Parametric Feature Detection

Simon Baker[†], Shree K. Nayar[†], and Hiroshi Murase[‡]

[†]Department of Computer Science
Columbia University
New York, NY 10027, U.S.A.
Email: {simonb,nayar}@cs.columbia.edu

[‡]NTT Basic Research Laboratory
Morinosato Wakamiya, Atsugi-shi
Kanagawa 243-01, Japan
Email: murase@siva.ntt.jp

Abstract

Most visual features are parametric in nature, including, edges, lines, corners, and junctions. We propose an algorithm to automatically construct detectors for arbitrary parametric features. To maximize robustness we use realistic multi-parameter feature models and incorporate optical and sensing effects. Each feature is represented as a densely sampled parametric manifold in a low dimensional subspace of a Hilbert space. During detection, the vector of intensity values in a window about each pixel in the image is projected into the subspace. If the projection lies sufficiently close to the feature manifold, the feature is detected and the location of the closest manifold point yields the feature parameters. The concepts of parameter reduction by normalization, dimension reduction, pattern rejection, and heuristic search are all employed to achieve the required efficiency. Detectors have been constructed for five features, namely, step edge (five parameters), roof edge (five parameters), line (six parameters), corner (five parameters), and circular disc (six parameters). The results of detailed experiments are presented which demonstrate the robustness of feature detection and the accuracy of parameter estimation.

Index Terms: Feature detection, parametric features, feature modeling, optical effects, sensor effects, feature manifolds, parameter normalization, dimension reduction, nearest neighbor search, step edges, roof edges, corners, lines, circular discs, relaxation.

1 Introduction

Many applications in image processing and computational vision rely upon the robust detection of parametric image features. The standard example of a parametric feature is the *step edge*. It is by far the most frequently studied feature due to its abundance in natural scenes, its high information content, and the fact that its simple one-dimensional structure makes analysis tractable. Nevertheless, the step edge is by no means the only feature of interest in image understanding. It is closely followed in significance by other ubiquitous features such as *lines, corners, junctions,* and *roof edges*[1]. This list is far from comprehensive, even if we restrict attention to features that can be defined analytically. Moreover, in any given application, the term feature may 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 bolt may be the feature of interest. The appearance of such a feature in an image may well depend upon a number of parameters such as orientation, localization, scale, and level of blurring. In short, parametric features are too numerous to justify the process of manually deriving a detector for each one.

The objective of this paper is to develop an algorithm that automatically constructs a feature detector for an arbitrary parametric feature. Further, during detection we also wish to recover the parameters of detected features and, in particular, do so with as high accuracy as possible. In many applications, precise estimates of feature parameters are of vital importance to higher levels of visual processing. A simple example is that of the generalized Hough transform where accurate knowledge of edge direction reduces the dimension of the Hough space by one. Likewise, the performance of boundary growing algorithms can be dramatically enhanced when the orientation of image edgels is used to guide the growth of the boundary.

To obtain high performance in both detection and parameter estimation, it is essential to accurately model the features as they appear in the physical world. Hence, we choose not

[1]Given the extent to which feature detection has been studied, a complete survey of previous work is well beyond the scope of this paper. In our discussion we will provide examples of existing detectors without attempting to mention all of them. Further, we focus on detectors that fit parametric feature models to the image intensities rather than detectors based on the gradient (e.g. [Prewitt 70]), Laplacian (e.g. [Marr and Hildreth 80]), second directional derivatives (e.g. [Canny 86] [Haralick 84]), Hessian determinant (e.g. [Deriche and Giraudon 93]), or other differential invariants. A concise description of one-dimensional image features (e.g. step edges, roof edges, and lines) and a survey of step edge detectors can be found in [Nalwa 93]. The papers by Rohr [Rohr 92] and Deriche and Giraudon [Deriche and Giraudon 93] cover most of the previous work concerning the two-dimensional problems of corner detection and junction detection.

to make any simplifications for analytic or efficiency reasons, but instead use realistic multi-parameter feature models. Whereas many step edge models assume that the edge passes directly through the center of the pixel and is a perfect step discontinuity, we include a 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. Our model of the step edge has five parameters, namely, the lower brightness level, the brightness difference across the step, the angle (orientation) of the edge, the intrapixel location, and the blurring (scaling) parameter. Our arguments in favor of highly descriptive feature models apply to other features as well. We use a five parameter model for roof edges, a six parameter model for lines, a five parameter model for corners, and a six parameter model for circular discs.

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. We argue that to fully optimize the performance of a detector, careful consideration must be given to how the sensor converts the continuous radiance function of a scene feature into its discrete image. For instance, the aspect ratio of an image sensor may significantly affect the appearance of a feature in an image. Perhaps less obvious are the effects of the shape and size of the photosensitive elements within a CCD image sensor. Our notion of a parametric feature model is a continuous one, but during detector construction we explicitly model the discretization of the sensor. The sensor model used is that of a standard CCD imaging device which integrates the radiance function over a sub-rectangle of each sensor pixel. The sub-rectangle corresponds to the pixel photosensitive area, which in general is not the entire pixel. In addition to the sensing discretization, we also model the blurring caused by the optical transfer function of the imaging optics.

When combined, a parametric feature model and an imaging system model allow us to accurately predict the pixel intensity values in a window around an imaged feature. All that is required are the parameters of the feature and the details of the imaging system. If we treat the pixel intensity values as real numbers, we can regard each parametric feature instance as a point in $\Re^N$, where N is the number of pixels in the window surrounding the feature. As the feature parameters vary, the point in $\Re^N$ 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 $\Re^N$ corresponding to the pixel intensity values in a novel image window. If the

closest manifold point is near enough, we register the presence of the feature. Then, 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 from the novel point, we declare the absence of the feature. This statement of the feature detection problem was first introduced by Hueckel [Hueckel 71], and was subsequently used by O'Gorman [O'Gorman 78], Hummel [Hummel 79], Hartley [Hartley 85], and Nalwa and Binford [Nalwa and Binford 86] for the detection of step edges. Hueckel [Hueckel 73] applied the same formulation to line detection and Rohr [Rohr 92] used it to detect corners. The same approach generalizes to three-dimensional image data as was used by Zucker and Hummel [Zucker and Hummel 81] and also by Lenz [Lenz 87] in the detection of three-dimensional step edges.

Hueckel [Hueckel 71] and Hummel [Hummel 79] both argued that to achieve the required efficiency, a closed form solution must be found for the parameters of the closest manifold point. To make their derivations possible, they used simplified feature models and neglected sensing effects. Our view of feature detection is radically different. We believe that the features we wish to detect are inherently complex visual entities. Hence, we willingly forego all hope of finding closed-form solutions for the best-fit parameters. Instead, we discretize the search problem by densely sampling the feature manifold. The closest point on the manifold is then approximated by finding the nearest neighbor amongst the sample points. Typically, this 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 window (centered around each pixel) in the image. Nalwa and Binford [Nalwa and Binford 86] and Rohr [Rohr 92] used more complex feature models than Hueckel and Hummel and also used numerical methods to find the best-fit parameters of their models to 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, we will show that our approach is indeed very practical. To obtain the required efficiency we used a number of different techniques. First, we introduce a set of simple normalizations that eliminate some of the parameters and so reduce the dimensionality of the manifold to 3 or 4 (for the five features which we experimented with). These normalizations cause no significant loss of information or reduction in the signal-to-noise ratio. Next, we apply the Karhunen-Loéve expansion [Oja 83], as a dimension reduction technique. This enables us to improve efficiency by projecting the feature manifold into a subspace of dimension, $d \ll N$. Dramatic dimension reduction is possible because most features of interest have significant structure and inherent symmetries. In practice, d

turns out to be in the range 5–15. Dimension reduction was first used in feature detection by Hummel [Hummel 79] and a similar compressed representation was proposed for 3-D object recognition and pose estimation in [Murase and Nayar 95].

During the search itself, we use a coarse-to-fine algorithm that exploits the local smoothness of the feature manifolds to quickly find the closest sample point. Further, we do not need to perform the search at every pixel in the image. Amongst other techniques, we use a recently developed rejection algorithm [Baker and Nayar 96] to quickly eliminate a vast majority of pixels without even needing to project fully into the low dimensional subspace. Such a rejection scheme is feasible and effective since most pixels in an image do not represent features of interest. With all the above efficiency enhancements in place, our feature detectors take only a few seconds on a standard single-processor workstation when applied to a 512×480 image. Given the enormous strides being made in memory and multi-processor technology, it is only a matter of time before real-time performance is achieved.

The remainder of this paper is organized as follows. In the next section, we introduce the notion of a parametric scene feature and discuss our sensor models. We 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, we introduce our five example features, namely, step edges, lines, corners, roof edges, and circular discs. In each case, we present the feature model, the result of dimension reduction, and the feature manifold. In Section 4, the detection algorithm is presented in detail. In particular, we describe manifold sampling, efficient search, and the use of rejection techniques. In Section 5, our experimental results are presented, which include comparisons with the Canny [Canny 86] and Nalwa-Binford [Nalwa and Binford 86] step edge detectors. We conclude in Section 6 with a discussion of several issues arising from our work.

2 Parametric Feature Representation

We begin by presenting the theoretical basis of our approach to feature detection. First, the notion of a parameterized scene feature is introduced. Then, we describe the artifacts introduced by the imaging system as it maps a scene feature to its discrete image. Finally, parameter normalization and dimension reduction techniques are used to obtain parametric feature manifolds in low-dimensional subspaces.

2.1 Parametric Scene Features

By a scene feature we mean a geometric or photometric phenomenon in the physical world that produces spatial radiance variations which, if detectable, can aid in visual perception. It is known that image brightness is proportional to scene radiance [Horn 86]. The image feature is therefore the continuous radiance function of the scene feature. It can be written as $F^c(x, y; \mathbf{q})$ where $(x, y) \in S$ are image points within a finite feature window, S, and $\mathbf{q}$ are the parameters of the feature. For instance, in the case of a step edge $\mathbf{q}$ would include edge orientation and the brightness values on the two sides of the edge. In the case of a corner $\mathbf{q}$ 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, we need to provide the feature radiance function, $F^c(x, y; \mathbf{q})$, the feature window, S, and the ranges of the parameters, $\mathbf{q}$.

