Vision*, 23(1), 1997.

[51] A.J. Tabatabai and O.R. Michell. Edge location to suxpixel values in digital imagery. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(2):188–200, March 1984.

[52] S. Tabbone and D. Ziou. A multi-scale edge detector. *Pattern Recognition*, 26(9), 1993.

[53] A. P. Witkin. Scale-space filtering. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence*, pages 1019–1022, 1983.

[54] D. Ziou and S. Tabbone. Adaptative elimination of false edges for first order detectors. In *Proceedings of the 8th International Conference, Image Analysis and Processing*, volume 974 of *Lecture Notes in Computer Science*, 1995.

[55] D. Ziou and S. Tabbone. Edge detection techniques- an overview. Technical Report AI Memo 833, Massachusette Institute of Technology, Artifical Intelligence Laboratory, 1997.