Design and Evaluation of Feature Detectors

Simon Baker

Submitted in partial fulfillment of the

requirements for the degree

of Doctor of Philosophy

in the Graduate School of Arts and Sciences

Columbia University

1998

Design and Evaluation of Feature Detectors

Simon Baker

Abstract

Many applications in both image processing and computational vision rely upon the robust detection of parametric image features and the accurate estimation of their parameters. In this thesis, I address three fundamental questions related to the design and evaluation of parametric feature detectors.

Most feature detectors have been designed to detect a single type of feature, more often than not, the step edge. A large number of other features are also of interest. Since the task of designing a feature detector is very time consuming, repeating the design effort for each feature is wasteful. To address this deficiency, in the first part of this thesis I develop an algorithm that takes as input a description of a parametric feature and automatically constructs a detector for it.

The development of many feature detectors begins with an ideal model of the feature. Since image data are noisy, feature detectors must actually detect features that are almost, but not quite, ideal. Many existing feature detectors can therefore be regarded as being defined by two components: (1) an ideal feature model, and (2) a function that measures how far the image data may deviate from ideal and still be regarded as the feature. For many detectors, little consideration has been given to the selection of the second of these two components. In the second part of

this thesis, I present a method of selecting the deviation function to maximize the performance of the general purpose feature detector described in the first part.

Essentially only two methods have actually been used to evaluate feature detectors empirically. The first consists of applying the detectors to a small number of real images and getting a human to evaluate the outputs. The second method involves generating a large number of synthetic images and then computing performance metrics from the outputs using knowledge of the way that the synthetic images were generated. Both of these approaches have their flaws. The first method is subjective. The second method does not use real images. In the third and final part of this thesis, I propose a class of evaluation techniques that use a large number of real images, and yet provide non-subjective performance metrics.

Contents

List of Figures

List of Tables

Acknowledgments

I would like to begin by thanking my dissertation committee: Peter Allen, Terrance Boult, John Kender, Shree Nayar, and Visvanathan Ramesh. In particular, I thank Peter and John for their numerous suggestions, questions, and comments at my thesis proposal that greatly improved both this thesis and my job talks.

I would also like to thank Joseph Traub for inviting me to study at Columbia and giving me my first exposure to research. I learnt a great deal from Joe during my first year at the information-based complexity group meetings.

The person I learnt the most from at Columbia was my advisor Shree Nayar. During my time as a Ph.D. student, I have come to believe that the one thing that distinguishes successful students from unsuccessful ones is their advisor. I certainly attribute all of my success, both current and future, to Shree. On my own, I would simply never have fully developed the skills that I now possess.

Most importantly, I would like to thank my close friends in the department: Clifford Beshers, Sabah al-Binali, Tiberiu Chelcea, Shu-Wie Chen, Sushil DaSilva, Samuel Fenster, Tobias Höllerer, Blair MacIntyre, Andrew Miller, Montek Singh, Gong Su, Michael Theobald, and Kazi Zaman. My time at Columbia would not have been anywhere nearly as enjoyable or productive without them.

To Mum, Dad, and Sandra

Chapter 1

Introduction

Feature detection is one of the fundamental tasks in computer vision. It has received widespread coverage in both the research literature and in vision textbooks such as [48], [35], and [79]. Some of the major applications of feature detection include:

Stereo: One of the most popular methods of performing correspondence matching along epipolar lines consists of matching detected edge features.

Object Recognition: Many object recognition algorithms start by detecting edge or corner features. Some algorithms use the detected features directly to perform object recognition, whereas others first aggregate the features into extended contours or groups of features.

Segmentation: Segmentation algorithms frequently begin by detecting edge features because these features are indicative of depth discontinuities or object boundaries in the scene.

Motion: Most structure from motion algorithms rely upon the ability to detect

and track corner features in image sequences.

Hough Transform: The conventional Hough transform takes as input a collection of edge features and aggregates them into extended line segments.

In this thesis, I restrict attention to features that are detected just using a single window in the image. A prototypical window would be square and of size 5×5 pixels. However, the window could be approximately circular, and might contain anywhere from 4 to 100 pixels. In particular, I explicitly rule out all feature aggregation and adaptive thresholding algorithms. The most well known example of such a technique is Canny's hysteresis [20]. Many such techniques dramatically improve the performance of all feature detectors. Here, I am solely interested in how well feature detection can be performed without them.

Once the decision has been made to restrict computation to a fixed size window, the key step in both the design and evaluation of feature detectors is characterizing the intensity distributions in the window that constitute the feature. The development of many, but not all, feature detectors therefore begins with an ideal model of the feature. Major examples include the model matching detectors surveyed in Section 2.1 and the optimal filtering detectors covered in Section 2.3.

Most existing feature detectors have been designed to detect a single type of feature, more often than not, the step edge. A large number of other features are also of interest, including, lines, corners, and junctions. Since the task of designing a feature detector is very time consuming, repeating the design effort for each individual feature is wasteful. To address this deficiency, in the first part of this thesis I propose an algorithm that takes as input a description of an arbitrary parametric feature, and automatically constructs a detector for it. The approach taken is the

model matching approach described in Section 2.1. I include comprehensive experimental results that demonstrate: (1) the generality of the algorithm, and (2) that a step edge detector constructed using my algorithm performs favorably to state of the art detectors designed specifically for the step edge.

Since image data are noisy, feature detectors must be able to cope with this noise and correctly detect features that are almost, but not quite, ideal. Many existing feature detectors can therefore be regarded as actually being defined by two components: (1) an ideal feature model, and (2) a function that measures how far the image data may deviate from ideal and still be regarded as an instance of the feature. This is the case for all of the model matching feature detectors surveyed in Section 2.1. However, for all of these model matching detectors, little or no consideration has been given to the selection of the second component in the definition, namely, the matching function. In the second part of this thesis, I propose a method of selecting the matching function to maximize the performance of the general purpose feature detector described in the first part.

Of the various methods of evaluating a feature detector reviewed in Section 2.5, only two have been used extensively. The first consists of applying the feature detector to a small number of real images and having a human evaluate the outputs. The second method involves generating a large number of synthetic images and computing standard performance metrics from the outputs using knowledge of the way that the synthetic images were generated. Both of these approaches have their flaws. The first method is subjective. The second does not use real images.

It seems exceedingly unlikely that there is a single, simple method of "fairly" evaluating a feature detector. Even in a field as mature as computer architecture,

there is no universally agreed upon way of measuring performance [93]. Instead, the usual approach is to apply a large number of benchmarks, each of which is designed to be typical of a certain range of applications. The overall performance on the benchmarks is then used to compare the different architectures. In the final part of this thesis, I advocate a similar approach for the evaluation of edge detectors. In particular, I propose a set of benchmarks for edge detectors specifically designed for applications requiring precise estimates of the edge orientation and the sub-pixel localization. Good examples of such applications include stereo matching, industrial inspection, and the computation of projective invariants. These benchmarks use a large number of real images, and yet provide non-subjective performance metrics.

The remainder of this thesis is organized as follows. I begin in Chapter 2 with a survey of feature detection. In the following chapter, I describe the general purpose feature detection algorithm. In Chapter 4, I present the results of an extensive study into the performance of the detectors constructed using this algorithm. In Chapter 5, I investigate the selection of the matching function. In particular, I show how to choose the matching function to maximize the performance of the detectors constructed using the general purpose feature detection algorithm. In Chapter 6, I describe my non-subjective benchmarks for edge detector evaluation. Finally, I conclude in Chapter 7 with a summary of my contributions, a brief discussion, and a number of suggestions for future work.

Chapter 2

Related Work

In this chapter, I review the feature detection literature. In particular, I concentrate on work performed since 1975. Much of the earlier work is covered by existing surveys such as those by Davis [26] and Brady [16]. The best sources of information about developments since 1975 are modern texts such as Pratt [100], Nalwa [79], and Faugeras [35]. I classify feature detectors into four major types: (1) model matching detectors, (2) differential invariant detectors, (3) optimal filtering detectors, and (4) statistical detectors. From Section 2.1 to Section 2.4, I cover each of these four types of detector in turn. Afterwards, I discuss the evaluation of feature detectors in Section 2.5 and several other important issues in Section 2.6.

2.1 Model Matching Feature Detectors

Model matching feature detectors, such as [50], [51], [80], and [106], are one of the predominant types of detector, as categorized by Nalwa [79]. The basis of a model matching detector is an ideal parametric model of the feature. For example,

the ideal model of a step edge would normally include parameters such as the orientation of the edge and the intensity levels on the two sides of the edge. Given the feature parameters, the feature model defines the intensity distribution of an ideal feature. For the step edge, the model parameters define which pixels are at the upper intensity level and which are at the lower one.

A feature is detected by a model matching feature detector if there are valid parameter values such that the corresponding ideal feature instance and the image data are sufficiently "similar." Exactly how a feature detector determines whether such parameter values exist is the major decision during the design of a detector. Some detectors use closed form solutions for the best matching parameters whereas others use numerical techniques. To measure how similar the feature model and the image data are, a detector requires a matching function. Sometimes the choice of the matching function is an explicit part of the feature definition, whereas for other detectors the choice is simply implicit in the final design of the detector.

2.1.1 Model Matching Step Edge Detectors

The first model matching detector was Hueckel's step edge detector [50]. The most important issue for Hueckel was to find a closed form solution for the model parameters that give the closest match to the image data. He was able to do this, finding an expression for the best fitting parameters in terms of the projection of the image data onto a low dimensional subspace. This subspace is spanned by a set of low order polynomials, and was chosen to reduce high frequency noise.

A number of similar approaches and enhancements followed [50], including [88], [52], [74], and [46]. O'Gorman's major contribution in [88] was to improve the

efficiency of [50] by using a low dimensional basis of Walsh functions. Hummel [52] applied the Karhunen-Loéve expansion [89] instead of Hueckel's ad-hoc dimension reduction. Further analysis of the application of the Karhunen-Loéve expansion to feature detection was subsequently performed by Lenz [64]. Morgenthaler [74] generalized the approach of Hummel to detect step edges superimposed on a low order polynomial, rather than on a constant function. Finally, Hartley [46] redesigned the Hueckel edge detector to use a Gaussian weighted L^2 norm for the matching function, as was suggested in the appendix of [51].

Probably the most sophisticated model matching edge detector is the Nalwa-Binford detector [80]. Nalwa and Binford used a much more realistic edge model than previous detectors. Their edge model incorporated both sub-pixel localization and image blur, whereas most previous detectors simply assumed a pure discontinuity passing directly through the center of a pixel. Due to the complexity of their edge model, Nalwa and Binford were unable to find a closed form solution for the best fitting parameters. Instead they used a numerical algorithm.

Step edges can occur in 3-D volumetric data as well as in 2-D images. Zucker and Hummel [122] generalized Hummel's 2-D step edge detector [52] to detect step edges in volumetric data. Some of Lenz's results in [64] concerning the application of the Karhunen-Loéve expansion are also applicable to 3-D edge models.

Finally, note that the Nevatia-Babu detector [83] can be regarded as a primitive kind of model matching detector. The Nevatia-Babu detector operates by convolving the image with six edge masks, each oriented in a different direction. If the response to any of the masks is sufficiently strong, an edge is detected and the orientation of the mask that caused the strongest response is used as an estimate

of the orientation of the detected edge. The six masks can be regarded as a very simple edge model, and selecting the mask with the strongest response as finding the best fitting model instance.

2.1.2 Other Model Matching Detectors

Model matching detectors have been proposed for several other types of features. For example, Hueckel [51] generalized his step edge detector of [50] to detect a six parameter line in addition to the original four parameter step edge. Another example is Rohr's corner and Y-junction detector [106]. In Rohr's detector, the model parameters that give the closest match between the feature model and the image data are found in a two stage process. First, initial estimates are made based upon the output of applying an edge detector in the immediate vicinity. These estimates are then used to initialize a numerical, gradient descent algorithm.

2.1.3 Selection of the Matching Function

Although the selection of an appropriate matching function is crucial to the performance of a model matching detector, the issue has never been studied in a systematic manner. Most existing detectors use the Euclidean L^2 norm, often without any discussion of the decision, including [88], [52], [74], [122], [80], and [106].

Non-uniformly weighted L^2 norms have been used in a small number of detectors. The first use dates back to the work of Hueckel [50]. In the continuous domain of [50], Hueckel used the weighting function $w(x, y) = [1 - (x^2 + y^2)]^{1/2}$, where (x, y) are coordinates relative to the center of a circular window with unit radius. The justifications provided for this choice were: (1) the weighting function

should be continuous, including at the periphery of the window, and (2) the value of the weighting function should decrease monotonically with distance from the center of the window. In [51], the weighting function remains the same to allow a closed form solution for the best fitting parameters. In the appendix, however, Hueckel suggests that a Gaussian weighting function may be more appropriate.

Few other detectors actually use non-uniform weighting functions. An exception is Hartley [46], who followed Hueckel's suggestion and used a Gaussian weighting function. Lenz [64] concentrates on the Euclidean L^2 norm, but extends some of his results to the uniformly weighted L^2 norm in polar coordinates. In Cartesian coordinates, this corresponds to using the weighting function $w(x, y) = 1/(x^2+y^2)^{1/2}$. Another example of the use of non-uniformly weighted L^2 norms is [1]. In this paper, Abdou and Pratt mention that weighting the pixels so as to reduce the influence of pixels that are distant from the center pixel improves Pratt's Figure of Merit [100], but few details are given. Finally, Paton [92] proposed a number of non-uniform weighting functions, although he never actually used any of them. These include a sequence of functions similar to Hueckel's weighting function, and an annular stop function that assigns zero weight to the center of the feature window:

$$w(x,y) \;=\; \begin{cases} k & \text{if } 0 < r^2 \leq x^2 + y^2 \leq 1 \\ 0 & \text{otherwise.} \end{cases} \tag{2.1}$$

Here, $r \in (0, 1)$ is the radius of the inner boundary of the annular stop. A related approach is that of Parida *et al.* in [91].

Non-uniformly weighted L^2 norms have also been used in a number of differential invariant feature detectors that use the surface fitting approach described in Section 2.2.1. For instance, in [71] both unweighted and Guassian weighted L^2

norms were used to define the best fitting polynomial surface that is subsequently differentiated to estimate the differential invariants.

2.1.4 Dimension Reduction

Since weighted L^2 norms are derived from underlying Hilbert spaces, it is possible to apply dimension reduction techniques, such as the Karhunen-Loéve (K-L) expansion [89] [38], to improve the efficiency of feature detection. The use of the K-L expansion was first proposed for feature detection by Hummel [52]. Subsequently, the K-L expansion was studied by Lenz [64], and used in a number of other detectors such as [122] and [81]. More ad-hoc dimension reduction was incorporated into earlier detectors, beginning with Hueckel [50], but also including Morgenthaler [74]. The effect on performance of using a subspace with very low dimension was investigated empirically by Nevatia in [82]. Nevatia found that using a subspace with too low a dimension does indeed reduce the performance of a detector.

2.2 Differential Invariant Feature Detectors

The second major class of feature detectors consists of those based upon differential invariants. Well known examples include the Deriche corner detector [30], the Haralick step edge detector [44], and the Marr-Hildreth step edge detector [70]. As indicated by the name, these detectors base their detection decisions upon differential invariants estimated from the image data. For example, the Deriche detector is based upon the Hessian determinant, the Haralick detector upon the second directional derivative, and the Marr-Hildreth detector upon the Laplacian.

Essentially, there are just two ways to estimate the differential invariants required by differential invariant detectors. In the first, a parameterized surface is fit to the image data. The partial derivatives of this surface are then used as estimates of the partial derivatives of the underlying image. I describe this first approach in Section 2.2.1. The second approach uses discrete approximations to differential filters. I describe this second approach in Section 2.2.1. In Sections 2.2.1 and 2.2.1, I only consider step edge detectors. Afterwards, in Section 2.2.3, I discuss differential invariant approaches to corner detection.

Finally, note that a small number of papers have focused on how to estimate the partial derivatives required by differential invariant feature detectors, rather than on feature detection itself. Such papers include [117], [71], and [41].

2.2.1 Surface Fitting Differential Invariant Detectors

One way to describe the intensity values in a feature window "is to fit a surface to the data and use the derivatives of the surface as characteristic descriptors" [17]. The major contribution of [17] was to show that a number of early edge detectors can be regarded as doing exactly this. Examples include the Robert's cross operator [105] for which the detection decision is based upon the gradient of the best fitting plane, and the Prewitt operator [102] for which the decision is based upon the gradient of the best fitting quadratic surface [79]. Later, Prewitt's approach was extended by Morgenthaler and Rosenfeld to detect edges in 3-D volumetric data [75].

Once a polynomial surface has been fit to the image data, almost any differential invariant can be computed, be it the gradient, Laplacian, or higher order

invariant. For example, Haralick proposed an edge detector that detects edges at negatively slopped zero crossings of the second directional derivative, taken in the direction of the gradient [44]. These zero crossings correspond to local maxima of the first order directional derivative taken in the direction of the gradient and are computed from the best fitting cubic surface. In a related paper [43], Haralick used a surface fitting approach to estimate the zero crossings of the first directional derivative in the direction that extremizes the second directional derivative. Haralick argues that these feature points correspond to ridges and valleys.

Surface fitting differential invariant detectors are closely related to the model matching detectors described in Section 2.1. Both fit a surface to the image data. The major difference, however, is that, whereas surface fitting differential invariant detectors base the detection decision upon differential invariants computed from the image, most model matching detectors base the decision entirely upon the degree of fit. One common issue is the selection of the function used to measure the degree of fit between the image data and the surface being fit to it. Just as for model matching detectors, the unweighted least squares fit is the usual choice. There are a few exceptions, however. For example, in [71] both unweighted and Guassian weighted L^2 norms were used to define the best fitting surface.

2.2.2 Filtering Differential Invariant Detectors

The other way of computing the partial derivatives of the image is by filtering. As pointed out by Torre and Poggio [115], the differentiation of a discrete image is an ill posed problem that needs to be regularized. One way to regularize the problem is by filtering, or smoothing, the image before differentiation. This filtering step

can be regarded as implicitly interpolating the image data to create a continuous surface. For this reason, there is a close relationship with surface fitting differential invariant detectors. As pointed out by Faugeras in [35], surface fitting can also be regarded as another form of regularization

To design a filtering differential invariant detector, decisions must be made on two major points: (1) the shape of the filter, and (2) the differential invariant. Perhaps the most popular choices have been the Gaussian filter and the Laplacian. Laplacian of Gaussian detectors date back to the Marr-Hildreth detector [70]. Since then, the Laplacian of Gaussian has been studied in great depth. For example, its efficiency [22] [54], accuracy [11] [54], information content [69] [70], and topological properties [115] have all been investigated.

Naturally, other choices are possible. For example, Modestino and Fries [72] investigated least mean square filters for the Laplacian, and Castan *et al.* [21] used the Symmetric Exponential Filter with both the gradient and the second directional derivative in the direction of the gradient. Hashimoto and Sklansky [41] and Weiss [117] both just considered the problem of computing the partial derivatives, as opposed to edge detection per se. Hashimoto and Sklansky [41] considered the Wiener filter, and Weiss [117] studied filters that preserve the derivatives of polynomials of a certain degree. A final example of a filtering based differential invariant detector is the half edge and vertex detector of [39]. Here, Gennert uses a modified directional derivative of a Gaussian operator to create an edge detector that performs more robustly at vertices and corners. The major disadvantage, however, is that the operator must be applied at a large number of different orientations.

2.2.3 Differential Invariant Corner Detectors

Whereas much of the literature on edge detection has concentrated on how to compute the three major differential invariants (the gradient, the Laplacian, and the second directional derivative in the direction of the gradient), the work on corner detection has largely focused on the differential invariants themselves [30].

Perhaps the first differential invariant corner detector was the Beaudet detector [8]. The differential invariant used as the measure of "cornerness" by Beaudet was the determinant of the Hessian:

$$\text{DET} \;=\; I_{xx}I_{yy} - I_{xy}^2 \tag{2.2}$$

The reason for this choice is that the DET operator can be shown to be equivalent to the product of the Gaussian curvature ($\kappa_{\max}\kappa_{\min}$) and a term that is an increasing function of the magnitude of the gradient:

$$\text{DET} \;=\; (\kappa_{\max}\kappa_{\min}) \cdot (1 + I_x^2 + I_y^2)^2. \tag{2.3}$$

Essentially, corners are detected at points where both the curvature and the contrast, or gradient magnitude, are high. A slightly different choice was made by Kitchen and Rosenfeld [58], that being to use the rate of change of the gradient direction multiplied by the gradient magnitude, a measure which simplifies to:

$$K \;=\; \frac{I_{xx}I_y^2 + I_{yy}I_x^2 - 2I_{xy}I_xI_y}{I_x^2 + I_y^2} \tag{2.4}$$

Torre and Poggio [115] later showed that this expression is also the second directional derivative in the direction orthogonal to the gradient.

Two corner detectors closely related to the Kitchen-Rosenfeld detector are the Dreschler-Nagel detector [33] and the Zuniga-Haralick detector [123]. Nagel

showed that the Dreschler-Nagel detector and the Kitchen-Rosenfeld detector are equivalent if the heuristic of nonmaximum suppression along the gradient is applied to the gradient before multiplying by the gradient magnitude [76]. Shah and Jain showed that the Zuniga-Haralick detector is the same as the Kitchen-Rosenfeld detector divided by the magnitude of the gradient [110]. Finally, another corner detector is the Plessey corner detector [45]. The Plessey detector is based upon first order invariants of a smoothed image. An explanation of the Plessey detector in terms of differential geometry was later provided by Nobel [84].

2.3 Optimal Filtering Feature Detectors

In the previous section, I discussed differential invariant approaches to feature detection. For example, one way to detect step edges is to apply the Laplacian operator and declare edges at its zero crossings [70]. The major goal of the work surveyed in that section was to compute the differential invariants as accurately as possible. The development of optimal filtering detectors can be seen as an attempt to answer the question of whether computing these differential invariants is really the best way to detect features. Although thresholding the magnitude of the gradient undeniably gives a reasonable edge detector, it is unclear whether this is the best that can be done, even if the gradient is computed perfectly.

The second method of computing the differential invariants in the previous section was by filtering. Optimal filtering edge detectors operate by declaring edges at local "extrema in the output of the convolution of the image with a [filter]" of an appropriate shape [35]. Then, the key question for optimal filtering detectors is the shape of filter. Unlike Section 2.2.2, where the goal is to choose the filter to

compute a specific invariant as accurately as possible, the goal for optimal filtering detectors is to choose the shape of the filter to maximize the feature detection performance. Not surprisingly, the optimal filter typically ends up looking similar to a derivative operator, however the exact shape may differ somewhat.

To design an optimal filtering detector, there are two major steps: (1) propose optimality criteria, and (2) optimize these criteria to derive the optimal filter. Various optimality criteria have been proposed in the literature, however the most well known and thoroughly studied are the three criteria proposed by Canny in [20]. The remainder of this section is organized as follows. I begin in Section 2.3.1 by describing Canny's three optimality criteria. In Section 2.3.2, I discuss the various ways that these criteria have been combined and the alternative approaches that have be used to optimize them. Finally, in Section 2.3.3, I discuss some of the other optimality criteria that have been proposed in the literature.

2.3.1 Canny's Optimality Criteria

To make his analysis tractable, Canny initially studied edges in 1-D signals. To extend to 2-D images he rotates the optimal filter so that it is aligned with the gradient. Before optimality criteria can be proposed, an ideal model is needed for the feature. Canny was interested in the step edge. He used a perfect discontinuity, or Heaviside function, as his model of a step edge. In addition to the feature model, a noise model is also required. Canny assumed zero mean white Gaussian noise.

Optimality criteria should be chosen so that they are closely related to performance measures. As discussed in Section 2.5, the three most fundamental elements of feature detection performance are: (1) the rate of occurrence of false positives,

(2) the rate of occurrence of false negatives, and (3) the parameter estimation accuracy. Canny's first two criteria correspond directly to these aspects of performance:

Good Detection: "There should be a low probability of failing to mark real edge points (ie. false negatives) and low probability of falsely marking non edge points (ie. false positives)" [12]. Canny argued that both of these criteria are strongly correlated with the signal to noise ratio (SNR). Therefore, he used the SNR as his first optimality criterion.

Good Localization: "The points marked by the operator should be as close as possible to the center of the true edge" [12]. Canny derived an estimate of the root mean squared (RMS) displacement of an ideal edge perturbed with independently and identically distributed Gaussian noise, and used it as his second optimality criterion.

In [20], Canny initially tried to optimize the product of these first two criteria. The optimal filter is the matched filter, or difference of boxes operator [35]. Unfortunately, this detector is well known to perform quite poorly. In particular, it tends to produce many local maxima in the vicinity of noisy step edges [35]. Hence, Canny introduced a third optimality criteria to address this problem:

Few Multiple Responses: For an ideal detector, there should be "only one response to a single edge" [12]. Canny derived an estimate for the expected distance between adjacent edges and used it as his third optimality criterion.

2.3.2 Deriving the Optimal Filter

There are various different ways of combining Canny's three criteria. Canny himself optimized the product of the first two criteria, while keeping the third one fixed. As discussed by Faugeras in [35], this method naturally corresponds to the use of a filter with finite extent. On the other hand, Deriche considers infinite impulse response (IIR) filters by allowing the width of the Canny filter to tend to infinity [29]. The result is a filter with a better value for the product of Canny's first two optimality criteria [35]. Note that Deriche's IIR filter can be implemented very efficiently using recursive filtering [29]. See Section 2.6.3 for more details.

A number of other authors have studied Canny's three criteria, and variants thereof. Rather than considering the product of the first two criteria while keeping the third one fixed, Spacek chose to optimize the product of all three criteria [113]. Later, Petrou and Kittler used Spacek's approach to derive optimal filters for ramp edges [97]. Another optimal (IIR) filter is the Sarkar-Boyer detector [109]. Rather than using Canny's third criterion, Sarkar and Boyer used a slightly different criterion designed to measure the likelihood of spurious responses to noise. Finally, Demigny and Kamlé [27] derived discrete versions of Canny's three criteria and used them to compare the performance of various step edge detectors.

2.3.3 Other Optimality Criteria

Several other optimality criteria have been proposed. These include the energy in the vicinity of the edge studied by Shanmugam *et al.* [111] and by Lunscher [66], and the Discriminative Signal to Noise Ratio studied by Rao and Ben-Arie [104].

2.4 Statistical Feature Detectors

A small number of feature detectors have been proposed that are based upon statistical techniques. Although the results obtained using these detectors are on the whole quite poor, statistical feature detectors constitute a significant subclass of feature detectors. The general approach taken by statistical feature detectors is as follows. First, the image data is assumed to have been generated by either a feature process or a non-feature process, corrupted by a noise process. The initial goal of the statistical feature detector is to estimate the *a posteriori* probability that the image data was generated by the feature process. A decision rule is then used to decide whether to detect the feature or not.

Probably the first statistical feature detector is that of Griffith [40]. The Griffith detector was designed to detect boundary features, which can be either simple lines or simple edges. The non-features modeled include homogeneous regions, skewed lines, and parts of lines. Later, Nahi and Jahanshahi developed an edge detector using a replacement processes to model image formation as the replacement of a background process with an object process [77]. In [42], Haralick proposed an edge detector using the F-statistic to test the statistical significance of the difference between the parameters of the best fitting sloped surfaces in neighboring pixels. Bovik *et al.* [14] proposed three different statistical tests for edge detection, two based upon linear rank sums and one based upon fitting order statistics. Finally, Huang and Tseng [49] proposed a statistical theory of edge detection based upon the change-point problem.

2.5 Evaluation of Feature Detectors

In the first part of this section, I present the various measures of performance that have been proposed in the literature, without discussing how, or even whether, they can actually be evaluated. In the second part, I describe how the measures can be evaluated. For a slightly different categorization of previous work on the evaluation of feature detectors, the reader is referred to [47].

2.5.1 Performance Measures

Most of the measures of performance can be placed into one of five categories:

Feature Detection Robustness: The first category consists of measures that attempt to characterize how likely the detector is to miss a feature that appears in the image (a false negative), and those that attempt to characterize the likelihood that a detector will produce a spurious response where there is no corresponding feature in the image (a false positive). Naturally, there is an inherent trade off between these two measures, as is seen for example in [6]. These two measures are of fundamental importance in most applications and have been widely studied, including in [36], [1], [80], [103], and [32]

Parameter Estimation Accuracy: Another fundamental class of measures consists of those that assess the accuracy with which the parameters of the feature (e.g. orientation, sub-pixel localization, and step magnitude) are estimated. These measures are important for any application that actually uses the feature parameters. These measures have also been widely studied, including in [31], [1], [11], [94], and [114].

Combinations of Robustness and Parameter Estimation: A third class of measures consists of those that are simple combinations of the above two types. Perhaps the most well known example is Pratt's Figure of Merit [1] [95]. Canny's optimality criterion can be regarded as another example, combining a robustness component and a localization accuracy component, with a third component that penalizes multiple responses to the same edge [20] [27].

Performance of Applications: The fourth class consists of measures that directly assess the performance of applications. Examples include, the computation of projective invariants [24], object recognition [47], structure from motion [112], industrial inspection [114], and boundary extraction [56] [90].

Local Measures of Coherence: The fifth and final class consists of measures that are based upon desirable local properties of the output feature map, for example, continuation and thinness [57] [95] [121]. Such properties are particularly important for applications that first aggregate individual features into groups before performing any subsequent processing.

2.5.2 Evaluation Techniques

If ground truth data existed for a large collection of images, estimating any of the measures would be straightforward. Unfortunately, none of the methods of generating ground truth data are entirely satisfactory. Getting a human to mark ground truth is unsuitable because: (1) it is a tedious task and hence error prone, and (2) humans can bring higher level processing to bear, and so there is no guarantee that what is perceived is actually present in the local image data, as is demonstrated

by phenomena such as subjective contours [55]. In fact, in a recent paper in which this approach is taken, Dougherty and Bowyer allowed the human to mask out any regions for which it was too difficult for the human to say which pixels contain edges [32]. An alternative approach is to apply a number of detectors and use the consensus as the ground truth [19]. However, the assumption that the consensus gives a good estimate of the ground ruth is questionable. There are at least four ways of estimating the measures in the absence of ground truth:

Mathematical Analysis: If the feature detection algorithm is simple enough, it is sometimes possible to derive analytical expressions for some of the performance measures. For example, Abdou and Pratt [1] analyze the robustness of several simple edge detectors, Berzins [11] analyzes the localization estimation of a Laplacian edge detector, and Ramesh and Haralick [103] analyze the robustness and parameter estimation accuracy of two different detectors. There are two major weaknesses of this approach: (1) it cannot be easily applied to any detector, and (2) the analysis requires using simplified models of what is a feature, what is not an feature, and the noise processes.

Statistical Tests: A solution to the first of these two problems is to use numerical techniques instead of analytical ones. Then, the complexity of the detector does not cause a problem. A number of papers have performed statistical tests using synthetically generated data, including, [36], [31], [1], [95], [80], and [6]. However, these approaches still have the limitation that they use ideal models of both the signals and the noise. A partial solution, recently proposed by Cho *et al.* [23], is to use a statistical technique known as bootstrap. Bootstrap, although it can be applied to real image data, still makes several

strong assumptions about the nature of a feature and the noise processes.

Subjective Human Evaluation: Another possibility is to get a human to analyze the outputs when applied to either real or synthetic images. Although nearly all feature detection papers do this, they typically do it in a very informal manner. It is possible to perform such a comparison in a more formal way, as was done in [36] and [47]. Even when done scientifically, such techniques still have the inherent weakness that they rely upon the subjective opinion of humans who can bring higher level processing to bear.

Direct Computation from Real Images: Some performance measures do not rely upon ground truth and can be computed directly from the detector output, for example, the local measures of coherence proposed in [57] and [121]. The major weakness of these approaches is that the output could totally misrepresent the structure of the scene and still be rated highly. So long as the local properties are good, the detector will be highly rated.

2.6 Miscellaneous Issues

2.6.1 Unifying Frameworks

A number of authors have attempted to develop unifying frameworks for edge detection. Some of these studies have attempted to show the similarities of specific detectors. For example, both Rosenfeld [107] and Abramatic [2] studied the Hueckel [50] and Roberts [105] detectors. They showed that if the Hueckel operator is implemented in a 2×2 window, it turns out to be the same as a variant of the Roberts'

cross operator. In [67] and [68], Lunscher and Beddoes presented a unified view of the Marr-Hildreth detector [70] and the detector of Shanmugam *et al.* [111].

Other authors have attempted to unify entire classes of detectors. Both Brooks [17] and Haralick [42] tried to provide a unified view of model matching and surface fitting differential invariant detectors through the surface fitting step inherent in both types of detector. Torre and Poggio presented a unified theory of differential invariant detector through regularization theory [115]. Further, Torre and Poggio also examined the relationship between Laplacian edge detectors such as [70] and second directional derivative operators such as [20] and [42].

2.6.2 Sensor Modeling and Sub-Pixel Localization

Most of the previous work on feature detection has assumed that the artifacts introduced by the imaging system are negligible and can be ignored. One exception is [94]. In this paper, Pedersini *et al.* are interested in maximizing the sub-pixel localization accuracy. They claim that to do so, the details of the imaging system must be taken into consideration. In particular, they model image formation as non-linear radial distortion, followed by low-pass filtering, and then ideal sampling. A simpler approach to sub-pixel localization was proposed by Nalwa [78]. Nalwa simply suggested that the image is sub-sampled. Another example of sensor modeling is [97]. Petrou and Kittler argue that ideal step edges do not occur in real images. Instead, real edges are actually ramp edges. They follow the approach of Canny [20] and derive optimal filters for ramp edges, which they claim yield better performance than filters designed for ideal step edges. Finally, Demigny and Kamlé [27] derived discrete (ie. pixel based) versions of Canny's three optimality criteria.

They used these criteria to compare the performance of various edge detectors.

2.6.3 Efficient Filtering

There are a number of techniques that can be used to implement 2-D filtering operations efficiently. See any good image processing text such as [100] for a survey. Perhaps the most well known and simplest example is separability. If the 2-D filter is separable (ie. it can be written as the product of two functions each just dependent on one of the two coordinate variables, as for example the Gaussian can be) it can be implemented as two 1-D filters in the two coordinate directions. Another example is processing in the Fourier transform domain, where a convolution can be implemented as a multiplication. One example of a paper that uses the fast Fourier transform to implement a filtering operation efficiently is Shanmugam *et al.* [111].

Another technique, that can often be even more efficient than Fourier domain processing, is recursive filtering [100] [35]. Perhaps even more importantly, recursive filtering can be used to implement certain infinite impulse response (IIR) filters, which otherwise could only be truncated and approximated as a finite input response (FIR) filter. Recursive filtering is based upon a recursive relationship between the filtered image and the input image. The z-transform of such a relationship is a rational function of z. If the z-transform of the filter also happens to be of this form, it can be implemented using the recursive relationship. A constant amount of computation is needed per pixel to implement these recursive relationships. For example, the infinite Deriche filter [29] can be implemented with 5 additions and 5 multiplications per pixel [35]. Other examples of recursive filtering include the Modestino and Fries detector [72] and the Sarkar and Boyer detector [109].

Another example of an efficient filtering technique is the approach of Chen *et al.* [22], a technique that was later refined by Sotak and Boyer [54]. Chen *et al.* [22] proposed decomposing the Laplacian of Gaussian operator [70] into the product of a Gaussian with smaller standard deviation, and a Laplacian of Gaussian with standard deviation chosen to make up the difference. Chen *et al.* showed that, because the Laplacian of the Gaussian is a bandpass filter, it is possible to reduce the resolution of the image before it is applied. The final computation therefore consists of four steps: (1) convolve the image with the (separable) Gaussian filter, (2) reduce the resolution by decimating the convolved image, (3) convolve with the smaller 2-D Laplacian of Gaussian filter, and (4) restore the full resolution using a simple bilinear interpolation scheme. It turns out that the complexity of this algorithm actually decreases as the standard deviation of the Gaussian increases.

2.6.4 Scale and Information Content

Scale has become an important issue in edge detection and has received a great deal of attention [79] [65]. The notion of the scale of an edge detector dates back to the Marr-Hildreth detector [70]. In particular, in their Laplacian of Gaussian detector, the standard deviation of the Guassian is used to control the scale by changing the amount of blurring. Later, Canny used the same technique to define the scale of his detector [20]. Moreover, an entire theory of scale space has been developed, beginning with the work of Witken [118] and then of Koenderink [60]. These initial papers established the desirable properties of the Gaussian smoothing for scale space. Koenderink [60] also pointed out that smoothing the image with a Gaussian is equivalent to solving the heat equation, also known as the diffusion equation.

Much later, this fact lead Perona and Malik to propose anisotropic diffusion as a mechanism to detect edges [96]. Diffusion is discouraged in the direction of large image gradients, thereby encouraging the formation of a small number of uniform intensity regions separated by boundaries with large gradient magnitude.

The primary goal of scale space theory, at least as far as feature detection is concerned, is to combine the outputs of operators at multiple scales in a coherent manner [79]. A number of attempts have been made at studying the output of detectors across scales, including, [10], [61], and [30]. Bergholm [10] tracked edges across scales to obtain high localization accuracy and also to restore junctions. Korn [61] studied the selection of the appropriate scale at which to apply detectors. Finally, Deriche [30] studied the affects of scale space smoothing on the localization of corners and junctions. He proceeded to show how the systematic bias of the Laplacian of Gaussian can be corrected. A related study, with somewhat different goals, is that of Berzins [11].