2.2 Modeling Image Formation and Sensing

Previous work on feature detection has implicitly assumed that the artifacts induced by the imaging system are negligible and can be ignored. There are two possible reasons for this. First, some of the artifacts are nonlinear in nature and would make the derivation of the detector, as approached before, more cumbersome. Second, the effects introduced by the imaging system are typically less pronounced than those that result from the feature parameters themselves. For reasons that will become clear shortly, we are able to incorporate both linear and nonlinear effects in our feature model. Hence, we choose to make our feature models as precise as possible by incorporating image formation effects.

The first such effect is the blurring of the continuous feature image. 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 bandlimited in its spatial resolution. Finally, the feature itself, even before imaging, may be somewhat blurred. For instance, a real scene edge would not be a perfect step but rather rounded. The magnitude of this affect in image space is spatially variant and also depends upon the magnification of the imaging system. Moreover, the level of defocus is not constant across the image and so we will develop an approach that can handle spatially varying blur. The defocus factor can be approximated by a pillbox function [Born and Wolf 65], the optical transfer function by the square of the first-order Bessel func-

tion of the first kind [Born and Wolf 65], and the blurring due to imperfections in the feature by a Gaussian [Koenderink 84]. We combine all three effects in a single blurring factor that is assumed to be a 2-D Gaussian:

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

The continuous image on the sensor plane is converted (typically by a CCD detector) to a discrete image through two processes. First, the light flux falling on each sensor element is averaged, or integrated. If the pixels are rectangular [Barbe 80] [Norton 82], the averaging function is simply the rectangular function [Bracewell 78]:

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

where, w_x and w_y are the x and y dimensions of the pixel, respectively. Next, the pixels are sampled, which can be modeled by a rectangular grid:

$$s(x, y) = {}^2\mathrm{III}(\tfrac{1}{p_x}x, \tfrac{1}{p_y}y) \tag{3}$$

where, p_x and p_y are spacings between discrete samples in the two spatial dimensions. The final discrete image of a feature may then be written as:

$$F(x, y; \mathbf{q}) = \{\, F^c(x, y; \mathbf{q}) \,*\, g(x, y) \,*\, a(x, y) \,\} \,.\, s(x, y) \tag{4}$$

where $*$ is the 2-D convolution operator. (Note that $g(x, y)$, $a(x, y)$, and $s(x, y)$ are all unit volume and so the feature image is not scaled.) Since the above image is simply a weighted sum of Kronecker delta functions [Bracewell 78], it can also be written as $F(m, n; \mathbf{q})$, where $(m, n) \in S$ are the (integer valued) pixel coordinates.

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

2.3 Parametric Feature Manifolds

If the total number of pixels in the feature window is N, then each feature instance, $F(m, n; \mathbf{q})$, may be regarded as a point in the N-dimensional Hilbert space, $\Re^N$. Suppose the feature has

k parameters ($\dim(\mathbf{q})=k$). Then, as the parameters vary over their ranges, the corresponding feature instances trace out a k-parameter manifold in $\Re^N$. Therefore, any parametric feature may be represented as a multivariate manifold in a high-dimensional space. Feature detection can then be posed as finding the closest point on the feature manifold to each novel candidate window in the image. Performing this task directly using the feature manifold is impractical for reasons of efficiency due to the high dimensionality of the Hilbert space (N) and the manifold itself (k). In the following two subsections, we present techniques that dramatically reduce the dimensionality of the manifold and the space in which it lies, thereby making the feature manifold a viable representation for feature detection and parameter estimation.

2.4 Parameter Reduction by Normalization

For each feature instance $F(m,n;\mathbf{q})$, we compute its mean $\mu(\mathbf{q})=\frac{1}{N}\sum_{(n,m)\in S} F(m,n;\mathbf{q})$, and its magnitude $\nu^2(\mathbf{q}) = \sum_{(n,m)\in S}\left(F(m,n;\mathbf{q})-\mu(\mathbf{q})\right)^2$. The following brightness normalization is then applied:

$$\overline{F}(m,n;\mathbf{q}) = \frac{1}{\nu(\mathbf{q})}\left(F(m,n;\mathbf{q}) - \mu(\mathbf{q})\right) \tag{5}$$

This simple normalization proves to be very valuable. For all of the features we implemented, it reduced the dimensionality of the feature manifold by two. This is because $\overline{F}(m,n;\mathbf{q})$ turns out to be approximately independent of two of the brightness parameters in $\mathbf{q}$. For instance, in the case of the step edge the normalized feature $\overline{F}(m,n;\mathbf{q})$ is invariant to the brightness values on either side of the step. It is only the values of μ and ν that change with the two brightness parameters. There are three important points to note about this brightness normalization: (a) it does not alter the signal-to-noise ratio of the feature, (b) the normalization must be applied not only during the construction of the feature manifold but also during feature detection, and (c) once a normalized feature has been detected, its mean μ and magnitude ν can be used to recover the two brightness parameters eliminated during normalization. See Appendix A for the details.

2.5 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 $\Re^N$ without significant loss of information[2]. If correlation between feature instances is the preferred measure of similarity, the Karhunen-Loéve (K-L) expansion [Oja 83][Fukunaga 90] yields the optimal subspace.

The covariance matrix $\mathbf{R} = E[(\overline{F} - E[\overline{F}])(\overline{F} - E[\overline{F}])^T]$ represents the correlation between corresponding pixels in the different feature instances. The normalized feature instances $\overline{F}$ are N-dimensional vectors, and so $\mathbf{R}$ is a symmetric $N \times N$ matrix. The reduced space is computed by solving the eigenstructure decomposition problem:

$$\mathbf{R}\,\mathbf{e} = \lambda\,\mathbf{e} \tag{6}$$

The result is the 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 parametrized features, the first few eigenvalues tend to be significantly larger than the remaining ones. This allows us to represent features in a low-dimensional subspace spanned by the few most prominent eigenvectors. Suppose we use the first d eigenvectors, then a measure of the information discarded is the K-L *residue* defined by:

$$R(d) = \sum_{j=d+1}^{N} \lambda_j \tag{7}$$

To give an idea of the data compression possible, a step edge manifold in a 49-D Hilbert space can be represented in a 3-D subspace with K-L residue of less than 10%. Moving to an 8-D subspace reduces the residue to less than 2%.

The parametric feature manifold is constructed by projecting all feature instances into the subspace. This requires the dot products (convolution) of each feature instance with the prominent eigenvectors that serve as a basis for the subspace. Since such a parameterized feature manifold is easy to compute for any feature, we have at our disposal 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 2.4 allow us to compactly represent features and detect them efficiently.

[2]This idea was first explored by Hummel [Hummel 79] and later by Lenz [Lenz 87]. Whereas Hummel derived closed-form solutions for the optimal subspace based upon simplified feature models, our approach is to use elaborate feature models and numerical methods. This results in higher precision and greater generality [Nayar et al. 96]. A similar approach has been adopted by Nandy et al. [Nandy et al. 96] in concurrent work.

3 Example Features

We now illustrate the manifold representations of 5 parametric features. For each feature, we provide a definition of the feature, list its parameters, discuss the effects of brightness normalization, and present the results of dimension reduction. The features we have chosen are merely examples that happen to be important in machine vision. The techniques are not restricted to brightness images, but may also be applied to features found in data produced by most other types of sensors.

3.1 Step Edge

Our first example feature is the familiar *step edge.* Parametric models for edges date back to the work of Hueckel [Hueckel 71]. Since then, the edge has been studied in more detail than any other visual feature (see [Nalwa 93]). Figures 1(a) and 1(b) show the isometric and plan views of the step edge model which we use. This model is a generalization of those used in [Hueckel 71], [Hummel 79], and [Lenz 87]. It is closest to the one used by Nalwa and Binford [Nalwa and Binford 86] in terms of the number and type of parameters, but differs slightly in its treatment of smoothing effects. The basis for the 2-D step edge model is the 1-D unit step function:

$$u(t) = \begin{cases} 1 & \text{if } t \geq 0 \\ 0 & \text{if } t < 0 \end{cases} \tag{8}$$

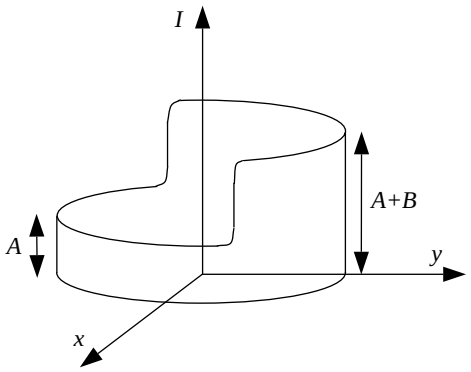
A step with lower intensity level, A, and upper intensity level, $A + B$, can be written as $A + B \cdot u(t)$. To extend the model to 2-D, we assume that the step edge is of constant cross section (step size along its length), is oriented at an angle θ, and lies at a distance ρ from the origin. Then,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

$$z = y \cdot \cos\theta - x \cdot \sin\theta - \rho \tag{9}$$

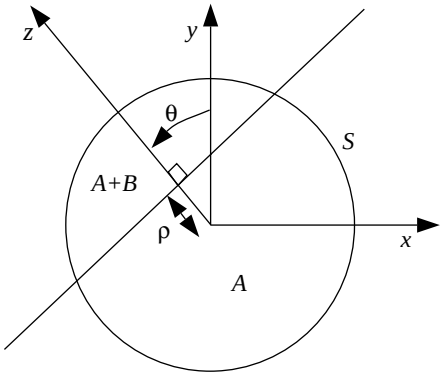
Therefore, an ideal step edge is given by $A + B \cdot u(z)$. For the reasons given in Section 2.2, we need to incorporate the Gaussian blur and the integration over each pixel performed by the sensor. The resulting step edge model is:

$$F_{SE}(x, y; A, B, \theta, \rho, \sigma) = \{\, (A + B \cdot u(z)) * g(x, y; \sigma) * \, a(x, y) \,\} \,.\, s(x, y) \tag{10}$$

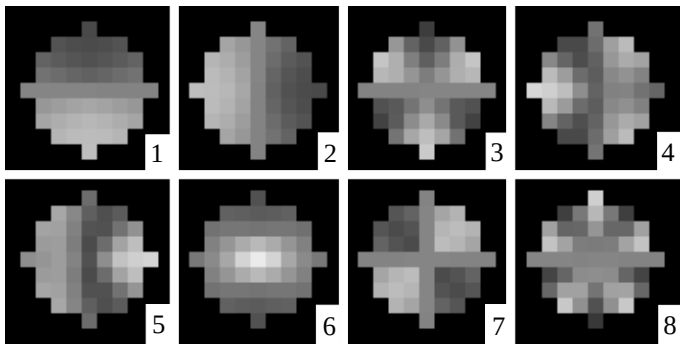
where z is given by equation (9). Note that the step edge model has 5 parameters, namely, orientation θ, localization ρ, blurring or scaling σ, and the brightness values A and B.



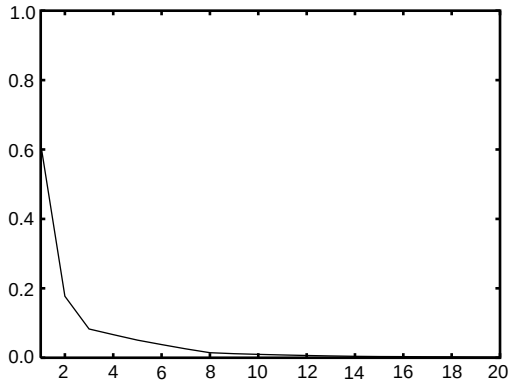
(a) Step edge model



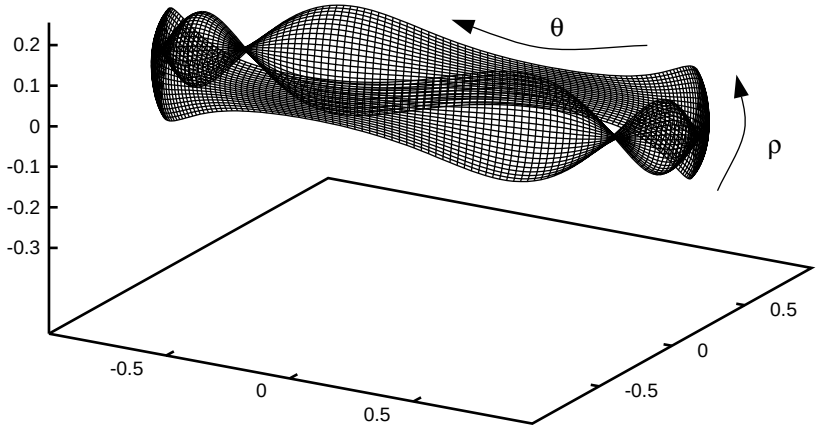
(b) Plan view



(c) First 8 eigenvectors



(d) Decay of the K-L residue



(e) Step edge parametric manifold

Figure 1: The step edge model includes two constant intensity regions of brightness A and $A+B$. Its orientation and intrapixel displacement are given by the parameters θ and ρ, respectively. The fifth parameter (not shown) is the blurring factor σ. The decay of the K-L residue shows that 90% of the edge image content is preserved by the first 3 eigenvectors and 98% by the first 8 eigenvectors. The step edge manifold is parameterized by orientation and intrapixel localization for a fixed blurring value and is displayed in a 3-D subspace constructed using the first three K-L eigenvectors.

To complete our definition of the step edge, we need to specify the ranges that the parameters may take. Distances are measured in units of the distance between two neighboring pixels and angles are given in degrees. The orientation parameter, θ, is drawn from $[0^o, 360^o]$. We restrict the localization parameter, ρ, 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. The blurring parameter, σ, is drawn from $[0.3, 1.5]$. As described in [Nalwa and Binford 86], substantially larger values of σ could be used, but really represent an edge at a much higher magnification. 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 brightness normalization described in Section 2.4. The structure of a normalized step edge, given by the parameters θ, ρ, and σ, is independent of A and B. Further, the values of A and B may be recovered from the mean μ and the magnitude ν as described in Appendix A.

The results of applying the Karhunen-Loéve expansion are displayed in Figures 1(c) and 1(d). In Figure 1(c) we display the 8 most important eigenvectors, ordered by their eigenvalues. The similarity between the first 4 eigenvectors and the ones derived analytically by Hummel in [Hummel 79] is immediate. On closer inspection, we notice that while Hummel's eigenvectors are radially symmetric, the ones we computed are not. This is to be expected since the introduction of the parameters ρ and σ breaks the radial symmetry that Hummel's edge model assumes. While Hummel's eigenvectors are optimal for his edge model, our results imply that they are not optimal for our, more complex, edge model. It is also interesting to note that the first two eigenvectors resemble first-order spatial derivative operators that constitute the basis of many popular edge detectors (for instance the Sobel operator [Prewitt 70]).

The window chosen for our edge model includes 49 pixels. To avoid unnecessary non-linearities induced by a square window we used a disc shaped one. In Figure 1(d), the decay of the Karhunen-Loéve residue is plotted as a function of the number of eigenvectors. As can be seen from the residue plot, the first two eigenvectors capture only about 80% of the information in an edge. Consequently, edge detectors that rely solely on the two first-order derivatives can only suggest the possible existence of an edge but not guarantee it. To reduce the residue to 10% we need to use 3 eigenvectors. To reduce it further to 2% we need 8 eigenvectors. These results illustrate a significant data compression factor of 5-15 times. As a result, the efficiency of feature detection and parameter estimation is greatly enhanced. Hummel [Hummel 79] predicts that for his continuous step edge model, the eigenvalues should decay like $1/n^2$. Our results are consistent with this. By plotting λ_n against n on logarithmic scales and analyzing

the slope of the curve, we found that our eigenvalues initially decay like $1/n^2$. Because we are working in $\Re^N$ rather than the infinite dimensional Hilbert Space of [Hummel 79], the rate of decay increases somewhat with increasing n.

The step edge manifold is displayed in Figure 1(e). Naturally, we are only able to display a 3-D projection of it into a subspace. This subspace is spanned by the 3 most important eigenvectors. Also, for clarity, we only display a 2 parameter slice through the manifold, by keeping σ constant and varying θ and ρ. As mentioned earlier, the first 3 eigenvectors capture more than 90% of the information. This is reflected in Figure 1(e), where most points on the manifold are seen to lie close to unit distance from the origin. Note that the four apparent singularities of the manifold are simply artifacts of the projection into the 3-D subspace. If we were able to visualize a higher dimensional projection, the singularities would disappear.

3.2 Roof Edge

The roof edge is similar to the step edge. However, unlike the step edge, it has not been explored much in the past despite having been acknowledged as a pertinent feature [Nalwa 93]. The difference between the two edge models is that the step discontinuity is replaced by a uniform intensity gradient as shown in Figure 2(a). A formal definition is obtained by replacing $A + B \cdot u(z)$ with $A - M \cdot z \cdot u(z)$, where A is the upper intensity level of the roof, and M is the gradient, or slope, of the roof. The result is a 5 parameter model:

$$F_{RE}(x, y; A, M, \theta, \rho, \sigma) = \{ \, (A - M \cdot z \cdot u(z)) * g(x, y; \sigma) * \, a(x, y) \, \} \, . \, s(x, y) \qquad (11)$$

where $u(z)$ and z are as defined for the step edge. The parameter ranges which we used for the roof edge are: $\theta \in [0^o, 360^o]$, $\rho \in [-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$, and $\sigma \in [0.4, 1.0]$. The range of sigma 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 sigma. The parameters A and M are free. As with the step edge, the structure of the normalized roof edge is independent of A and M, and their values are easily recovered from the normalization coefficients μ and ν. See Appendix A for the details.

The results of applying the Karhunen-Loéve expansion, as shown in Figures 2(c) and 2(d), are similar to those for the step edge. The K-L residue decays slightly faster as should be expected since the roof edge more closely resembles a constant intensity region than the step edge. (The residue of a constant intensity region would decay immediately to zero.) The