A very important issue related to that of scale is the information content of edges. One of the major reasons for detecting edges is their supposedly high information content [70]. The most important question in this context is whether certain edge maps provide a complete representation of the image, in the sense that the image could be uniquely recovered from the edge map. As early as the work of Marr and Hildreth [70], it was conjectured that the zero crossings of the Laplacian of the Gaussian do provide a complete representation. A number of positive results have been found in which it has been shown that the zero crossings at multiple scales are complete for large classes of images [53] [120] [115]. However, recently counterexamples have been found to the most general statement of the problem

by Meyer, as described by Mallat in [69]. In spite of this result, a number of algorithms have been proposed to invert the edge detection process that work very well in practice [69] [34].

Chapter 3

Parametric Feature Detection

3.1 Introduction

As can be seen from the literature survey in Chapter 2, the most frequently studied image feature is the step edge. There are several reasons for this: (1) step edges are very abundant in natural scenes, (2) as discussed in Section 2.6.4, they possess high information content, and (3) their simple 1-dimensional structure makes analysis tractable. Nevertheless, the step edge is by no means the only feature of interest. It is closely followed in significance by other ubiquitous features such as lines, corners, junctions, and roof edges. Even if one restricts attention to features that can be defined analytically, this list is still not comprehensive. Moreover, in any given application, the term "feature" may well take on a meaning that is specific to that application. For instance, during the inspection or recognition of a manufactured part, a subpart such as a nut may be the feature of interest. The appearance of such features will depend upon a number of parameters, for example, orientation, sub-pixel localization, scale, and degree of blurring. In short, parametric features

are too numerous to justify the process of manually deriving a detector for each one. The major contribution of this chapter is to develop an algorithm that automatically constructs a feature detector for an arbitrary parametric feature.

To obtain high performance, it is essential to accurately model the features as they appear in the scene. Therefore, I deliberately choose not to make any simplifications for analytic or efficiency reasons. Instead I use realistic, multi-parameter feature models. Whereas many step edge models assume that the edge passes directly through the center of a pixel and is a perfect step discontinuity, I include a sub-pixel localization parameter and a blurring parameter. These parameters enhance the robustness of detection, while at the same time being useful parameters to recover in their own right. My step edge model has five parameters, namely, the lower brightness level, the brightness difference across the step, the orientation of the edge, the sub-pixel localization, and the blurring parameter. The arguments in favor of highly descriptive feature models apply to other features as well. I use a five parameter roof edge model, a six parameter line model, a five parameter corner model, and a six parameter circular disc model.

In most previous work, feature detectors have been designed in the continuous domain based upon continuous feature models. The detectors developed are only sampled as a final step before their application to discrete images. I argue that to optimize the performance of a detector fully, careful consideration must be given to how the sensor converts the continuous radiance function of a feature into its discrete image. For instance, the aspect ratio of an sensor may significantly affect the appearance of a feature. Perhaps less obvious is the effect that the shape and size of the photosensitive elements within a CCD image sensor have on the appear-

ance of a feature. My notion of a parametric feature model is a continuous one, but during detector construction I explicitly model the discretization of the sensor. The model I use is that of a standard CCD imaging device that integrates the radiance function over a sub-rectangle of each pixel. The sub-rectangle corresponds to the pixel photosensitive area, which, in general, is not the entire pixel. In addition to the sensing discretization, I also model the blurring caused by the optical transfer function of the lens and aperture. For a discussion of other approaches to sensor modeling in feature detection, the reader is referred Section 2.6.2.

When combined, a parametric feature model and a sensor model allow one to predict the pixel intensity values in a window around an ideal feature. All that is required are the parameters of the feature and the details of the imaging system. If the pixel intensity values are treated as real numbers, each feature instance can be regarded as a point in $\mathbf{R}^N$, where N is the number of pixels in a window surrounding the feature. As the feature parameters vary, the point in $\mathbf{R}^N$ corresponding to the feature traces out a k-dimensional manifold, where k is the number of feature parameters. In this setting, feature detection can be posed as finding the closest point on the feature manifold to the point in $\mathbf{R}^N$ corresponding to the pixel intensity values in a novel image window. If the closest manifold point is near enough, the feature is detected and the exact location of the closest point on the manifold reveals the parameters of the feature just detected. On the other hand, if the nearest manifold point is too far away, no feature is detected. This statement of the feature detection problem was first proposed by Hueckel [50], and was subsequently used by O'Gorman [88], Hummel [52], Hartley [46], and Nalwa and Binford [80] for the detection of step edges. Hueckel [51] applied the same formulation to line detection

and Rohr [106] used it to detect corners. The same approach generalizes to 3-dimensional data, as was used by Zucker and Hummel [122] and by Lenz [64]. See Section 2.1 for more discussion of these so called model-matching feature detectors.

Hueckel [50] and Hummel [52] both argued that to achieve the required efficiency, a closed form solution must be found for the parameters of the closest point on the manifold. To make their derivations possible, they used simplified feature models and completely neglected sensing effects. My view of feature detection is radically different. I believe that most features of interest are inherently complex visual entities. Hence, I willingly forgo any hope of finding closed-form solutions for the best-fit parameters. Instead, I discretize the search problem by densely sampling the feature manifold. The closest point on the manifold is then approximated by its nearest neighbor among the sample points. Typically the sampling will result in the order of 10^5 points, which lie in a space of dimension $N = 25$–100. Further, the search for the closest manifold point must be repeated for each each pixel in the image. Nalwa and Binford [80] and Rohr [106] also used more realistic feature models than Hueckel and Hummel. Likewise, they used numerical algorithms to find the parameters of their models that best fit the image data.

At first glance, applying a high dimensional search for every pixel in an image seems inefficient to the point of impracticality. However, I will show that this approach is indeed practical. To obtain the required efficiency, I use a number of techniques. First, I introduce a simple normalizations that eliminates two of the parameters and so reduces the dimensionality of the manifold to three or four (for the five features that I experimented with). This normalization causes no significant loss of information or reduction in the signal to noise ratio. Next, I apply the

Karhunen-Loéve expansion [89] as a dimension reduction technique, which enables me to improve the efficiency by projecting the feature manifold into a low dimensional subspace. Dramatic dimension reduction is possible because most features have significant structure and inherent symmetries. In practice, the dimension of the subspace required turns out to be in the range 5–15. Dimension reduction was first used in feature detection by Hueckel [50]. See Section 2.1.4 for a review of the use of dimension reduction in feature detection.

To perform the search for the closest point on the manifold, I use a coarse-to-fine algorithm that exploits the local smoothness of the feature manifolds to find the closest sample point very quickly. Further, the search does not need to be performed at every pixel. Amongst other techniques, I use a pattern rejection algorithm [4] [5] to eliminate a vast majority of pixels without even needing to project fully into the low dimensional subspace. Such a rejection scheme is effective since most pixels in an image will typically not exhibit the feature of interest. Using the above efficiency enhancements, the feature detectors take only a few seconds on a standard single processor workstation when applied to a 512×480 image.

The remainder of this chapter is organized as follows. In the next section, I introduce the notion of a parametric feature and discuss the sensor model. I show how features may be represented as parametric manifolds, and then describe the efficiency enhancements achievable through parameter normalization and dimension reduction. In Section 3.3, I introduce the five example features that I considered, namely, the step edge, the symmetric line, the corner, the roof edge, and the circular disc. For each feature, I present the feature model, the result of dimension reduction, and the feature manifold. In Section 3.4, I describe the detection al-

gorithm in detail. In particular, I describe manifold sampling, the coarse-to-fine search, and the use of rejection techniques. In the following chapter, I present experimental results obtained for the five example features, including comparisons with a Canny-like operator [20] and the Nalwa-Binford [80] detector.

3.2 Parametric Feature Representation

I begin this section by first introducing the notion of a parametric scene feature. I then describe my model of the imaging system, and how this model leads to imaged features being represented as parametric manifolds. Finally, I describe the parameter normalization and dimension reduction techniques that I used to make the manifold representation more compact and to enhance efficiency.

3.2.1 Parametric Scene Features

By a scene feature, I mean a geometric or photometric phenomenon in the scene that produces spatial radiance variations which can aid in visual perception. It is known that irradiance on the image plane is proportional to scene radiance [48]. I assume that the continuous scene feature can be parameterized and can therefore be written as the continuous irradiance function $F^c(x, y; \mathbf{q}^c)$ where $(x, y) \in S^c$ are points within a bounded feature window S^c on the image plane, and $\mathbf{q}^c$ are the parameters of the scene feature. In the case of a step edge, $\mathbf{q}^c$ would include the edge orientation and the brightness values on the two sides of the edge. In the case of a corner, $\mathbf{q}^c$ would include the orientation of the corner, the angle subtended by the corner, and the brightness values inside and outside the corner. To fully specify

a feature, one must provide the feature irradiance function $F^c(x, y; \mathbf{q}^c)$, the feature window S^c, and the ranges of the parameters $\mathbf{q}^c$.

3.2.2 Modeling Image Formation and Sensing

Most previous work on feature detection has implicitly assumed that the artifacts introduced by the imaging system are negligible and can be ignored. See Section 2.6.2 for a discussion of the use of sensor models in feature detection. An additional reason for ignoring sensor artifacts is that they can be quite complex and nonlinear in nature. Hence, modeling such effects can often make the derivation of a detector far more cumbersome. For reasons that will become clear shortly, modeling both linear and non-linear sensor effects does not involve any extra difficulty in my detection algorithm. Hence, I make the feature models as realistic as possible by explicitly modeling image formation and sensing effects.

The first such effect is blurring of the feature irradiance function. If the scene feature lies outside the focused plane of the imaging system, its image will be defocused. Further, the finite size of the lens aperture causes the optical transfer function of the imaging system to be band-limited in its spatial resolution. Finally, the feature itself, even before imaging, may be somewhat blurred. For instance, a real step edge may not be a perfect step, but instead somewhat rounded. The magnitude of these effects varies spatially across the image plane. In particular, the level of defocus will depend upon the depth of the feature in the scene, and the degree of blurring will depend upon the nature of the geometric or photometric phenomenon causing the feature. Hence, I develop an approach that can handle spatially varying blur. The defocus factor can be approximated by a pillbox function

[13], the optical transfer function by the square of the first-order Bessel function of the first kind [13], and the blurring due to imperfections in the feature by a Gaussian [60]. I combine all three effects in a single blurring factor that is assumed for simplicity to be a 2-D Gaussian:

$$g(x, y; \sigma) \;=\; \frac{1}{2\pi\sigma^2} \exp(-\frac{1}{2} \cdot \frac{x^2 + y^2}{\sigma^2}). \tag{3.1}$$

The continuous irradiance function on the sensor plane is converted by a CCD sensor into a discrete image through two processes. First, the light flux falling on each sensor element is averaged, or integrated. If the pixels are rectangular [7] [85], the averaging function is simply the rectangular function [15]:

$$a(x, y) \;=\; \frac{1}{w_x\, w_y}{}^2\Pi(\frac{1}{w_x}x, \frac{1}{w_y}y) \tag{3.2}$$

where w_x and w_y are the x and y dimensions of the pixel. Second, the pixels are sampled. The final discrete image of a feature can therefore be written as:

$$F(n, m; \mathbf{q}) \;=\; F^c(x, y; \mathbf{q}^c) * g(x, y) * a(x, y)|_{x=n, y=m} \tag{3.3}$$

where $*$ is the 2-D convolution operator, $\mathbf{q} = \mathbf{q}^c \cup \{\sigma\}$ is the vector of discrete feature parameters which includes the spatially varying blur parameter σ, and $(n, m) \in S$ are pixel coordinates in a discrete approximation to the feature window S^c.

It is important to note that the blurring, averaging, and sampling functions vary from sensor to sensor. Above, I have assumed the pixels and the sampling are rectangular. In practice, these functions should be selected based on the details of the actual sensor used. In general, a model of the imaging system is a functional that takes the continuous irradiance function $F^c(x, y; \mathbf{q}^c)$ as input and maps it to a discrete function $F(n, m; \mathbf{q})$.

3.2.3 Parametric Feature Manifolds

If N is the total number of pixels in the discrete feature window S, then each feature instance $F(n, m; \mathbf{q})$ can be regarded as a point in $\mathbf{R}^N$. Suppose the feature has k parameters; ie. $\dim(\mathbf{q})=k$. Then, as these parameters vary over their ranges, the corresponding feature instances trace out a k-parameter manifold in $\mathbf{R}^N$. The feature can therefore be represented as a parametric manifold in $\mathbf{R}^N$. Constructing this manifold is straightforward using Equation (3.3). It just requires the feature model $F^c(x, y; \mathbf{q}^c)$, the range of the blur parameter σ, and the details of the sensor.

In this setting, feature detection can be posed as finding the closest point on the feature manifold to the point in $\mathbf{R}^N$ corresponding to a novel candidate window in the image. If the closest manifold point is near enough, the feature is detected and the exact location of the closest point on the manifold reveals the parameters of the feature just detected. On the other hand, if the nearest manifold point is too far away, no feature is detected.

Performing this task directly using the feature manifold is impractical for reasons of efficiency due to the high dimensionality of the manifold k and the space that it lies in N. In the following sections, I present two techniques that dramatically reduce the dimensionality of the manifold and the space that it lies in, thereby making the feature manifold a viable representation for feature detection.

3.2.4 Parameter Reduction by Normalization

The first efficiency enhancement is a normalization. For each feature instance $F(n, m; \mathbf{q})$, compute the coordinate mean $\mu(\mathbf{q})= \frac{1}{N}\sum_{(n,m)\in S} F(n, m; \mathbf{q})$ and the total coordinate variance $\nu^2(\mathbf{q}) = \sum_{(n,m)\in S} [F(n, m; \mathbf{q}) - \mu(\mathbf{q})]^2$. The following

brightness normalization is then applied:

$$\overline{F}(n, m; \mathbf{q}) \;=\; \frac{1}{\nu(\mathbf{q})}\,[\,F(n, m; \mathbf{q}) \;-\; \mu(\mathbf{q})\,]\,. \tag{3.4}$$

This simple normalization proves to be very valuable. For all of the features considered, it reduces the dimensionality of the manifold by two. This occurs because the normalized feature $\overline{F}(n, m; \mathbf{q})$ turns out to be approximately independent of two of the brightness parameters in $\mathbf{q}$. In the case of the step edge, the normalized feature is invariant to the brightness values on either side of the step; just the values of μ and ν change with the brightness parameters.

There are three important points to note about this normalization: (1) it does not alter the signal to noise ratio, (2) the normalization must be applied both to the feature instances during the construction of the feature manifold and also to the image data during feature detection, and (3) once a normalized feature has been detected, the coordinate mean μ and total variance ν can be used to recover the parameters eliminated during normalization. In the next section, I describe how this inversion can be performed.

3.2.5 Recovering the Normalized Parameters

The brightness normalization described in the previous section can be used to eliminate two of the parameters. I now describe how to recover the normalized parameters. The computation requires as input, the coordinate mean μ and total variance ν computed during normalization, and the unnormalized parameters estimated from the parameters of the closest point on the manifold. For the step edge, line, corner, and circular disc, the two normalized parameters are the base

intensity A and the intensity step B. For the roof edge, they are the peak intensity A and the intensity gradient M. See Section 3.3 for a complete description of the five example features. Although I describe the recovery technique in terms of A and B, the same approach works for the roof edge by replacing B with M.

In the continuous domain, and if the parameters $\mathbf{q} - \{A, B\}$ are fixed, it is easy to see that, for all of the features considered in Section 3.3, the coordinate mean μ and total variance ν are linear functions of A and B:

$$\mu = c_{11} \cdot A + c_{12} \cdot B \tag{3.5}$$

$$\nu = c_{21} \cdot A + c_{22} \cdot B \tag{3.6}$$

where $c_{ij} = c_{ij}(\mathbf{q} - \{A, B\})$ are coefficients that just depend upon the unnormalized parameters. After allowing for the discretization performed by the CCD sensor, these relationships will not be exactly linear, however the deviation from linearity can be neglected. If the coefficients c_{ij} are known, these equations can be used to recover the normalized parameters because they can be inverted to give:

$$A = \frac{c_{22}}{\Delta} \cdot \mu - \frac{c_{12}}{\Delta} \cdot \nu \tag{3.7}$$

$$B = -\frac{c_{21}}{\Delta} \cdot \mu + \frac{c_{11}}{\Delta} \cdot \nu \tag{3.8}$$

where $\Delta = c_{11} \cdot c_{22} - c_{12} \cdot c_{21}$ is the determinant of the matrix (c_{ij}).

The coefficients $c_{ij} = c_{ij}(\mathbf{q} - \{A, B\})$ can be easily precomputed during the construction of the manifold. For each vector of unnormalized parameters $\mathbf{q} - \{A, B\}$ used to sample the manifold, the feature is evaluated for $A = 0$, $B = 1$ and then normalized as described in Section 3.2.4 to give mean μ_1 and total variance ν_1. Repeating for $A = 1$, $B = 1$, gives mean μ_2 and total variance ν_2. Finally, the

coefficients can be computed using:

$$c_{11} = \mu_2 - \mu_1 \tag{3.9}$$

$$c_{12} = \mu_1 \tag{3.10}$$

$$c_{21} = \nu_2 - \nu_1 \tag{3.11}$$

$$c_{22} = \nu_1. \tag{3.12}$$

The coefficients c_{ij} are stored in a lookup table indexed by $\mathbf{q} - \{A, B\}$. As soon as the parameters $\mathbf{q} - \{A, B\}$ have been recovered from the closest manifold point, the coefficients c_{ij} can be easily found. Then, A and B can be recovered using Equations (3.7) and (3.8).

I tested the accuracy of normalization inversion for each of the five features considered in Section 3.3. After computing the coefficients c_{ij} for every manifold sample point using the method described above, I randomly generated a sequence of ideal feature instances. The normalized parameters A and B of the feature were generated uniformly at random in the interval $[0, 1]$. To generate the unnormalized parameters, a point on the manifold was chosen uniformly at random and its unnormalized parameters used. Then I generated the ideal feature using the feature and sensor models described above. After normalizing the feature, I used Equations (3.7) and (3.8) to recover A and B. The results showed that the normalization inversion works almost completely without error. The worst performance across all five features, and over every feature instance generated, gave an error of less that 0.02%. The average error was an order of magnitude lower.

3.2.6 Dimension Reduction

For several reasons, such as feature symmetries and high correlation between feature instances with similar parameter values, it is possible to represent the feature manifold in a low dimensional subspace of $\mathbf{R}^N$ without significant loss of information. This idea was first explored by Hummel [52] and later by Lenz [64]. See Section 2.1.4 for a discussion of the use of dimension reduction in feature detection. If correlation between feature instances is the preferred measure of similarity, the Karhunen-Loéve (K-L) expansion [38] [89] yields the optimal subspace.

The covariance matrix $\mathbf{C} = \mathrm{E}_{\mathbf{q}}[(\overline{F} - \mathrm{E}_{\mathbf{q}}[\overline{F}])(\overline{F} - \mathrm{E}_{\mathbf{q}}[\overline{F}])^{\mathrm{T}}]$ represents the correlation between the pixels in the different feature instances. Since the normalized feature instances $\overline{F}$ are N-dimensional vectors, $\mathbf{C}$ is a symmetric $N \times N$ matrix. The optimal subspace is then computed by solving the eigenvalue problem:

$$\mathbf{C}\,\mathbf{e} \;=\; \lambda\,\mathbf{e}. \tag{3.13}$$

The result is a set of eigenvalues $\{\,\lambda_j \mid j = 1, 2, ..., N\,\}$ where $\lambda_1 \geq \lambda_2 \geq ..... \geq \lambda_N \geq 0$, and a corresponding set of orthonormal eigenvectors $\{\,\mathbf{e}_j \mid j = 1, 2, ..., N\,\}$. Due to the inherent structure and symmetries of most parametric features, the first few eigenvalues tend to be significantly larger than the remaining ones. Hence, the feature manifolds can be represented in a low dimensional subspace spanned by the most prominent eigenvectors. If the first d eigenvectors are used, a measure of the information discarded is the K-L residual defined by:

$$R(d) \;=\; \sum_{j=d+1}^{N} \lambda_j. \tag{3.14}$$

To give an idea of the data compression possible, a step edge manifold in a 49-D space can be represented in a 3-D subspace with K-L residual of less than 10%.

Moving to an 8-D subspace reduces the residual to less than 2%.

3.2.7 Computation of the Feature Manifolds

The parametric feature manifold is constructed by projecting all of the normalized feature instances into the K-L subspace. This just requires the dot product of each feature instance with the eigenvectors that serve as a basis for the subspace. Since such a parameterized feature manifold is easy to compute for any feature, it serves as a generic tool for designing feature detectors. Further, the dramatic dimension reduction produced by the K-L expansion, together with the parameter elimination achieved through the brightness normalization described in Section 3.2.4, allows features to be represented compactly and detected efficiently.

3.3 Example Features

I now illustrate the parametric manifold representations of my five example features. For each feature, I provide a definition of the feature irradiance function, list its parameters, and present the results of applying parameter normalization and dimension reduction. Note that the features I have chosen are merely examples that happen to be important in machine vision. The techniques themselves are not restricted to intensity images and can also be applied to detect features found in the output of almost any type of sensor.

3.3.1 The Step Edge

The first example feature is the step edge. Parametric models of step edges date back to the work of Hueckel [50]. Since then, the edge has been studied in more depth than any other visual feature. See Chapter 2 for a comprehensive review of the feature detection literature. Figures 3.1(a) and 3.1(b) show isometric and plan views of the step edge model that I used. This model is a generalization of the models used in [50], [52], and [64]. It is closest to the one used by Nalwa and Binford [80] in terms of the number and type of parameters, but differs slightly in its treatment of smoothing and blurring effects. The basis for my 2-D step edge model is the 1-D unit step function:

$$u(t) \;=\; \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{if } t < 0. \end{cases} \tag{3.15}$$

A 1-D step with lower intensity level A and upper intensity level $A + B$ can be written as $A + B \cdot u(t)$. To extend this model to 2-D, I assume that the step edge is of constant cross-section and step size along its length, is oriented at an angle θ to the y-axis, and lies at a distance ρ from the origin. Then, as is shown in Figure 3.1(b), the signed distance of a point $(x, y) \in S^c$ from the step is given by:

$$d \;=\; y \cdot \cos\theta - x \cdot \sin\theta - \rho. \tag{3.16}$$

So, an ideal 2-D step edge is given by $A + B \cdot u(d)$. After incorporating the sensing and image formation models proposed in Section 3.2.2, the final model is:

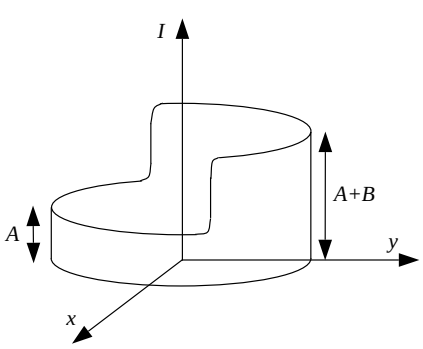
$$F_{SE}(n, m; A, B, \theta, \rho, \sigma) \;=\; [A + B \cdot u(d)] * g(x, y; \sigma) * \; a(x, y)|_{x=n, y=m} \tag{3.17}$$

where d is given by Equation (3.16). As can be seen, the step edge model has five parameters, namely, the orientation angle θ, the sub-pixel localization ρ, the

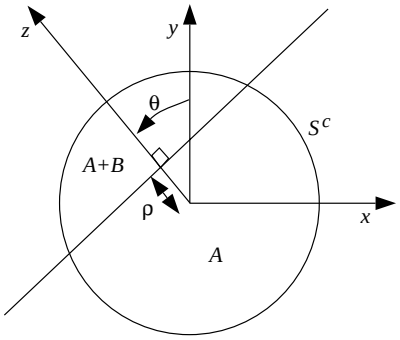
blurring parameter σ, and the brightness values A and B.

To complete the definition, the ranges of the parameters need to be specified. I measure distances in units of the distance between two adjacent pixels and angles in degrees. Then, the orientation parameter θ is drawn from $[0°, 360°]$, the localization parameter ρ is restricted to lie in $[-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$ since any edge must pass closer than a distance $\frac{\sqrt{2}}{2}$ from the center of at least one pixel in the image, and the blurring parameter σ is drawn from $[0.3, 1.5]$. As described in [80], substantially larger values of σ could be used but represent an edge at a much higher scale. Such cases require the use of a larger image window. The intensity parameters A and B are free to take any value. This is because of the parameter normalization described in Section 3.2.4; the structure of a normalized step edge is independent of A and B and is just determined by the other three parameters θ, ρ, and σ. Given a normalized step edge, the values of A and B can be recovered from the coordinate mean μ and total variance ν^2 using the method described in Section 3.2.5.

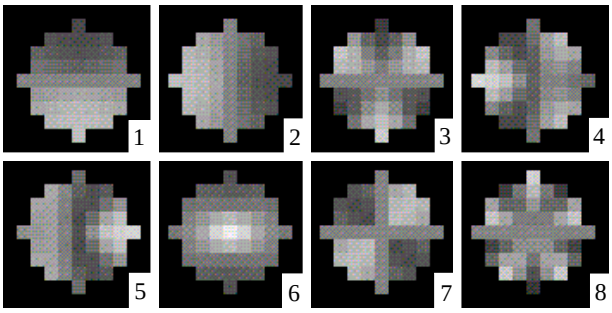
The results of applying the Karhunen-Loéve expansion are displayed in Figures 3.1(c) and 3.1(d). For these figures, I chose to use a feature window S with 49 pixels. In my implementation, any size and shape window can be used. Moreover, to avoid the unnecessary non-linearities induced by a square window, I used a disc shaped one. In Figure 3.1(c), I display the eight most prominent eigenvectors, ordered by their eigenvalues. The similarity between the first four eigenvectors and the ones derived analytically by Hummel in [52] is immediate. Notice, however, that while the eigenvectors of [52] are radially symmetric, the ones in Figure 3.1(c) are not. This is to be expected since the introduction of the parameters ρ and σ breaks the radial symmetry of Hummel's edge model. While the eigenvectors in



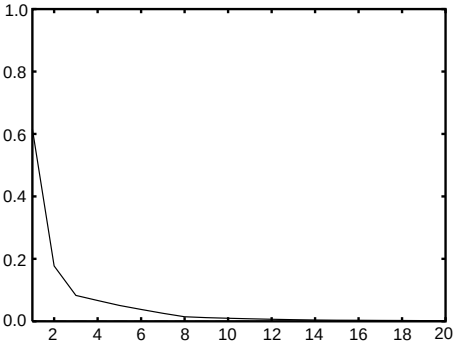
(a) Step edge model



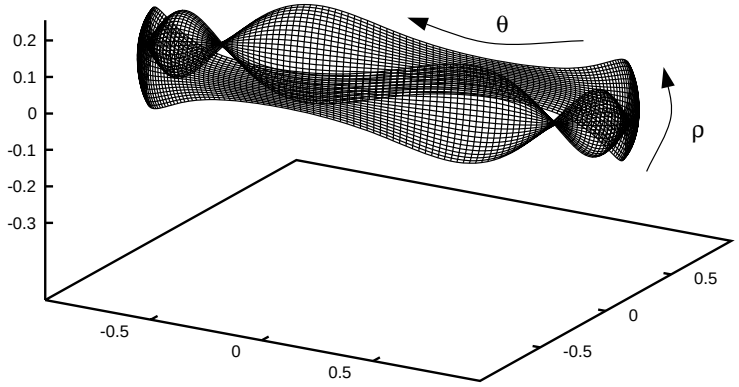
(b) Plan view



(c) First 8 eigenvectors



(d) Decay of the K-L residue



(e) Step edge parametric manifold

Figure 3.1: The step edge model includes two constant intensity regions of brightness A and $A + B$. Its orientation and sub-pixel localization are given by the parameters θ and ρ respectively. The fifth parameter (not shown) is the blur parameter σ. The decay of the K-L residual shows that 90% of the edge image content is preserved by the first three eigenvectors and 98% by the first eight eigenvectors. The step edge manifold is parameterized by orientation and sub-pixel localization for a fixed blurring value and is displayed in a 3-D subspace constructed using the first three K-L eigenvectors.

[52] are optimal for the edge model used there, Figure 3.1(c) shows that they are not optimal for my, more realistic, edge model. Also, note the resemblance of the first two eigenvectors to the first-order spatial derivative operators that constitute the basis of many simple edge detectors, such as the Sobel operator [102].

In Figure 3.1(d), the decay of the Karhunen-Loéve residual is plotted as a function of the number of eigenvectors. As can be seen, the first two eigenvectors capture about 80% of the information. To reduce the residual to 10% three eigenvectors are needed, and to reduce it further to 2% eight eigenvectors must be used. These results represent a compression factor in the range 5-15. As a result, the efficiency of feature detection is greatly enhanced. In [52], Hummel derives the result that the eigenvalues for his continuous step edge model should decay like $1/n^2$. The results in Figure 3.1(d) are consistent with this prediction. By plotting λ_n against n on logarithmic scales and fitting a straight line to the curve, I found that the eigenvalues initially decay like $1/n^2$. However, because I am working in $\mathbf{R}^N$ rather than the infinite dimensional continuous domain considered in [52], the rate of decay increases for larger n.

The step edge manifold is displayed in Figure 3.1(e). Naturally, I only display a projection of it into a 3-D subspace. The subspace chosen is the one spanned by the three most prominent eigenvectors. Also, for clarity I only display a two parameter slice through the manifold, keeping σ constant while varying θ and ρ. As mentioned earlier, the first three eigenvectors capture more than 90% of the information. This is reflected in Figure 3.1(e), where most points on the manifold are seen to lie close to unit distance from the origin. Finally, note that the four apparent singularities of the manifold are simply artifacts of the projection into the

3-D subspace. If it was possible to visualize a higher dimensional projection, these apparent singularities would disappear.

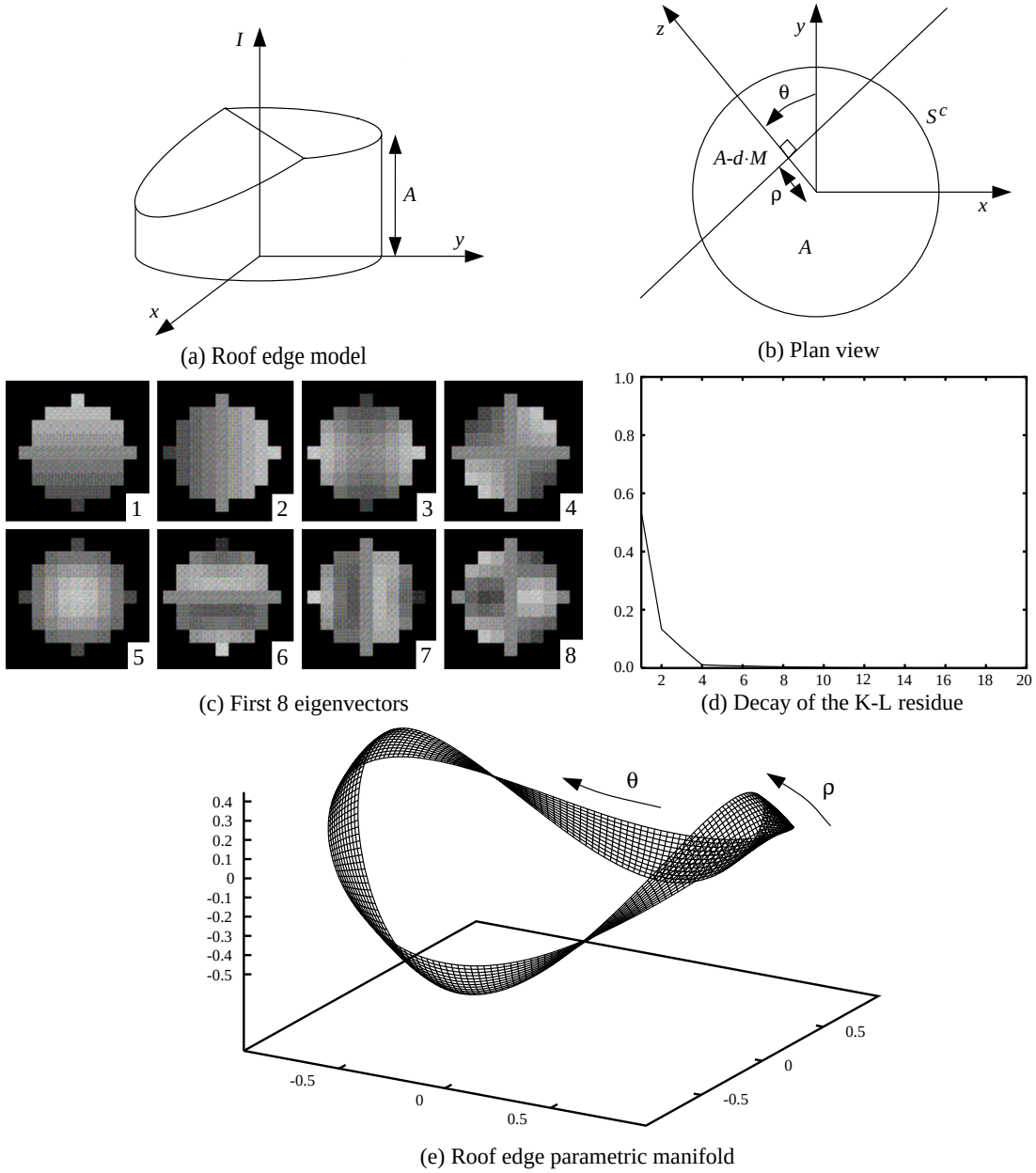
3.3.2 The Roof Edge

Unlike the step edge, the roof edge has not been studied much in the past despite having been acknowledged as a pertinent feature [79]. The only difference between the two edge models is that the step discontinuity in the step edge is replaced by a uniform intensity slope in the roof edge, as is shown in Figure 3.2(a). An algebraic definition is obtained by replacing $A + B \cdot u(d)$ with $A - M \cdot d \cdot u(d)$, where A is the upper intensity level of the roof, and M is the gradient of the slope. The result is a five parameter model:

$$F_{RE}(n, m; A, M, \theta, \rho, \sigma) \;=\; [A - M \cdot d \cdot u(d)] * g(x, y; \sigma) * a(x, y)|_{x=n, y=m} \quad (3.18)$$

where $u(d)$ and d are as defined for the step edge. The parameter ranges that I used for the roof edge are: $\theta \in [0°, 360°]$, $\rho \in [-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$, and $\sigma \in [0.4, 1.0]$. The range of σ is less than that for the step edge since blur has more effect on the roof edge for the same size window. Increasing the size of the window would allow a larger range for σ. The parameters A and M are free to take any value. As for the step edge, the structure of the normalized roof edge is independent of A and M, and their values can be recovered from the normalization coefficients μ and ν. See Section 3.2.5 for the details of how to invert the parameter normalization.

The results of applying the Karhunen-Loéve expansion, presented in Figures 3.2(c) and 3.2(d), are similar to those for the step edge. In fact, the first two eigenvectors are almost exactly the same as those for the step edge, at least up to a



(a) Roof edge model

(b) Plan view

(c) First 8 eigenvectors

(d) Decay of the K-L residue

(e) Roof edge parametric manifold

Figure 3.2: The roof edge has a region of constant intensity A on one side of the edge and a uniform intensity slope of gradient M on the other. Both parameters A and M are removed by the parameter normalization. The orientation parameter θ, the sub-pixel localization parameter ρ, and the blur parameter σ are similar to those used in the step edge. After the first two eigenvectors, the K-L residual decays marginally faster for the roof edge than for the step edge. The displayed slice through the roof edge manifold is parameterized by orientation and intrapixel displacement for a fixed blurring value.

sign change. The K-L residual decays slightly faster for the roof edge, as might be expected since the roof edge more closely resembles a constant intensity region than the step edge. (The residual of a constant intensity region would decay immediately to zero.) For the roof edge, three eigenvectors are also needed to capture 90% of the information, but only five eigenvectors for 98%. The parametric manifold for the roof edge is displayed in Figure 3.2(e). The significant difference in appearance compared to the step edge manifold is due to the difference between the third eigenvectors of the two features. The projection onto the first two eigenvectors is similar; it is approximately a circle in both cases.

3.3.3 The Symmetric Line

A line can be thought of as a pair of parallel step edges separated by a short distance w, the width of the line [51]. The line model which I used is illustrated in Figure 3.3(a). I assume that the step edges are both of the same magnitude and so the line is symmetric. It is possible to generalize this model to lines with different intensities on the two sides of the line by adding one more parameter [51]. The symmetric line model that I used has six parameters and is given by:

$$\begin{aligned} F_L(n,m;A,B,\theta,\rho,w,\sigma) &= [A + B \cdot u(d + w/2) - B \cdot u(d - w/2)] \\ &\quad * g(x,y;\sigma) * a(x,y)|_{x=n,y=m}. \end{aligned} \tag{3.19}$$

The ranges of the parameters ρ and σ are exactly the same as for the roof edge: $\rho \in [-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$ and $\sigma \in [0.4, 1.0]$. Given the symmetry of the line model, the range of the orientation parameter can be halved to $\theta \in [0°, 180°]$. The width of the line is restricted to $w \in [1.0, 3.5]$. The brightness parameters A and B are free to take

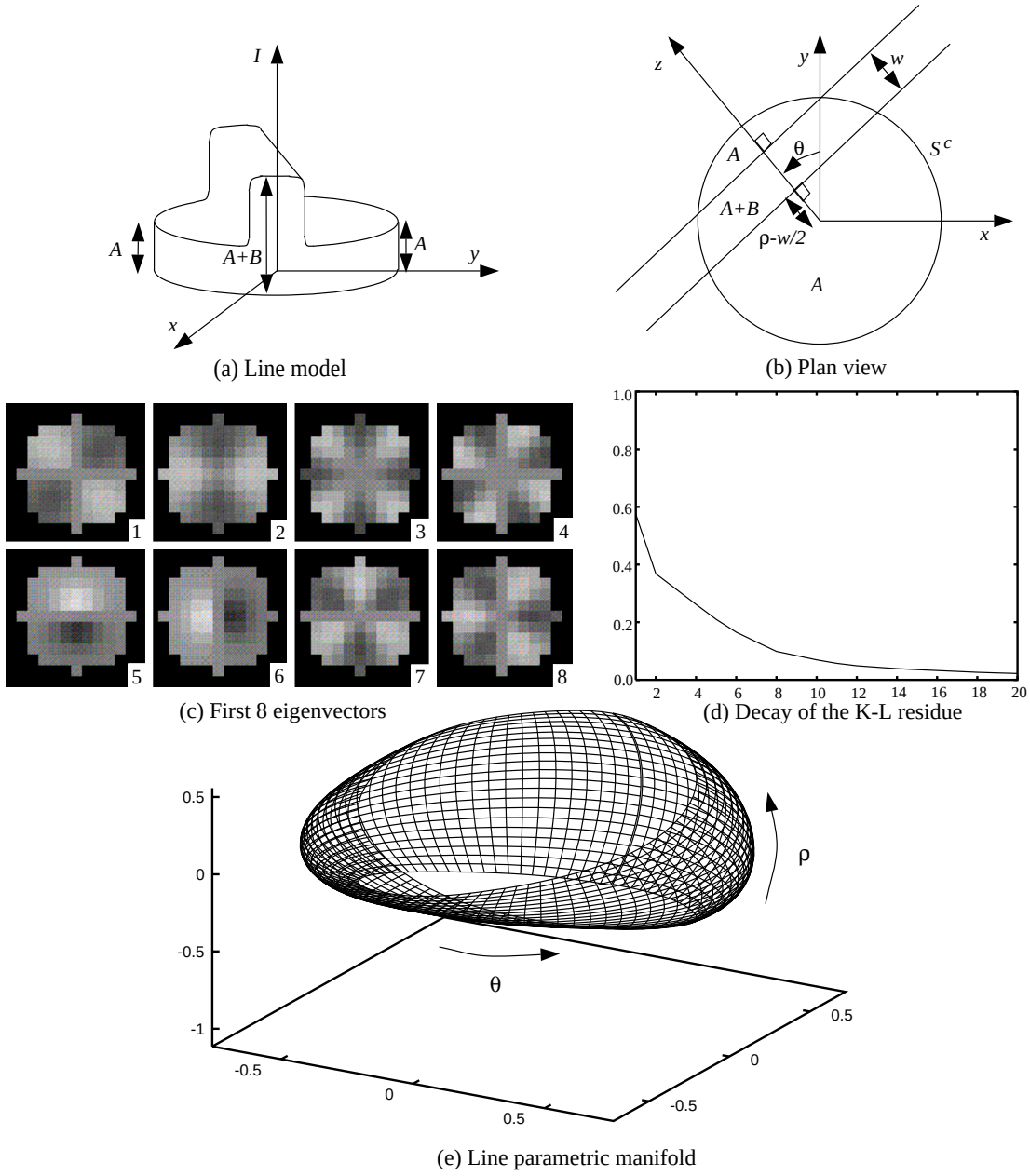


Figure 3.3: The line is of width w, has intensity $A + B$ on the line itself, and has regions of intensity A on either side of the line. In addition, there is an orientation parameter θ, a sub-pixel localization parameter ρ, and a blur parameter σ. Eight eigenvectors are need to capture 90% of the information and twenty-two eigenvectors for 98%. By this measure, the line is a more complex feature than either of the edges. The line manifold is displayed for fixed values of σ and w and has the structure of a Möbius band.

any value, just as for both edge models. They can be eliminated by applying the normalization described in Section 3.2.4, and recovered from μ and ν using exactly the same algorithm as for the step edge.

The result of applying the Karhunen-Loéve expansion is a little different from the results for the previous features. Most significant is the lower rate of decay in the residual, as seen in Figure 3.3(d). To reduce the residual to 10% eight eigenvectors are required, and to reduce it to 2% twenty-two must be used. By this measure, the line is a considerably more complex feature than either of the edges. However, the data compression is still large, and in the range 3-5. Finally, note that the line manifold in Figure 3.3(e) has the structure of a Möbius band. This fact follows from the following symmetry in the line model:

$$F_L(n, m; A, B, \theta + 180°, \rho, w, \sigma) = F_L(n, m; A, B, \theta, -\rho, w, \sigma). \tag{3.20}$$

3.3.4 The Corner

The corner is a common and important feature [84]. Most existing corner detectors are based upon differential invariant based measures of curvature [30], but Rohr [106] recently proposed a parametric model matching approach to corner detection. The simplest way to think about a corner is as the intersection of two non-parallel step edges. In my corner model, shown in Figure 3.4(a), θ_1 is the angle one of these edges makes with the y-axis, and θ_2 the angle subtended by the corner. That is, the corner lies at the intersection of its bounding edges at angles θ_1 and $180° + \theta_1 + \theta_2$. This situation is illustrated in Figure 3.4(b). Algebraically, the intersection can be expressed as the product of two unit step functions. The final corner model has

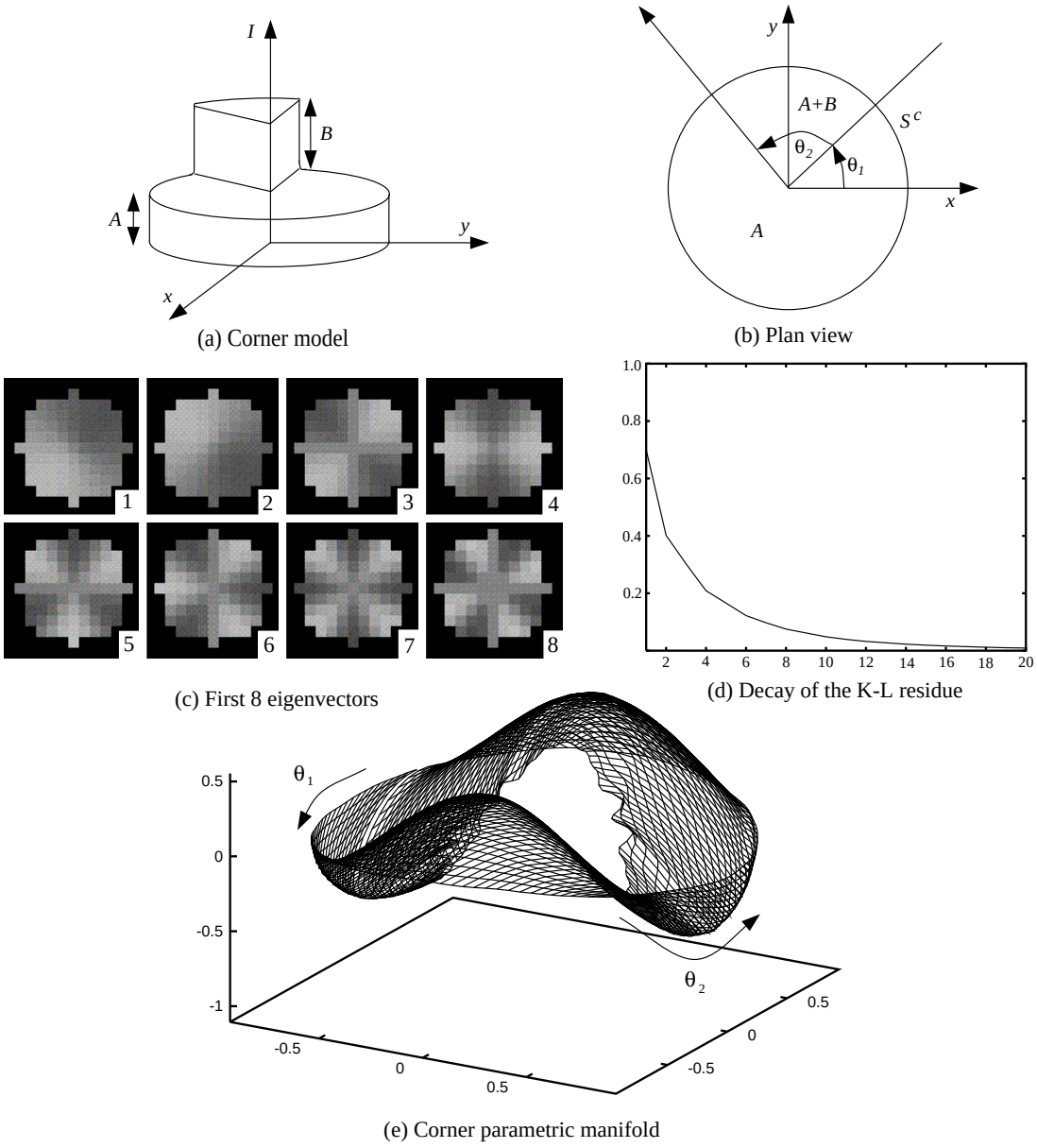


Figure 3.4: The corner is described by the intensity values $A + B$ inside and A outside the corner, the angles θ_1 and θ_2 made by its edges, and the blur parameter σ. Seven eigenvectors are needed to preserve 90% of the information and fifteen eigenvectors for 98%. The corner manifold is shown for a fixed value of σ.

five parameters and is written as:

$$F_C(n,m;A,B,\theta_1,\theta_2,\sigma) = [A + B \cdot u(d(\theta_1)) \cdot u(d(180^\circ + \theta_1 + \theta_2))] * g(x,y;\sigma) * a(x,y)|_{x=n,y=m} \quad (3.21)$$

where $d(\theta) = y \cdot \cos\theta - x \cdot \sin\theta$. The parameter ranges are: $\theta_1 \in [0^\circ, 360^\circ]$, $\theta_2 \in [30^\circ, 120^\circ]$, and $\sigma \in [0.4, 1.0]$. Again, parameter normalization eliminates the parameters A and B. The decay of the K-L residual, shown in Figure 3.4(d), is similar to that of the line. In this case, seven eigenvectors reduce the residual to below 10%, and fifteen eigenvectors are needed to reduce it to less than 2%. The corner manifold is displayed for fixed σ in Figure 3.4(e).

3.3.5 The Circular Disc

My final example feature, the circular disc, is illustrated in Figures 3.5(a) and 3.5(b). Its parameters are its radius r, the direction θ that the center P of the disc makes with the y axis, the sub-pixel localization ρ, and the level of blurring σ. The intensity values inside and outside the disc are $A + B$ and A, respectively. Algebraically, the circular disc can be expressed as:

$$F_{CD}(n,m;A,B,\theta,\rho,r,\sigma) = [A + B \cdot u(r - \mathrm{d}(x,y))] * g(x,y;\sigma) * a(x,y)|_{x=n,y=m} \quad (3.22)$$

where $\mathrm{d}(x,y) = \sqrt{(x + [r+\rho]\sin\theta)^2 + (y - [r+\rho]\cos\theta)^2}$ is the distance of (x,y) from the point $P = (-[r+\rho]\sin\theta, [r+\rho]\cos\theta)$. The parameter ranges are: $\theta \in [0^\circ, 360^\circ]$, $\rho \in [-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$, $r \in [3.0, 12.0]$, and $\sigma \in [0.4, 1.0]$. Again, parameter normalization removes the effects of A and B. The rate of decay of the K-L residual in Figure 3.5(d) is slightly slower than that of the step edge. In this case, four

eigenvectors are needed to reduce the residual to 10%, and eleven eigenvectors to reduce it below 2%. The first eight eigenvectors are shown in Figure 3.5(c) and the feature manifold is displayed in Figure 3.5(e).

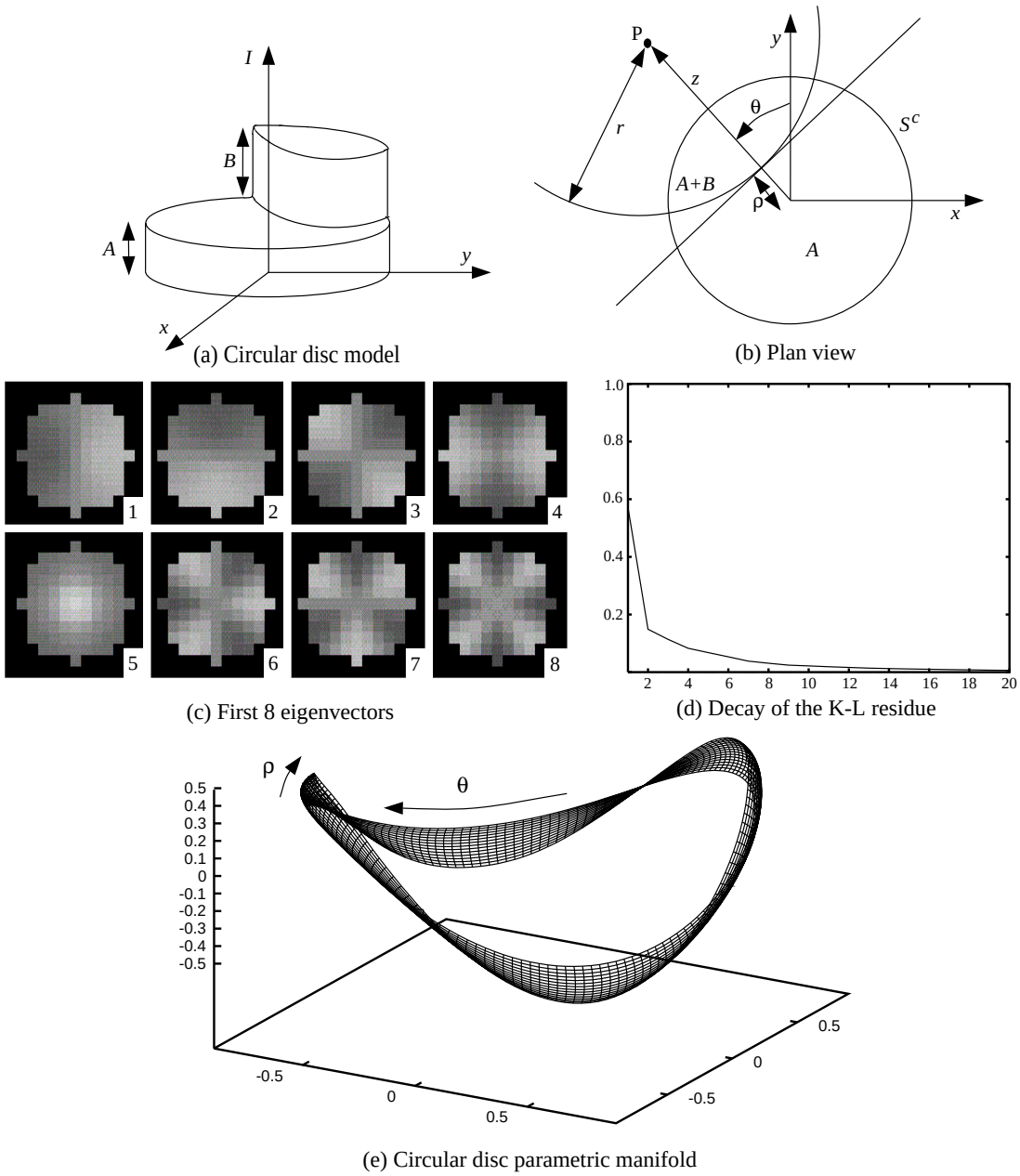
3.4 Feature Detection and Parameter Estimation

In Section 3.2, I introduced parametric manifolds as a representation for arbitrary parametric features. I also showed how this representation can be made much more compact through parameter normalization and dimension reduction. I now explain how feature detection is actually performed using the feature manifold. The first step consists of sampling the manifold.

3.4.1 Sampling the Parametric Manifold

After two parameters have been eliminated by applying the parameter normalization described in Section 3.2.4, the feature manifold is typically parameterized by $k = 3$–4 parameters. To sample the manifold, I first sample each parameter separately, and at equally spaced intervals across its range. Then, I sample the manifold at the cartesian product of these points; ie. I sample the manifold on a k-dimensional grid. One question remains unanswered: how densely should each individual parameter be sampled?

The answer to this question depends upon how much varying each parameter affects the feature. If changing a particular parameter causes the feature to vary rapidly, it should be sampled densely to capture the complete variation in the feature. On the other hand, if changing a parameter results in only a small change



(a) Circular disc model

(b) Plan view

(c) First 8 eigenvectors

(d) Decay of the K-L residue

(e) Circular disc parametric manifold

Figure 3.5: The circular disc is described by the intensity parameters A and B, the radius r of the disc, the angle θ subtended by the center of the disc, the sub-pixel localization ρ, and the blur parameter σ. Four eigenvectors are needed to preserve 90% of the information and eleven eigenvectors for 98%. The manifold is displayed for fixed values of σ and r.

Table 3.1: Automatically generated sampling intervals for the five example features. These sampling intervals were generated by attempting to ensure that the distance between each pair of neighboring sample points is the same, while at the same time limiting the total to 50,000 sample points. The results can also be used to assess the importance of each parameter to the feature. The most important parameters are the ones with the smallest sampling intervals.

Feature	Step Edge		Roof Edge		Line		Corner		Circular Disc	
No. Sample Points	49500		45300		41040		45650		41328	
	$\Delta\theta$	2.007^o	$\Delta\theta$	0.796^o	$\Delta\theta$	2.532^o	$\Delta\theta_1$	2.171^o	$\Delta\theta$	2.928^o
	$\Delta\rho$	0.082	$\Delta\rho$	0.081	$\Delta\rho$	0.106	$\Delta\theta_2$	2.930^o	$\Delta\rho$	0.144
	$\Delta\sigma$	0.136	$\Delta\sigma$	0.161	$\Delta\sigma$	0.376	$\Delta\sigma$	0.122	$\Delta\sigma$	0.174
					Δw	0.218			Δr	2.282

in the feature, there is little point in sampling it densely since the noise inherent in the image will fundamentally limit how accurately that parameter can be estimated anyway. As a rough guideline, the distance between between neighboring samples should be of the same order of magnitude as the average change caused by noise.

In the absence of an accurate estimate of the noise level, I first decide how many sample points can be afforded, for either time or space complexity reasons. Then, I sample the manifold as densely as possible with approximately that number of sample points. Table 3.1 contains the output of an algorithm that first estimates the average rate of change of the feature location with respect to each parameter, and then uses these estimates to derive sampling intervals. The input to the algorithm was the request to generate manifold samplings containing approximately 50,000 sample points. The output is displayed in a separate column for each feature and consists of the sampling interval determined for each parameter.

3.4.2 Search for the Closest Sample Point

Once the manifold has been sampled, feature detection can be performed by finding the closest sample point to the point corresponding to the pixel intensity values in a novel image window. If the closest sample point is near enough, the feature is detected and the parameters of the sample point can be used as estimates of the parameters of the feature just detected. On the other hand, if the closest sample point is too far away, the feature is not detected.

Finding the nearest neighbor amongst a fixed set of points to a given novel point is a well studied problem that was first posed by Knuth [59]. A recent paper by Yanilos [119] contains a comprehensive survey of algorithms developed since then. The task of finding the closest sample point on the manifold has more structure than the general nearest neighbor problem since the sample points lie on the manifold. Rather than using any of the general purpose algorithms, I take advantage of the locally smooth nature of the feature manifolds and use a less general but faster search technique. In particular, I used a 4-level heuristic coarse-to-fine search. It does not guarantee finding the closest sample point for pathological manifolds, but through statistical tests I found that it performs very well in practice. For each of the five manifolds sampled using the intervals in Figure 3.1, the coarse-to-fine search results in a speed-up of about 50-100 times over linear search, with average error less than the spacing between neighboring samples.

The coarse-to-fine search is both conceptually simple as well as very easy to implement. The manifold is sampled several times, giving a sequence of grids, from a very coarse grid with few points up to the finest grid containing the most points. The finest grid consists of the sample points given by the intervals in Figure 3.1.

The search begins by finding the closest point on the coarsest grid using a brute force linear search. This does not take long since the coarsest grid does not contain many points. The search then moves to the next finest grid. This grid is searched locally in the region of the result of the previous level. This search is also a linear brute force search. Again, it does not take long since it is only a local search and on a relatively coarse grid. This approach is repeated for each grid in turn, reducing the size of the local search at each step, until the finest grid is reached. The result of the local search on the finest grid is output as the final result.

3.4.3 Further Efficiency Improvements

On a 1993 DEC Alpha 3600 workstation with no additional hardware, the coarse-to-fine search for a 3 parameter manifold sampled at 50,000 points in a 10-D subspace takes approximately 1 milli-second. So, applying the detector to every pixel in a 512×480 image takes around 4 minutes. This figure is by no means the best that can be achieved in terms of efficiency:

Pattern Rejection: The coarse-to-fine search does not need to be applied at every pixel in the image. This observation is almost as old as edge detection itself and is explicitly mentioned in [50]. Combining a variety of techniques, I have already reduced the time to process a 512×480 image to less than a minute. I first threshold on the total coordinate variance ν^2 computed during parameter normalization. Avoiding feature windows with small total variance in this way is similar to the use of the interest operator [73] to assess the reliability of potential stereo correspondence matches. Next, I threshold on the distance of the input from the K-L subspace. Since the distance from the subspace

is approximately a lower bound on the distance from the manifold, the pixel can be eliminated if the input is too far from the subspace. Finally, using the pattern rejection techniques in [4] and[5], it is even possible to eliminate most of the cost of computing the distance to the K-L subspace.

Parallel Implementation: Feature detection is inherently a parallelizable task because a detector can be applied to each pixel independently. An implementation on a multi-processor workstation could easily cut the times mentioned above by a factor of 3-4, or more. Moreover, it is reasonable to expect continuing performance increases for the individual processors, thereby further improving the efficiency. It is safe to expect that, within a few years, a standard workstation will be able to apply these detectors in close to real-time.

Chapter 4

Experimental Evaluation

In this chapter, I present the results of an experimental evaluation of the feature detector developed in Chapter 3. Of the four evaluation methodologies described in Section 2.5.2, I applied two: (1) statistical tests on synthetically generated data, the results of which are presented in Section 4.1, and (2) subjective human evaluation, the results of which are presented in Section 4.2. Later in this thesis, I propose a class of benchmarks for the evaluation of edge detectors that can be computed directly from the output of the detector. In Chapter 6, I include the results obtained by my step edge detector on these benchmarks.

4.1 Statistical Tests

As I described in Section 2.5.1, there are a large number of different performance measures that can be estimated using statistical tests. Here, I just consider the two most fundamental: (1) feature detection robustness (ie. the rates of occurrence of false positives and false negatives) in Section 4.1.1, and (2) the parameter esti-

mation accuracy in Section 4.1.2. In both cases, I compare the step edge detector developed in the previous chapter with a Canny-like operator [20] and the Nalwa-Binford [80] detector. In doing so, the aim is to demonstrate that the parametric manifold step edge detector performs comparably to these well known and highly regarded detectors. I also compare the performance of the parametric manifold technique across the five example features. The goal of this second comparison is to demonstrate the generality of the algorithm by showing that the performance is similar for all five features. In the remainder of this section, I largely follow the expermental approach taken by Nalwa and Binford in [80].

4.1.1 Feature Detection Robustness

The statistical test for feature detection robustness consists of two phases. In the first one, I generate a large number of ideal features, add zero-mean Gaussian noise to them, and then apply the detectors. Whenever a detector fails to detect a feature, I increment a count of false negatives. The second phase consists of generating a large number of windows that do not contain the feature, adding noise, and again applying the detectors. Whenever a detector responds erroneously and mistakenly detects a feature, I increment a count of false positives.

Although the basic idea behind the comparison is simple enough, there are a number of difficult decisions that need to be made. The first problem arises because each detector is based upon its own model of a feature. My step edge model and the Nalwa-Binford model are similar, but the Canny-like operator is based upon a differential invariant rather than a parametric model. Since I took great care modeling both image formation and the features themselves, I used my feature

models in all of the tests. Doing so slightly biases the comparison with Canny and Nalwa-Binford in favor of my step edge detector. However, since my goal is only to demonstrate that the detectors perform similarly, this decision is not as important as it would be if my aim was to claim superiority.

For fairness, I changed some of the details of my feature models. Both the Canny and Nalwa-Binford detectors assume a constant amount of blur, so I fixed the value of σ in my step edge model to be 0.6 pixels. Secondly, the Nalwa-Binford detector is based upon a square 5×5 window, as is the Canny-like operator in the implementation[1] that I used. Hence, I used a square window containing $N = 25$ pixels for the comparison with Canny and Nalwa-Binford, that is as opposed to the $N = 49$ pixel disc-shaped window used in Figure 3.1.

Another difficult issue is the lack of a model for a window of data not containing a feature [79]. I resolve this issue, as was done in [80], by taking a constant intensity window as the characteristic non-feature. During my investigation into the selection of optimal weighting functions in Chapter 5, I generalize this notion of what is not a feature. The finally difficulty is the need to be able to measure the amount of noise in a consistent way across all five features. I define the S.N.R. of an arbitrary feature to be $\frac{2\times\nu}{\sigma_{noise}}$, where ν^2 is the coordinate total variance of the feature instance defined in Section 3.2.4, and σ_{noise} is the standard deviation of the added Gaussian noise. The reason for this definition is that, for a step edge with no blur in a window where half of the pixels are on each side of the edge, the value

[1] I used an implementation of the Canny operator provided by Geoff West of Curtin University, Western Australia. This implementation is publically available on the Web from the URL http://www.cs.curtin.edu.au/~geoff/. Geoff West's implementation only computes the Gaussian smoothed gradient, which I simply threshold to detect edges. For simplicity, I do not find the zero crossing of the second directional derivative. Neither do I perform hysteresis [20] since it uses information derived from neighboring windows, something I explicitly outlawed in this thesis.

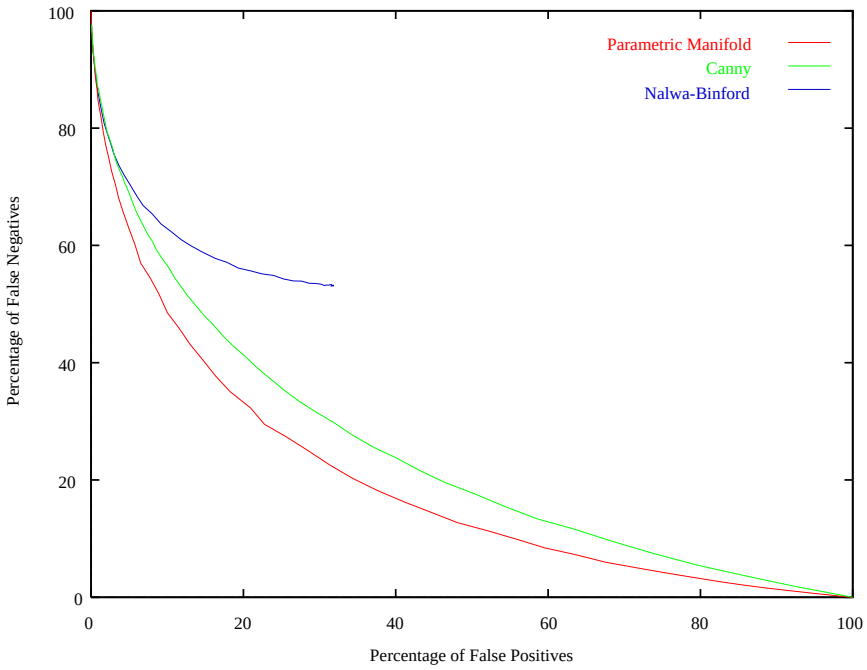


Figure 4.1: A comparison of the feature detection robustness for the Canny, Nalwa-Binford, and parametric manifold step edge detectors for S.N.R. = 1.0. I plot false positives against false negatives. For each detector, the result is a curve parameterized by the threshold inherent in that detector. The closer a curve lies to the origin, the better the performance. It can be seen that the Canny detector and the parametric manifold detector perform comparably, with the parametric manifold algorithm doing marginally better. The results for the Nalwa-Binford detector are consistent with those presented in [80], but are of a fundamentally different nature. See the text for further discussion.

of this expression is the same as the definition used in [80].

In Figure 4.1, I compare the feature detection robustness of the Canny, Nalwa-Binford, and parametric manifold step edge detectors for S.N.R. 1.0. Inherent in each detector is a threshold. The Canny operator thresholds on the gradient magnitude, the Nalwa-Binford detector thresholds on the estimated step size, and the parametric manifold detector thresholds on the distance from the manifold. As the threshold is varied, for a fixed level of noise, the relative number of false positives and false negatives changes. So, I plot a curve of false positives against

false negatives parameterized by the appropriate threshold. The closer a curve lies to the origin in Figure 4.1, the better the performance. Hence, my detector and the Canny detector perform comparably, with my algorithm doing slightly better.

The results for the Nalwa-Binford detector are consistent with those presented in [80]. I did not use step 2)' of the algorithm. The percentage of false positives for the Nalwa-Binford detector never exceeds about 32%. This is in agreement with Figure 8 of [80]. Secondly, for S.N.R. of 1.0 the number of false negatives in Figure 4.1 never drops below about 56%, whereas in Figure 9 of [80] its lowest level is 77%. These two numerical results are slightly different because: (1) I use a different model to generate the ideal step edges, and (b) my definition of S.N.R. yields a slightly lower value than the definition in [80] due to blurred and off-center edges. Comparing the results with those in Figure 9 of [80], it can be seen that the curve in Figure 4.1 corresponds to a curve somewhere between S.N.R. 1.0 and 2.0. The reason that the Nalwa-Binford detector performs qualitatively differently to the Canny and parametric manifold detectors is its inherent conservatism, as enforced by steps 4) and 5) of the algorithm. See page 704 of [80].

In Figure 4.2, I compare the feature detection robustness of the five example features introduced in Chapter 3. In the figure, the curves are all plotted for S.N.R. 1.0 and for a disc shaped window containing 61 pixels. It can be seen immediately that the performance for the step edge and the circular disc is marginally superior to that for the other three features, but four of the five features perform similarly. The roof edge is somewhat more noise sensitive than the other four. One method of reducing the noise sensitivity is to use a slightly larger window. If the window size is increased to a disc containing 89 pixels, the performance is greatly enhanced.

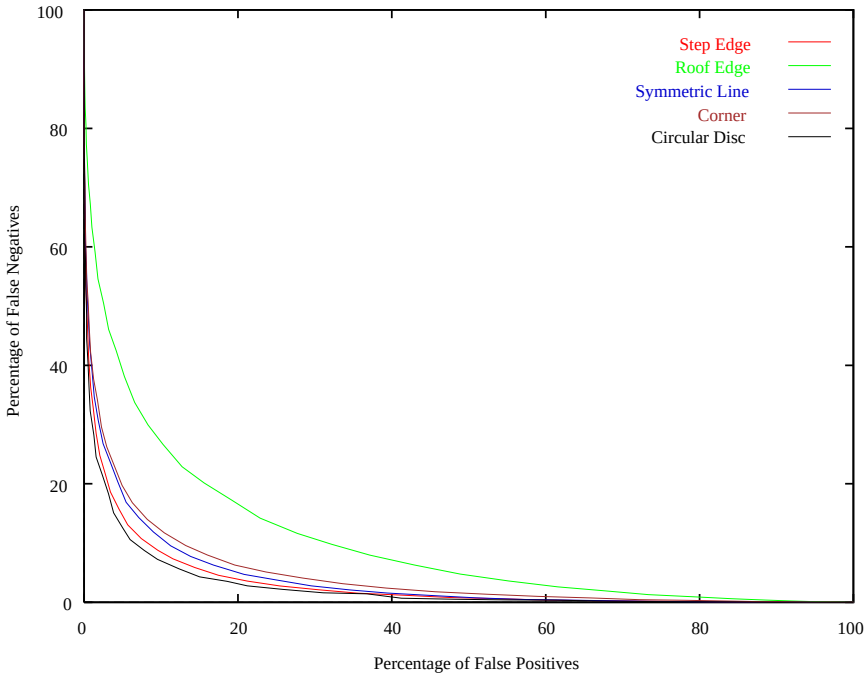


Figure 4.2: A comparison of the feature detection robustness for the five example features of Chapter 3. All of the results are for S.N.R. = 1.0 and for a disc shaped window containing 61 pixels. It can be seen that the step edge and circular disc are slightly less noise sensitive than the other features and that the roof edge is the most noise sensitive.

Naturally, the performance for all five feature also improves rapidly with S.N.R.

4.1.2 Parameter Estimation Accuracy

Assessing the parameter estimation accuracy of a feature detector is relatively straightforward when compared to how difficult it is for the feature detection robustness. Again, I follow the approach taken in [80]. Conducting the experiments consists of generating a large number of ideal features, adding a known amount of zero-mean white Gaussian noise, applying the detectors, and finally measuring the accuracy of the estimated parameters. The question of which models should be used to generate the features is still problematic. For the same reasons as above, I

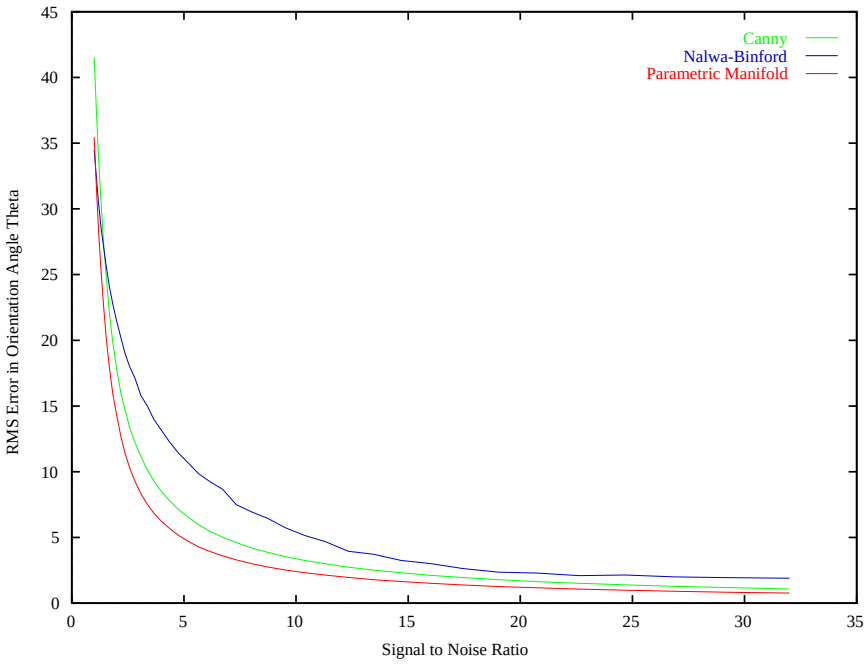


Figure 4.3: A comparison of the orientation estimation accuracy for the three step edge detectors. I took synthesized step edges, added noise to them, and then applied the three detectors. I plot the R.M.S. error in the orientation estimate against the S.N.R. At all noise levels, the parametric manifold detector slightly outperforms both the Nalwa-Binford and Canny detectors.

used my feature models with the same modifications for fairness.

Figures 4.3–4.6 contain the results of the comparison of my step edge detector with the Canny and Nalwa-Binford detectors. In each of the four figures, I plot the R.M.S. error in the estimate of one of the parameters against the S.N.R. Figure 4.3 contains the results for the orientation parameter θ. Since the implementation of the Canny detector that I used does not provide estimates of the other parameters (the sub-pixel localization ρ, the base intensity level A, and the intensity step B), the other three figures only contain comparisons with the Nalwa-Binford detector. In terms of orientation estimation, the parametric manifold detector performs slightly better than the other detectors. For the sub-pixel localization the perfor-

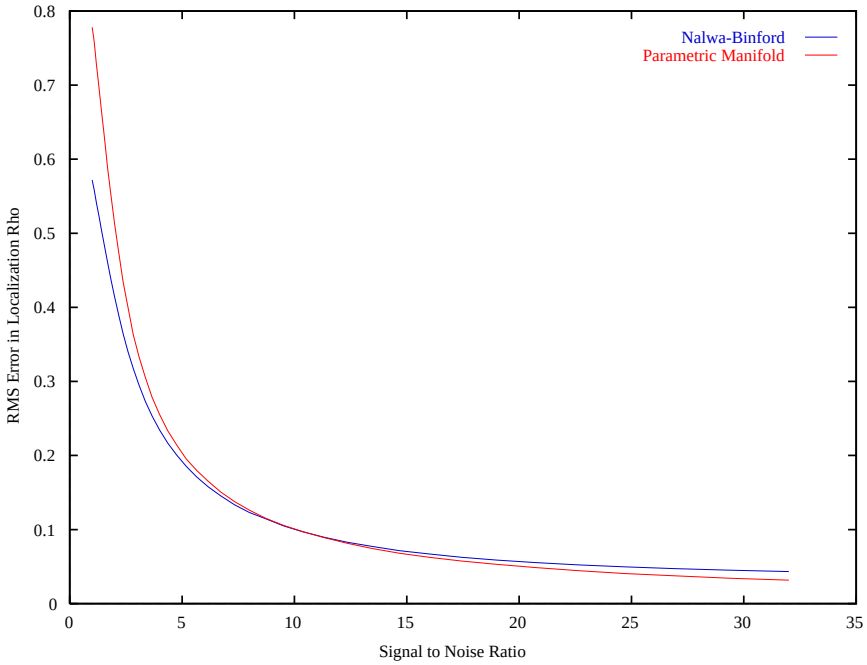


Figure 4.4: A comparison of the localization estimation accuracy for the Nalwa-Binford and parametric manifold step edge detectors. Both detectors perform similarly.