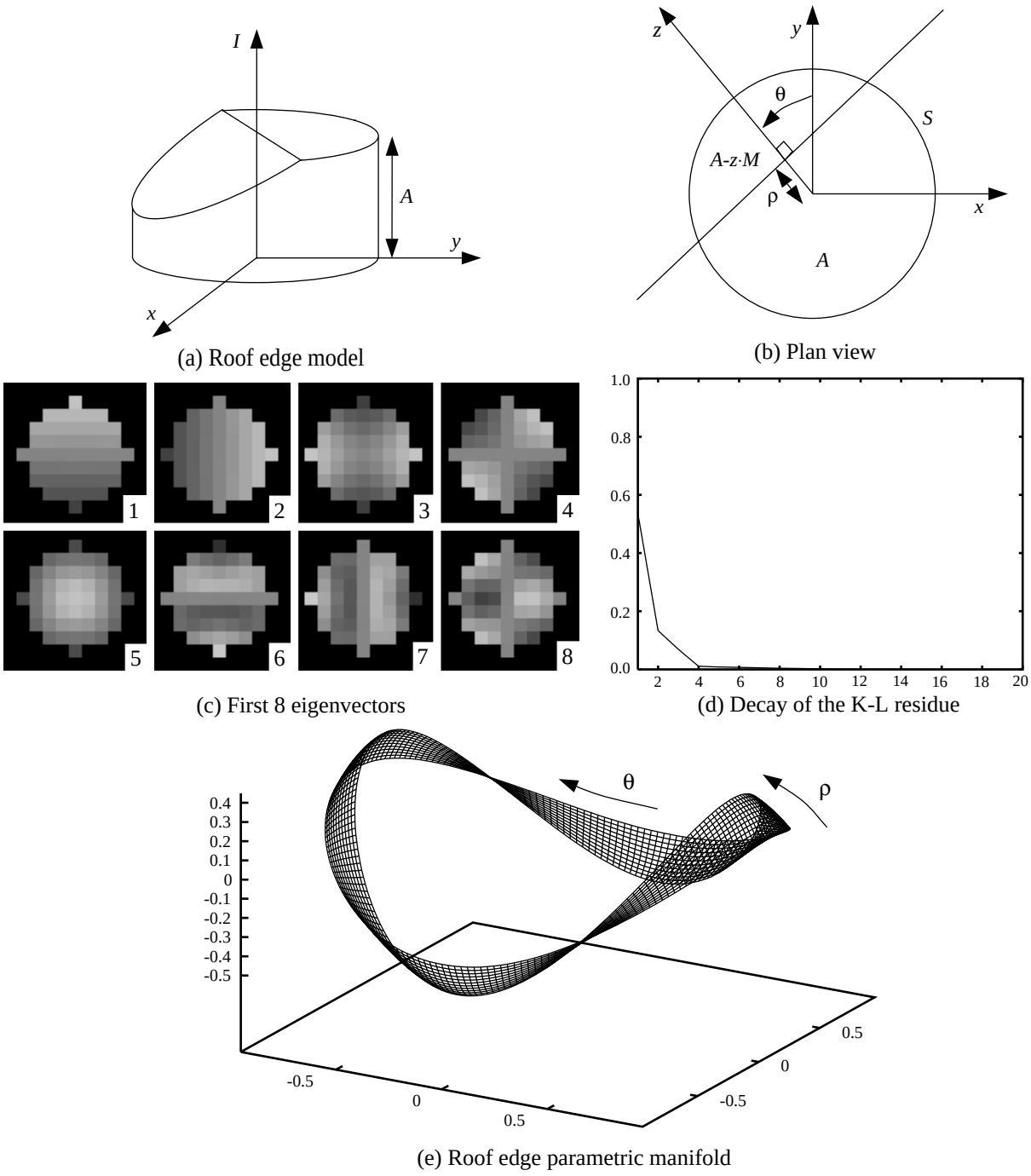


Figure 2: The roof edge model has a region of constant brightness A on one side and a uniform brightness slope of gradient M on the other. Both parameters A and M are removed by the brightness normalization. The orientation parameter θ, the localization parameter ρ, and the blur parameter σ are similar to those used for the step edge. The first two eigenvectors of the roof edge are similar to those of the step edge, but after that the K-L residue decays marginally faster. The displayed slice through the roof edge manifold is parameterized by orientation and intrapixel displacement for a fixed blurring value.

first two eigenvectors are approximately the same as those for the step edge (at least, up to a sign change). For the roof edge 3 eigenvectors are needed to capture 90% of the edge content, and 5 eigenvectors for 98%. The parametric manifold for the roof edge is displayed in Figure 2(e). The significant difference in appearance from 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.

3.3 Line

The line consists of a pair of parallel step edges separated by a short distance, namely, the width w of the line [Hueckel 73]. Our line model is illustrated in Figure 3(a). In our definition, we assume that the intensity steps are both of the same magnitude. It is possible to generalize the model to lines with different intensities on either side of the line with the addition of one extra parameter [Hueckel 73]. The symmetric line model which we use has 6 parameters and is given by: $F_L(x, y; A, B, \theta, \rho, w, \sigma) =$

$$\{ (A + B \cdot u(z + w/2) - B \cdot u(z - w/2)) * g(x, y; \sigma) * a(x, y) \} . s(x, y) \tag{12}$$

The ranges of the parameters ρ and σ are exactly as for the roof edge: $\rho \in [-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$, and $\sigma \in [0.4, 1.0]$. Given the brightness symmetry in our line model, the orientation range can be halved to $\theta \in [0^o, 180^o]$. We restrict the width of the line to $w \in [1.0, 3.5]$. The brightness parameters A and B are free, as for the step and roof edge models, and can be eliminated by applying the normalization procedures presented in Section 2.4. Again, during detection, A and B can be recovered from the normalization coefficients 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 residue, as seen in Figure 3(d). To reduce the residue to 10% we require 8 eigenvectors, and to reduce it to 2% we need 22. By this measure, the line is a considerably more complex feature to detect than an edge. However, the data compression factor is still relatively large, and in the range of 3-5. It is interesting to note that the line manifold in Figure 3(e) has the structure of a Möbius band. This follows directly from the following symmetry in the line model:

$$F_L(x, y; A, B, \theta + 180^o, \rho, w, \sigma) = F_L(x, y; A, B, \theta, -\rho, w, \sigma) \tag{13}$$

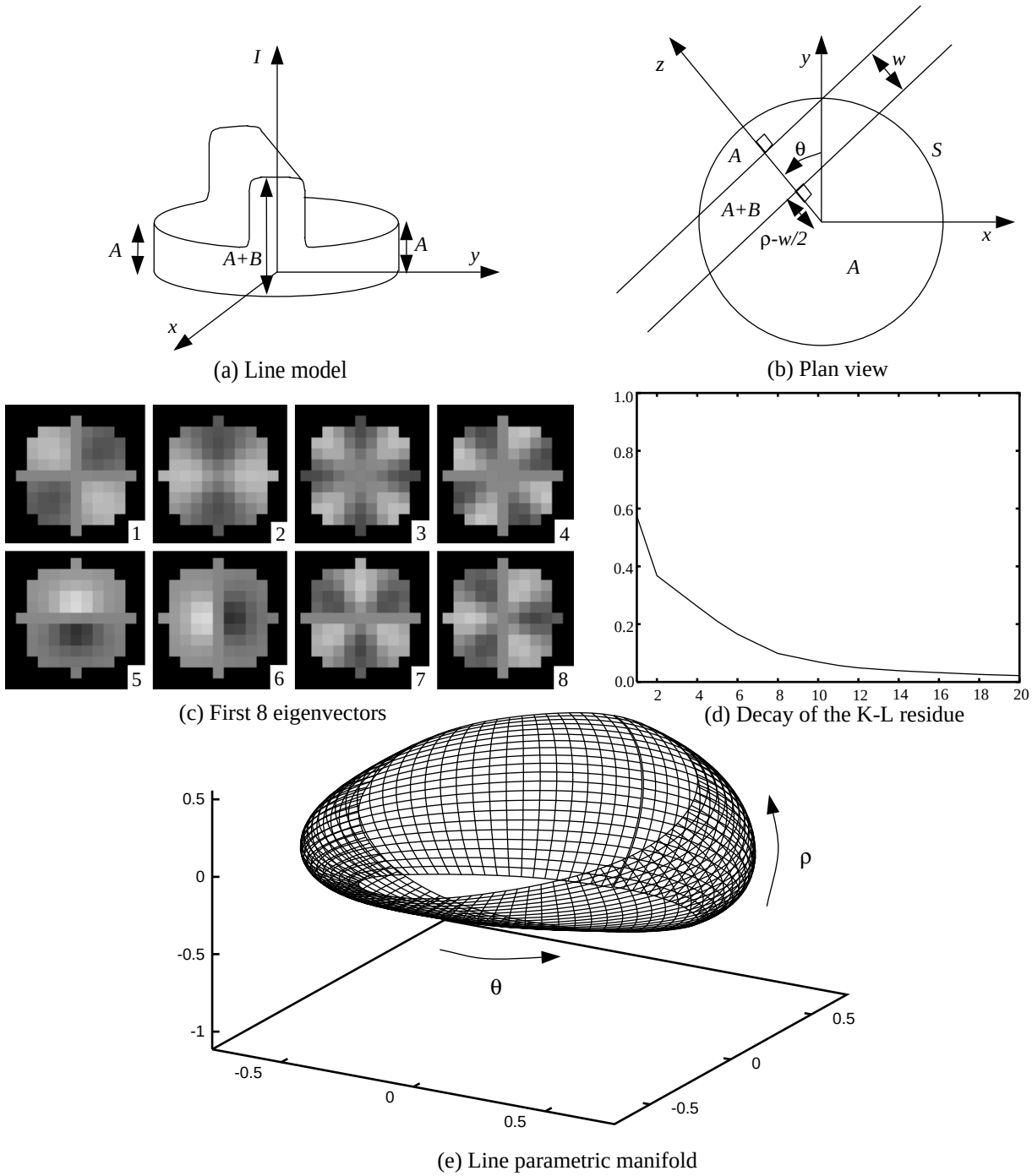


Figure 3: The line is of width w, has brightness $A+B$ on the line itself, and has regions of brightness A on either side of the line. In addition, there is the orientation parameter θ, the localization parameter ρ, and the blur parameter σ. 8 eigenvectors are need to capture 90% of the feature content and 22 eigenvectors for 98%. By this measure the line is a more complex feature than an edge. The line manifold is displayed for fixed values of σ and w. It has the structure of a Möbius band.

3.4 Corner

The corner is a common and hence important image feature [Nobel 88]. Most existing approaches to corner detection are based upon differential geometric measures of curvature such as the determinant of the Hessian or the second directional derivative orthogonal to the gradient [Deriche and Giraudon 93]. Recently, Rohr [Rohr 92] proposed a parametric model fitting approach to detect corners. The simplest way to think about a corner is as the intersection point of two non-parallel lines. In our corner model, shown in Figure 4(a), θ_1 is the angle one of the edges of the corner makes with the y-axis, and θ_2 the angle subtended by the corner itself. That is, the corner lies at the intersection of its bounding edges at angles θ_1 and $180^o + \theta_1 + \theta_2$. This is illustrated in Figure 4(b). Mathematically, this intersection can be expressed as the product of two unit step functions. The complete corner model has 5 parameters and is written as: $F_C(x, y; A, B, \theta_1, \theta_2, \sigma) =$

$$\{ (A + B \cdot u(z(\theta_1)) \cdot u(z(180^o + \theta_1 + \theta_2))) * g(x, y; \sigma) * a(x, y) \} . s(x, y) \tag{14}$$

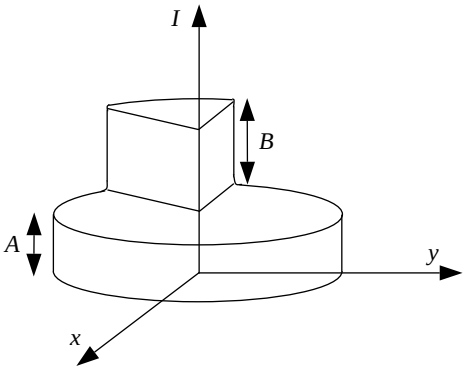
where, $z(\theta) = y \cdot \cos\theta - x \cdot \sin\theta$. The parameter ranges which we used are: $\theta_1 \in [0^o, 360^o]$, $\theta_2 \in [30^o, 120^o]$, and $\sigma \in [0.4, 1.0]$. Again, brightness normalization eliminates the parameters A and B. The decay of the K-L residue, shown in Figure 4(d), is similar to that of the line. In this case, 7 eigenvectors reduce the residue to below 10%, and 15 eigenvectors are needed to reduce it to less than 2%. The corner manifold is displayed for fixed σ in Figure 4(e).