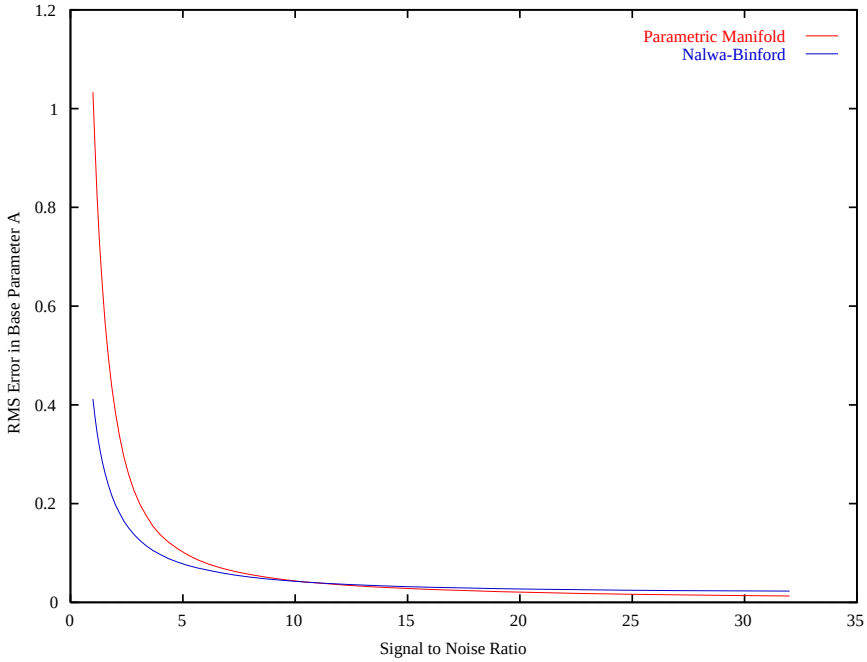


Figure 4.5: A comparison of the base intensity estimation accuracy for the Nalwa-Binford and parametric manifold step edge detectors. The Nalwa-Binford performs better than the parametric manifold detector for high noise levels.

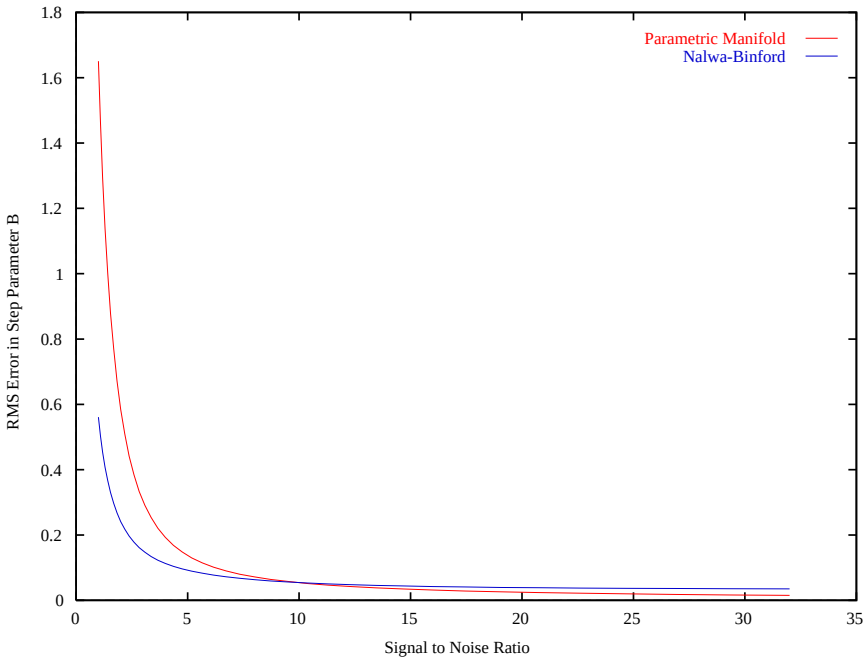


Figure 4.6: A comparison of the step intensity estimation accuracy for the Nalwa-Binford and parametric manifold step edge detectors. The Nalwa-Binford performs better than the parametric manifold detector for high noise levels.

mance is about the same, but for the estimation of the two brightness parameters the Nalwa-Binford is slightly better. The fact that the Nalwa-Binford detector does somewhat better than the parametric manifold detector when estimating the intensity parameters suggests that it may be possible to improve on the algorithm used to recover the normalized parameters described in Section 3.2.5. Finally, note that the results for the Nalwa-Binford detector are consistent with those presented in [80], after allowing for the slightly different feature models and definition of S.N.R.

Next, I compare the performance of four of my five example features. The results for the circular disc are almost identical to the step edge and so are omitted for clarity. Since all of the feature models have an orientation parameter, in Figure 4.7, I plot the R.M.S. error in the orientation estimate against the S.N.R.

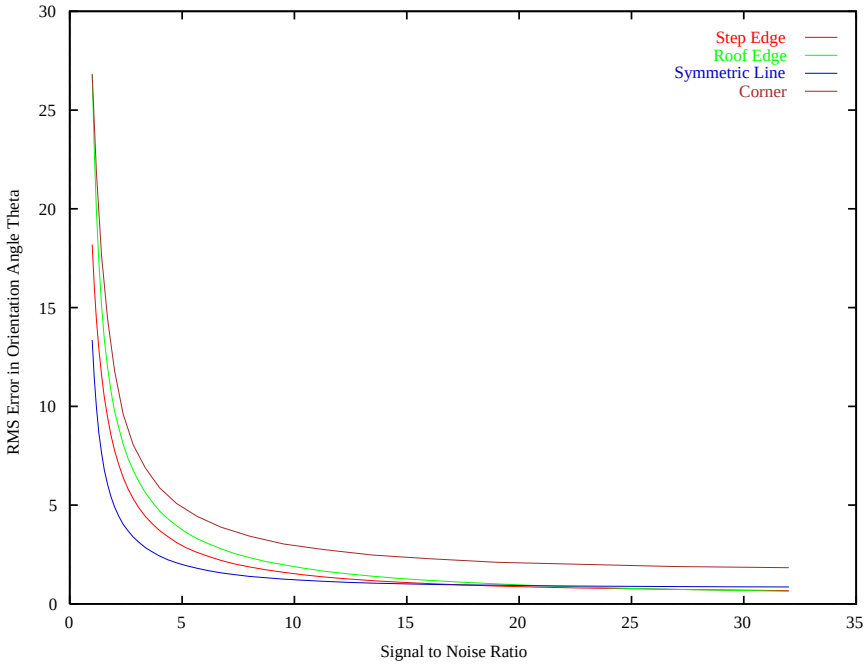


Figure 4.7: A comparison of the orientation estimation accuracy for the step edge, the roof edge, the symmetric line, and the corner. All four features perform similarly.

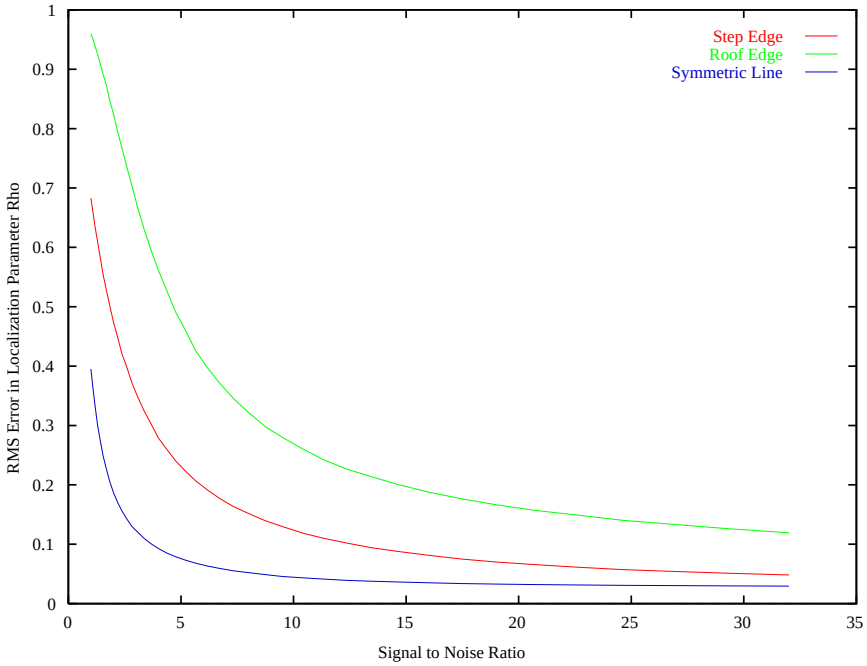


Figure 4.8: A comparison of the localization estimation accuracy for the step edge, the roof edge, and the symmetric line. The line performs the best, followed by the step edge, with the roof edge performing by far the worst.

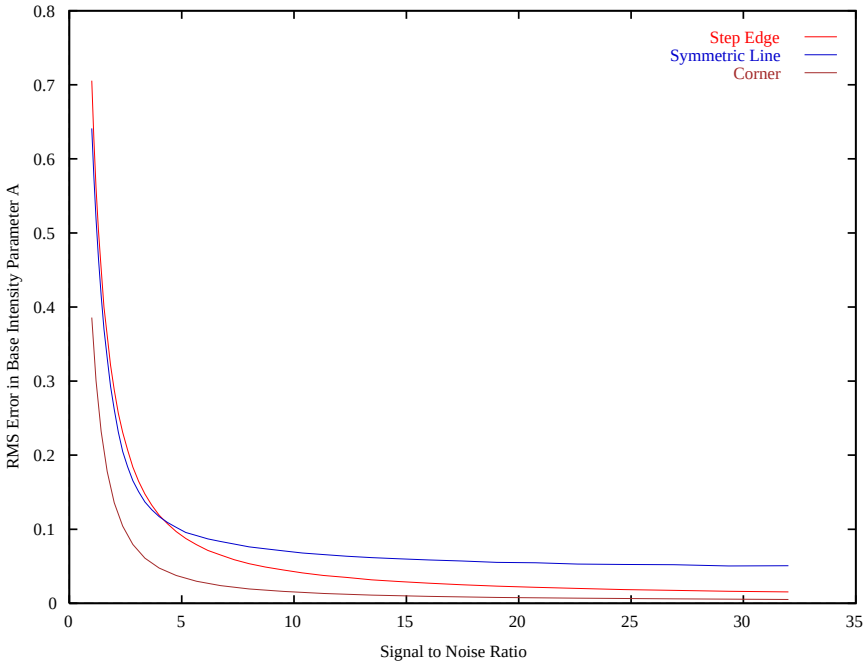


Figure 4.9: A comparison of the base intensity estimation accuracy for the step edge, the symmetric line, and the corner. All three features perform similarly.

As can be seen, all of the features perform similarly, with the symmetric line doing the best, followed closely by the step edge and the roof edge. Only for the corner is the orientation estimation accuracy somewhat worse than for the other features. In Figure 4.8, a similar graph is plotted for the localization estimation accuracy of the three features that have this parameter. Here, there is a dramatic variation in the performance across the three features, with the symmetric line doing the best followed by the step edge. The roof edge does very poorly. The good performance of the line can be explained by the fact that its localization estimate can be regarded as the average of the estimates for the two step edges it is composed of. In Figures 4.9 and 4.10, I plot the estimation accuracy for the base and step intensity levels. The results for the roof edge are omitted since the intensity levels have a different meaning. The results are quite similar for all three features, with one

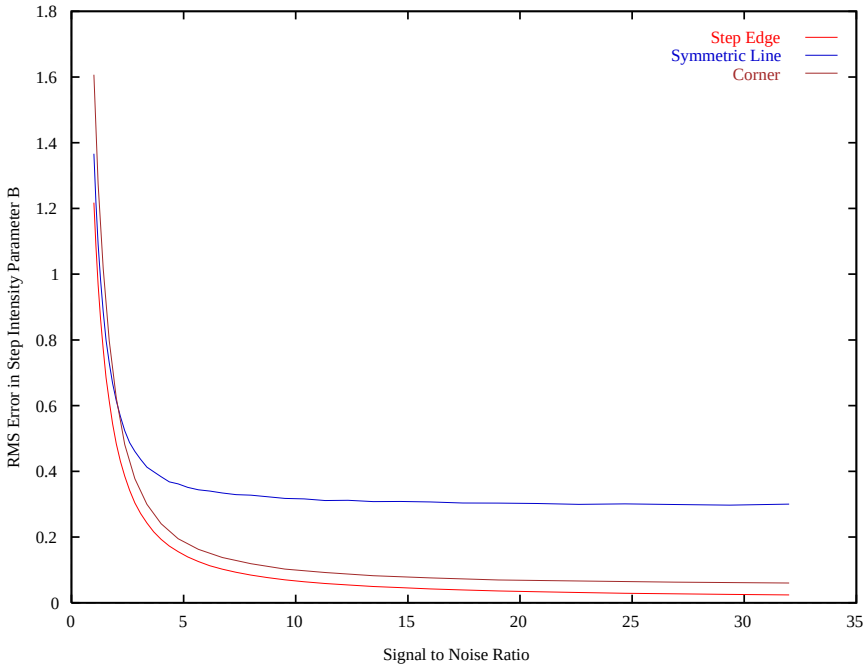
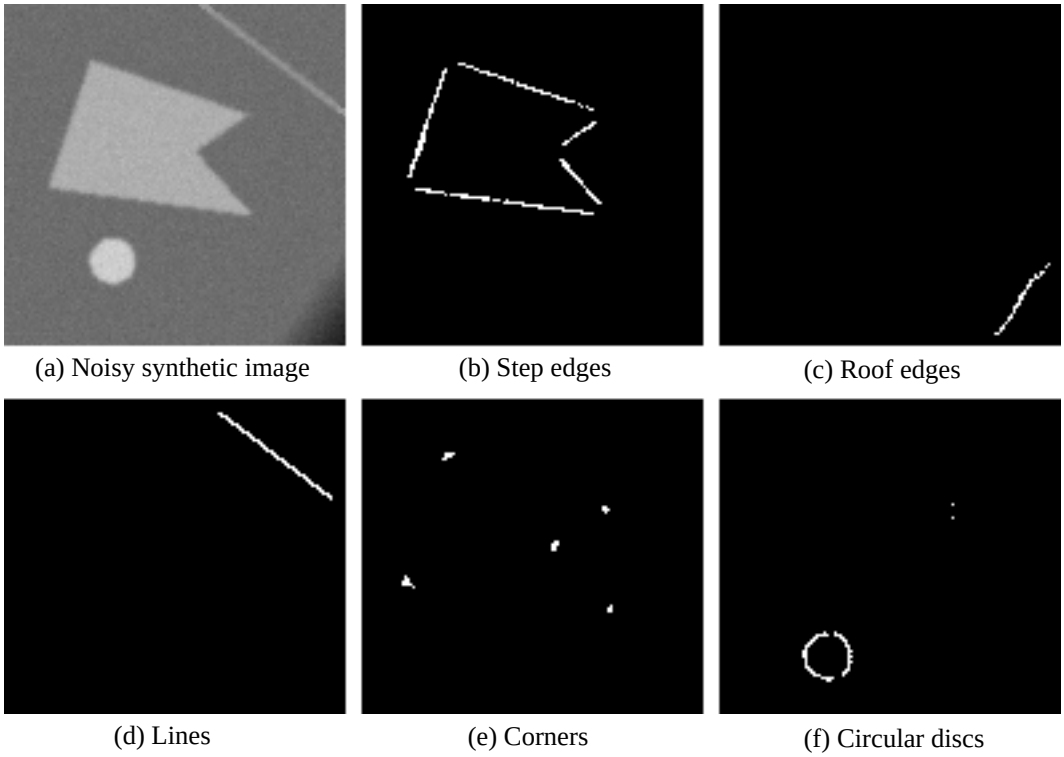


Figure 4.10: A comparison of the intensity step estimation accuracy for the step edge, the symmetric line, and the corner. The symmetric line performs worse than the other two features, since, for thin lines, very few of the pixels are at the upper intensity level.

major exception. The estimation of the intensity step for the line is significantly worse than for the step edge and the corner. The explanation of this fact is that, for thin lines, very few of the pixels are at the upper intensity level.

4.2 Subjective Human Comparison

The most frequently used evaluation technique consists of applying the detectors to a number of real or synthetic images, and then presenting the output to a human for their subjective evaluation. The advantage of using synthetic images is that there can be little argument as to which features should be detected in the images. The disadvantage is that artificial noise processes need to be assumed to generate the images. Typically, there is no way to justify that the noise added is representative of



(a) Noisy synthetic image (b) Step edges (c) Roof edges

(d) Lines (e) Corners (f) Circular discs

Figure 4.11: The results of applying the five feature detectors to a noisy synthetic image. The five different features are detected and discriminated in an image using the same algorithm. The only difference between the detectors is the parametric model of the feature used to build the detector.

the noise actually found in real images. Here, I present results on synthetic images in Section 4.2.1, on images scanned from [86] in Section 4.2.2, and on images taken from the INRIA image database in Section 4.2.3.

4.2.1 Application to Synthetic Images

In Figures 4.11(b)–(f), I display the results of applying my five example detectors to the noisy synthetic image in Figure 4.11(a). The synthetic image is of size 128×128 pixels and contains a irregular pentagon (intensity 175 grey levels), a circular disc (radius 8.5 pixels, intensity 206 grey levels), a line (width 2.3 pixels, intensity

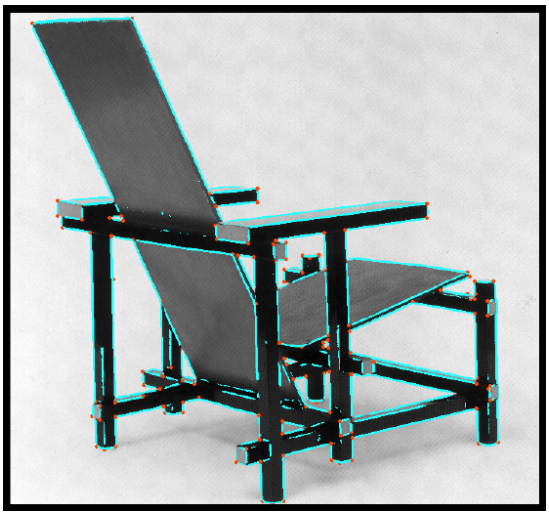
153 grey levels), and a roof edge (slope 4 grey levels per pixel). The background intensity is 110 grey levels. The image was first blurred with a Guassian ($\sigma = 0.6$ pixels) and then white zero-mean Gaussian noise ($\sigma = 4.0$ grey levels) added.

The results shown in Figures 4.11(b)–(f) are obtained by simply thresholding the distance to the feature manifold. At pixels where two or more feature detectors register the presence of a feature, only the one with the closest manifold is detected. As can be seen, the five features are detected and discriminated very well. The only slight confusion is the false detection of circular discs close to one of the corners of the pentagon. This effect is also seen in the real images in the following section. The reason is quite simply that even after only a moderate amount of blurring, corners and circular discs appear very similar.

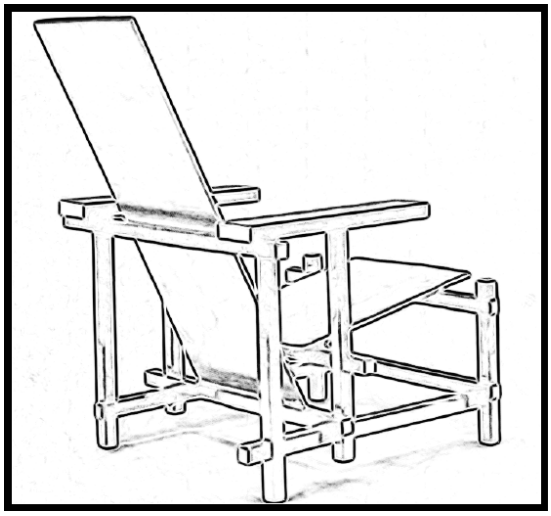
4.2.2 Application to Scanned Images

In Figures 4.12(b)–(d), 4.13(b)–(d), and 4.14(b)–(d), I present some of the results obtained by applying my example feature detectors to the three greyscale images in Figures 4.12(a), 4.13(a), and 4.14(a). The original images were all scanned from [86] using an Envisions 6600S scanner at 200dpi. (Note that the image formation process for a scanner is somewhat different to that for a camera. Hence, the image formation model described in Section 3.2.2 should be regarded as only an approximation. The feature detection performance would be further enhanced if a more accurate image formation model were used.) Feature detection was accomplished by simply thresholding on the distance from the feature manifold. No further post-processing or sophisticated thresholding techniques were applied. One slight change was made to the raw feature maps for clarity of presentation. To make the detected

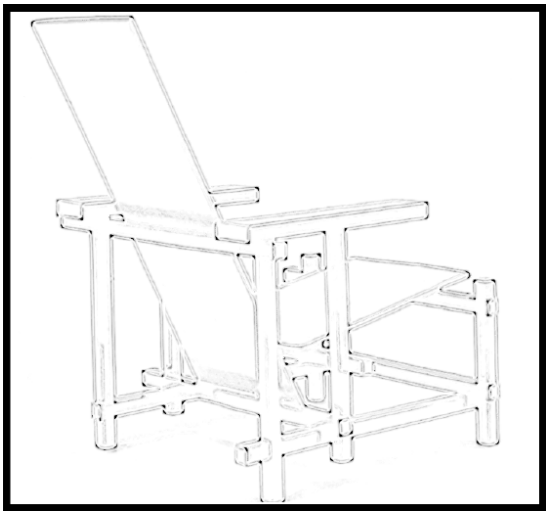
(a) Original image



(b) Step edges (blue) & corners (red)

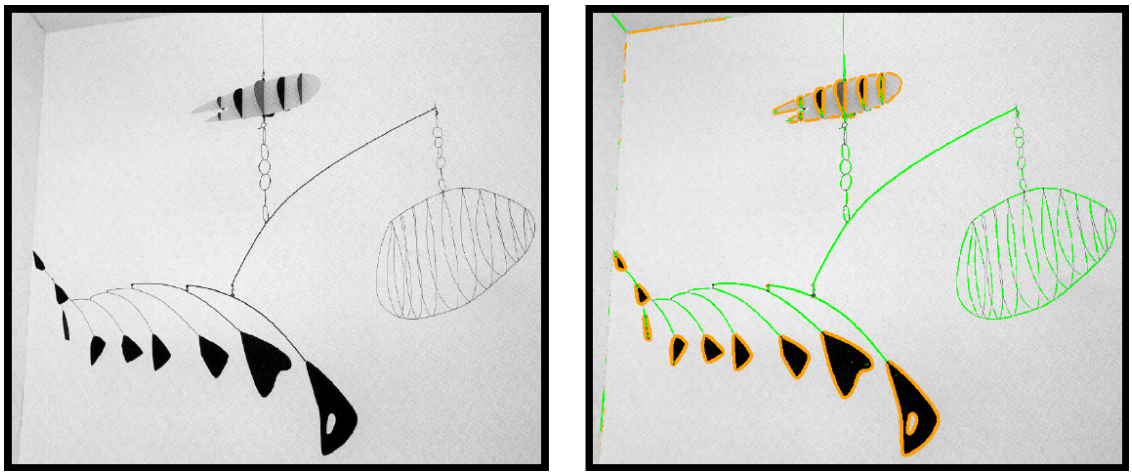


(c) Distance to step edge manifold



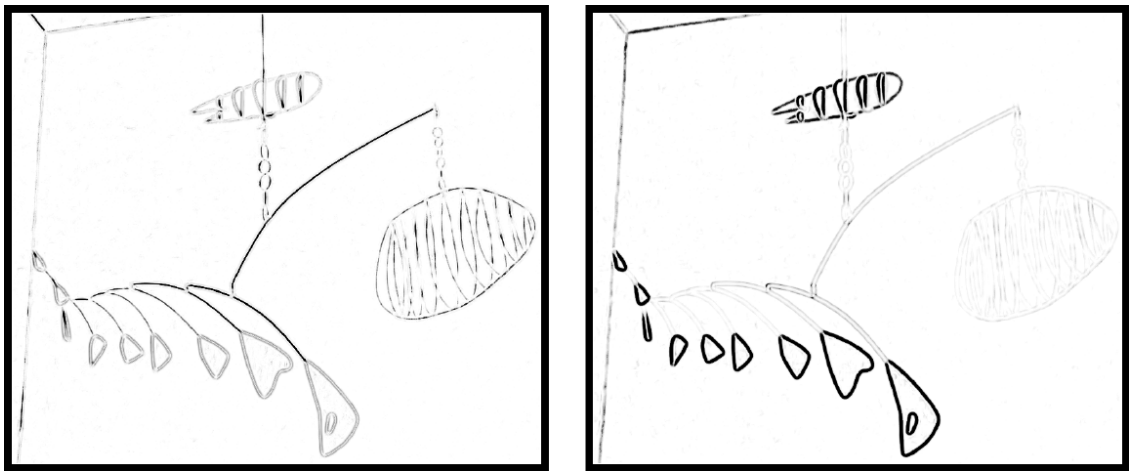
(d) Distance to corner manifold

Figure 4.12: Results of step edge and corner detection for a 711 × 661 pixel image of "Red and Blue," by *Gerrit Rietveld*, circa 1918, scanned from [86]. The raw unthresholded detector outputs in (c) and (d) reflect high accuracy in both detection and localization. Note also that that these results reflect the similarly between the definition of a corner and a step edge as the angle subtended by the corner nears 180°. In (b), simple thresholds were used to find the dominant feature (if any) at each pixel. Some corners remain undetected because their angles are less than the 30° minimum that I imposed in the definition of a corner in Section 3.3.4.



(a) Original image

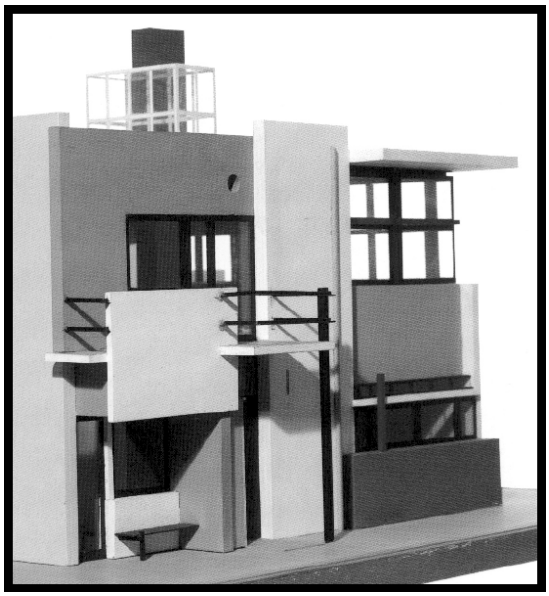
(b) Lines (green) and discs (yellow)



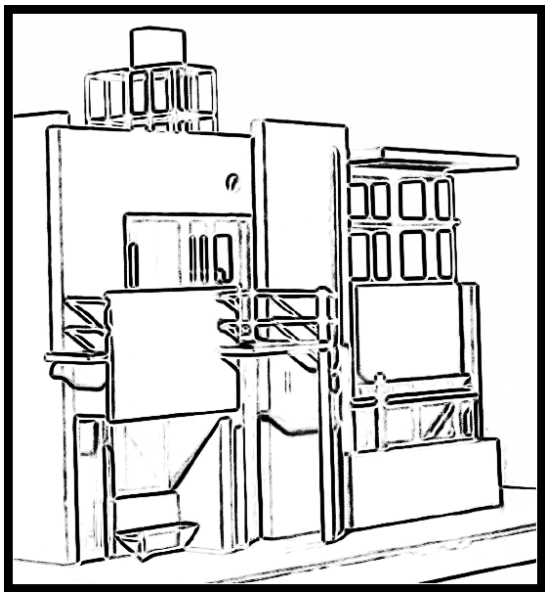
(c) Distance to line manifold

(d) Distance to disc manifold

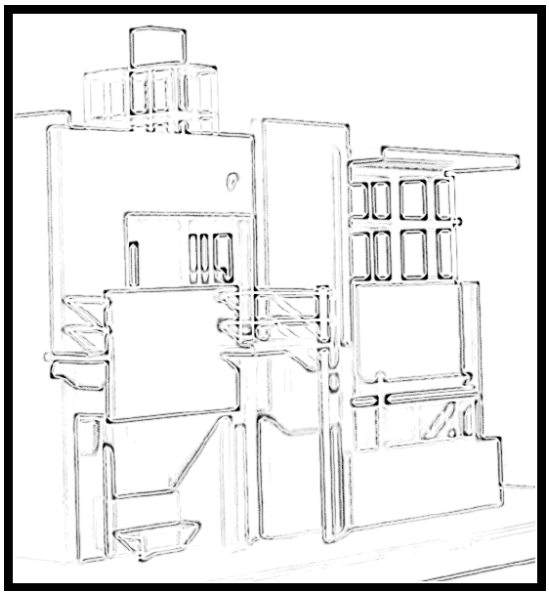
Figure 4.13: Results of line and disc detection for a 796×679 pixel image of "Lobster Trap and Fish Tail," by *Alexander Calder,* 1939, scanned from [86]. Though many of the lines in the image are faint, thin, and incomplete, the line detector does a good job extracting them. In (b), simple thresholds were used to find the dominant feature at each pixel. Again, (c) and (d) indicate considerable similarity between the feature definitions. In this case, a thick line and a disc with a large radius are similar in appearance.



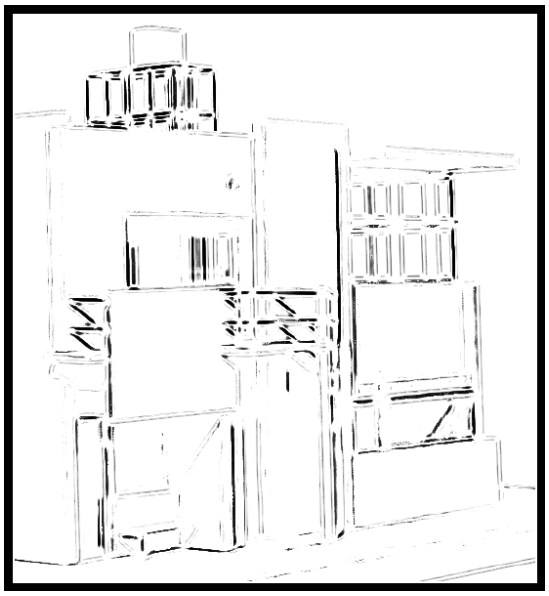
(a) Original image



(b) Distance to step edge manifold



(c) Distance to corner manifold



(d) Distance to line manifold

Figure 4.14: Results of step edge, corner, and line detection for a 564×611 pixel image of "Schröder House," by *Gerrit Rietveld,* 1924, scanned from [86]. These results convey the richness of information obtained when multiple feature detectors are applied to an image, as well as the similarities in some of the feature definitions for extreme parameter values. The outputs (b), (c), and (d), together with the parameter estimates, could serve as the basis for a multi-feature relaxation scheme. See Section 7.3.1 for more discussion.

corners in Figure 4.12(b) more visible on the printed page, I first applied a non-maximum suppression algorithm to localize the corners, and then replaced each detected corner with a 5×5 disc of highlighted pixels.

The outputs of the step edge and corner detectors in Figures 4.12(b)–(d), and the line and circular disc detectors in Figures 4.13(b)–(d) are consistent with the structures of the input images. The results of the step edge, corner, and line detectors in Figure 4.14(b)–(d) are included to convey the richness of information obtained when multiple detectors are applied to an image. The distances from several feature manifolds, together with estimates of the feature parameters, could be very valuable in reinforcing or inhibiting the existence of other features at neighboring pixels. Further discussion of this point is contained in Section 7.3.1.

4.2.3 Application to INRIA Images

In Figures 4.15, 4.16, and 4.17, I present the results of applying the step edge, corner, line, and circular disc detectors to three images taken from the INRIA image database, located at ftp://krakatoa.inria.fr/pub/IMAGES_ROBOTVIS/. The detected edges are overlaid in blue, the detected corners in red, the detected lines in green, and the detected circular discs in yellow. As above, feature detection was accomplished by simply thresholding on the distance from the feature manifold, with no further post-processing applied except for clarity of presentation. As before, the detected corners are replaced with a 5×5 disc so they can be seen. As can be seen, the detected features capture the structure of the images well. Again, note that much more information is available to subsequent processing than that actually shown in the three figures. This information includes the distances to the

Figure 4.15: The results of applying the step edge, corner, line, and circular disc detectors to the 900×900 pixel image "INRIA Bulding." Step edges are overlaid in blue, corners in red, lines in green, and circular discs in yellow.

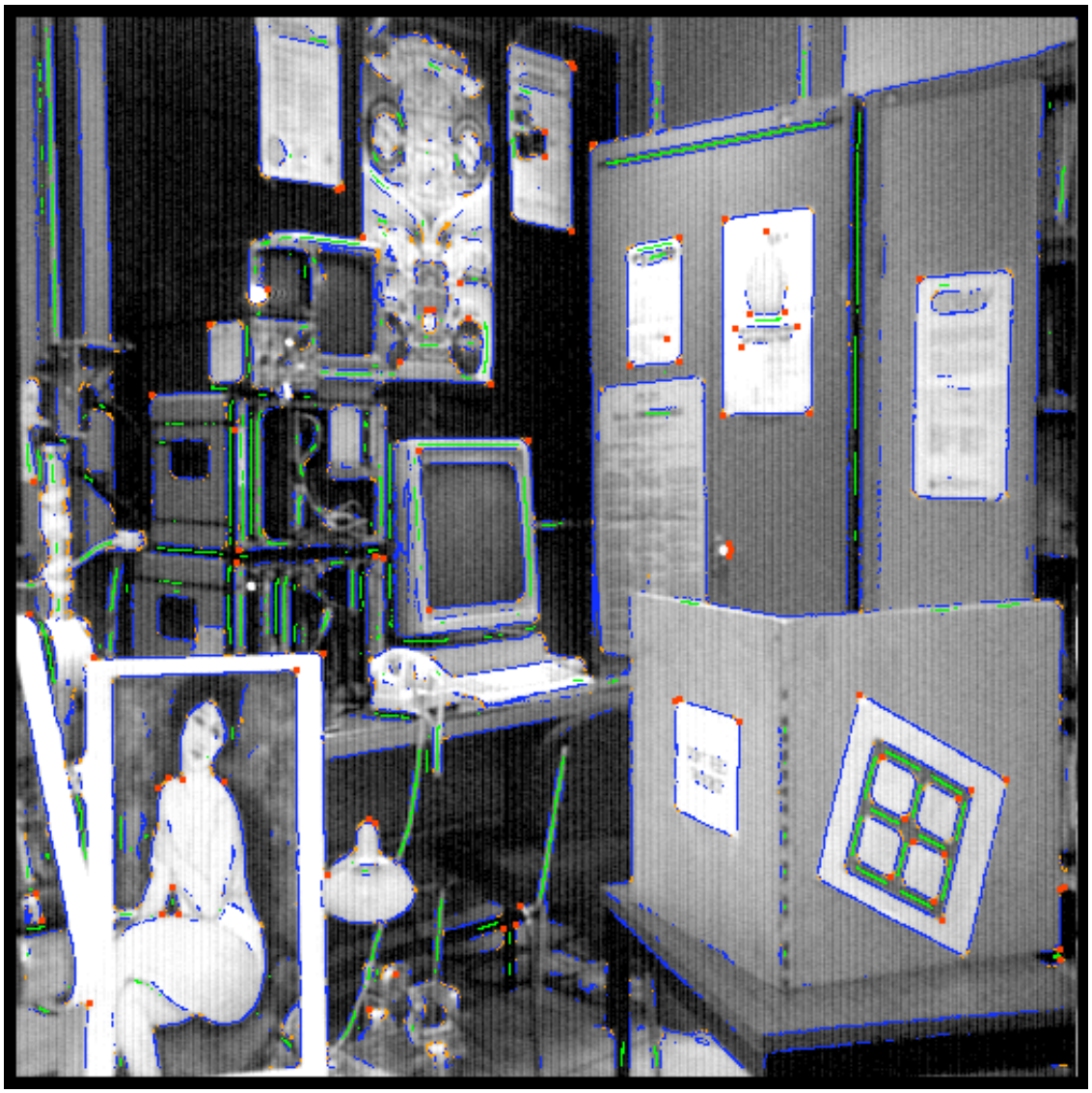


Figure 4.16: The results of applying the step edge, corner, line, and circular disc detectors to the 512×512 pixel image "INRIA Office." Step edges are overlaid in blue, corners in red, lines in green, and circular discs in yellow.