3.5 Circular Disc

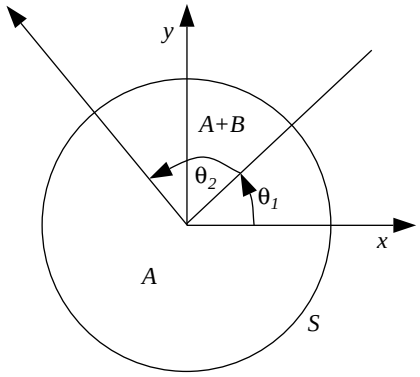
Our final example feature is the circular disc which is illustrated in Figure 5(a) and Figure 5(b). The parameters of the circular disc are its radius r, the direction θ that the center P of the disc makes with the y axis, the disc localization ρ, and the level of blurring σ. The brightness values inside and outside the disc are $A + B$ and A, respectively. Mathematically, the circular disc model can be expressed as:

$$F_{CD}(x, y; A, B, \theta, \rho, r, \sigma) = \{(A + B \cdot u(r - \mathrm{d}(x, y))) * g(x, y; \sigma) * a(x, y) \} . s(x, y) \tag{15}$$

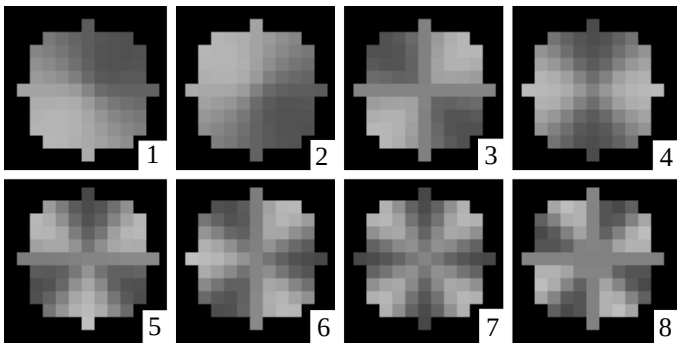
where $\mathrm{d}(x, y) = \sqrt{((x + (r - \rho)\sin\theta)^2 + (y - (r - \rho)\cos\theta)^2)}$ is the distance of (x, y) from the point P. The parameter ranges are: $\theta \in [0^o, 360^o]$, $\rho \in [-\frac{\sqrt{2}}{2}, \frac{\sqrt{2}}{2}]$, $r \in [3.0, 12.0]$, and $\sigma \in [0.4, 1.0]$. Again, brightness normalization removes the effects of A and B. The rate of decay of the K-L residue, as seen from Figure 5(d), is slightly less than that of the step



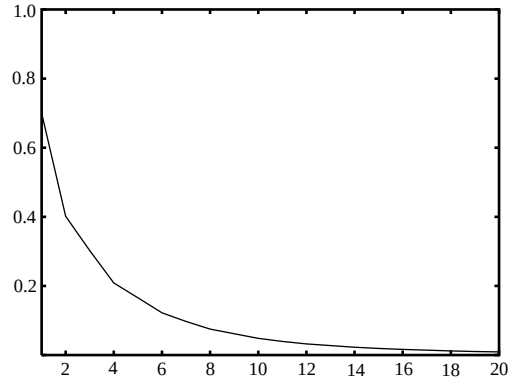
(a) Corner model



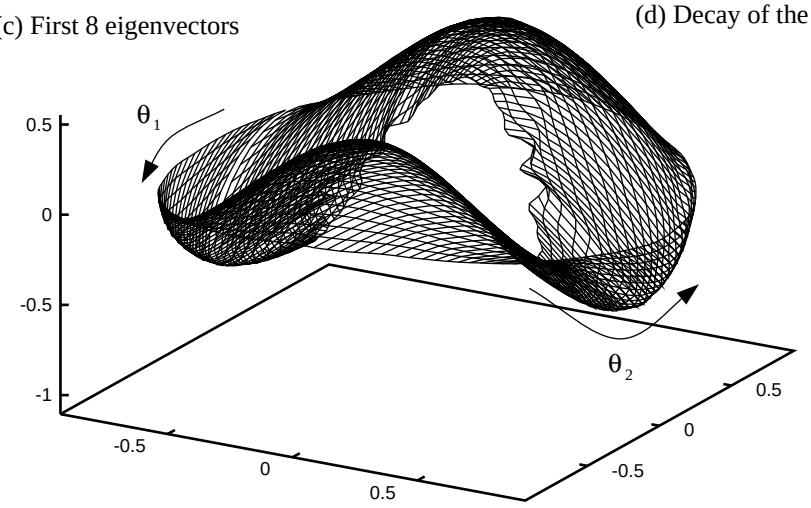
(b) Plan view



(c) First 8 eigenvectors



(d) Decay of the K-L residue



(e) Corner parametric manifold

Figure 4: The corner is described by the brightness values (A and $A + B$) inside and outside the corner, the angles θ_1 and θ_2 made by its edges, and the blur parameter σ. 7 eigenvectors are needed to preserve 90% of the information and 15 eigenvectors for 98%. The corner manifold is shown for a fixed value of σ.

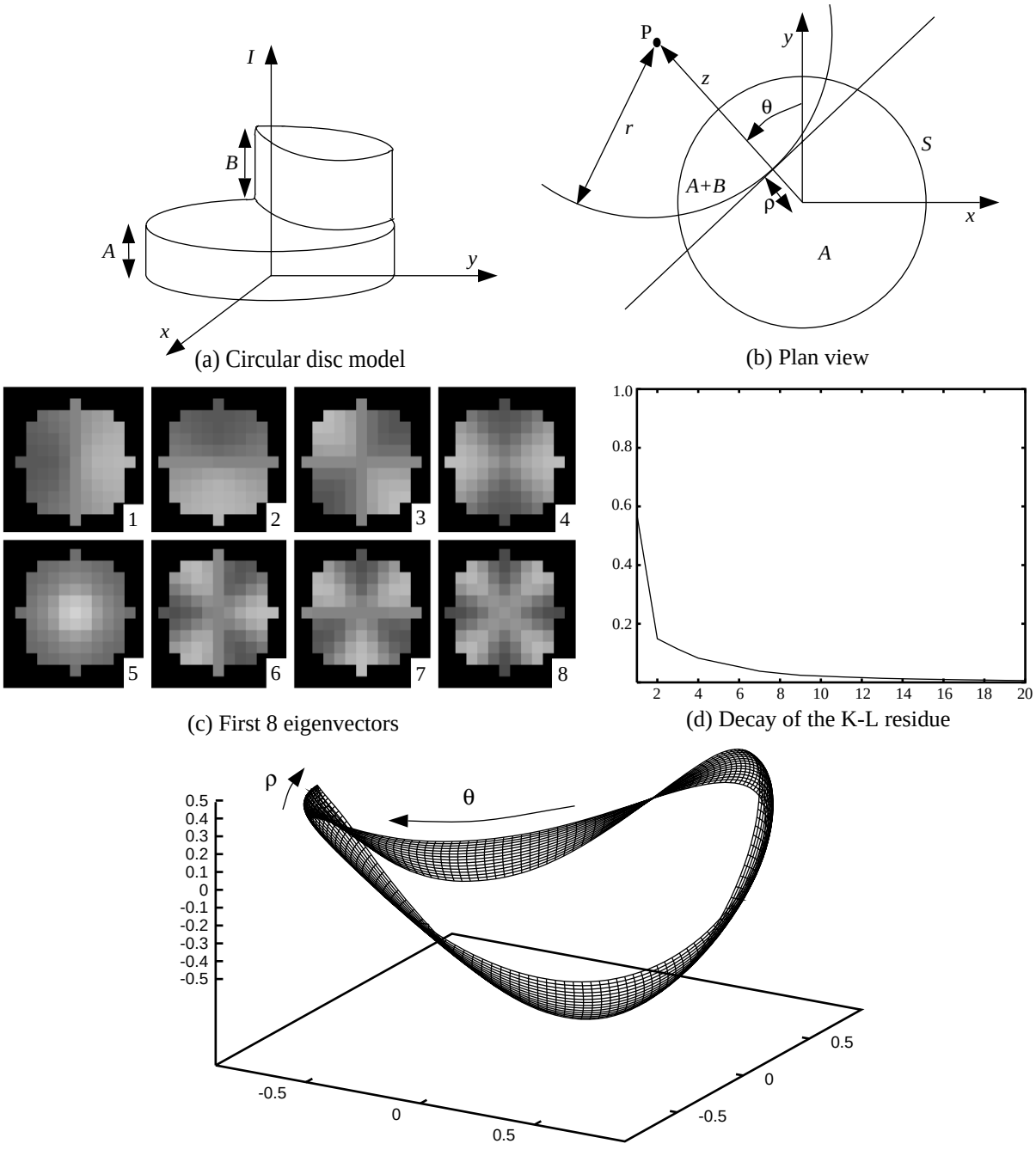


Figure 5: The circular disc is described by the brightness parameters A and B, the radius r of the disc, the angle θ subtended by the center of the disc, the localization ρ, and the blur parameter σ. 4 eigenvectors are needed to preserve 90% of the information, and 11 eigenvectors for 98%. The circular disc manifold is displayed for fixed values of σ and r.

edge. In this case, we need 4 eigenvectors to reduce the residue to 10%, and 11 eigenvectors to reduce it to below 2%. The first 8 eigenvectors are shown in Figure 5(c), and the manifold in Figure 5(e).

4 Feature Detection and Parameter Estimation

In Section 2 we introduced parametric feature manifolds as a representation for parametric features, and then in Section 3 we presented 5 example features together with their parametric manifolds. We now describe how feature detection and parameter estimation are accomplished in terms of the feature manifolds.

4.1 Sampling the Parametric Manifold

As we saw in Section 3, after eliminating two parameters by applying the brightness normalization of Section 2.4, we are typically left with a manifold of dimension in the range 2–4. For convenience, we sample each parameter independently and at equally spaced intervals across its range. Then, the Cartesian product of the parameter sample points is taken and used to sample the manifold. If the dimension of the manifold is k, the result is sampling at a k-dimensional mesh of parameter values. For example, we might sample the angle θ of the step edge every 2^o, the localization ρ every 0.1 pixels, and the blurring parameter σ every 0.2 pixels. This leaves one question unanswered: How densely should we sample each parameter?

The answer to this question depends on how much varying each parameter affects the visual appearance of the feature. Suppose we measure change in appearance by the Euclidean distance traveled on the manifold in $\Re^N$. Then, if changing a particular parameter causes the appearance to vary rapidly, we should sample it densely in order to capture the full variation in appearance. If varying a parameter results in only a small change in appearance, then there is little point in sampling it densely, since the noise inherent in the image will fundamentally limit the accuracy with which we can estimate that parameter anyway. As a rough guideline, we should aim for the change in appearance between two neighboring sample points to be approximately the same as the change in appearance we can expect due to noise.

In the absence of an estimate of the level of noise, we specify the number of sample points that can be afforded, for either time or space complexity reasons. Then, we sample the manifold as uniformly and densely as possible, with approximately that number of sample

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

Figure 6: Automatically generated sampling intervals for the 5 example features. The intervals are generated by attempting to ensure that the appearance change (Euclidean distance) between each pair of neighboring sample points is the same, while at the same time trying to limit the total to 50,000 sample points. This figure may be used to assess the importance of each parameter to the model. The most important parameters should be those with the smallest sampling intervals.

points. In Figure 6, we present the output of an algorithm that estimates the average rate of change of appearance with respect to each parameter, and then uses the estimates to derive sampling intervals for each parameter. The input to the program was the request to generate manifold samplings each 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.

4.2 Search for the Closest Manifold Point

Finding the nearest neighbor amongst a fixed set of points to a given novel point is a well studied problem in computational geometry, and was first posed by Knuth [Knuth 81]. The more recent paper [Yianilos 93] contains a pretty comprehensive bibliography of algorithms developed since then. Our problem has more structure than the general nearest neighbor problem since we know that the points lie on a parametric manifold. So rather than using any of the general purpose algorithms, we attempt to take advantage of the locally-smooth nature of the feature manifolds and develop a less general but faster search technique. We used a 4-level heuristic coarse-to-fine search. It does not guarantee finding the closest point for pathological manifolds, but we found empirically that is performs very well for our 5 example features all of which have smooth manifolds as can be seen in Figures 1–5. In particular, for the manifolds sampled at approximately 50,000 points, the coarse-to-fine search results in a speed-up by a factor of 50-100 times over linear search through the 50,000 points. The average error for each parameter of every feature was always less than the spacing between

neighboring samples.

The coarse-to-fine search is both conceptually simple as well as very easy to implement. We sample the manifold several times, giving a sequence of meshes, from a very coarse one with few points up to the finest one containing the most points. The finest mesh is the one we really wish to search. We begin by finding the closest point on the coarsest mesh by using a brute force linear search. This does not cost much in terms of time since the coarsest mesh does not contain many points. We then move to the next finer mesh. We search this mesh locally in the region of the result of the previous level. This search is also a linear brute force search. It again does not cost much since it is only a local search, and on a relatively coarse mesh. We repeat this for each mesh in turn, reducing the size of the local search at each step, until we reach the finest mesh. The result at the finest search gives us the answer we are looking for.

4.3 Further Efficiency Improvements

On a single-processor DEC Alpha 3600 workstation with no additional hardware, the coarse-to-fine search for a 3-D manifold in a 10-D space that is sampled at 50,000 points takes approximately 1ms. So, applying the detector to every pixel in a 512×480 image takes around 4mins. This figure is by no means the best we can do in terms of efficiency.

Rejection: We do not need to apply the coarse-to-fine search at every pixel in the image. This observation is almost as old as edge detection itself and is explicitly mentioned in [Hueckel 71]. Combining a variety of techniques, we have already reduced the time to process a 512×480 image to less than a minute. In particular, we currently threshold on the magnitude, ν, obtained during normalization. This technique is similar to Moravec's interest operator [Moravec 77] used to predict the usefulness of potential stereo correspondence matches. We also threshold on the distance of the novel point from the K-L subspace. Since the distance from the subspace is (approximately) a lower bound on the distance from the manifold, if the distance is too large, we can immediately decide that the pixel does not contain the feature. Using the techniques in [Baker and Nayar 96], we can even avoid most of the cost of computing the distance from the K-L subspace.

Parallel Implementation: Feature detection is inherently a parallelizable task. As high performance multi-processor workstations become commonplace, the times mentioned above will easily be cut by factors on the order of 10 or more. Also, it is reasonable to expect

performance increases for the individual processors, further increasing the overall performance. It is safe to expect that, within a few years, a standard workstation will be able to apply the proposed detectors in real-time.

5 Experimental Results

Upon surveying the literature, we found a number of methods that have been used to compare the performance of feature detectors:

- Examine the rates of occurrence of false positives and false negatives when applied to synthetically generated feature instances. (See, for example, [Fram and Deutsch 75], [Abdou and Pratt 79], and [Nalwa and Binford 86].)
- Study the accuracy of parameter estimation, either using statistical tests or through an analytical investigation of systematic biases. (See, for example, [Deutsch and Fram 78], [Abdou and Pratt 79], [Berzins 84], and [Nalwa and Binford 86].)
- Evaluate measures that combine feature detection rates with parameter estimation accuracy. (One example is Pratt's Figure of Merit [Abdou and Pratt 79] [Pratt 90].)
- Subjectively analyze detector outputs when applied to real or synthetic images. (Almost all feature detection papers do this.)

We begin this experimental section by presenting the results of a sequence of statistical tests. In Section 5.1 we study feature detection rates, and then move on to investigate parameter estimation accuracy in Section 5.2. In both cases, we compare our step edge detector with those of Canny [Canny 86] and Nalwa-Binford [Nalwa and Binford 86]. In this, our aim is to demonstrate that the parametric manifold method performs comparably to these well-known step edge detectors. We also compare the performance of our technique across the five example features, the goal of which is to demonstrate the generality of the approach by showing that the performance is similar for all five features. In Sections 5.3 and 5.4 we present the results of applying our feature detectors to a number of real and synthetic images.

5.1 Feature Detection Rates

We first statistically compare our step edge detector with the Canny [Canny 86] and Nalwa-Binford [Nalwa and Binford 86] detectors. For reasons of consistency with previous work, we follow the approach taken in [Nalwa and Binford 86]. The statistical analysis consists of two phases. In the first phase we generate a number of ideal step edges, add zero-mean Gaussian noise to them, and then apply the three step edge detectors. Whenever a detector fails to detect an edge, we increment a count of false negatives. The second phase consists of generating windows not containing edges, adding noise, and again applying the detectors. If a detector responds positively to a not-edge we register a false positive.

Although the basic idea behind the comparison is simple, 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 an edge. Our model and the Nalwa-Binford model are closely related, but the Canny operator is based upon differential invariants rather than a parametric model. Since we took great care modeling both the features and the imaging system, we used our step edge model in the comparison. For fairness we changed some of the details slightly. Both the Canny and Nalwa-Binford detectors assume a constant amount of blur, so we fixed the value of σ in the 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 detector in the implementation[3] that we used. Hence, we used a square window containing $N = 25$ pixels for our detector, rather than the 49 pixel disc window presented in Figure 1. Another issue is the lack of a model for a window not containing an edge [Nalwa 93]. We resolve this, as in [Nalwa and Binford 86], by taking a constant intensity window as our characteristic non-edge. Finally, we need to be able to measure the amount of noise in a consistent way across the five features. We define the S.N.R. of a feature to be $\frac{2\times\nu}{\sigma_{noise}}$, where ν is the magnitude of the feature as defined in Section 2.4 and σ_{noise} is the standard deviation of the added Gaussian noise. The reason for this definition is that for an step edge with no blur in a window and where half of the pixels are on each side of the edge, the value of the S.N.R. is the size of the step (i.e. the value of parameter B), which is the measure of S.N.R. used in [Nalwa and Binford 86]

[3]We used an implementation of the Canny operator provided by Geoff West of Curtin University, Western Australia, which is publically available from URL `http://www.cs.curtin.edu.au/ geoff/`. This implementation computes the Gaussian smoothed gradient which we then immediately threshold to detect edges. For simplicity, we do not find the zero crossing of the second directional derivative. Neither do we perform hysteresis [Canny 86] since this uses information derived from neighboring windows.