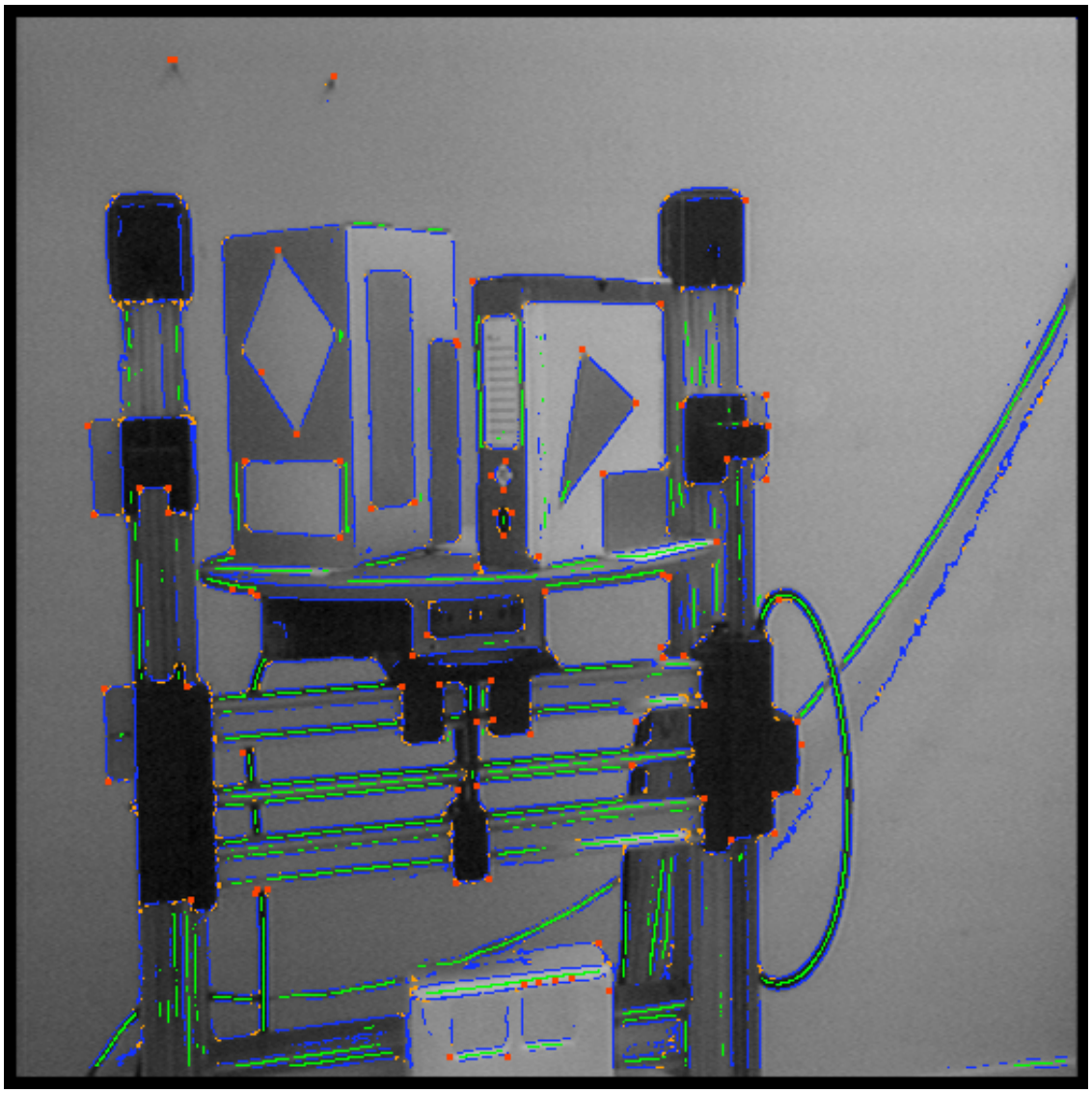


Figure 4.17: The results of applying the step edge, corner, line, and circular disc detectors to the 512×512 pixel image "INRIA Tourn." Step edges are overlaid in blue, corners in red, lines in green, and circular discs in yellow.

four feature manifolds, and estimates of all the feature parameters.

Chapter 5

Optimal Weighting Functions for Feature Detection

5.1 Introduction

In Chapter 3, I proposed a general purpose feature detection algorithm that uses the model matching approach described in Section 2.1. A feature is detected by a model matching detector if there exist valid parameter values such that the image data and the ideal feature instance with those parameters are sufficiently "similar." To measure the degree of similarity, a matching function is required. For convenience, I simply used the Euclidean L^2 norm as the matching function, without any discussion of the decision. In this chapter, I show how the matching function can be selected to maximize the performance of the detector.

To the best of my knowledge, the selection of the matching function for a model matching feature detector has never before been studied in a systematic manner. In fact, most detectors simply use the Euclidean L^2 norm. Other model

matching feature detectors have used weighted L^2 norms, but in all cases the weighting function was chosen in an ad-hoc manner. See Section 2.1.3 for a survey of the use of weighted L^2 norms in feature detection. In this chapter, I restrict attention to the class of weighted L^2 norms as possible matching function, and investigate the choice of the weighting function. In studying this question, I am effectively trying to decide how to weight the contributions of the image data over the feature window. As will be seen, some pixels provide more reliable information than others. For instance, when estimating the parameters of a corner, the center pixels provide little reliable information while the pixels on the periphery are vital for high performance. On the other hand, the center pixels are the most important when estimating the sub-pixel localization of a step edge.

The approach that I take in this chapter consists of first proposing optimality criteria and then optimizing the criteria to derive optimal weighting functions. Optimality criteria should ideally be chosen so that they are closely related to performance. The three most important elements of feature detection performance are: (1) the rate of occurrence of false positives, (2) the rate of occurrence of false negatives, and (3) the parameter estimation accuracy. A perfect detector would produce no false positive, no false negatives, and have zero parameter estimation error. See Section 2.5.1 for a more detailed discussion of performance measures for feature detectors. In this chapter, I propose separate optimality criteria for each of these three key aspects of performance. I then show how they can be combined to form optimality criteria that are more appropriate for specific applications.

Once I have decided upon my optimality criteria, I first investigate analytic solutions for the optimal weighting functions. In particular, I derive the optimal

weighting function for parameter estimation accuracy, under the approximating assumption that the feature manifold is linear. The other optimality criteria are all highly nonlinear and no analytic solution appears possible. Instead, I propose a numerical optimization algorithm that can be used to find the optimal weighting functions. This optimization algorithm can be used for any of the optimality criteria and for arbitrary parametric features. I include the results of applying this algorithm for three of the features studied in Chapter 3.

The remainder of this chapter is organized as follows. I begin in Section 5.2 by describing how weighted L^2 norms can be used as the matching function. In particular, I show how the feature detection algorithm proposed in Chapter 3 can be extended to use an arbitrarily weighted L^2 norm. In Section 5.3, I introduce my optimality criteria and in Section 5.4 show how they can be optimized. First, I present my analytical results. Afterwards, I describe the numerical optimization algorithm, and present the results of applying it to three of features studied in Chapter 3. I conclude this chapter with a discussion of a number of important issues for the design and evaluation of optimal feature detectors.

5.2 Feature Detection using Weighted L^2 Norms

In this section, I describe how the general purpose feature detection algorithm that I proposed in Chapter 3 can be extended to use a weighted L^2 norm, without any significant loss of efficiency. The key step of the algorithm is finding the feature parameters $\mathbf{q}$ that minimize the distance from the point on the manifold with those parameters to the vector of image data. The algorithm to find the parameters consists of three substeps:

1. Apply the parameter normalization described in Section 3.2.4 to eliminate two of the intensity parameters from $\mathbf{q}$.

2. As described in Section 3.2.6, use the Karhunen-Loéve expansion to reduce the dimension of space that the normalized feature manifold lies in.

3. Find the closest point on the normalized feature manifold in the low dimensional subspace using the coarse-to-fine search described in Section 3.4.2.

Using a weighted L^2 norm poses no additional difficulty for the coarse-to-fine search. Therefore, after introducing weighted L^2 norms, I describe how parameter normalization and dimension reduction can be performed using a weighted L^2 norm.

5.2.1 Weighted L^2 Norms

Both the feature detection robustness and the parameter estimation accuracy of the algorithm in Chapter 3 depend upon the function used to measure the distance from the vector of input image data to the feature manifold. If a different function is used, not only will the distance to the manifold change, but also the closest point will be perturbed. Hence, both the features detected and their parameters will depend upon the choice of the matching function. I now introduce weighted L^2 norms as a class of possible matching functions to choose from.

Every measure w on the pixels leads to a different L^2 norm, denoted by either $L^2(w)$ or $\|\cdot\|_w$ [25]. A measure is defined by the weight $w = w(n,m) \geq 0$ that it assigns to each of the pixels $(n,m) \in S$. If $\mathbf{v}_1 = (v_1(n,m))$ and $\mathbf{v}_2 = (v_2(n,m))$ are two vectors of pixel intensity values, their weighted inner product is:

$$\langle \mathbf{v}_1, \mathbf{v_2} \rangle_w \;=\; \sum_{(n,m)\in S} w(n,m) \cdot v_1(n,m) \cdot v_2(n,m) \tag{5.1}$$

and the $L^2(w)$ norm of $\mathbf{v}_1$ is:

$$\|\mathbf{v}_1\|_w \;=\; \sqrt{\langle \mathbf{v}_1, \mathbf{v}_1 \rangle_w}. \tag{5.2}$$

Strictly speaking, for Equation (5.1) to define an inner product rather than a semi-inner product, it is also required that $w(n,m) > 0$ for all $(n,m) \in S$. This is a minor technical point that can be ignored since feature detection does not require the definiteness property of the resulting distance function. The Euclidean L^2 norm, denoted by $\|\cdot\|$, is distinguished as the weighted L^2 norm for which the measure of each pixel is 1.0. Closely related to the Euclidean L^2 norm is the sum of squared differences, or SSD. The SSD of two vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ is the square of the Euclidean L^2 norm of their difference; ie. it is $\|\mathbf{v}_1 - \mathbf{v}_2\|^2$.

Any weighted $L^2(w)$ norm can be used to measure the distance between the vector of image data $\mathbf{I}(a,b) = (I(a+n, b+m))$ and the ideal feature instance $\mathbf{F}(\mathbf{q}) = (F(n,m;\mathbf{q}))$ using:

$$\| \mathbf{F}(\mathbf{q}) - \mathbf{I}(a,b) \|_w \;=\; \left[\sum_{(n,m)\in S} w(n,m) \cdot [F(n,m;\mathbf{q}) - I(a+n, b+m)]^2 \right]^{1/2}. \tag{5.3}$$

Feature detection could be based upon the distance between the vector of image data and the closest ideal feature instance on the manifold. However, since computing the square of an L^2 norm is easier than computing the L^2 norm itself, and since the square root function is monotonic, in Chapter 3 feature detection is equivalently based upon the square of the distance to the closest point on the manifold. Using a weighted L^2 norm, this squared distance is given by:

$$\min_{\mathbf{q}} \| \mathbf{F}(\mathbf{q}) - \mathbf{I}(a,b) \|_w^2 \;=\; \min_{\mathbf{q}} \sum_{(n,m)\in S} w(n,m) \cdot [F(n,m;\mathbf{q}) - I(a+n, b+m)]^2 . \tag{5.4}$$

Once a feature has been detected, its parameters are estimated using the parameter values that actually minimize the expression in Equation (5.4).

5.2.2 Parameter Normalization

In Section 3.2.4, a simple normalization was applied that reduces the number of parameters on the feature manifold and hence the amount of computation required. For each feature instance, the mean coordinate $\mu(\mathbf{q}) = \frac{1}{N}\sum_{(n,m)\in S} F(n,m;\mathbf{q})$ and the total coordinate variance $\nu^2(\mathbf{q}) = \sum_{(n,m)\in S}[F(n,m;\mathbf{q}) - \mu(\mathbf{q})]^2$ are computed. Then, the feature instance is normalized:

$$\overline{F}(n,m;\mathbf{q}) \;=\; \frac{1}{\nu(\mathbf{q})}\,\left[F(n,m;\mathbf{q}) \;-\; \mu(\mathbf{q})\right]. \tag{5.5}$$

For most features, this simple normalization reduces the number of parameters by two because $\overline{F}(n,m;\mathbf{q})$ turns out to be approximately independent of two of the brightness parameters in $\mathbf{q}$. These two parameters can be ignored during the construction of the manifold and recovered once a feature has been detected. See Section 3.2.5 for a description of how to recover the normalized parameters.

If the "all one" vector $\mathbf{c} \in \mathbf{R}^N$ is defined by $\mathbf{c} = (1, 1, \ldots, 1)$ and $\hat{\mathbf{c}} = \mathbf{c}/\|\mathbf{c}\|$, then Equation (5.5) may be rewritten as:

$$\overline{F}(n,m;\mathbf{q}) \;=\; \frac{F(n,m;\mathbf{q}) \;-\; \langle F(n,m;\mathbf{q}), \hat{\mathbf{c}}\rangle\,\hat{\mathbf{c}}}{\|F(n,m;\mathbf{q}) \;-\; \langle F(n,m;\mathbf{q}), \hat{\mathbf{c}}\rangle\,\hat{\mathbf{c}}\|} \tag{5.6}$$

where $\|\cdot\|$ is the Euclidean L^2 norm and $\langle\,\cdot\,,\,\cdot\,\rangle$ is the Euclidean inner product. When using a weighted L^2 norm, normalization can be performed exactly as in Equation (5.6) except that the weighted inner product and weighted L^2 norm should be used in place of their Euclidean equivalents.

5.2.3 Dimension Reduction

In the coarse-to-fine search of Section 3.4.2, it is vitally important that the matching function can be evaluated efficiently. Examining Equation (5.3), it can be seen that $4 \cdot N - 1$ arithmetic operations are needed to evaluate $\|\overline{F}(n, m; \mathbf{q}) - \overline{I}(a+n, b+m)\|_w^2$. In the Euclidean case, this computation is an SSD which can be performed using only $3 \cdot N - 1$ arithmetic operations. In Chapter 3, the Karhunen-Loéve expansion is applied as a dimension reduction technique to reduce the cost of evaluating the SSD even further. I now describe how dimension reduction can be performed when a weighted L^2 norm is being used.

Applying dimension reduction when using a non-uniformly weighted L^2 norm is performed as follows. If $\{\mathbf{e}_j \mid j = 1, 2, \ldots, N\}$ is an orthonormal basis, with respect to the underlying inner product $\langle \cdot, \cdot \rangle_w$, then:

$$\|\overline{F}(n, m; \mathbf{q}) - \overline{I}(a+n, b+m)\|_w^2 \;=\; \sum_{j=1}^{N} \Big[\langle \overline{F}(n, m; \mathbf{q}), \mathbf{e}_j \rangle_w - \langle \overline{I}(a+n, b+m), \mathbf{e}_j \rangle_w \Big]^2 . \tag{5.7}$$

After a suitable change of basis vectors, dimension reduction corresponds to discarding a number of the basis vectors, without loss of generality the last few, and restricting attention to the low dimensional subspace spanned by $\{\mathbf{e}_j \mid j = 1, 2, \ldots, d\}$, where d is the dimension of the low dimensional subspace. Hence, after applying dimension reduction it is possible to approximate:

$$\|\overline{F}(n, m; \mathbf{q}) - \overline{I}(a+n, b+m)\|_w^2 \;\approx\; \sum_{j=1}^{d} \Big[\langle \overline{F}(n, m; \mathbf{q}), \mathbf{e}_j \rangle_w - \langle \overline{I}(a+n, b+m), \mathbf{e}_j \rangle_w \Big]^2 . \tag{5.8}$$

Since $\langle \mathbf{a}, \mathbf{e}_j \rangle_w$ is the j^{th} component of the vector $\mathbf{a}$ in the low dimensional subspace, it follows that the square of the weighted L^2 norm can be estimated with an SSD

in the low dimensional subspace. This result assumes the use of an orthonormal basis, but orthonormality can easily be obtained using Gram-Schmidt [101]. So, the weighted L^2 norm can be computed with exactly the same cost as the Euclidean L^2 norm: ie. using $3 \cdot d - 1$ arithmetic operations. Hence, the only additional computation cost of using a non-uniformly weighted L^2 norm is that incurred during parameter normalization and projection into the low dimensional subspace. The coarse-to-fine search incurs the same computational cost.

The equivalent of the Karhunen-Loéve expansion for a weighted L^2 norm is performed as follows. The $N \times N$ weighted covariance matrix $\mathbf{C} = (C_{nm,rs})$ is computed using:

$$C_{nm,rs} = \mathrm{E}_{\mathbf{q}} \left[\overline{G}(n, m; \mathbf{q}) \cdot w(r, s) \cdot \overline{G}(r, s; \mathbf{q}) \right] \tag{5.9}$$

where $\mathrm{E}[\cdot]$ is the expectation operator and $\overline{G}(n, m; \mathbf{q}) = \overline{F}(n, m; \mathbf{q}) - \mathrm{E}_{\mathbf{q}}[\overline{F}(n, m; \mathbf{q})]$. Then, the d eigenvectors of $\mathbf{C}$ with the largest eigenvalues are used as the basis $\{\mathbf{e}_j \mid j = 1, 2, \ldots, d\}$ for the low dimensional subspace.

5.3 Optimality Criteria

A number of optimality criteria for feature detectors have been studied in the literature. The most well known are the three criteria proposed by Canny in [20]:

Good Detection: "There should be a low probability of failing to mark real edge points (ie. false negatives) and low probability of falsely marking non-edge points (ie. false positives)" [12]. Canny argued that both of these criteria are strongly correlated with the signal to noise ratio (SNR). Hence, he used the SNR as his first optimality criterion.

Good Localization: "The points marked by the operator should be as close as possible to the center of the true edge" [12]. Canny derived an estimate of the root mean squared (RMS) displacement of an ideal edge perturbed with independently and identically distributed Gaussian noise, and used it as his second optimality criterion.

Few Multiple Responses: For an ideal detector, there should be "only one response to a single edge" [12]. Canny derived an estimate for the expected distance between adjacent edges and used it as his third optimality criterion.

Besides Canny, a number of other authors have studied his three criteria, and variants thereof, combining them in various ways. See, for example, [12], [113], [28], [29], [109], and [97]. Other optimality criteria that have been considered include the energy in the vicinity of the edge [111] [66] [67] [68] and the Discriminative Signal to Noise Ratio [104]. See Section 2.3 for more discussion of optimal edge detection.

All of the above optimality criteria were developed for feature detectors based on filtering rather than for the model matching detectors considered in this thesis. In the remainder of this section, I propose three optimality criteria that are more appropriate for model matching detectors. In Section 5.3.1, I consider "Good Detection" and derive estimates of the probability of a false positive and the probability of a false negative. I use the term feature detection robustness to refer to these two optimality criteria together. In Section 5.3.2, I consider parameter estimation accuracy, a generalization of the "Good Localization" criterion to include all of the feature parameters. In Section 5.3.3, I discuss how these three optimality criteria can be combined. I do not consider the "Few Multiple Responses" criterion in this paper since its introduction in [20] was for technical reasons, rather than

because it is a fundamental element of feature detection performance.

5.3.1 Feature Detection Robustness

Feature detection is not robust, both when the detector misses features (false negatives), and when the detector mistakenly detects features that are not present (false positives). Evaluating the robustness of a feature detector is known to be difficult, amongst other reasons because of the lack of a characteristic model of a "non-feature" [80] [103]. Canny avoided this problem when defining his optimality criteria by using the signal to noise ratio, which he claimed to be correlated with robustness. Both Nalwa and Binford [80] and Ramesh and Haralick [103] simply used a constant intensity vector as their characteristic non-feature.

To derive my optimality criteria for robustness, I assume that, in addition to a parametric model of the feature $F^c(x, y; \mathbf{q}^c)$, there is also a parametric model of the non-feature $NF^c(x, y; \mathbf{nq}^c)$. A suitable non-feature for the step edge might be a constant gradient slope with three parameters:

$$NF^c_{SE}(x, y; A, g, \theta) \;=\; A + g \cdot (y \cdot \cos\theta - x \cdot \sin\theta) \tag{5.10}$$

where A is the intensity value of the center pixel, g is the gradient of the slope, and θ is the angle which the direction of steepest ascent makes with the positive y-axis. The discretized non-feature model $NF_{SE}(n, m; \mathbf{nq})$ is then defined in the same way that the discretized feature model was in Equation (3.3):

$$NF_{SE}(n, m; \mathbf{nq}) \;=\; NF^c_{SE}(x, y; A, g, \theta) * g(x, y; \sigma) * a(x, y)|_{x=n, y=m}. \tag{5.11}$$

Suppose that there is an ideal feature instance with parameters $\mathbf{q}$ in the scene. If this feature is projected onto a window surrounding the pixel (a, b) in the

image I, the vector of image data $\mathbf{I}(a,b) = (I(a+n, b+m))$ should differ from $\mathbf{F}(\mathbf{q}) = (F(n,m;\mathbf{q}))$ only because of the noise introduced by the imaging process. If the noise is modeled by the additive random vector $\boldsymbol{\eta} = (\eta(n,m))$, it follows that:

$$I(a+n, b+m) \;=\; F(n,m;\mathbf{q}) + \eta(n,m). \tag{5.12}$$

Assume that pixel (a,b) is mistakenly classified as a non-feature if $\mathbf{I}(a,b)$ is closer to the non-feature manifold than it is to the feature manifold, ie. if:

$$\min_{\mathbf{nq}} \| \mathbf{I}(a,b) - \mathbf{NF}(\mathbf{nq}) \|_w^2 \;<\; \min_{\mathbf{q}} \| \mathbf{I}(a,b) - \mathbf{F}(\mathbf{q}) \|_w^2. \tag{5.13}$$

I therefore estimate the probability of a false negative for a feature in the scene with parameters $\mathbf{q}$ to be:

$$\mathrm{P}_{\boldsymbol{\eta}}(\mathrm{FN} \mid \mathbf{q}) \;=\; \mathrm{P}_{\boldsymbol{\eta}}\left[\min_{\mathbf{nq'}} \|\mathbf{F}(\mathbf{q}) + \boldsymbol{\eta} - \mathbf{NF}(\mathbf{nq'})\|_w^2 \;<\; \min_{\mathbf{q'}} \|\mathbf{F}(\mathbf{q}) + \boldsymbol{\eta} - \mathbf{F}(\mathbf{q'})\|_w^2\right]. \tag{5.14}$$

Now, $\mathrm{P}_{\boldsymbol{\eta}}(\mathrm{FN} \mid \mathbf{q})$ is still a function of the parameters $\mathbf{q}$. Hence, I allow the designer of the feature detector to supply a probability distribution $\mathrm{P}(\mathbf{q})$ to specify the *a priori* likelihood of a feature with parameters $\mathbf{q}$ appearing in the scene. Given $\boldsymbol{\eta}$ and $\mathrm{P}(\mathbf{q})$, I propose the following as my optimality criterion for false negatives:

$$\mathbf{FN} \;=\; \int \mathrm{P}(\mathbf{q})\, \mathrm{P}_{\boldsymbol{\eta}}(\mathrm{FN} \mid \mathbf{q})\, \mathrm{d}\mathbf{q}. \tag{5.15}$$

Similarly, by reversing the roles of the feature and the non-feature, I propose the following as my optimality criterion for false positives:

$$\mathbf{FP} \;=\; \int \mathrm{P}(\mathbf{nq})\, \mathrm{P}_{\boldsymbol{\eta}}(\mathrm{FP} \mid \mathbf{nq})\, \mathrm{d}\mathbf{nq} \tag{5.16}$$

where:

$$\mathrm{P}_{\boldsymbol{\eta}}(\mathrm{FP} \,|\, \mathbf{nq}) \;=\; \mathrm{P}_{\boldsymbol{\eta}}\left[\min_{\mathbf{q'}} \|\mathbf{F}(\mathbf{nq}) + \boldsymbol{\eta} - \mathbf{F}(\mathbf{q'})\|_w^2 \;<\; \min_{\mathbf{nq'}} \|\mathbf{F}(\mathbf{nq}) + \boldsymbol{\eta} - \mathbf{NF}(\mathbf{nq'})\|_w^2\right] \tag{5.17}$$

and P($\mathbf{nq}$) is a probability distribution that specifies the *a priori* likelihood of the occurrence of a non-feature in the scene with parameters $\mathbf{nq}$.

5.3.2 Parameter Estimation Accuracy

Parameter estimation is inaccurate when the closest point on the manifold is perturbed from its correct position. Again, suppose that there is an ideal feature instance with parameters $\mathbf{q}$ in the scene. If this feature is projected onto a window surrounding pixel (a, b) in image I, the vector of image data $\mathbf{I}(a, b)$ will differ from $\mathbf{F}(\mathbf{q})$ because of the noise introduced in the imaging process. If the imaging noise is again modeled with the additive random vector $\boldsymbol{\eta} = (\eta(n, m))$, it follows that $I(a+n, b+m) = F(n, m; \mathbf{q}) + \eta(n, m)$. Then, if the closest point on the manifold to $\mathbf{I}(a, b)$ is $\mathbf{F}(\mathbf{q} + \Delta\mathbf{q})$, the error in estimating the parameters will be:

$$\Delta\mathbf{q} = \Delta\mathbf{q}(\boldsymbol{\eta}) = \arg\min_{\mathbf{q}'} \|\mathbf{F}(\mathbf{q}) + \boldsymbol{\eta} - \mathbf{F}(\mathbf{q}')\|_w^2 - \mathbf{q}. \tag{5.18}$$

Here, the error in estimating parameter q_i is $\Delta q_i = \Delta q_i(\boldsymbol{\eta})$. To obtain a measure that is independent of the noise added, but not the distribution of the noise, I average over the noise distribution by taking the root mean squared error:

$$\mathrm{RMS}_{\boldsymbol{\eta}}[\Delta q_i(\boldsymbol{\eta})] = \sqrt{\mathrm{E}_{\boldsymbol{\eta}}[(\Delta q_i(\boldsymbol{\eta}))^2]}. \tag{5.19}$$

Again, $\mathrm{RMS}_{\boldsymbol{\eta}}[\Delta q_i(\boldsymbol{\eta})]$ is still a function of the parameters $\mathbf{q}$. Hence, I propose the following as my optimality criterion for the estimation of parameter q_i:

$$\mathbf{PE}_{q_i} = \left[\int \mathrm{P}(\mathbf{q}) \left(\mathrm{RMS}_{\boldsymbol{\eta}}[\Delta q_i(\boldsymbol{\eta})]\right)^2 \mathrm{d}\mathbf{q}\right]^{1/2} \tag{5.20}$$

where P($\mathbf{q}$) is the designer supplied probability distribution specifying the *a priori* likelihood of a feature appearing in the scene with parameters $\mathbf{q}$.

5.3.3 Combinations of Optimality Criteria

In addition to the probability of a false positive **FP**, the probability of a false negative **FN**, and the parameter estimation accuracy $\mathbf{PE}_{q_i}$, it is straightforward to combine these three optimality criteria to form others. One example is the feature detection robustness:

$$\mathbf{FDR} \;=\; \sqrt{\mathbf{FP} \times \mathbf{FN}}, \tag{5.21}$$

another example is the combined parameter estimation accuracy:

$$\mathbf{CPE} \;=\; \left(\prod_{i=1}^{|\mathbf{q}|} \mathbf{PE}_{q_i} \right)^{1/|\mathbf{q}|}, \tag{5.22}$$

and a final example is the overall feature detection performance:

$$\mathbf{ODP} \;=\; \mathbf{FDR} \times \mathbf{CPE}. \tag{5.23}$$

Other combinations are possible and could also be optimized using the numerical algorithm that I will propose in Section 5.4.2. One example is $\mathbf{FDR}^p \times \mathbf{CPE}^{1-p}$, where $p \in [0, 1]$ is a number that can be used to adjust the relative importance of feature detection robustness and parameter estimation accuracy. The value of p can be selected based upon the requirements of a specific application.

5.4 Optimization of the Optimality Criteria

I now describe how the optimality criteria just proposed can actually be optimized. I begin in Section 5.4.1 by discussing an analytical solution for the parameter estimation accuracy criterion under the approximating assumption that the feature manifolds are linear. In Section 5.4.2, I describe a numerical algorithm that can be

used to optimize the other criteria. Finally, in Section 5.4.3, I present the results obtained applying this algorithm for three of the features studied in Chapter 3.

5.4.1 Analysis for Linear Manifolds

Analyzing the optimality criteria proposed in Section 5.3 is, in general, very difficult because the feature manifolds are nonlinear. Moreover, the feature detection robustness criteria are nonlinear functions of the manifolds due to the $\min(\cdot,\cdot)$ functions that they contain. It turns out, however, that it is possible to derive optimal weighting functions for the parameter estimation accuracy criteria $\mathbf{PE}_{q_i}$ under the assumption that the feature manifolds can be approximated by their first order Taylor expansions. This assumption is particularly reasonable when the noise level is not too high. For linear manifolds, finding the closest point on the manifold is simply a weighted least squares problem. So, finding the optimal weighting function corresponds to selecting the weighting function that gives the best linear unbiased estimate of the solution to a weighted least squares problem [63] [99]. The answer to this problem was found by Aitken in [3]. The optimal weighting function is:

$$w(n,m) \;=\; \frac{1}{\mathrm{Var}_{\boldsymbol{\eta}}[\eta(n,m)]} \tag{5.24}$$

where $\mathrm{Var}_{\boldsymbol{\eta}}[\eta(n,m)] = \mathrm{E}_{\boldsymbol{\eta}}[\eta^2(n,m)] - \mathrm{E}_{\boldsymbol{\eta}}[\eta(n,m)]^2$ is the variance of the noise $\boldsymbol{\eta}$ in pixel (n,m). This result assumes that the noise in each pixel in independent. It is possible to generalize this result to the non-independent case, but it requires that the weighting function $w(n,m)$ be replaced with a positive definite quadratic form. The optimal choice for the positive definite quadratic form is the inverse of the covariance matrix of the noise [99]. See Section 5.5.1 for a discussion of the use of positive definite quadratic forms in feature detection. Finally, note that

Equation (5.24) implies that under the linear manifold assumption, if the noise $\boldsymbol{\eta}$ is independently and identically distributed across the pixels, then the Euclidean L^2 norm is optimal for parameter estimation accuracy.

5.4.2 Numerical Optimization

As mentioned above, analytically optimizing my optimality criteria is very difficult. The use of realistic multi-parameter feature models, combined with a nonlinear sensing model, leads to very complicated expressions for the discretized feature models $F(n, m; \mathbf{q})$. After taking into account both parameter normalization and dimension reduction, the optimality criteria become extremely complex. Even Canny resorted to a numerical algorithm to optimize his relatively simple optimality criterion for arbitrary features [20]. Here, I follow the same approach.

If there are N pixels in the discrete feature window S, the optimization is $N-1$ dimensional rather than N dimensional because the optimality criteria are unchanged if the weighting function is multiplied by a positive constant. Since N is typically in the range 25–100, to make the problem more tractable I assume that the weighting function is rotationally symmetric; i.e. it is only a function of the distance from the center of the window. Such an assumption is particularly reasonable if there is no *a priori* knowledge about the likely orientation of features in the image. I also assume that the weighting function can be approximated by a low order polynomial in the distance from the center of the window. I found that a degree 5 polynomial was sufficient, and that the best way to parameterize this polynomial is in terms of the value of the polynomial at equally spaced distances from the center of the feature window. The radius of the feature window in my

experiments was 4 pixels. Hence, I parameterized the polynomial in terms of its value at distance $0, 1, 2, 3, 4$ pixels from the center of the window. The result is a 5 dimensional optimization which proved to be tractable, even for the most complex optimality criteria. For larger window sizes, a 4–5 degree polynomial should still be sufficient, so a larger spacing between the parameterization points can be used.

I decided to use an optimization algorithm that does not evaluate the partial derivatives of the optimality criterion. The partial derivatives are so complicated that probably the only way to estimate them would be numerically anyway. I implemented Powell's Method directly from Chapter 10.5 of [101]. Powell's Method works by repeatedly performing 1-dimensional optimizations though the current best estimate of the minimum. Each 1-dimensional optimization is performed using Brent's Method, which iteratively samples the optimality criterion 3 times, performs a parabolic fit, and then replaces one of the sample points with the minimum of the parabola [101]. After each application of Brent's Method, the best estimate of the minimum is updated and a new direction is chosen. The key decision in Powell's Method is how to choose and update the set of directions in which the 1-dimensional optimizations are performed. I used the technique suggested in [101] of starting with the basis vectors and "discarding the direction of largest decrease."

As noted several times, my criteria are very complex; even evaluating them can be time-consuming. Using Monte Carlo integration to perform the averaging over the parameter space and the noise distribution, it takes approximately 1–2 minutes to evaluate any of the optimality criteria at a single sample point. Typically, Powell's Method samples the optimality criteria a few hundred times resulting in an overall running time of a few hours, normally around 12 hours. This length

of time is reasonable because it is offline computation performed during the design of a feature detector. In fact, it is exceedingly short compared to the usual length of time that it takes to design a feature detector.

5.4.3 Numerical Results

I applied Powell's algorithm to three of the features considered in Chapter 3, namely, the step edge, the corner, and the symmetric line. Some of the optimality criteria require non-feature models as well as feature models. I used the constant gradient slope defined in Equation (5.10) as the non-feature for the step edge and the step edge itself as the non-feature for both the corner and the symmetric line. The optimality criteria also require probability distributions to be provided for the feature parameters and the noise. I assumed that P($\mathbf{q}$) and P($\mathbf{nq}$) are both uniformly distributed. The noise $\boldsymbol{\eta}$ was chosen to be Gaussian and independently distributed across the pixels, with signal to noise ratio 2.0, using the definition of signal to noise ratio for arbitrary features proposed in Section 4.1.1.

The results of applying Powell's algorithm to the three features are displayed in Tables 5.1–5.3. In each table, I present the numerical value of the optimality criteria both for the Euclidean L^2 norm and for the optimal L^2 norm. There are a number of points that should be noted:

- The parameter estimation accuracy results for the non-normalized parameters (ie. all parameters except A and B) are in close agreement with the analysis in Section 5.4.1. There I showed that the Euclidean L^2 norm would be optimal if the manifolds were linear. Locally the manifolds are approximately linear. Therefore it is no surprise that the optimal weighted L^2 norm

Table 5.1: Computed values of the optimality criteria for the step edge.

	$\mathbf{PE}_A$	$\mathbf{PE}_B$	$\mathbf{PE}_\theta$	$\mathbf{PE}_\rho$	$\mathbf{PE}_\sigma$	**CPE**	**FDR**	**ODP**
Euclidean	0.261	0.438	6.982	0.441	0.509	0.709	0.390	0.825
Optimal	0.234	0.372	6.957	0.438	0.507	0.706	0.376	0.822

Table 5.2: Computed values of the optimality criteria for the corner.

	$\mathbf{PE}_A$	$\mathbf{PE}_B$	$\mathbf{PE}_{\theta_1}$	$\mathbf{PE}_{\theta_2}$	$\mathbf{PE}_\sigma$	**CPE**	**FDR**	**ODP**
Euclidean	0.120	0.519	10.448	17.813	0.288	1.274	0.234	0.897
Optimal	0.112	0.459	10.398	17.738	0.287	1.251	0.221	0.854

Table 5.3: Computed values of the optimality criteria for the symmetric line.

	$\mathbf{PE}_A$	$\mathbf{PE}_B$	$\mathbf{PE}_\theta$	$\mathbf{PE}_\rho$	$\mathbf{PE}_\sigma$	$\mathbf{PE}_w$	**CPE**	**FDR**	**ODP**
Euclidean	0.276	0.690	4.510	0.216	0.355	0.992	0.635	0.00358	0.00681
Optimal	0.223	0.654	4.500	0.215	0.354	0.989	0.632	0.00238	0.00475

is only marginally better than the Euclidean L^2 norm. On the other hand, an improvement of 10-20% is possible for the normalized parameters A and B.

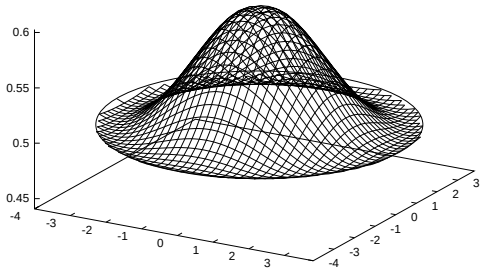
- For feature detection robustness some improvement is possible over the Euclidean L^2 norm. For the step edge and corner it is only around 5%, but for the symmetric line an improvement of around 35% is possible. For the overall detection performance, almost no improvement is possible for the step edge, a 5% improvement is possible for the corner, and a 30% improvement is possible for the symmetric line.

- Finally, note that estimating the step parameter B is harder than estimating the base parameter A. This effect is particularly noticeable for the line and the corner where the upper intensity region occupies a much smaller part of the feature window. In a similar vein, note that both the angle θ and the

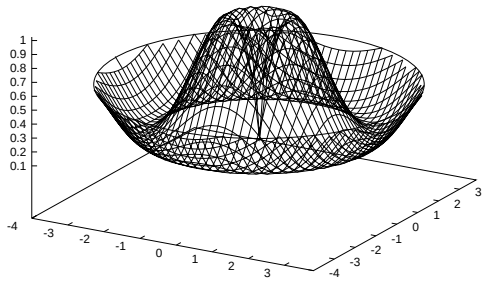
subpixel localization ρ of the line are easier to estimate than the corresponding parameters for the step edge. The intuitive reason for this fact is that the line can be regarded as two parallel step edges separated by the width of the line w. The estimates of θ and ρ for the line can be thought of as averages of the estimates for the two step edges. Therefore they are less noise sensitive.

In Figure 5.1, I display 3–D plots of several of the optimal weighting functions. The optimal weighting function for the parameter estimation accuracy of the sub-pixel localization ρ of the step edge is presented in Figure 5.1(a). The overall form of this optimal weighting function is as expected. Intuitively, the center-most pixels are the most important when estimating sub-pixel localization [20]. Hence, it is to be expected that they should be given more weight. The results in Figure 5.1(c) for the combined parameter estimation accuracy of the corner are also in agreement with intuition. Here, the central pixels do not change much as the parameters of the corner vary. This effect is particularly true for the two angles θ_1 and θ_2. So, one would not expect the central pixels to be as heavily weighted as those on the periphery of the window. The optimal weighting function for the combined parameter estimation accuracy of the symmetric line, displayed in Figure 5.1(e), is neither a monotonically increasing nor decreasing function of the distance from the center of the window. The intuitive reason for this effect is that the central pixels need to be heavily weighted to estimate the sub-pixel localization and the width of the line, whereas the peripheral pixels also need to be heavily weighted to estimate the orientation and the intensity parameters.