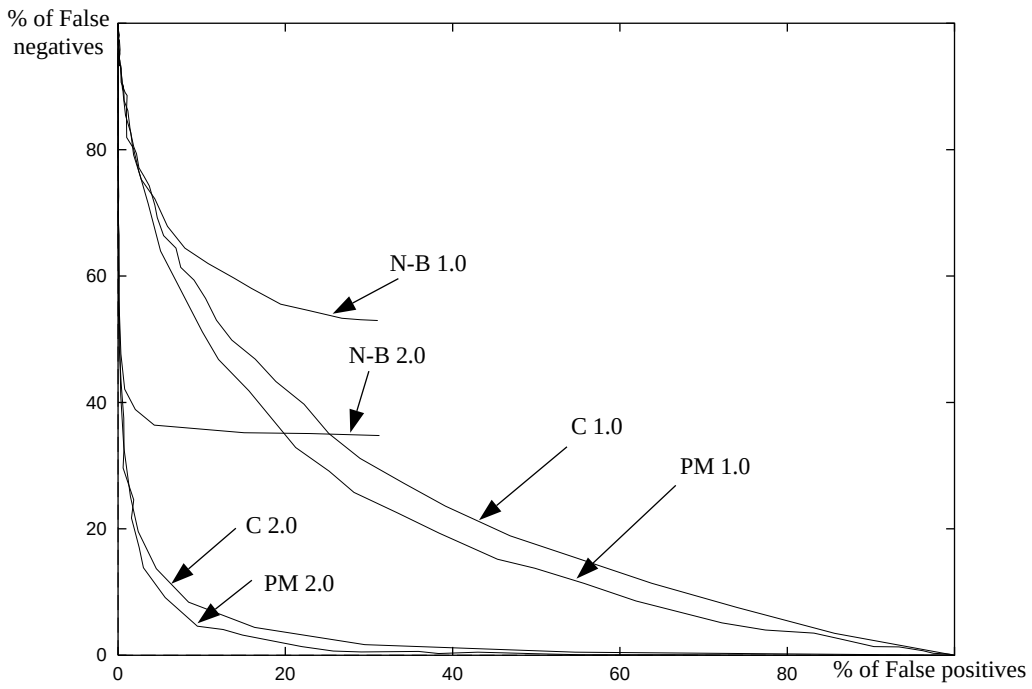


Figure 7: A comparison of edge detection rates. The Canny (C), Nalwa-Binford (N-B), and parametric manifold (PM) detectors are compared for S.N.R. = 1.0 and 2.0. We plot false positives against false negatives. For each detector and S.N.R., the result is a curve parameterized by the threshold inherent in that detector. The closer a curve lies to the origin, the better the performance. We see that the Canny detector and the parametric manifold technique perform comparably. The results for the Nalwa-Binford detector are consistent with those presented in [Nalwa and Binford 86] but are of a fundamentally different nature. See text for a discussion of this.

In Figure 7 we compare the detection performance of the three edge detectors. Inherent in each of the three detectors is a threshold. The Canny operator thresholds on the gradient magnitude, the Nalwa-Binford detector thresholds on the estimated step size, and our approach thresholds on the distance from the parametric manifold. As we vary the threshold, for a fixed level of noise, the relative number of false positives and false negatives changes. Hence, for each signal-to-noise ratio (S.N.R.), we can plot a curve of false positives against false negatives parameterized by that detector's threshold. The closer a curve lies to the origin in Figure 7, the better the performance. We see that both the Canny detector and our detector do increasingly well as the S.N.R. increases. Further, we note that the two detectors perform comparably, with our algorithm doing very marginally better[4].

[4]These results differ slightly from the ones presented in [Nayar et al. 96] as they reflect refinements made

The results for the Nalwa-Binford detector are consistent with those presented in [Nalwa and Binford 86]. (We did not use step 2)' of the algorithm.) Independently of the S.N.R., the percentage of false positives in Figure 7 never exceeds 32%. This validates Figure 8 of [Nalwa and Binford 86]. Secondly, for a S.N.R. of 1.0, the number of false negatives in Figure 7 never drops below 56%, whereas in Figure 9 of [Nalwa and Binford 86] its lowest level is 77%. These two numerical results are slightly different because (a) we use a different model to generate the ideal step edges, and (b) our definition of S.N.R. yields a slightly lower value than the definition in [Nalwa and Binford 86] due to blurred and off center edges. Comparing our results with those in Figure 9 of [Nalwa and Binford 86], we see that our curve for S.N.R. 1.0 lies somewhere between the two curves for S.N.R. 1.0 and 2.0. The reason that the Nalwa-Binford performs qualitatively differently to the Canny and Parametric Manifold detectors in Figure 7 is its inherent conservatism in detecting edges, as enforced by steps 4) and 5) of the Nalwa-Binford algorithm (see page 704 of [Nalwa and Binford 86]).

In Figure 8 we compare the detection rates of our five example features. In the figure, the curves are all plotted for S.N.R. 1.0, and for a disc shaped window containing 49 pixels. We see that the performances of the step edge and the circular disc are marginally superior to that of the other 3 features. We conclude, that the roof edge, corner, and line are slightly more sensitive to noise. One method of reducing the noise sensitivity is to use a slightly larger window. If we increase the window size to a disc containing 81 pixels, the performance is greatly enhanced. We also found that the performance of each of the 5 feature detectors improves with the S.N.R., just as it does for the step edge in Figure 7. For S.N.R. above about 3.0, all the detectors perform almost without error.

5.2 Parameter Estimation Accuracy

Assessing the performance of parameter estimation is relatively straightforward when compared to that of feature detection robustness. Again, generalizing the procedure used in [Nalwa and Binford 86], we randomly generate a vector of feature parameters, synthesize a feature with those parameters, add a known amount of zero-mean white Gaussian noise, apply the detector, and then measure the accuracy of the estimated parameters. If we repeat this procedure a statistically meaningful number of times, the results give a very good indication of parameter estimation performance when applied to a real image. The issue of which model

to our detector, the definition of the S.N.R., and the manner in which the experiments were conducted.

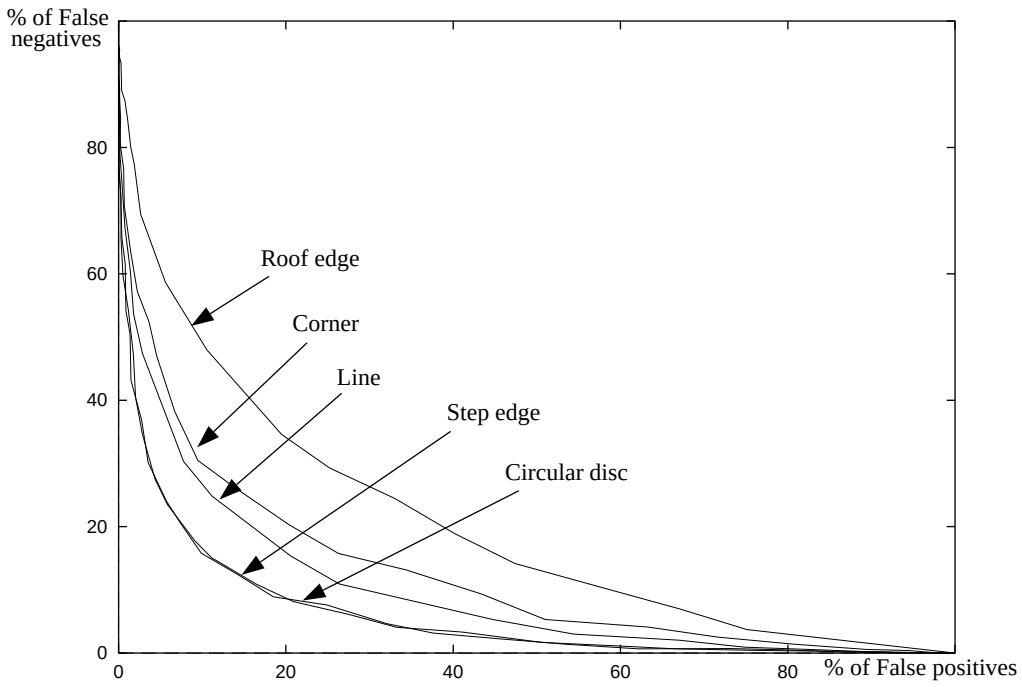


Figure 8: A comparison of feature detection rates for our five example features. All results are for S.N.R. = 1.0 and for a disc shaped window containing 49 pixels. We see that the step edge and circular disc are less noise sensitive than the other features. Note however that the noise sensitivity of all features may be reduced by increasing the size of the window. Further, for S.N.R. around 3.0 and above, all the feature detectors perform with very little error.

we should use to generate the features is still problematic. For the same reasons as before, we again used our feature models to generate the synthetic features.

In Figure 9 we compare the performance of our step edge detector with that of the Canny detector [Canny 86] and the Nalwa-Binford [Nalwa and Binford 86] detector. For fairness, as before, we used the parametric step edge detector computed for a 5×5 square window, and with the blurring parameter fixed at 0.6 pixels. In the figure, we plot the R.M.S. error in the estimate of the orientation θ against the S.N.R. The plot is consistent with the performance figures for the Nalwa-Binford detector presented in [Nalwa and Binford 86], after allowing for the different feature models used and definition of S.N.R. We see that for low S.N.R. the performance of all detectors is severely limited by the noise. However, for all noise levels, the parametric manifold detector marginally outperforms both of the other detectors. If we plot a similar graph for the localization parameter ρ we find the behavior to be similar. In this case however, the performance of the Nalwa-Binford detector is slightly better than

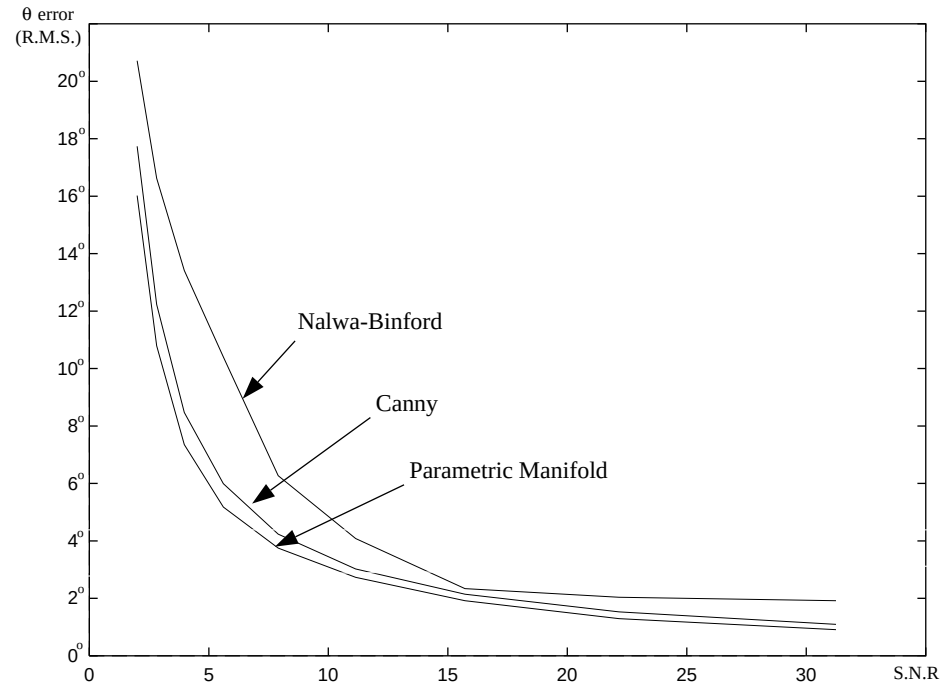


Figure 9: A comparison of the orientation estimation accuracy for the three step edge detectors. We took synthesized step edges, added noise to them, and then applied the edge detectors. We plot the R.M.S. error of the orientation estimate against the S.N.R. At all noise levels, the parametric manifold approach slightly outperforms both the Nalwa-Binford and Canny detectors.

that of the parametric manifold detector, except at low noise levels.