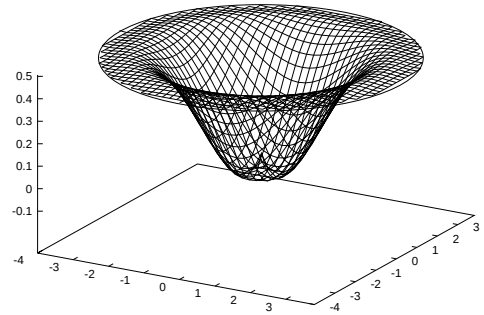
In Figures 5.1(b), (d), and (f), I display optimal weighting functions for feature detection robustness and overall detection performance. Providing intuitive



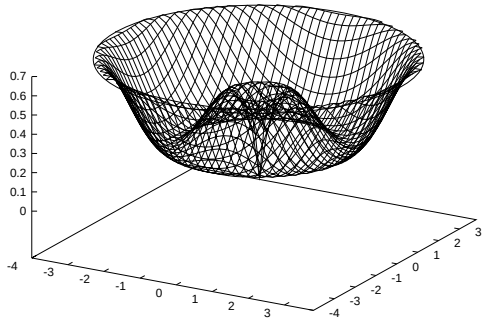
(a) $\mathbf{PE}_\rho$ for the Step Edge



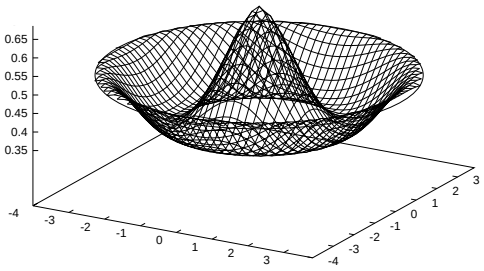
(b) **FDR** for the Step Edge



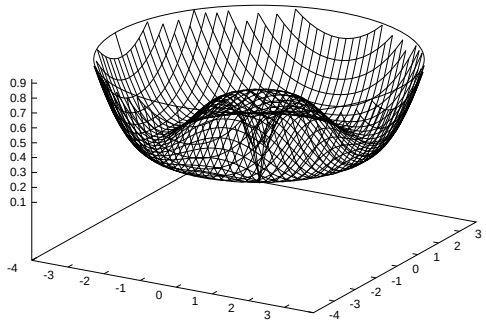
(c) **CPE** for the Corner



(d) **ODP** for the Corner



(e) **CPE** for the Symmetric Line



(f) **FDR** for the Symmetric Line

Figure 5.1: A selection of optimal weighting functions computed by applying Powell's algorithm to the optimality criteria proposed in Section 5.3.

explanations for the shape of these optimal weighting functions is more difficult than for parameter estimation accuracy. At first glance, the fact that these weighting function are not monotonic functions of the distance from the center of the window may appear strange. The explanation is straightforward however. The robustness criteria can be thought of as trying to maximize the discriminability of the feature and its corresponding non-feature. The difference between each feature and the "nearest" non-feature is usually not a monotonic function of the distance from the center of the window. Hence, it is no surprise that the optimal weighting functions are also not monotonic functions of the distance from the center of the window.

5.5 Discussion

In this chapter, I investigated the choice of the matching function for the model matching detector described in Chapter 3. In particular, I restricted attention to weighted L^2 norms and presented a general framework for the selection of the weighting function. I proposed optimality criteria for the three key aspects of feature detection performance: the probability of a false positive, the probability of a false negative, and the parameter estimation accuracy. I also showed how to combine these three optimality criteria to form ones more appropriate for specific applications. Finally, I derived an approximate expression for the optimal weighting function for parameter estimation accuracy, and proposed a numerical algorithm for the optimization of the other criteria.

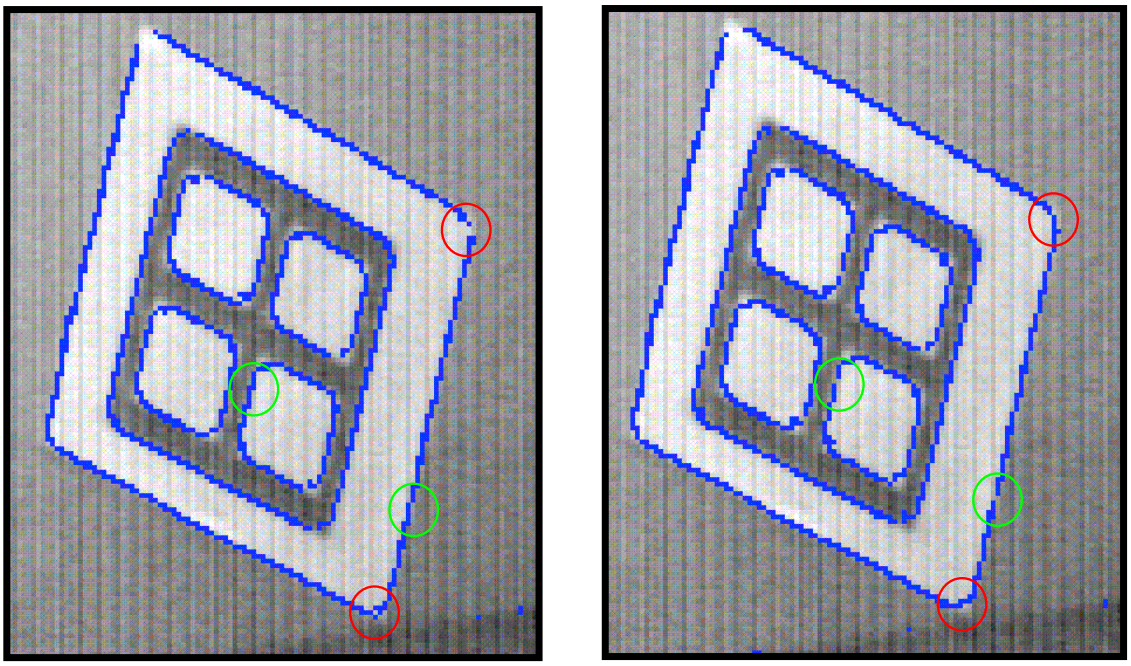
It turns out that the optimal weighted L^2 norms are only slight improvements over the Euclidean L^2 norm. The improvement in the optimality criteria was typically around 5%, and was always less than about 30%. So, the Euclidean L^2 norm is

close to optimal, both for parameter estimation accuracy and for feature detection robustness. This conclusion, however, is highly dependent upon the assumptions made about the noise; ie. that it is independently and identically distributed across the pixels. This noise model is undoubtedly simplistic, but it is unclear how to derive a noise model that is more realistic of the noise present in real images.

A related difficulty is how to evaluate the optimal weighting functions. One possibility would be to conduct statistical tests on synthetic data, similar to those performed in Section 4.1. I decided not to conduct these tests, since little useful information would be obtained: the tests are just too closely related to the optimality criteria. Another possibility is to look at experimental results on real images. As is demonstrated by Figure 5.2, doing so can be difficult. In this figure there are a number of points where the Euclidean L^2 norm appears to be superior to the optimal weighted L^2 norm and a number of points where it does worse. It is hard to conclude which detector performs better by simply looking at the edge map. I surveyed the literature (see Section 2.5), but could not find any appropriate ways of evaluating the optimal weighting functions. In the next chapter, I propose a collection of benchmarks for non-subjectively evaluating edge detectors. Beforehand, I briefly discuss two additional issues arising from the work presented in this chapter.

5.5.1 Other Classes of Matching Functions

An immediate question that arises from this chapter is whether it is possible to generalize the class of matching functions considered. The two major reasons for restricting attention to weighted L^2 norms were: (1) weighted L^2 norms have been used in a number of model matching detectors (see Section 2.1.3), and (2) the



(a) (b)

Figure 5.2: Detected step edges when using (a) the Euclidean L^2 norm and (b) the optimal weighted L^2 norm for overall detection performance. The results shown are for the lower right hand part of the image "INRIA Office" displayed in Figure 4.16. The results demonstrate how difficult it is to compare weighting functions on real image data. The red circles highlight points where the optimal weighted L^2 norm does better, and the green circles highlight points where the Euclidean L^2 norm does better.

feature detector described in Chapter 3 can be easily extended to use an arbitrarily weighted L^2 norm. A natural extension of this chapter is to consider the class of positive definite quadratic forms. It is straightforward to extend the detector described in Chapter 3 to use a quadratic form, however the number of parameters in the optimization will grow from N, the number of pixels, to roughly $N^2/2$. Another issue that would need to be addressed is the noise model used in the optimality criteria. To take full advantage of the additional cross terms in the quadratic form, the noise model would have to be generalized to model correlation

of the noise across the pixels.

5.5.2 Relationship with Canny

I now discuss the relationship between the approach described in this chapter and the optimal filtering approach to edge detection best exemplified by [20]. Taking Equation (5.4) and replacing unnormalized vectors with their normalized counterparts, it can be seen that the model matching approach to feature detection is based upon finding the parameters $\mathbf{q}$ that minimize:

$$\sum_{(n,m)\in S} w(n,m) \cdot \left[\overline{F}(n,m;\mathbf{q}) - \overline{I}(a+n,b+m)\right]^2. \tag{5.25}$$

Since the vectors are normalized, this expression simplifies to:

$$2 \; - \sum_{(n,m)\in S} w(n,m) \cdot \overline{F}(n,m;\mathbf{q}) \cdot \overline{I}(a+n,b+m). \tag{5.26}$$

So, model matching feature detectors can be thought of as choosing $\mathbf{q}$ to maximize:

$$\sum_{(n,m)\in S} w(n,m) \cdot \overline{F}(n,m;\mathbf{q}) \cdot \overline{I}(a+n,b+m). \tag{5.27}$$

In this chapter, I studied the selection of the weighting function $w(n,m)$ to optimize feature detection performance.

On the other hand, Canny [20] studied the selection of the filter $f(x)$ that optimizes the performance of a 1-D step edge detector that declares edges at local maxima of:

$$\int_{-W}^{+W} I(a+x) \cdot f(-x)\, dx \tag{5.28}$$

where $I(x)$ is the continuous 1-D input image and W is the width of the 1-D feature window. If this continuous 1-D setting is translated into the discrete 2-D setting

used in this thesis, the expression in Equation (5.28) becomes:

$$\sum_{(n,m)\in S} f_\theta(n,m) \cdot I(a+n, b+m) \tag{5.29}$$

where $f_\theta(n,m)$ is $f(x)$ rotated by an angle θ and discretized. Comparing Equations (5.27) and (5.29), the following similarities and differences become apparent.

The major similarity is the form of the expression that is maximized. In both cases, it is a discrete convolution. In model matching the data is convolved with $w_{nm} \cdot \overline{F}(n,m;\mathbf{q})$ and in optimal filtering it is convolved with $f_\theta(n,m)$. A second similarity is the relationship between the selection of an optimal weighting function and the selection of a optimal convolution filter. Interestingly, selecting a uniform weighting function, which is almost optimal, corresponds to choosing a matched filter, which is optimal for the first two components of Canny's criterion.

The major difference is the domain over which the maximum is taken. In model matching it is over all of the parameters, whereas in optimal filtering it is just over the location of the edge. So, if there is a sub-pixel localization parameter, model matching implicitly performs the same local non-maximum suppression that the Canny detector does. If there is a rotation parameter, model matching also optimizes over it. There is no equivalent in [20] because the formulation is entirely 1-D. Note, however, that steerable filters provide a way of optimizing over the rotation parameter [37]. A final difference is the normalization of the image data. Like most other model matching detectors, the algorithm in Chapter 3 uses normalized data, whereas optimal filtering approaches typically use the raw pixel values.

Chapter 6

Global Measures of Coherence for Edge Detector Evaluation

6.1 Introduction

Assessing the performance of an edge detector is an important, yet difficult, task. Of the four evaluation methodologies described in Section 2.5.2, the only one that has actually been used by a significant number of authors is subjective human evaluation. In particular, [47] contains a survey of recent articles on edge detection. None of the twenty-one papers considered used any other method of performance evaluation. Subjective human comparison consists of simply applying the detectors to a small number of images and then displaying the output edge maps for evaluation by a human. Although the limitations of such an approach are widely acknowledged, none of the the other techniques that have been proposed in the literature have either been universally accepted or frequently used.

One of the major reasons that authors have not used any of the other eval-

uation techniques is the difficulty of doing so. Applying any of the other methods typically involves extensive effort by the author. Another major obstacle is obtaining the results for other detectors, without which the results are often meaningless. So, a researcher who wishes to evaluate a new detector must, not only implement and conduct the tests on their own detector, but implement several other detectors and conduct the tests on those detectors as well. For a performance evaluation technique to be widely used, it must be both straightforward to use and the results for a large number of other detectors must be made available.

It seems exceedingly unlikely that there is a single, simple method of "fairly" evaluating an edge detector. Even in a field as mature as computer architecture, there is no universally agreed upon way of measuring performance [93]. Instead, the usual approach is to apply a large number of simple benchmark tests, each of which is designed to be typical of a range applications. The overall performance on the benchmarks is then used to compare the architectures. The results on any specific benchmark are not assumed to generalize to the other benchmarks, or to applications that the benchmark is unrepresentative of. However, if enough benchmarks are applied, the overall results are accepted as indicative of the general performance one could expect on a novel application.

In this chapter, I advocate a similar approach for the evaluation of edge detectors. In particular, I propose a set of benchmarks designed specifically for applications requiring precise sub-pixel localization and orientation estimation. Good examples of such applications include the Hough transform, stereo matching, structure from motion, and the computation of projective invariants. More generally, these benchmarks provide useful information for any application in which vision

is used in a quantitative manner, or as a measurement tool. The benchmarks are less representative of more qualitative tasks such as object recognition, segmentation, and edge grouping. Other evaluation techniques have been proposed that are somewhat more suited to such tasks, including [32], [47], [121], and [57]. My benchmarks are meant to supplement, not replace, these existing techniques. Note that several new evaluation techniques for quantitative applications, such as structure from motion [112] and industrial inspection [114], have been proposed recently.

Each of my benchmarks is based upon a constraint on the edges in the scene, for example, that they are colinear. To capture the benchmark images, I carefully create a scene for which the constraint holds. An example scene where all the edges are colinear can be constructed by placing a convex polygonal object with constant albedo lambertian faces in front of a perfectly black background. If an image of such a scene is cropped so that only one surface normal discontinuity is visible, all of the edges will be colinear. Each benchmark consists of a large number of images of the scene captured by varying the viewing and illumination directions. Because the images are captured in a controlled environment, it is simple to sample the set of images very widely by varying the viewing pose, the illumination conditions, and the reflectance properties of the objects. Although it would also be possible to sample the space of images just as widely using synthetic images, these images would not take into acccount the sources of noise that corrupt real images.

After applying the edge detector, I estimate the degree to which the constraint holds in the output edge map and use it as the measure of edge detection performance. For the colinearity constraint just described, each detected edge can be used to estimate the projective geometric representation of the line that all of the

detected edges should lie on. I estimate the variances of the three components of the line from the positions and orientations of the detected edges. I then average these variances in a simple way and use the result as the performance measure. Since such performance measures are functions of how well the detected edges satisfy the scene constraint, I refer to them as global measures of edge map coherence.

Computing the global measures of coherence from the output edge maps is relatively straightforward, although there are a few caveats that are discussed in Section 6.3. I have written the required code to compute four different global measures of coherence, together with several variants of them. All of the programs and benchmark images are available online on the World Wide Web. To use any of the benchmarks is very easy. Quite simply, all a user has to do is download the code and associated images, modify their edge detector to output the detected edges in the correct format, and then run a script with the name of their edge detector. In all, at most 2-3 hours is required to conduct the tests for a new detector. All of the results for the detectors that I tested are also available online to allow a user to immediately compare their results with a large number of other detectors.

I begin the remainder of this chapter in Section 6.2 by presenting the four global measures of coherence in detail. For each one, I first describe the scene constraint that it is based upon, and then define the measure itself. In Section 6.3, I describe the details of how the edge map is actually sampled to compute the global measures of coherence. In Section 6.4, I present experimental results obtained by applying the benchmarks to four well known edge detectors, as well as the step edge detector developed in Chapter 3. Next, in Section 6.5, I show how global measures of coherence can be used to investigate how quickly edge detector performance

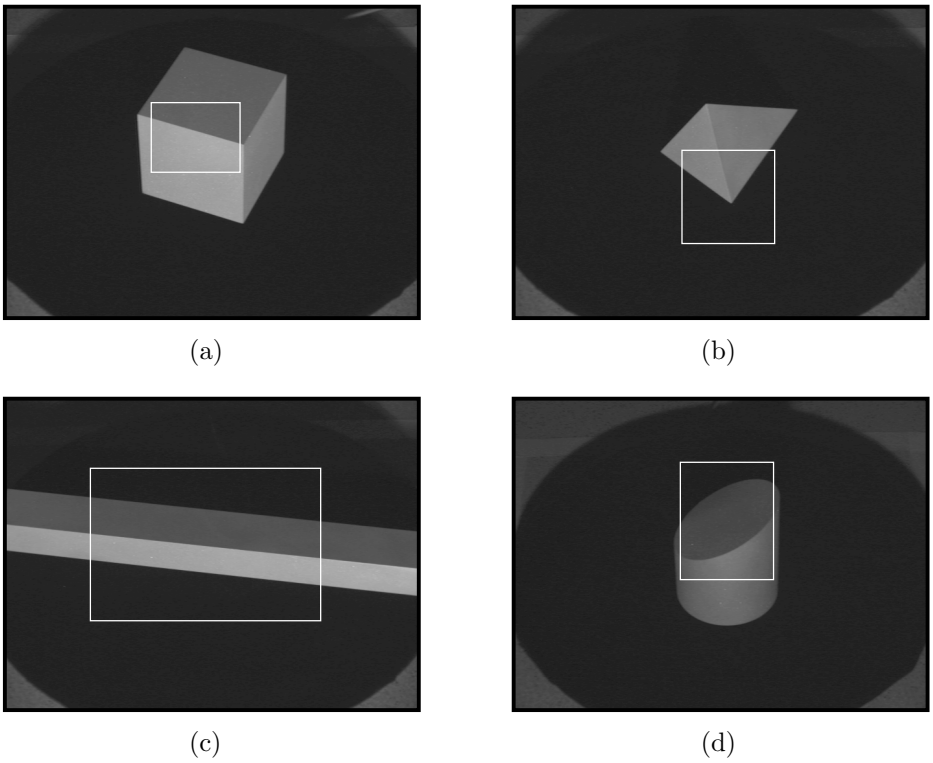


Figure 6.1: Cropped regions exhibiting the four constraints: (a) all of the edges are colinear, (b) all of the edges intersect at a single point, (c) all of the edges are parallel in the scene, and (d) all of the edges lie on an ellipse.

degrades as various camera parameters are varied. Finally, in Section 6.6, I discuss the strengths and weaknesses of global measures of coherence.

6.2 Global Measures of Coherence

In this section, I introduce my four global measures of coherence. Each measure is derived from a constraint on the edges in the scene. These four constraints are illustrated in the cropped regions of Figure 6.1: in (a) all of the edges are colinear, in (b) all of the edges intersect at a single point, in (c) all of the edges are parallel in the scene, and in (d) all of the edges lie on an ellipse.

Given a cropped image for which one of these constraints holds, the corresponding global measure of coherence is computed as follows. First, the edge detector is applied. Suppose the output is a set of edges:

$$E \;=\; \{e_i = (x_i, y_i, \theta_i) \mid i = 1, \ldots, n\}, \tag{6.1}$$

where n is the number of edges. Suppose the i^{th} edge $e_i = (x_i, y_i, \theta_i)$ passes through the image point (x_i, y_i), and the normal to this edge make an angle θ_i with the positive y–axis, as is shown in Figure 3.1(b). Then, given an appropriate number of edges from E, certain quantities are estimated that should be constant if the constraint held exactly. For example, given two edges in the cropped region of Figure 6.1(b) that are not approximately parallel, the coordinates of their point of intersection are estimated. If the edge map E were perfect, the point of intersection would be the same for all such pairs of edges. Each global measure of coherence is defined to be a monotonically increasing function of the variances of the quantities that would be constant if the edge map were perfect. The global measures of coherence are computed by sampling E to estimate this function of the variances.

In the rest of this section, I describe each of the four constraints in turn. For each one, I define the corresponding global measure of coherence. Two of the measures also have variants that do not require knowledge of the orientations of the edges θ_i. In these cases, I also provide definitions for the alternative measures. In the following section, I discuss the exact details of how the edge map E is actually sampled to estimate the global measures of coherence.

6.2.1 All Edges are Colinear 1: Known θ_i

The first constraint is that all of the edges are colinear. Such a scene can be constructed by placing a convex polygonal object with constant albedo lambertian faces in front of a perfectly black background. If an image of such a scene is cropped so that only one depth or surface normal discontinuity is visible, all of the edges will be colinear. See Figure 6.1(a) for an example image of such a scene. The fact that colinear edges in the scene are projected onto colinear edges in the image relies upon the implicit assumption that the camera is linear. In Section 6.3.1, I describe how simple nonlinear effects such as radial distortion can be compensated for.

Given just one of the detected edges $e_i = (x_i, y_i, \theta_i) \in E$, it is possible to estimate the line that of all the edges lie on. In the projective geometric notation of [35], the vector representation of this line is:

$$\begin{aligned} \mathbf{L}_i \equiv \left(l_i^1, l_i^2, l_i^3\right)^{\mathrm{T}} &= (x_i, y_i, 1)^{\mathrm{T}} \wedge (x_i + \cos\theta_i, y_i + \sin\theta_i, 1)^{\mathrm{T}} \\ &= (-\sin\theta_i, \cos\theta_i, x_i \sin\theta_i - y_i \cos\theta_i)^{\mathrm{T}} . \end{aligned} \tag{6.2}$$

A minor difficulty that needs to be addressed at this point is that equality is only defined up to a constant multiplicative factor in projective spaces [35]. There are two aspects to this problem:

Sign of $\mathbf{L}_i$: Adding 180° to θ_i does not change the line, but reverses the sign of $\mathbf{L}_i$. Enforcing $\theta_i \in [0, 180^\circ)$ so that l_i^1 is always negative does not solve this problem. Then, a small perturbation to the line, say from $\theta_i = 0.1^\circ$ to $\theta_i = 179.9^\circ$, results in a large change in l_i^2, from approximately $+1.0$ to -1.0. To solve this problem, I choose θ_i to be whichever of θ_i or $\theta_i + 180^\circ$ makes $l_i^3 \geq 0$. This can also cause a problem when the line almost passes through

the origin. Then, a small perturbation to the line can result in a large change in both l_i^1 and l_i^2. I avoid this situation by making sure that the correct line does not pass close to the origin in any of the benchmark images.

Scale of $\mathbf{L}_i$: Multiplying $\mathbf{L}_i$ by a positive scalar does not change the line. The natural way to restrict the scale of a line vector, that is known not to be the line at infinity, is to require $(l_i^1)^2 + (l_i^2)^2 = 1.0$. As can be seen from Equation (6.2), this is already the case.

As the basis for my first global measure of coherence, I would like to use the sum of the variances of the three line coordinates l_i^1, l_i^2, and l_i^3. A natural question, however, is: how should the three components of this sum be weighted? A somewhat related question is whether the global measure of coherence is independent of the choice of units for x_i and y_i. The answer to the second of these questions is straightforward: the estimate of l_i^3 does depend upon the choice of units for x_i and y_i, whereas the choice does not affect the variances of l_i^1 and l_i^2.

Since the natural use of $\mathbf{L}_i$ is to test whether a point $\mathbf{x} = (x, y, 1)^{\mathrm{T}}$ lies on the line using $\mathbf{x}^{\mathrm{T}} \cdot \mathbf{L}_i = 0$, a sensible choice for the ratio of the weights in the sum is $(\mathrm{E}x)^2 : (\mathrm{E}y)^2 : 1$, where $\mathrm{E}x$ denotes the expected value of $|x|$ and $\mathrm{E}y$ denotes the expected value of $|y|$. To avoid any dependence on the distance units, I define the first global measure of coherence as follows:

$$\mathbf{GMC}_1 \;=\; \frac{1}{\mathrm{E}x \cdot \mathrm{E}y}\left[(\mathrm{E}x)^2 \cdot \sigma^2(l^1) + (\mathrm{E}y)^2 \cdot \sigma^2(l^2) + \sigma^2(l^3)\right] \tag{6.3}$$

where for $k = 1, 2, 3$:

$$\sigma^2(l^k) \;=\; \frac{1}{n}\sum_{i=1}^{n}\left[l_i^k - \frac{1}{n}\sum_{j=1}^{n} l_j^k\right]^2 \tag{6.4}$$

is the variance of the k^{th} coordinate of $\mathbf{L}_i$. There are a number of ways that Ex and Ey could be estimated, but for simplicity I use:

$$\text{E}x \;=\; \frac{1}{n}\sum_{i=1}^{n}|x_i| \tag{6.5}$$

and similarly for Ey.

6.2.2 All Edges are Colinear 2: Unknown θ_i

A variant of the first global measure of coherence can be estimated without using the angle θ_i. Suppose $e_i = (x_i, y_i, \theta_i) \in E$ and $e_j = (x_j, y_j, \theta_j) \in E$ are two detected edges. The line that passes through them can be estimated using:

$$\begin{aligned}\mathbf{L}_{ij} \;\equiv\; \left(l_{ij}^1, l_{ij}^2, l_{ij}^3\right)^{\text{T}} &= (x_i, y_i, 1)^{\text{T}} \wedge (x_j, y_j, 1)^{\text{T}} \\ &= (y_i - y_j, x_j - y_j, x_i \cdot y_j - x_j \cdot y_i)^{\text{T}}.\end{aligned} \tag{6.6}$$

Note that $\mathbf{L}_{ij}$ must be normalized so that $l_{ij}^3 \geq 0$ and $\left(l_{ij}^1\right)^2 + \left(l_{ij}^2\right)^2 = 1.0$. Then, Equation (6.3) can be used again to re-estimate the first global measure of coherence, but the way the variances were computed in Equation (6.4) must be modified. If the two edges that are used to compute $\mathbf{L}_{ij}$ are very close to each other, the estimate of $\mathbf{L}_{ij}$ will be unnecessarily noisy. Hence, I only allow edges to contribute to the variance of $\mathbf{L}_{ij}$ if they are more than 5 pixels apart. In particular, consider the set of edge index pairs:

$$P_1 \;=\; \left\{(i,j) \mid i < j \text{ and } \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2} > 5 \text{ pixels}\right\}. \tag{6.7}$$

Then, I estimate the variance of the k^{th} coordinate of $\mathbf{L}_{ij}$ using:

$$\sigma^2(l^k) \;=\; \frac{1}{|P_1|}\sum_{(i,j)\in P_1}\left[l_{ij}^k - \frac{1}{|P_1|}\sum_{(l,m)\in P_1} l_{lm}^k\right]^2 \tag{6.8}$$

rather than Equation (6.4).

6.2.3 All Edges Intersect at a Single Point

My second constraint is that all of the edges intersect at a single point. Such a scene can be constructed exactly as before, by placing a convex polygonal object with constant albedo lambertian faces in front of a perfectly black background. If an image of such a scene is cropped so that only the neighborhood of one of the vertices is visible, all of the edges should intersect at the image of the vertex. See Figure 6.1(b) for an example image of such a scene. Again, the fact that edges that intersect at a single point in the scene are projected onto edges that intersect at a single point in the image relies upon the assumption that the camera is linear.

Given two of the edges $e_i = (x_i, y_i, \theta_i), e_j = (x_j, y_j, \theta_j) \in E$ that are not parallel, it is possible to estimate the point where all of the edges intersect. If $\mathbf{L}_i = (l_i^1, l_i^2, l_i^3)^{\mathrm{T}}$ and $\mathbf{L}_j = \left(l_j^1, l_j^2, l_j^3\right)^{\mathrm{T}}$ are the vector representations of the lines passing through the two edges, they intersect at the point:

$$\mathbf{P}_{ij} \equiv \left(p_{ij}^1, p_{ij}^2, p_{ij}^3\right)^{\mathrm{T}} = \left(l_i^1, l_i^2, l_i^3\right)^{\mathrm{T}} \wedge \left(l_j^1, l_j^2, l_j^3\right)^{\mathrm{T}}. \tag{6.9}$$

Again, since equality is only defined up to a constant multiplicative factor, this expression for the point of intersection needs to be normalized. In this case, normalization is easy. Since the lines $\mathbf{L}_i$ and $\mathbf{L}_j$ are not parallel, $p_{ij}^3 \neq 0$. So, $\mathbf{P}_{ij}$ can be divided by its third coordinate to give the normalized point of intersection:

$$\overline{\mathbf{P}}_{ij} \equiv \left(\overline{p}_{ij}^1, \overline{p}_{ij}^2, 1\right)^{\mathrm{T}} = \left(\frac{p_{ij}^1}{p_{ij}^3}, \frac{p_{ij}^2}{p_{ij}^3}, 1\right)^{\mathrm{T}}. \tag{6.10}$$

Just as before, I would like to use the sum of the variances of $\overline{p}_{ij}^1$ and $\overline{p}_{ij}^2$ as the basis for my second global measure of coherence. The same two questions arise of: (1) how to weight the two components in the sum, and (2) how to avoid any dependence on the units of x and y. In addition, a pair of edges should contribute

to the variance only if they are sufficiently far away from parallel. Hence, I define the second global measure of coherence to be:

$$\mathbf{GMC}_2 \;=\; \frac{1}{(\mathrm{E}x)^2}\sigma^2(\overline{p}^1) + \frac{1}{(\mathrm{E}y)^2}\sigma^2(\overline{p}^2) \tag{6.11}$$

where for $k = 1, 2$:

$$\sigma^2(\overline{p}^k) \;=\; \frac{1}{|P_2|}\sum_{(i,j)\in P_2}\left[\overline{p}^k_{ij} - \frac{1}{|P_2|}\sum_{(l,m)\in P_2}\overline{p}^k_{lm}\right]^2 \tag{6.12}$$

is the variance of the k^{th} coordinate of $\overline{\mathbf{P}}_{ij}$,

$$P_2 \;=\; \{(i,j) \;:\; |\theta^i - \theta^j| > 15^\circ\}, \tag{6.13}$$

is the set of edge index pairs which are not too close to parallel, and Ex and Ey are defined in the same way as before.

6.2.4 All Edges are Parallel

My third constraint is that all of the edges are parallel in the scene. Such a scene can be constructed by placing a lambertian cuboid with constant albedo faces in front of a perfectly black background. An image of such an object is then cropped so that only one set of parallel edges is visible. See Figure 6.1(c) for an example image of such an scene. Unlike the previous two constraints, parallel edges in the scene do not always remain parallel in an image because of perspective foreshortening effects. However, if the camera is linear, parallel edges in the scene are projected onto edges that intersect at a single point in the image, known as the vanishing point. This constraint is unique amongst the four that I consider in that the constraint on the edges in the scene leads to a different constraint in the image.

Since the image constraint is the same as in the previous section, I begin in the same way by estimating the intersection of two lines passing through two edges $e_i = (x_i, y_i, \theta_i), e_j = (x_j, y_j, \theta_j) \in E$. If $\mathbf{L}_i = (l_i^1, l_i^2, l_i^3)^{\mathrm{T}}$ and $\mathbf{L}_j = \left(l_j^1, l_j^2, l_j^3\right)^{\mathrm{T}}$ are the vector representations of the two lines, they intersect at the vanishing point:

$$\mathbf{V}_{ij} \equiv \left(v_{ij}^1, v_{ij}^2, v_{ij}^3\right)^{\mathrm{T}} = \left(l_i^1, l_i^2, l_i^3\right)^{\mathrm{T}} \wedge \left(l_j^1, l_j^2, l_j^3\right)^{\mathrm{T}}. \tag{6.14}$$

The vanishing point $\mathbf{V}_{ij}$ may or may not lie at infinity, so v_{ij}^3 may or may not equal 0. If $\mathbf{V}_{ij}$ does lie at infinity, which is approximately the case in the benchmark images, the only useful information is the angle that it makes with the x-axis:

$$\phi_{ij} = \arctan\left(\frac{v_{ij}^2}{v_{ij}^1}\right). \tag{6.15}$$

I would like to use the variance of ϕ_{ij} as the third global measure of coherence. The questions of weighting and distance units are not an issue here, but the cyclic range of the arctan function is somewhat problematic; a vanishing point at one end of the range can easily be perturbed by noise to lie at the other end. The solution I adopted is to use the range $[-90°, 90°]$, making sure that the correct vanishing point never lies close to either $90°$ or $-90°$ in any of the benchmark images. Finally, I define my third global measure of coherence to be:

$$\mathbf{GMC}_3 = \frac{1}{(180°)^2} \cdot \frac{1}{|P_3|} \sum_{(i,j)\in P_3} \left[\phi_{ij} - \frac{1}{|P_3|} \sum_{(l,m)\in P_3} \phi_{lm}\right]^2 \tag{6.16}$$

where

$$P_3 = \{(i,j) \ : \ |l_3^i - l_3^j| > 20 \text{ pixels}\} \tag{6.17}$$

and $l_3^i = x^i \sin\theta^i - y^i \cos\theta^i$ is the third line coordinate. Since l_3^i is the closest distance in pixels from the origin of the image to the line through the edge, P_3 excludes any edge pairs that might lie on the same line in the image.

6.2.5 All Edges Lie on an Ellipse: Unknown θ_i

The fourth and final constraint is that all of the edges lie on an ellipse. Such a scene can be constructed by cutting off a constant albedo lambertian circular cylinder with a plane and then placing it in front of a perfectly black background. An image of the cylinder is cropped so that the sides of the cylinder are not visible and the only edges that can be seen are on the ellipse. An example of such as scene is presented in Figure 6.1(d). Again, the fact that edges lying on an ellipse in the scene are projected onto edges lying on an ellipse in the image relies upon the implicit assumption that the camera is linear. See Section 6.3.1 for a description of how nonlinear effects such as radial distortion can be compensated for.

In this section, I derive the fourth global measure of coherence, assuming that the orientation of the edge is unknown. In the next section, I describe two variants that use the orientation information. An ellipse is defined by the equation:

$$\mathbf{x}^{\mathrm{T}}\mathbf{A}\mathbf{x} \;=\; 0 \tag{6.18}$$

where $\mathbf{x} = (x, y, 1)^{\mathrm{T}}$ is a homogeneous vector of image coordinates, and:

$$\mathbf{A} \;=\; \begin{pmatrix} a_{11} & \frac{1}{2}a_{12} & \frac{1}{2}a_{13} \\ \frac{1}{2}a_{12} & a_{22} & \frac{1}{2}a_{23} \\ \frac{1}{2}a_{13} & \frac{1}{2}a_{23} & a_{33} \end{pmatrix} \tag{6.19}$$

is a 3×3 symmetric matrix, as usual only defined up to a scale factor [35]. So, the matrix $\mathbf{A}$ has just five independent parameters. The fact that the i^{th} edge $e_i = (x_i, y_i, \theta_i) \in E$ lies on the ellipse provides one constraint on $\mathbf{A}$:

$$(x_i, y_i, 1)\mathbf{A}(x_i, y_i, 1)^{\mathrm{T}} \;=\; 0. \tag{6.20}$$

Since this constraint is linear in the six unknowns (a_{11}, a_{12}, a_{13}, a_{22}, a_{23}, and a_{33}), the ellipse can be recovered using Gauss-Jordan elimination [101] if the locations of five edges on the ellipse are known. Since the vector of ellipse parameters is only determined up to a scale factor, the problem that appeared in Section 6.2.1 of setting the sign and scale of the vector of parameters also arises here. I solve this problem by enforcing $a_{33} = 1.0$ to define the sign and scale, while using benchmark images for which the ellipse does not pass close to the origin to ensure that the sign of a_{33} cannot be accidentally changed by a small perturbation to the edges.

As the basis for the fourth global measure of coherence, I would like to use the sum of the variances of the estimates of the five ellipse parameters that are free; ie. all of them except a_{33}. Again, the questions of how to weight the sum and avoid any dependence upon the units of x and y are important. Since Equation (6.18) can be used to determine whether an image point $\mathbf{x} = (x, y, 1)^{\mathrm{T}}$ lies on the ellipse, I define the fourth and final global measure of coherence to be:

$$\begin{aligned} \mathbf{GMC}_4 &= (\mathrm{E}x)^4\sigma^2(a_{11}) + (\mathrm{E}x)^2\sigma^2(a_{13}) + \\ &\quad (\mathrm{E}y)^4\sigma^2(a_{22}) + (\mathrm{E}y)^2\sigma^2(a_{23}) + (\mathrm{E}x)^2(\mathrm{E}y)^2\sigma^2(a_{12}) \end{aligned} \tag{6.21}$$

where $\sigma^2(a_{ij})$ is the variance of the estimate of ellipse parameter a_{ij}. For example, the weight for $\sigma^2(a_{11})$ is Ex raised to the fourth power because the term involving a_{11} in Equation (6.18) is $a_{11} \cdot x^2$. Since it takes five edges to estimate the ellipse parameters, the variances of the ellipse parameters a_{ij} are all computed over the set of quintuples of edges, no pair of which are closer than 30 pixels apart.

6.2.6 All Edges Lie on an Ellipse: Known θ_i

If the orientation of each edge is known, it is possible to estimate the ellipse parameters using just three edges. Each edge $e_i = (x_i, y_i, \theta_i) \in E$ provides two constraints on the parameters. The first one is $(x_i, y_i, 1)\mathbf{A}(x_i, y_i, 1)^{\mathrm{T}} = 0$, as above. The second one is that the tangent to the ellipse must have orientation θ_i at the image point $(x_i, y_i, 1)^{\mathrm{T}}$. Differentiating Equation (6.18), setting $\frac{\mathrm{d}y}{\mathrm{d}x} = \tan\theta$, and reorganizing gives the tangency constraint:

$$[2x_i a_{11} + a_{13} + y_i a_{12}]\cos(\theta_i) + [2y_i a_{22} + a_{23} + x_i a_{12}]\sin(\theta_i) \;=\; 0. \qquad (6.22)$$

So, given three edges, three ellipse constraints and two tangency constraints can be used to estimate the ellipse parameters. The second variant of the fourth measure of coherence is then exactly as given above, but with the variance now being computed across all triples of edges, no pair of which are closer than 50 pixels apart.

Finally, I also considered a third variant of the fourth global measure of coherence computed using four edges, no pair of which are closer than 40 pixels apart. For this final variant, four ellipse constraints and one tangency constraint are used to estimate the ellipse parameters.

6.3 Computing the Global Measures

In this section, I describe how I actually compute the global measures of coherence. In particular, there are several details that I omitted to mention in the previous section. First, all of the measures rely upon the assumption that the camera is linear. In Section 6.3.1, I describe how the nonlinear effects of radial distortion can be corrected. Second, computing several of the measures in the obvious way is

intractable, because doing so leads to $\Theta(n^3)$ or slower algorithms. In Section 6.3.2, I show how Monte Carlo algorithms can be used to find good estimates of the measures in a reasonable amount of time. Third, most detectors have various thresholds that need to be set. In Section 6.3.3, I show how one of these thresholds can be left unset. The performance of the detector is then characterized by a curve parameterized by this free threshold. Fourth, to obtain a reliable estimate of the measures, a large number of images need to be used. In Section 6.3.4, I describe how these images are captured, and in Section 6.3.5 how the results are averaged.

6.3.1 Correcting for Radial Distortion

My four global measures of coherence are based upon constraints on the edges in the scene, for example that they are parallel. Converting such constraints into corresponding constraints on the edges detected in an image requires the assumption that the camera is linear. For example, parallel edges in the scene become edges which intersect at a single point, the vanishing point, only if the camera is linear. An ideal pinhole camera should be linear, however, the use of a lens may introduce various nonlinear distortions, the two major types being radial and tangential [87]. Both types of distortion can be modeled, but in practice the tangential component is negligible, and only the radial distortion is actually compensated for [116]. I applied Tsai's algorithm [116] to estimate the radial distortion, but found it also to be negligible ($\kappa_1 \approx -2.0 \times 10^{-6}\,\text{pixels}^{-2}$) for the focal length that I used. So, I assumed the camera to be linear for the experiments described in Sections 6.4 and 6.5. If a shorter focal length was ever needed, Tsai's algorithm could easily be used to compensate for any noticeable radial distortion. Finally, note that measures

similar to my global measures of coherence have actually been used in the past to perform camera calibration for the radial distortion [9] [18].

6.3.2 Efficient Computation using Monte Carlo

Since the number of detected edges is usually at most $n = 10^3$, it is possible to compute the first three measures by simply enumerating all pairs of edges. The fourth measure, however, requires all quintuples of edges on the ellipse to be enumerated. Doing so will, in general, be intractable. So, to compute the fourth measure I use a Monte Carlo algorithm [101]. Quite simply, I sample the set of quintuples of edges randomly a fixed number of times, typically around 1000 times. I then compute the variance over the samples and use it as an estimate of the variance over the the entire set of quintuples. The same approach can naturally be used for the other two variants of the fourth global measure of coherence.

6.3.3 Dealing with Detector Thresholds

Nearly all edge detectors have various thresholds that need to be set. For example, gradient based detectors (see Section 2.2) threshold on the magnitude of the gradient, and model-matching detectors (see Section 2.1) threshold on the degree of fit. The value of any performance metric, including my global measures of coherence, will depend upon the settings of these thresholds. In this section, I describe how one of these thresholds can be left unset. The performance of each edge detector is then characterized by a curve parameterized by the free threshold. In [32], Dougherty and Bowyer used a generalization of this technique to allow more than one threshold to be left unspecified. Such a generalization could also be used for my global

measures of coherence, however most edge detectors that do not include any form of feature aggregation or adaptive thresholding typically only have one threshold anyway. All of the detectors that I experimented on had just one threshold.

Given an image and a threshold setting, a certain number of edges will be detected. As the threshold is varied, more or less edges will be detected, depending on which way the threshold operates. For any particular threshold setting, it is possible to compute the global measures of coherence. So, if the threshold is changed so that more edges are detected, the global measures of coherence will also change, in all likelihood getting worse. So, by varying the threshold, a curve can be plotted of the number of edges detected against the global measure of coherence. For any specific number of edges, the smaller the measure the better. So, two detectors can be compared without setting their thresholds by plotting these curves. The closer the curve lies to the abscissa, the better the performance.

Because the global measures of coherence are all simple combinations of the variances of certain quantities, these curves can be computed easily without actually thresholding. Each edge detector is rewritten so that the detected edges are sorted by the quantity that is finally thresholded. The edges are output in this order. The program that computes the global measure of coherence can then keep a running estimate of the variances that compose the measure. As each edge is read in, the number of edges detected is incremented, the variances updated, and the global measure of coherence computed. It is then trivial to output a point on the curve; ie. a pair consisting of the number of edges and the global measure of coherence.

6.3.4 Capturing a Large Number of Images

To get a robust estimate of a performance measure, it is important to use a large number of images. The easiest way to capture these images for my benchmarks was by varying the illumination and camera geometry. In particular, I fixed the position of the light, used a turntable to rotate the object to five different angles, and independently moved the camera to fifteen different points in space with a six degree of freedom robot. The result was seventy-five images of each object under widely varying conditions. Some of these images contain more than one region that could be cropped for the benchmark. For example, in the image of the cube in Figure 6.1(a) there are three different extended surface normal discontinuity edges that could be cropped. Including all possible regions that are reasonably sized results in each of the benchmarks containing between seventy-five and two hundred images. This number of images is sufficient to give robust estimates of the global measures of coherence, although more images could easily have been captured by sampling the viewing and illumination conditions more densely.

6.3.5 Averaging over a Large Number of Images

For each cropped benchmark image, I compute a curve of the number of detected edges against the global measure of coherence, as described in Section 6.3.3. Naturally, the next question is how does one average these curves over the seventy-five to two hundred images that comprise the benchmark. One possibility would be to compute the average value of the measure for each setting of the number of image. The problem with this approach is that because the images comprising the benchmark are all different sizes, the point at which the detectors enter phase three

is different for each curve. Hence, all information beyond the size of the smallest image would be lost. Another possibility might be to average the values of the measure for different settings of the thresholded parameter. However, this approach would make it difficult to compare the detectors because the threshold parameters all have different meanings.

So, what I do is the following. For each possible value of the global measure of coherence, I sum up the number of edges than can be used in each of the benchmark images and still obtain that global measure of coherence. Afterwards, I divide the sum by the number of images in the benchmark. The result is a curve of the global measure of coherence against the average number of edges that can be used while still obtaining that measure of coherence. One minor problem with this approach is that sometimes the global measure of coherence is not quite a monotonically increasing function of the number of edges. I resolve this problem by counting the number of edge measure pairs on the curves for which the measure is below the level currently being evaluated, even if for a smaller number of edges the measure is above the current level.

6.4 Experimental Results

In this section, I present experimental results obtained by applying the benchmarks to five edge detectors. The first two are the Canny-like operator and the Nalwa-Binford detector that were considered in Chapter 4. The third is the step edge detector developed in Chapter 3. The final two are the Roberts' cross operator [105] and the Sobel operator [98], both of which are described in texts such as [79] and [100]. The Canny and Nalwa-Binford detectors are relatively modern and

highly regarded. The Roberts' cross and Sobel operators were amongst the first few edge detectors proposed, and are regarded as relatively poor detectors.

The main reason for selecting two relatively poor detectors was to demonstrate the range of performance that can be expected using the global measures of performance. For a measure of performance to be useful, there must be a marked difference between the performance of the best and worst detectors. One criticism that could be made of several recent performance evaluation papers such as [47] is that by only testing supposedly good detectors, the full range of performance was never completely sampled. The results presented in [47] show that the detectors perform fairly similarly. It is then unclear whether this is because the detectors are similar in quality, or whether the evaluation methodology is incapable of ever widely separating good detectors from bad ones. The goal of this experimental section is to demonstrate that a large range of different performance levels can be obtained using the global measures of coherence, not that any specific detector is superior to the others. Naturally, it is expected that the Roberts' cross and Sobel detectors will perform noticeable worse than the other three detectors.

As described in the introduction to this chapter, the global measures of coherence are specifically designed to be representative of tasks requiring precise sub-pixel localization and orientation estimation; ie. good parameter estimation accuracy. For this reason, two of the detectors considered (the Nalwa-Binford detector and my step edge detector) come with sub-pixel estimates of the location of the edge. The other three provide no sub-pixel estimation. By comparing the relative performance of the five detectors across the four measures, it will be possible to see for which measures estimating the location of the edge accurately is vitally

important, and for which measures only estimating the orientation is important.

As described in Sections 2.5 and 4.1, besides parameter estimation accuracy, there is one other fundamental measure of edge detection performance, namely, feature detection robustness; ie. the rates of occurrence of false positives and false negatives. If a detector generates spurious false positives randomly located in the image, the global measures of coherence will be very poor, as they should be. On the other hand, if a detector generates false positives close to the correct location, for example when a detector produces "thick" edges, the performance will degrade more slowly. In order to alleviate this problem, all five detectors were extended with the same non-maximum suppression step. Quite simply, any edges for which there is a more prominent edge in the direction normal to it were removed.

If a detector fails to detect some of the edges and produces false negatives, the global measures of coherence will unfortunately not be adversely affected. This failure to penalize false negatives is one of the weaknesses of the global measures of coherence, however there is a way to tell if a detector is missing a large number of edges. As will be seen later in this section, the graphs of the measures against the average number of edges used to estimate them increase dramatically at a certain point. This point corresponds to when the detector starts detecting spurious false positives all over the image. For a detector with non-maximum suppression, the point at which this occurs is a measure of the total number of correct edges detected before the detector began to detect false positives. For a detector that produces a large number of false negatives, this phenomenon will occur at a smaller number of edges than it would otherwise. So, it is possible to use the global measures to spot detectors that are generating a larger number of false positive.

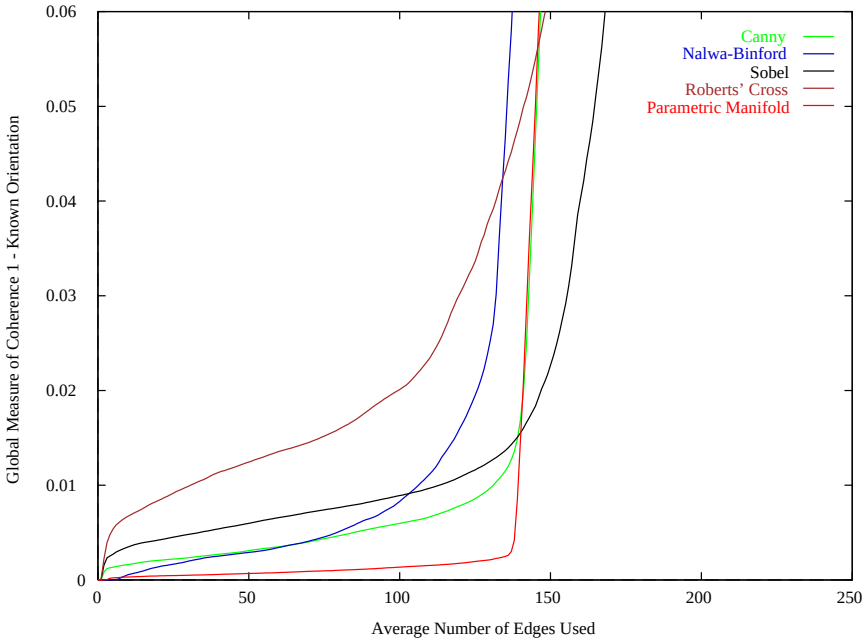


Figure 6.2: The results for the known orientation variant of the first global measure of coherence. The parametric manifold detector does the best. The Nalwa-Binford and Canny detectors perform slightly worse, followed by the Sobel detector. The Robert's cross performs by far the worst. The Nalwa-Binford detector suffers from a higher degree of false negatives than the others, as can be seen by the slow, early transition to the third rapid growth phase. These results are consistent with those in Figure 4.1 and in [80].

In the remainder of this section, I present the results obtained for the five detectors on the four global measures of coherence and their variants. I begin with the global measure of coherence for colinear edges.

6.4.1 GMC 1: All Edges Are Colinear

I present the results for the first global measure of coherence in Figures 6.2 and 6.3. Figure 6.2 contains the results for the first variant assuming known orientation, and Figure 6.3 contains the results for the second variant assuming unknown θ_i. The benchmark used consists of 150 cropped images of the depth discontinuities of the cube in Figure 6.1(a). Similar results were obtained using the surface normal

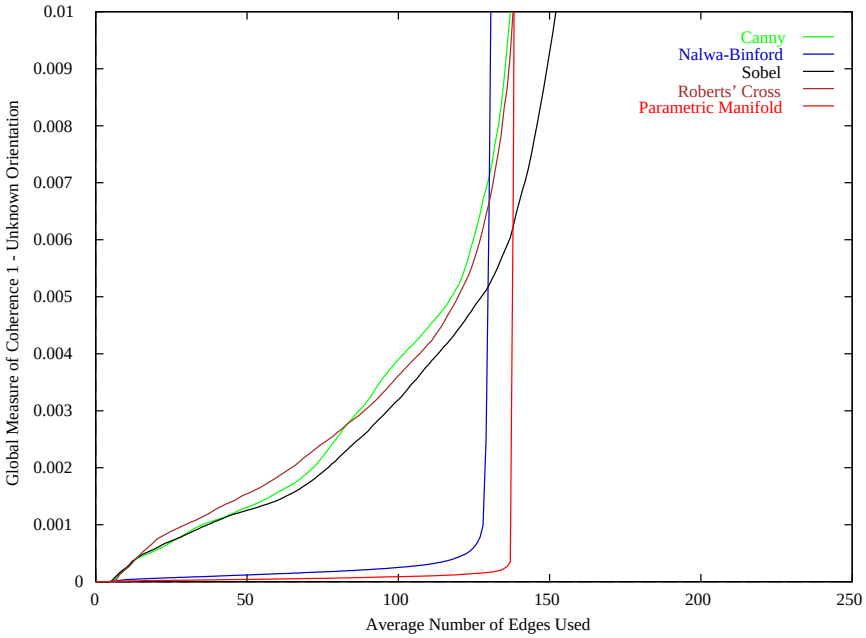


Figure 6.3: The results for the second, unknown orientation variant of the first global measure of coherence. The parametric manifold and Nalwa-Binford detectors perform significantly better than the other three detectors. These results demonstrate the clear distinction between the two detectors which provide subpixel localization (the Nalwa-Binford and parametric manifold detectors) and those that don't (the Sobel, Roberts' cross, and Canny detectors). Hence, this second variant of the first global measure of coherence is largely a measure of subpixel localization accuracy.

discontinuities but are not shown.

The way to interpret these results is as follows. Each curve can be divided into three phases. In the first phase where only a small number of edges are being used, the curves start at zero and then rapidly grow to a relatively stable value. The first phase is largely unimportant since the nature of the behavior just depends upon the order in which the first few easily detectable edges are actually detected. The second phase begins when the growth rate slows and continues until it starts rising rapidly again. The second phase is the most important in terms of parameter estimation accuracy because it corresponds to the situation where the detectors is

detecting most of the edges, and there are no false positives to degrade the global measures of coherence. In the second phase, the closer a curve lies to the abscissa the better. The third phase begins when the curve starts to rise rapidly again. This occurs when the detector starts detecting false positives. Since the measures are based on variances, they are not robust to outliers. Hence, as soon as any false positives are detected, the performance drops off rapidly. As discussed above, if the third phase begins for a relatively small number edges, it is indicative of a detector that suffers badly from false negatives, since it means many edges have been not yet been detected by the time the false positives start to be detected. So, the later the start of the third phase begins, the better the detector is performing. Also, the sharper the transition from the second to the third phase the better. In Figure 6.2, the first phase for the Roberts' cross operator lasts from 0 until about 10 edges, the second phase from about 10 edges until about 110 edges, and the third phase from 110 edges onwards. For the parametric manifold detector, the transition from the second to the third phase is very sharp and occurs at around 140 edges.

In the second phase of the results for the first variant of the first global measure of coherence displayed in Figure 6.2, the ranking of the detectors is very clear. The best detector is the parametric manifold detector. The Canny and Nalwa-Binford detectors perform next best, and roughly the same. Next is the Sobel detector, and the worst by a long way is the Roberts' cross detector. This ordering is pretty much exactly what one would expect. Looking at the transition from the second to the third phase, one sees that the order is similar, with one exception. The transition for the Nalwa-Binford detector is further to the left and is less sharp than for the Canny and Sobel detectors. I therefore conclude that the

Nalwa-Binford detector suffers from a relatively high percentage of false negatives. This fact is totally consistent with Figure 4.1 in the Statistical Tests section of Chapter 4. It is also consistent with the results presented in [80].

The results for the second (unknown orientation) variant of the first global measure of coherence displayed in Figure 6.3 are somewhat different. First, note that the scale on the ordinate has changed. So long as the two edges are far enough apart, estimating the line from two edges is far easier than from a single edge and its orientation. This is as one would expect. The results for the second variant demonstrate the clear distinction between the two detectors which provide subpixel localization (the Nalwa-Binford and parametric manifold detectors) and those that don't (the Sobel, Roberts' cross, and Canny detectors). The second variant of the first global measure of coherence is largely a measure of subpixel localization, whereas to do well on the first variant, both accurate subpixel localization and orientation estimation are required. Depending upon the requirements of a specific application, a user could easily decide which of the two variants to use.

6.4.2 GMC 2: All Edges Intersect at a Single Point

In Figure 6.4, I present the results for my second global measure of coherence. The benchmark used to generate these results consists of 150 cropped images of the vertices of the tetrahedron in Figure 6.1(b). The vertices used were the ones where two extended edges meet; ie. the ones formed by one depth discontinuity and one reflectance discontinuity. The results for the three edge vertices are very similar and are omitted. One minor difference from the previous results is that phase one lasts longer. This fact is of little significance, for the reasons described above. Quite

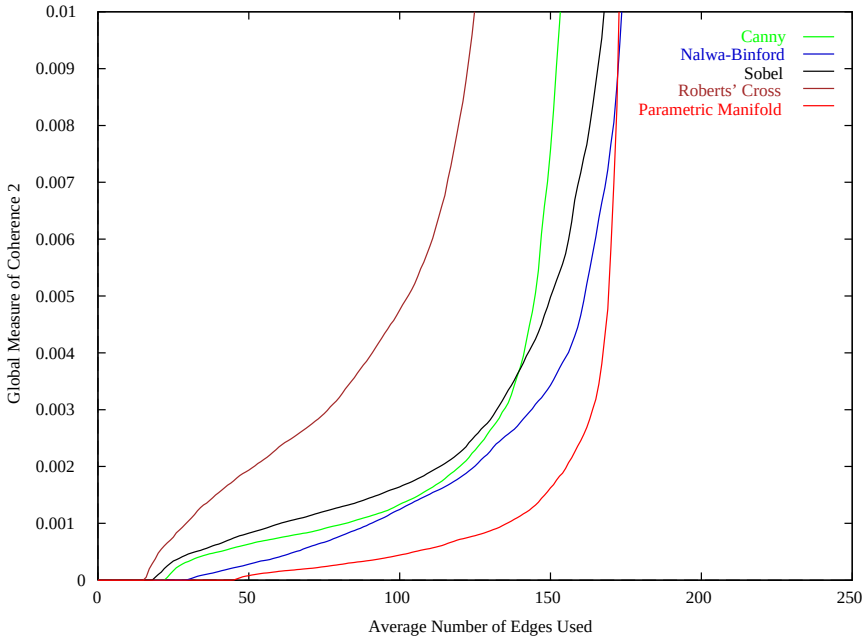


Figure 6.4: The results for the second global measure of coherence. The parametric manifold detector performs the best by a wide margin. Next, the Canny, Nalwa-Binford, and Sobel detectors perform similarly. Finally, the Roberts' cross operator performs by far the worst. This figure clearly demonstrates the wide range of performance levels that are possible with global measures of coherence, a fundamental requirement for a performance measure to be useful.

simply the detectors tend to detect most of the edges from one of the extended edges before the other. Therefore, between 20 and 50 edges need to be detected before there are a pair that can actually be used to estimate the intersection point. The results in Figure 6.4 again show that the parametric manifold detector performs the best. The Canny, Nalwa-Binford, and Sobel detectors all perform similarly, and as before, the Robert's cross operator is by far the worst. The most important point, however, is the huge difference in performance between the worst detector (the Roberts' cross operator) and the best one (the parametric manifold detector). Thus, the ability of my global measures of performance to discriminate between multiple performance levels is clearly demonstrated.

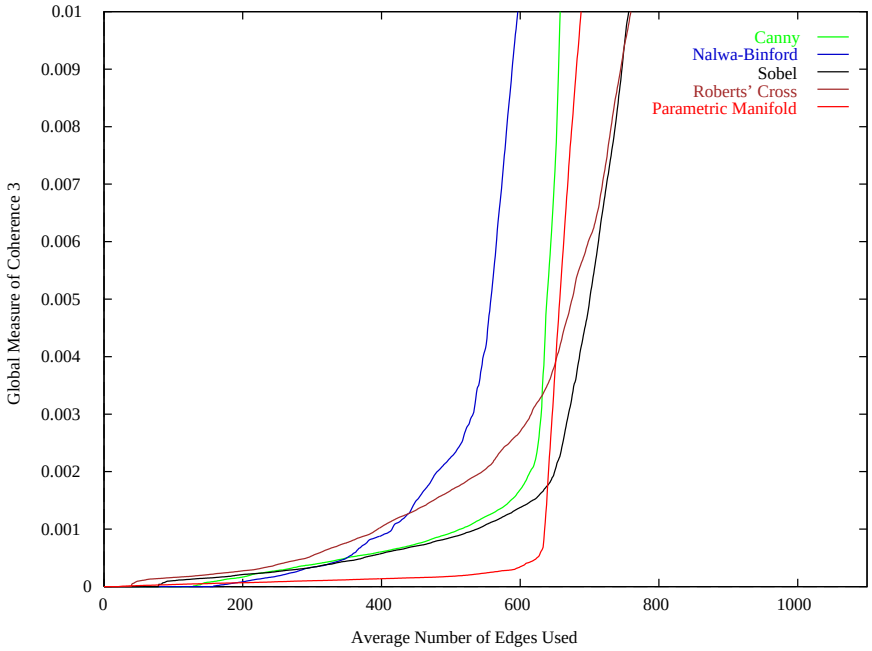


Figure 6.5: The results for the third global measure of coherence. The parametric manifold detector again performs the best, followed by the Canny and Sobel detectors. The Roberts' cross operator does by far the worst. The third phase of the Nalwa-Binford detector starts early, as in Figure 6.2, indicating a large number of false negatives.

6.4.3 GMC 3: All Edges Are Parallel in the Scene

In Figure 6.4, I present the results for the third global measure of coherence. The benchmark used to generate these results consists of 75 cropped images of the three parallel edges of the rectangular cuboid in Figure 6.1(c). As can be seen, one of the edges is a surface normal discontinuity, the second is a depth discontinuity, and the third is a reflectance discontinuity. As can be seen, the results are similar to before with the parametric manifold detector doing the best, followed by the Canny and Sobel detectors. Again, the Roberts' cross operator is by far the worst. As for the first measure, phase three for the Nalwa-Binford detector begins much earlier than for the other detectors, indicating that it is suffering from a large number of false negatives. Since there is almost no foreshortening in the benchmark images, the

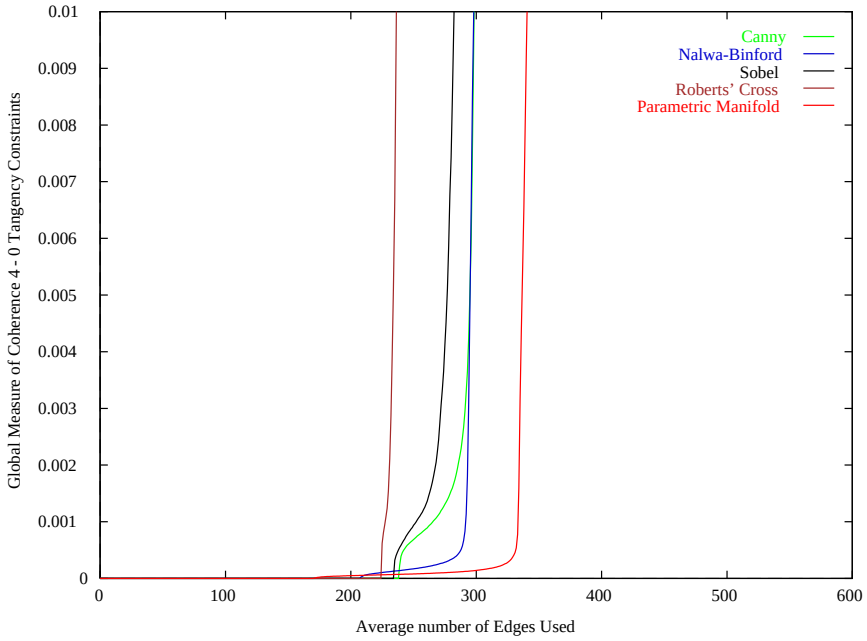


Figure 6.6: The results for the unknown orientation variant of the fourth global measure of coherence. The location of five edges are used to estimate the ellipse. The parametric manifold and Nalwa-Binford detectors perform significantly better than the other three detectors. Like Figure 6.3, these results demonstrate the clear distinction between the two detectors that provide subpixel localization and those that don't. This variant of the fourth global measure of coherence is largely a measure of subpixel localization accuracy.

vanishing point is close to lying at infinity. Hence, the most important element of the third global measure of performance is the orientation estimation accuracy. It is no surprise then that the Nalwa-Binford detector performs worse than both the Canny and parametric manifold detectors because this was what was discovered before in Figure 4.3 of Section 4.1.2.

6.4.4 GMC 4: All Edges Lie on an Ellipse

I present the results for the fourth global measure of coherence in Figures 6.6–6.8. Figure 6.6 contains the results for the first variant, the one that assumes unknown orientation and uses the location of five edges to estimate the ellipse. Figure 6.7

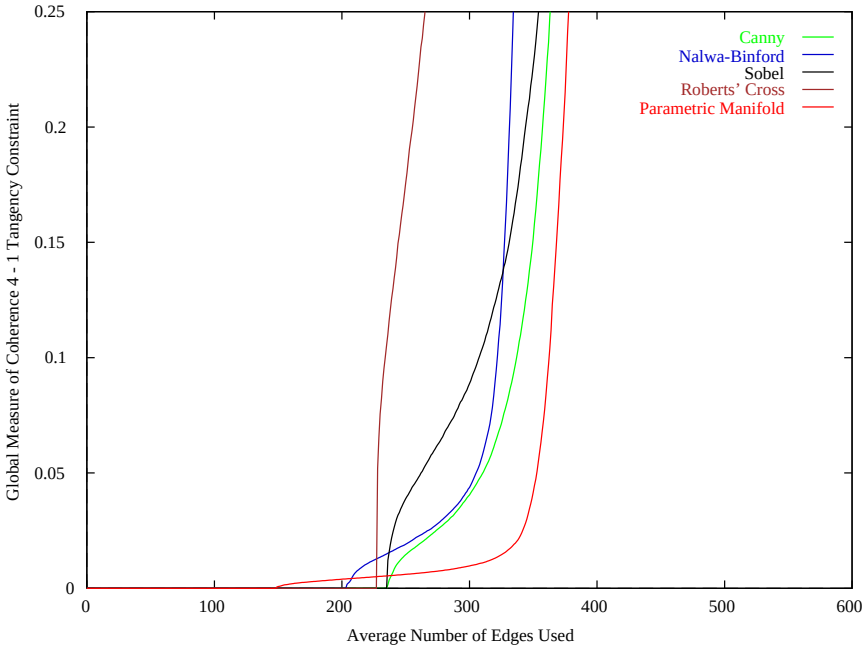


Figure 6.7: The results for the first known orientation variant of the fourth global measure of coherence. The location of four edges and one tangency constraint are used to estimate the ellipse. The parametric manifold detector does the best, followed by the Nalwa-Binford and Canny detectors doing about the same, then the Sobel operator, and finally the Roberts' cross operator.

contains the results for the second variant, the one that uses the location of four edges and one tangency constraint. Finally, Figure 6.8 contains the results for the third variant, the one that uses the location of three edges and two tangency constraints. The benchmark consists of 75 cropped images of a cylinder sliced with a plane, as shown in Figure 6.1(d).

The first thing to note about the results is that the first phase is much longer than before. The reason is that, to estimate the ellipse, between three and five edges are needed. Moreover, I require that the edges are well spread around the ellipse (see Section 6.2.5), because otherwise the estimates of the ellipse are very sensitive to noise. For similar reason to those given in Section 6.4.2, it often takes until over

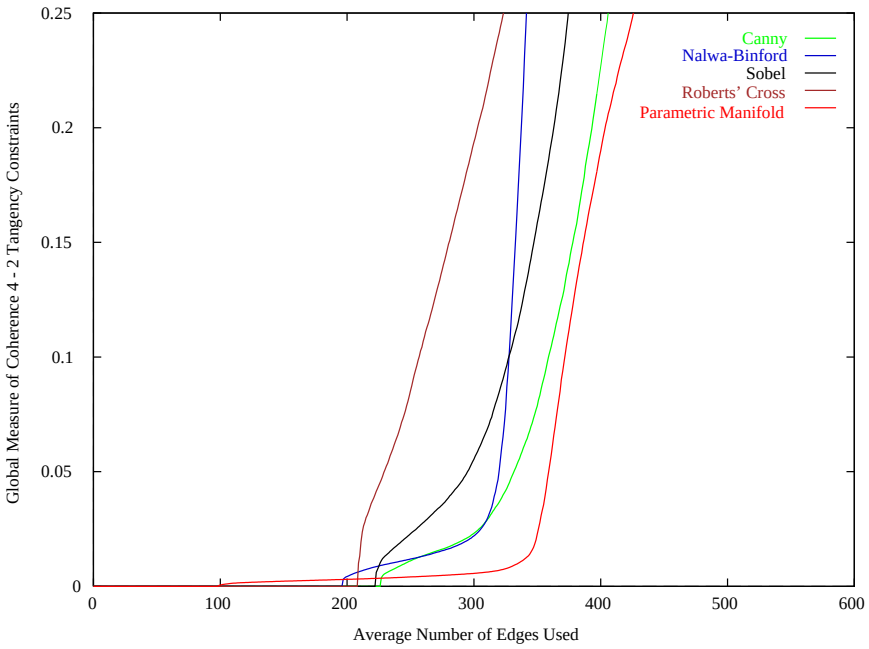


Figure 6.8: The results for the second known orientation variant of the fourth global measure of coherence. The location of three edges and two tangency constraints are used to estimate the ellipse. The parametric manifold detector does the best, followed by the Nalwa-Binford and Canny detectors doing about the same, then the Sobel operator, and finally the Roberts' cross operator.

two hundred edges have been detected before the edges are sufficiently well spread around the ellipse for the ellipse to be estimated reasonably accurately.

The results for the first variant in Figure 6.6 are similar to those in Figure 6.3 for the same reason that only the location of the edges is used. Like there, the two detectors that provide subpixel localization (the Nalwa-Binford and parametric manifold detectors) perform far better than the three that don't (the Sobel, Roberts' cross, and Canny detectors). As usual, the Robert's cross operator is even worse than Sobel and Canny detectors. In fact, it does so badly that the second phase appears to be non-existent. The Sobel and Canny detectors also do quite badly with their second phases being relatively short and steep.

The results for the two other variants in Figures 6.7 and 6.8 are very similar to each other. The parametric manifold detector does the best, followed by the Canny and Nalwa-Binford detectors doing about the same, then the Sobel operator, and finally the Roberts' cross operator. Note that the scales of the ordinate in both of these graphs have been changed from that used in Figure 6.6. As in Section 6.4.1, when using more edges with subpixel localization it is easier to estimate the ellipse than when using fewer edges and the tangency constraint. As was the case there, the first variant is largely a measure of the subpixel localization and the other two variants require both accurate subpixel localization and orientation estimation for performance to be good. A user could decide which of the three variants to use, based upon the requirements of the application at hand.

6.5 Varying Camera Parameters

Because the images used to estimate the global measures of coherence are captured in a controlled environment, it is very easy to vary the imaging conditions to investigate how performance degrades as the conditions become more difficult. In this section, I present the results of two experimental comparisons. In the first experiment, I varied the focus setting. In the second, I varied the aperture of the camera. A number of other experiments could have been performed. For example, the material or reflectance properties of the objects in the scene could have been changed. Alternatively, the sharpness of the surface normal discontinuities could have been reduced. The results presented here are meant to be illustrative of the type of information that global measures of coherence can provide about how edge detection performance degrades as imaging conditions become more difficult.

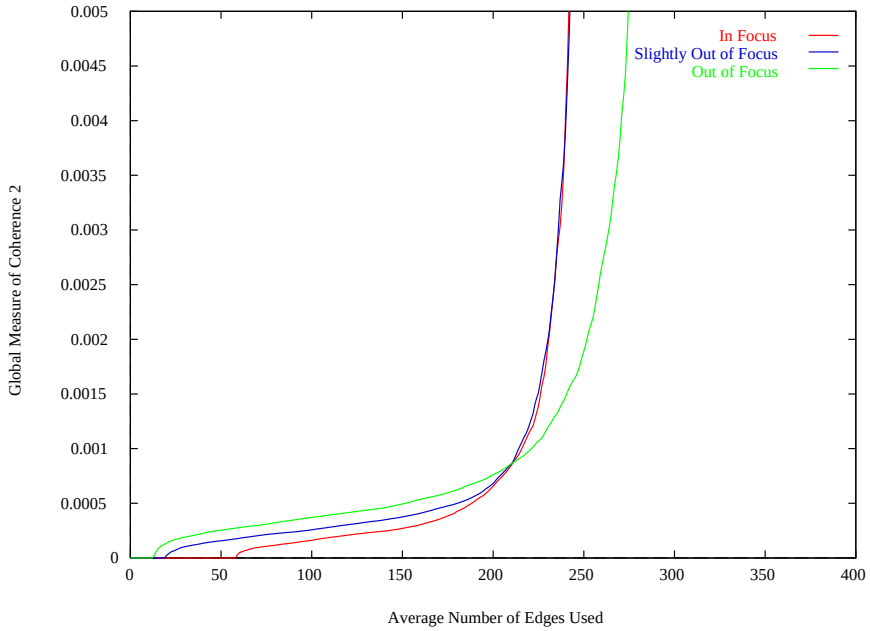


Figure 6.9: The results of varying the focus setting for the parametric manifold detector. As the images become more defocused the parametric detector performs worse. However, for defocused images more edges are detected before the performance falls off rapidly.

6.5.1 Varying the Focus Setting

In Figures 6.9–6.11, I present the results of varying the focus setting. I captured three sets of images for the second global measure of coherence, one fully focused, one slightly out of focus, and one even more out of focus. I then estimated the second global measure of coherence for all three sets of images. Figure 6.9 contains the results for the parametric manifold detector, Figure 6.10 the results for the Nalwa-Binford detector, and Figure 6.11 the results for the Canny detector.

As one would expect, the performance of all three detectors becomes worse as the images get more defocused. However, the drop off in performance is noticeably worse for the Canny detector than it is for the other two detectors. Another interesting point is the variation in the point at which the detectors enter the third rapid growth phase. For the parametric manifold and Nalwa-Binford detectors this

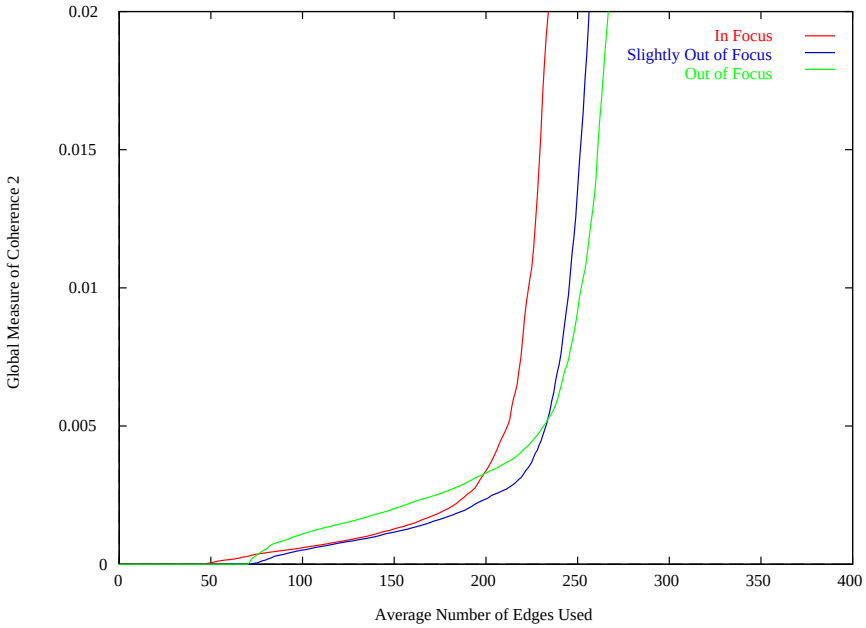


Figure 6.10: The results of varying the focus setting for the Nalwa-Binford detector. Just as for the parametric manifold detector the performance gets worse as the images get more defocused. Also, when the images are very defocused more edges are detected before the performance drops off completely.

point gets later as the images get more defocused, whereas for the Canny detector it gets earlier. These results indicate that as images become more defocused, model-matching detectors tend to start generating false positives and detecting thicker lines, whereas optimal filtering detectors like Canny tend to start generating false negatives and detecting broken edges.

6.5.2 Varying the Aperture Setting

In Figures 6.12–6.14, I present the results of varying the aperture setting. I captured three sets of images for the third global measure of coherence, one using an F1.8 aperture, one using an F2.8 aperture, and one using an F4.0 aperture. I estimated the third global measure of coherence for all three sets of images. Figure 6.12

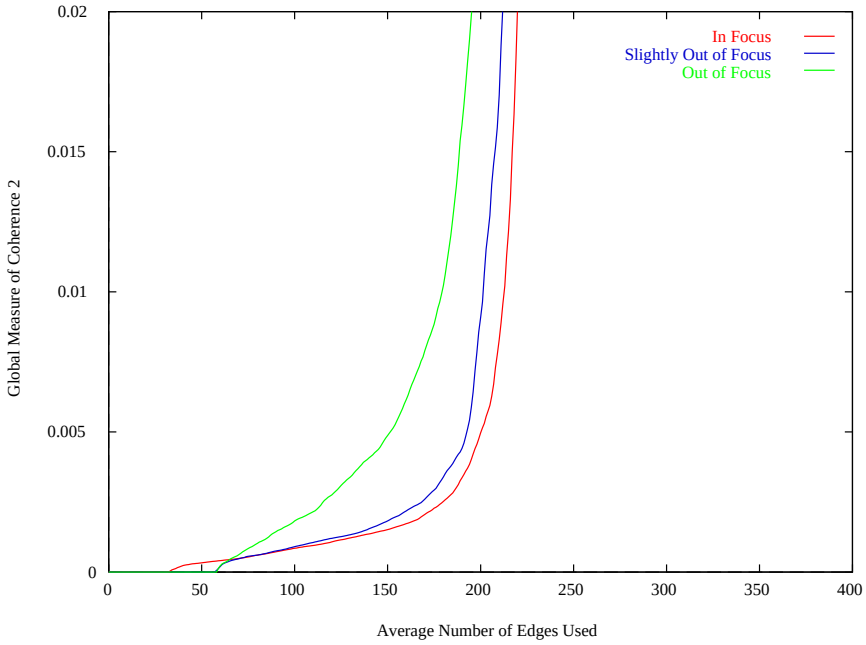


Figure 6.11: The results of varying the focus setting for the Canny detector. Just as for the other detectors the performance gets worse as the images get more defocused. However, unlike the other detectors, less edges are detected in the more defocused images.

contains the results for the parametric manifold detector, Figure 6.13 the results for the Nalwa-Binford detector, and Figure 6.14 the results for the Canny detector.

As one would expect, the performance of all three detectors gets worse as the aperture gets smaller. Just as for the focus setting, the drop off in performance is slightly worse for the Canny detector than it is for the other two detectors. However, for the parametric manifold detector, the number of edges at which the detector enters the third rapid growth phase stays roughly constant. On the other hand, for the other two detectors an increasing number of edges are missed as the aperture decreases. The reason the parametric manifold algorithm does not suffer from an increasing number of false negatives is probably the parameter normalization of Section 3.2.4. This normalization makes the detection decision largely independent

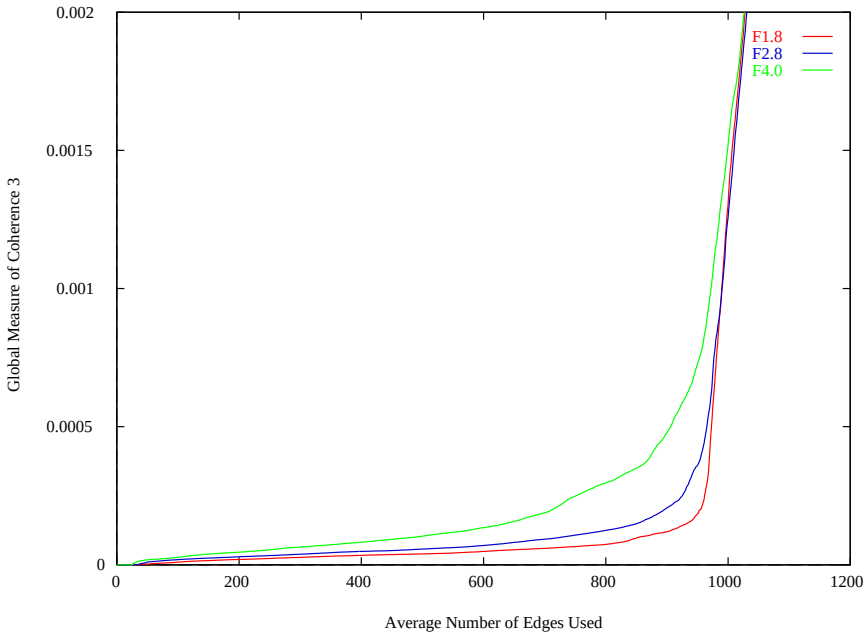


Figure 6.12: The results of varying the aperture setting for the parametric manifold detector. As the aperture gets smaller the performance gets worse, however the number of detected edges stays almost constant.

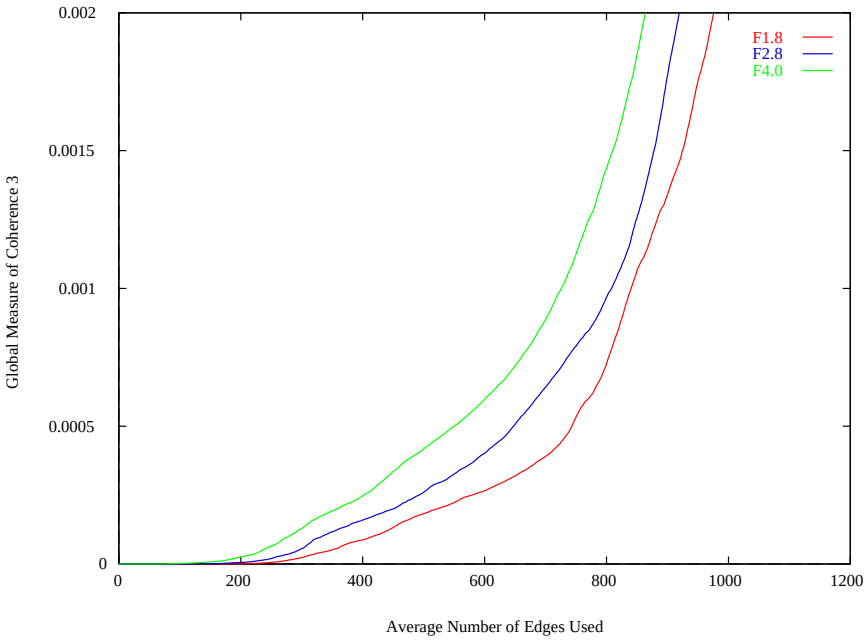


Figure 6.13: The results of varying the aperture setting for the Nalwa-Binford detector. As in Figure 6.12, the performance gets worse as the aperture gets smaller. In addition, for smaller apertures there are an increasing number of false negatives.

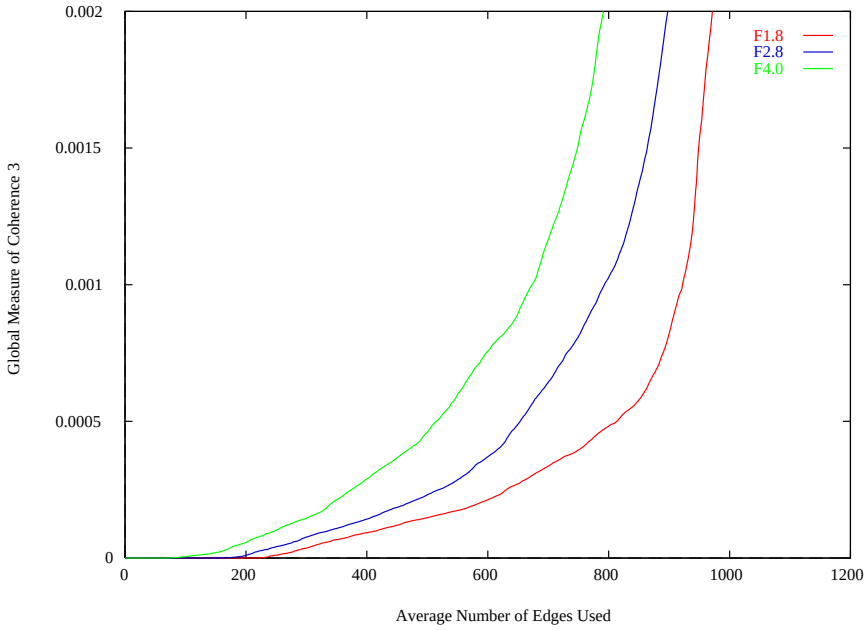


Figure 6.14: The results of varying the aperture setting for the Canny detector. As for the other detectors, the performance gets worse as the aperture gets smaller. Just as for the Nalwa-Binford detector, the number of false negatives also goes up.

of the size of the step, and hence the size of the aperture.

6.6 Discussion

In this chapter, I presented global measures of coherence as a method of benchmarking an edge detector for applications which require precise sub-pixel localization and orientation estimation. Examples of such applications include the Hough transform, structure from motion, industrial inspection, the computation of projective invariants, and stereo matching. The results show that global measures of coherence can clearly discriminate a wide range of different performance levels.

The major weakness of global measures of coherence is that the scenes used to capture the images contain carefully constructed man made objects. Although

they are somewhat representative of certain industrial environments, the scenes are very simple. However, it should be noted that defining what is, and moreover what is not, an edge in an image of a natural object such as a tree or a human face is very difficult, and arguably inherently subjective anyway. In fact, in a recent paper in which edge detectors are evaluated by getting a human to mark the edges in an image by hand, Dougherty and Bowyer allowed the human to mask out certain regions as too difficult for the human to say which pixels contain edges [32]. The three major advantages of using global measures of coherence are:

1. They use a very large number of real images. Although the benchmarks used in this chapter only contain between seventy-five and two hundred images, capturing an order of magnitude more images would be easy, although somewhat time consuming.

2. They can be used to measure how detection performance degrades as imaging conditions and physical properties of the scene change. This possibility could lead to important insights into what physical causes make edge detection difficult. This understanding in turn might lead to better edge models and noise distributions that could then be used to design better detectors. (Note that although my benchmarks only contained images of scenes with reflectance, depth, and surface normal discontinuity edges, it would be very easy to create similar scenes containing shadow edges, edges created when the surface normal turns away from a light source, and other physical causes of edges.)

3. They can be applied to any edge detector very easily. Applying the benchmarks consists of three simple steps: (1) download the images and compile my

code, (2) modify the detector to output the edge map in the correct format, and (3) run a script with the detector name as its first argument. This final point is perhaps the most significant. For a benchmark to be used, it has to be possible to obtain it freely, use it without significant effort, and the results of competing algorithms on the same benchmark must be available. My benchmarks are already available over the Internet with performance graphs for the five detectors I tested. In the future, I hope to obtain implementations of other detectors to test with my global measures of coherence, the results of which I will also make available.

Chapter 7

Conclusion

7.1 Summary of Contributions

In Chapter 3, I developed a feature detection algorithm applicable to arbitrary parametric features. This algorithm offers a level of generality that is uncommon in feature detection. As far as possible to ascertain, there is no other technique that is capable of detecting the five features considered in Chapter 3. Moreover, the construction of a new detector just consists of two simple steps: (1) writing a C/C++ function to define the continuous feature model, and (2) compiling the new feature model and linking it with the existing implementation. Features could also be constructed from imaged features by writing a defining function that appropriately transforms (e.g. interpolates, scales, rotates, shifts, and blurs) the image data. A similar approach to object recognition was recently proposed by Krumm [62]. Finally, note that although I have only considered features in visible light images, the same approach is directly applicable to any other sensing modality, including, X-ray, MR, infrared, ultrasound, and range.

A second major contribution of Chapter 3 is the use of realistic multi-parameter feature models and the incorporation of an explicit sensor model. As discussed in Section 2.6.2, careful modeling of sensing and optical effects is necessary to maximize feature detection robustness and obtain the best possible estimates of the parameters. In this respect, the approach of Chapter 3 should be contrasted with previous feature detectors which typically use relatively simple feature models and completely ignore sensing effects. Such simplified models do not capture the full variation in the appearance of imaged features. Hence, the performance of the resulting detectors is sub-optimal.

In Chapter 5, I investigated the choice of the matching function for the feature detector described in Chapter 3. In particular, I restricted attention to weighted L^2 norms and presented a general framework for the selection of the weighting function. I proposed optimality criteria for the three key aspects of feature detection performance and showed how they can be combined to yield optimality criteria for specific applications. I derived an approximate expression for the optimal weighting function for the parameter estimation accuracy criterion and suggested a numerical algorithm for the optimization of the other criteria. This algorithm can be used to compute the optimal weighting function for any of my optimality criteria and for arbitrary parametric features.

In Chapter 6, I proposed a collection of edge detector benchmarks appropriate for applications requiring precise sub-pixel localization and orientation estimation, such as the Hough transform, stereo matching, and the computation of projective invariants. Each benchmark consists of a very large number of real images, albeit of carefully constructed scenes in the laboratory. These benchmarks

yield non-subjective performance metrics, but do not require ground truth data. Besides being useful for assessing the relative performance of edge detectors, these benchmarks can also be used to study how feature detection performance degrades as camera parameters, such as the focus and aperture settings, vary.

7.2 Discussion

The most important requirement when designing a feature detector is a specification of the intensity distributions that constitute the feature. There are a number of ways of providing this information. One commonly used method is through an explicit feature model, the approach I took in Chapter 3. Other ways define the distributions implicitly, for example, using differential invariants or optimality criteria, as in Sections 2.2 and 2.3. Another possibility might be to define the distributions empirically by giving a large number of example features.

The same statement is equally true when evaluating a feature detector: the key requirement is a specification of the intensity distributions that constitute the feature. Again, this information can be provided in several ways. One way is using an explicit feature model, as I did in Section 4.1. Another method is to provide a large number of example features, usually embedded in a collection of images, as was done in both Section 4.2 and Chapter 6.

An underlying theme of this thesis has also been an investigation into how to specify the intensity distributions that constitute a feature, both when designing and evaluating a feature detector. In this respect, several points should be noted:

- In Chapter 3, I proposed a feature detection algorithm applicable to essen-

tially any feature. Since the most important goal was generality, I used explicit parametric feature models to specify the features. With most implicit methods it would have been very difficult, if possible at all, to provide specifications for arbitrary features. For this reason, differential invariants and optimality criteria were immediately ruled out as possibilities.

- Just specifying the intensity distributions that constitute an ideal feature is not enough, on its own, either to define or evaluate a feature detector. In addition, a description of the distributions that do not constitute the feature is also needed. This information can be provided explicitly in the form of a non-feature model, as was done in Section 4.1 and Chapter 5, or in the form of a large number of example non-features. As above, example non-features are usually embedded in a collection of images, particularly when evaluating a detector. This was the approach taken in both Section 4.2 and Chapter 6. Alternatively, it is possible to specify the non-features implicitly using either an optimality criterion, as in Section 2.3, or a function that measures the distance from ideal. See Chapter 5 for more discussion of this point.

- In Chapter 6, I studied the evaluation of step edge detectors. Using explicit models of features and non-features for detector evaluation is problematic. As discussed in Section 4.1.1 and Section 5.5, the selection of appropriate feature models, non-feature models, and noise models is difficult to do without biasing the comparison. Ideally one would like to use example features and non-features taken from real images to avoid these issues. As discussed in Section 2.5.2, doing so is unfortunately problematic because deciding which pixels in an image exhibit a feature is tedious, error prone, and inherently

subjective. In Chapter 6, I proposed a class of evaluation benchmarks that avoid this problem by using images for which a constraint is known on the feature instances. The detectors are evaluated using the extent to which this constraint holds in the output edge map, rather than in terms of the extent to which individual features are detected or not.

7.3 Future Work

This thesis suggests a number of possibilities for future work. Here, I just mention two of them. In Section 7.3.1, I discuss the possibility of using the output of multiple feature detectors as input into a multi-feature aggregation algorithm. In Section 7.3.2, I describe how my global measures of coherence might be used to study which physical effects actually make edge detection difficult.

7.3.1 Multi-Feature Aggregation

Simple post-processing can dramatically improve the performance of a feature detector. A number of post-processing and feature aggregation algorithms have been proposed in the literature, perhaps the most well known being Canny's adaptive thresholding technique, hysteresis [20]. In this thesis, I focused exclusively on how well feature detection can be performed without using such techniques. In particular, I assumed that the decision of whether to detect a feature is based solely upon the distribution of the pixel intensity values in the feature window, and independently of whether features are detected in neighboring windows.

Existing post-processing and feature aggregation algorithms, such as relax-

ation [108], typically assume that a single feature detector has been applied to the image. Not surprisingly, the feature detector is usually a step edge detector. The results of applying the five detectors proposed in Section 3.3 not only consists of detected features, but also estimates of their parameters and a measure of how well the image data fits the feature model in terms of the distance to the closest point on the manifold. This yields a huge amount of information that might be valuable to a higher level multi-feature aggregation algorithm.

There are at least two effects that such an algorithm should take advantage of: (1) Powerful constraints result from the use of multiple feature detectors. For instance, a corner cannot exist in isolation, but instead must have edges in the vicinity. (2) There are inherent correlations between the responses of the feature detectors. For example, a corner with an angle above around 150° also gives a weak response as a step edge. Similarly, a line that is wide enough should also gives a weak responses as a pair of parallel step edges. This correlation between detector responses is clearly demonstrated in Figure 4.14. The incorporation of these two effects into a multi-feature aggregation algorithm should lead to much improved performance over algorithms that only use a single feature.

7.3.2 Investigation into the Physical Causes of Edges

In Section 6.5, I demonstrated how global measures of coherence can be used to investigate how quickly detector performance degrades as camera parameters are altered. In particular, I varied both the focus and aperture settings. A large number of other experiments could have been performed. For example, the material properties of the objects in the scene could have been changed, the sharpness of

the surface normal discontinuities could have been reduced, and additional camera parameters such as the zoom setting could have been varied.

Conducting these experiments may lead to far greater understanding of which physical effects actually make edge detection difficult. Many important questions might be answered: At what point do low contrast edges become undetectable? Are depth discontinuity edges fundamentally easier to detect than surface normal discontinuities? What about the other types of discontinuities and causes of edges? How big an effect do material and reflectance properties have on performance? How robust can edge detection ever be to the setting of camera parameters?

The understanding gained by conducting such experiments into the physical causes of edges should hopefully lead to more realistic edge models, physically validated noise distributions, and eventually better edge detectors.

Bibliography

[1] I.E. Abdou and W.K. Pratt. Quantitative design and evaluation of enhancement/thresholding edge detectors. *Proceedings of the IEEE*, 67(5):753–763, May 1979.

[2] J.F. Abramatic. Why the simplest "Hueckel" edge detector is a Roberts operator. *Computer Graphics and Image Processing*, 17:79–83, 1981.

[3] A.C. Aitken. On least squares and linear combinations of observations. *Proceedings of the Royal Society of Edinburgh A*, 55:42–47, 1934.

[4] S. Baker and S.K. Nayar. Algorithms for pattern rejection. In *Proceedings of the 13th International Conference on Pattern Recognition*, volume II Track B, pages 869–874, Vienna, Austria, August 1996. IAPR.

[5] S. Baker and S.K. Nayar. Pattern rejection. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition*, pages 544–549, San Francisco, California, June 1996. IEEE Computer Society.

[6] S. Baker, S.K. Nayar, and H. Murase. Parametric feature detection. *International Journal of Computer Vision*, 27(1):27–50, 1998.

[7] D.F. Barbe. *Charge-Coupled Devices*. Springer-Verlag, 1980.

[8] P.R. Beaudet. Rotational invariant image operators. In *Proceedings of the 4th International Conference on Pattern Recognition*, pages 579–583, Tokyo, Japan, 1978.

[9] S. Becker and Jr. V.M. Bove. Semiautomatic 3-D model extraction from uncalibrated 2-D camera views. In *Proceedings of SPIE Visual Data Exploration and Analysis II*, volume 2410, pages 447–461, San Jose, California, February 1995.

[10] F. Bergholm. Edge focusing. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(6):726–741, November 1987.

[11] V. Berzins. Accuracy of Laplacian edge detectors. *Computer Vision, Graphics, and Image Processing*, 27:195–210, 1984.

[12] R.A. Boie, I.J. Cox, and P. Rehak. On optimum edge recognition using matched filters. In *Proceedings of the 1986 Conference on Computer Vision and Pattern Recognition*, pages 100–108, 1986.

[13] M. Born and E. Wolf. *Principles of Optics*. Permagon Press, 1965.

[14] A.C. Bovik, T.S. Huang, and D.C. Munson Jr. Nonparametric tests for edge detection in noise. *Pattern Recognition*, 19(3):209–219, 1986.

[15] R.N. Bracewell. *The Fourier Transform and Its Applications*. McGraw Hill, second edition edition, 1978.

[16] M. Brady. Computational approaches to image understanding. *Computing Surveys*, 14(1):2–71, March 1982.

[17] M.J. Brooks. Rationalizing edge detectors. *Computer Graphics and Image Processing*, 8:277–285, 1978.

[18] D.C. Brown. Close-range camera calibration. In *Symposium on close-range photogrametry*, Urbana, Illinois, January 1971.

[19] D.J. Bryant and D.W. Bouldin. Evaluation of edge operators using relative and absolute grading. In *Proceedings of the IEEE Conference on Pattern Recognition and Image Processing*, pages 138–145, Chicago, IL, 1979.

[20] J. Canny. A computational approach to edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, November 1986.

[21] S. Castan, J. Zhao, and J. Shen. New edge detection methods based on exponential filter. In *Proceedings of the 10th International Conference on Pattern Recognition*, pages 709–711, 1990.

[22] J.S. Chen, A. Huertas, and G. Medioni. Fast convolution with Laplacian-of-Gaussian masks. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(4):584–590, July 1987.

[23] K. Cho, P. Meer, and J. Cabrera. Performance assessment through bootstrap. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1185–1198, November 1997.

[24] C. Coehlo, A. Heller, J.L. Mundy, D.A. Forsyth, and A. Zisserman. An experimental evaluation of projective invariants. In J.L Mundy and A. Zisserman,

editors, *Geometric Invariants for Machine Vision*, chapter 4, pages 87–104. MIT Press, 1992.

[25] J.B. Conway. *A Course in Functional Analysis*. Springer-Verlag, 1985.

[26] L.S. Davis. A survey of edge detection techniques. *Computer Graphics and Image Processing*, 4:248–270, 1975.

[27] D. Demigny and T. Kamlé. A discrete expression for Canny's criteria for step edge detector performance evaluation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1199–1211, November 1997.

[28] R. Deriche. Optimal edge detection using recursive filtering. In *Proceedings of the First International Conference on Computer Vision*, pages 501–505, 1987.

[29] R. Deriche. Using Canny's criteria to derive a recursively implemented optimal edge detector. *International Journal of Computer Vision*, 1:167–187, 1987.

[30] R. Deriche and G. Giraudon. A computational approach for corner and vertex detection. *International Journal of Computer Vision*, 10(2):101–124, 1993.

[31] E.S. Deutsch and J.R. Fram. A quantitative study of the orientation bias of some edge detector schemes. *IEEE Transactions on Computers*, 27(3):205–213, March 1978.

[32] S. Dougherty and K.W. Bowyer. Objective evaluation of edge detectors using a formally defined framework. In *Proceedings of the 1998 Workshop on*

Empirical Evaluation Techniques in Computer Vision, pages 211–234, Santa Barbara, California, June 1998. IEEE Computer Society.

[33] L. Dreschler and H.H. Nagel. On the selection of critical points and local curvature extrema of region boundaries for interframe matching. In *Proceedings of the 6th International Conference on Pattern Recognition*, pages 542–544, 1982.

[34] J. Elder and S. Zucker. Scale space localization blur and contour-based image coding. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition*, pages 27–34, San Francisco, California, June 1996. IEEE Computer Society.

[35] O.D. Faugeras. *Three–dimensional Computer Vision: A Geometric Viewpoint*. MIT Press, 1993.

[36] J.R. Fram and E.S. Deutsch. On the quantitative evaluation of edge detection schemes and their comparison with human performance. *IEEE Transactions on Computers*, 24(6):616–628, June 1975.

[37] W.T. Freeman and E.H. Adelson. The design and use of steerable filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13:891–906, 1991.

[38] K. Fukunaga. *Introduction to Statistical Pattern Recognition*. Academic Press, 1990.

[39] M. Gennert. Detecting half-edges and vertices in images. In *Proceedings of the 1986 Conference on Computer Vision and Pattern Recognition*, pages 552–557, 1986.

[40] A.K. Griffith. Mathematical models for automatic line detection. *Journal of the Association for Computing Machinery*, 20(1):62–80, January 1973.

[41] M. Hahsimoto and J. Sklansky. Multiple-order derivatives for detecting local image characteristics. *Computer Vision, Graphics, and Image Processing*, 39:28–55, 1987.

[42] R.M. Haralick. Edge and region analysis for digital image data. *Computer Graphics and Image Processing*, 12:60–73, 1980.

[43] R.M. Haralick. Ridges and valleys on digital images. *Computer Vision, Graphics, and Image Processing*, 22:28–38, 1983.

[44] R.M. Haralick. Digital step edges from zero crossing of second directional derivatives. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6(1):58–68, January 1984.

[45] C. Harris. Determination of ego-motion from matched points. In *Proceddings of the 3rd Alvey Vision Conference*, Cambridge, UK, 1987.

[46] R. Hartley. A Gaussian-weighted multiresolution edge detector. *Computer Vision, Graphics, and Image Processing*, 30:70–83, 1985.

[47] M.D. Heath, S. Sarkar, T. Sanocki, and K.W. Bowyer. A robust visual method for assessing the relative performance of edge-detection algorithms. *IEEE*

Transactions on Pattern Analysis and Machine Intelligence, 19(12):1338–1359, December 1997.

[48] B.K.P. Horn. *Robot Vision*. McGraw Hill, 1996.

[49] J.S. Huang and D.H. Tseng. Statistical theory of edge detection. *Computer Vision, Graphics, and Image Processing*, 43:337–346, 1988.

[50] M.H. Hueckel. An operator which locates edges in digitized pictures. *Journal of the Association for Computing Machinery*, 18(1):113–125, January 1971.

[51] M.H. Hueckel. A local visual operator which recognizes edges and lines. *Journal of the Association for Computing Machinery*, 20(4):634–647, October 1973.

[52] R.A. Hummel. Feature detection using basis functions. *Computer Graphics and Image Processing*, 9:40–55, 1979.

[53] B.F. Logan Jr. Information in the zero crossings of bandpass signals. *Bell Systems Technical Journal*, 56(4):487–510, April 1977.

[54] G.E. Sotak Jr. and K.L. Boyer. The Laplacian-of-Gaussian kernal: A formal analysis and design procedure for fast, accurate convolution and full-frame output. *Computer Vision, Graphics, and Image Processing*, 48:147–189, 1989.

[55] G. Kanizsa. Subjective contours. *Scientific American*, 234(4):48–52, 1976.

[56] T. Kanungo, M.Y. Jaisimha, J. Palmer, and R.M. Haralick. A methodology for quantitative performance evaluation of detection algorithms. *IEEE Transactions on Image Processing*, 4(12):1667–1673, December 1995.

[57] L. Kitchen and A. Rosenfeld. Edge evaluation using local edge coherence. *IEEE Transactions on Systems, Man, and Cybernetics*, 11(9):597–605, September 1981.

[58] L. Kitchen and A. Rosenfeld. Gray-level corner detection. *Pattern Recognition Letters*, 1:95–102, December 1982.

[59] D.E. Knuth. *The Art of Computer Programming, Volume II: Seminumerical Algorithms*. Addison-Wesley, 1981.

[60] J.J. Koenderink. The structure of images. *Biological Cybernetics*, 50:363–370, 1984.

[61] A.L. Korn. Towards a symbolic representation of intensity changes in images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 10(5):610–625, September 1988.

[62] J. Krumm. Eigenfeatures for planar pose measurement of partially occluded objects. In *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition*, San Francisco, California, June 1996. IEEE Computer Society.

[63] C.L. Lawson and R.J. Hanson. *Solving Least Squares Problems*. Prentice-Hall, 1974.

[64] R. Lenz. Optimal filters for the detection of linear patterns in 2-D and higher dimensional image. *Pattern Recognition*, 20(2):163–172, 1987.

[65] T. Lindenberg. *Scale-Space Theory in Computer Vision*. Kluwer Academic Publishers, 1994.

[66] W.H.H. Lunscher. The assymptotic optimal frequency domain filter for edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 5(6):678–680, November 1983.

[67] W.H.H.J. Lunscher and M.P. Beddoes. Optimal edge detector design 1: Parameter selection and noise effects. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):164–177, March 1986.

[68] W.H.H.J. Lunscher and M.P. Beddoes. Optimal edge detector design 2: Coefficient quantization. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):178–187, March 1986.

[69] S. Mallat and S. Zhong. Characterization of signals from multiscale edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 14(7):710–732, July 1992.

[70] D. Marr and E. Hildreth. Theory of edge detection. *Proceedings of the Royal Society of London, Series B*, 207:187–217, 1980.

[71] P. Meer and I. Weiss. Smoothed differentiation filters for images. *Journal of Visual Communication and Image Representation*, 3(1):58–72, March 1992.

[72] J.W. Modestino and R.W. Fries. Edge detection in noisy images using recursive digital filtering. *Computer Graphics and Image Processing*, 6:409–433, 1977.

[73] H.P. Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the Fifth Internation Joint Conference on Artificial Intelligence*, 1977.

[74] D.G. Morgenthaler. A new hybrid edge detector. *Computer Graphics and Image Processing*, 16:166–176, 1981.

[75] D.G. Morgenthaler and A. Rosenfeld. Multidimensional edge detection by hypersurface fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(4):482–486, July 1981.

[76] H.H. Nagel. Displacement vectors derived from second-order intensity variations in image sequences. *Computer Vision, Graphics, and Image Processing*, 21:85–117, 1983.

[77] N.E. Nahi and M.H. Jahanshahi. Image boundary estimation. *IEEE Transactions on Computers*, 26(8):772–781, August 1977.

[78] V.S. Nalwa. Edge detector resolution improvement by image interpolation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9(3):446–451, May 1987.

[79] V.S. Nalwa. *A Guided Tour of Computer Vision*. Addison-Wesley, 1993.

[80] V.S. Nalwa and T.O. Binford. On detecting edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(6):699–814, November 1986.

[81] D. Nandy, Z. Wang, J. Ben-Arie, K. Raghunath Rao, and N. Jojic. A generalized feature extractor using expansion matching and the karhunen-loeve transform. In *Proceedings of the 1996 DARPA Image Understanding Workshop*, pages 969–972, Palm Springs, CA, February 1996.

[82] R. Nevatia. Evaluation of a simplified Hueckel edge-line detector. *Computer Graphics and Image Processing*, 6:582–588, 1977.

[83] R. Nevatia and K.R. Babu. Linear feature extraction and description. *Computer Graphics and Image Processing*, 13:257–269, 1980.

[84] J.A. Nobel. Finding corners. *Image and Vision Computing*, 6(2):121–127, May 1988.

[85] H.N. Norton. *Sensor and Analyzer Handbook*. Prentice Hall, 1982.

[86] The Museum of Modern Art. *The Museum of Modern Art New York: The History and the Collection*. Harry N. Abrams, 1984.

[87] American Society of Photogrammetry. *Manual of Photogrammetry*. American Society of Photogrammetry, fourth edition edition, 1980.

[88] F. O'Gorman. Edge detection using Walsh functions. *Artificial Intelligence*, 10:215–223, 1978.

[89] E. Oja. *Subspace Methods of Pattern Recognition*. Research Studies Press, 1983.

[90] P.L. Palmer, H. Dabis, and J. Kittler. A performance measure for boundary detection algorithms. *Computer Vision and Image Understanding*, 63(3):476–494, May 1996.

[91] L. Parida, D. Geiger, and R. Hummel. Kona: A mulie-junction detector and classifier. In *Proceedings of the International Conference on Energy Minimization in Computer Vision and Pattern Recognition*, Venice, Italy, 1997.

[92] K. Paton. Picture description using Legendre polynomials. *Computer Graphics and Image Processing*, 4:40–54, 1975.

[93] D.A. Patterson and J.L. Hennessy. *Computer Architecture: A Quantitive Approach*. Morgan Kaufman, San Mateo, California, 1990.

[94] F. Pedersini, A. Sarti, and S. Tubaro. Estimation and compensation of subpixel edge localization error. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(11):1278–1284, November 1997.

[95] T. Peli and D. Malah. A study of edge detectional algorithms. *Computer Graphics and Image Processing*, 20:1–21, 1982.

[96] P. Perona and J. Malik. Scale-space and edge detection using anisotropic diffusion. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(7):629–639, July 1990.

[97] M. Petrou and J. Kittler. Optimal edge detectors for ramp edges. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(5):483–491, May 1991.

[98] K.K. Pingle. Visual perception by a computer. In A. Grasselli, editor, *Automatic Interpretation and Classification of Images*, pages 277–284. Academic Press, New York, 1969.

[99] R.L. Plackett. *Principles of Regression Analysis*. Oxford Univeristy Press, 1960.

[100] W.K. Pratt. *Digital Image Processing*. Wiley-Interscience, 1991.

[101] W.H. Press, S.A. Teukolsky, W.T. Vetterling, and B.P. Flannery. *Numerical Recipes in C*. Cambridge University Press, second edition, 1992.

[102] J.M. Prewitt. Object enhancement and extraction. In B.S. Lipkin and A. Rosenfeld, editors, *Picture Processing and Psychopictorics*. Academic Press, 1970.

[103] V. Ramesh and R.M. Haralick. Performance characterization of edge detectors. *SPIE Applications of Artificial Intelligence: Machine Vision and Robotics*, 1708:252–266, 1992.

[104] K. Raghunath Rao and J. Ben-Arie. Optimal edge detection using expansion matching and restoration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(12):1169–1182, December 1994.

[105] L.G. Roberts. Machine perception of three-dimensional solids. In J.T. Trippit, D.A. Berkowitz, L.C. Chapp, C.J. Koester, and A. Vanderburgh, editors, *Optical and Electro-Optical Information Processing*, pages 159–197. MIT Press, Cambridge, Massachusetts, 1965.

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

[107] A. Rosenfeld. The max Roberts operator is a Hueckel-type edge detector. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(1):101–103, January 1981.

[108] A. Rosenfeld, R.A. Hummel, and S.W. Zucker. Scene labeling by relaxation operations. *IEEE Transactions on Systems, Man, and Cybernetics*, 6:420–433, 1976.

[109] S. Sarkar and K.L. Boyer. On optimal infinite impulse response edge detection filters. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 13(11):1154–1171, November 1991.

[110] M.A. Shah and R. Jain. Detecting time-varying corners. *Computer Vision, Graphics, and Image Processing*, 28:345–355, 1984.

[111] K.S. Shanmugam, F.M. Dickey, and J.A. Green. An optimal frequency domain filter for edge detection in digital pictures. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1(1):37–49, January 1979.

[112] M.C. Shin, D. Goldgof, and K.W. Bowyer. An objective comparison methodology of edge detection algorithms using a structure from motion task. In *Proceedings of the 1998 Workshop on Empirical Evaluation Techniques in Computer Vision*, pages 235–254, Santa Barbara, California, June 1998. IEEE Computer Society.

[113] L.A. Spacek. Edge detection and motion estimation. *Image and Vision Computing*, 4:43–56, 1986.

[114] C. Steeger. Analytical and empirical performance evaluation of subpixel line and edge detection. In *Proceedings of the 1998 Workshop on Empirical Evaluation Techniques in Computer Vision*, pages 188–210, Santa Barbara, California, June 1998. IEEE Computer Society.

[115] V. Torre and T.A. Poggio. On edge detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8(2):147–163, March 1986.

[116] R.Y. Tsai. An efficient and accurate camera callibration technique for 3D machine vision. In *Proceedings of the 1986 Conference on Computer Vision and Pattern Recognition*, pages 364–374, 1986.

[117] I. Weiss. High-order differential filters that work. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 16(7):734–739, July 1994.

[118] A.P. Witken. Scale-space filtering. In *Proceedings of the 8th International Joint Conference on Artificial Intelligence*, pages 1019–1022, Karlsruhe, Germany, August 1983.

[119] P.N. Yianilos. Data structures and algorithms for nearest neighbor search in general metric spaces. In *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms*, 1993.

[120] A.L. Yuille and T.A. Poggio. Scaling theorems for zero crossings. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 1986.

[121] Q. Zhu. Efficient evaluation of edge connectivity and width uniformity. *Image and Vision Computing*, 14:21–34, 1996.

[122] S.W. Zucker and R.A. Hummel. A three-dimensional edge operator. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 3(3):324–331, May 1981.

[123] O.A. Zuniga and R.M. Haralick. Corner detection using the facet model. In *Proceedings of the 1983 Conference on Computer Vision and Pattern Recognition*, pages 30–37. IEEE Computer Society, 1983.