Next, we compare the performance of our five example features. Since all the feature models have an orientation parameter, in Figure 10 we plot the R.M.S. error in orientation estimate against the S.N.R. From Figure 10, we see the performance to be very similar for all our features. Only for the corner is parameter estimation accuracy significantly worst than the rest. Although we do not have room to show them, we plotted similar graphs for the other parameters. Qualitatively the results are all similar. We now summarize some of the more interesting points:

- The plot for the localization parameter ρ shows that the step edge, roof edge, and circular disc all perform very similarly. The performance for the line is noticeably better, presumably because it is an average over two estimates, one for each of the two parallel step edges.

- Estimating A is easier for the corner and line than for the step edge, but estimating B is

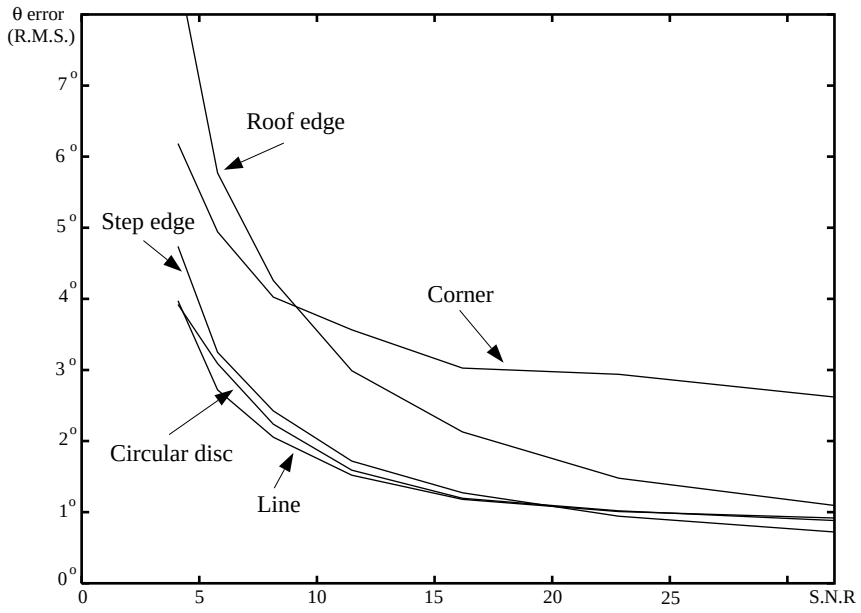


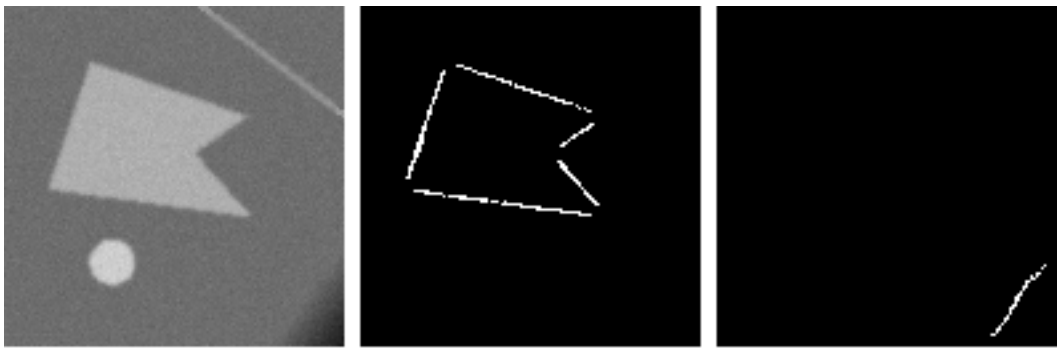
Figure 10: A comparison of orientation estimation accuracy across the five features. Since all the features have an orientation parameter, we use it to compare the performance. We plot the R.M.S. orientation estimation error against the S.N.R. The graph shows that the performance for all five features is approximately the same.

more difficult. This reflects the relative areas occupied by the lower and upper intensity levels in these features.

- It is harder to estimate θ_2 for the corner than it is to estimate θ_1. Probably for related reasons, estimating the width of the line is harder than estimating the localization.

5.3 Application to Synthetic Images

In Figures 11(b)–(f) we display the results of applying the five example detectors to the noisy synthetic image in Figure 11(a). The synthetic image is of size 128×128 pixels and contains a pentagonal region (intensity of 175), a circular disc (radius of 8.5 pixels, intensity of 206), a line (width of 2.3 pixels, intensity of 153), and a roof edge (slope of 4 intensity levels per pixel). The background intensity is 110. The image was first blurred with Guassian smoothing ($\sigma = 0.6$ pixels) and then we added white zero-mean Gaussian ($\sigma = 4.0$ pixels) noise. At pixels where two feature detectors register the presence of a feature, we choose the one with the closer manifold. Further, after blurring and the addition of noise, corners and



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

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

Figure 11: The application of our five feature detectors to a noisy synthetic image. Five different features have been detected and discriminated in the same image using the same technique.

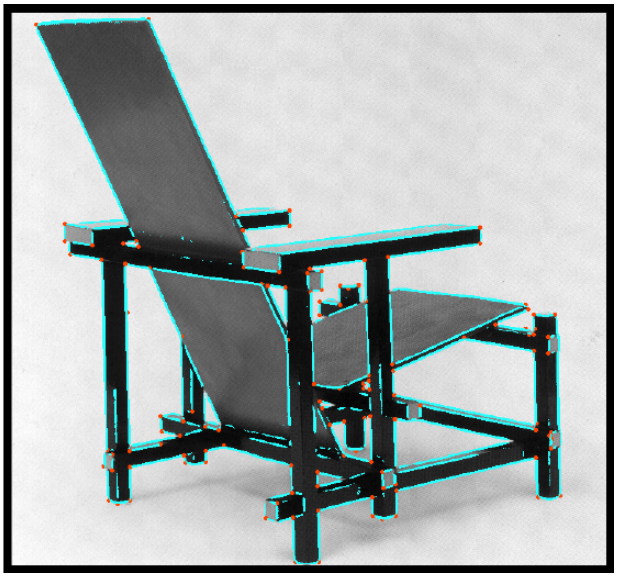
circular discs can appear very similar and so it required a limited amount of post-processing to discriminate these two features.

5.4 Application to Real Images

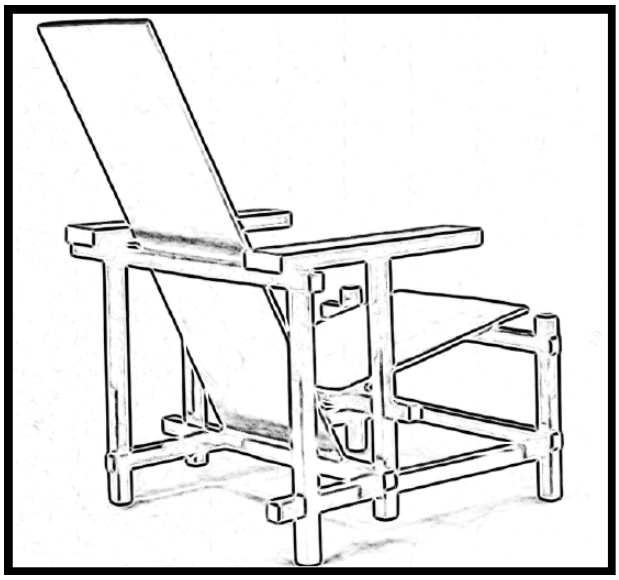
In Figures 12(b)–(d), 13(b)–(d), and 14(b)–(d) we present some of the results obtained by applying our feature detectors to three greyscale images in Figures 12(a), 13(a), and 14(a). The original images are all taken from [MOMA 84] and were digitized using an Envisions 6600S scanner at 200dpi. Feature detection was accomplished by thresholding on the distance from the feature manifold. No further post-processing or sophisticated thresholding techniques (e.g. hysteresis [Canny 86]) were applied. One slight change was made to the raw feature maps for clarity. To make the detected corners in Figure 12(b) visible on the printed page, we first applied non-maximal suppression to localize the corner, and then replaced each detected corner with a 5×5 disc of highlighted pixels.

The outputs of the step edge and corner detectors in Figure 12(b)–(d), and the out-

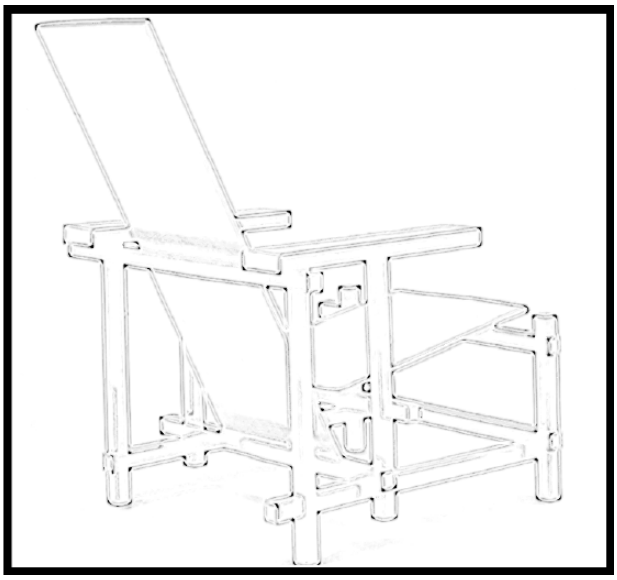
(a) Original image (711 × 661 pixels)



(b) Detected step edges (blue) & corners (red)

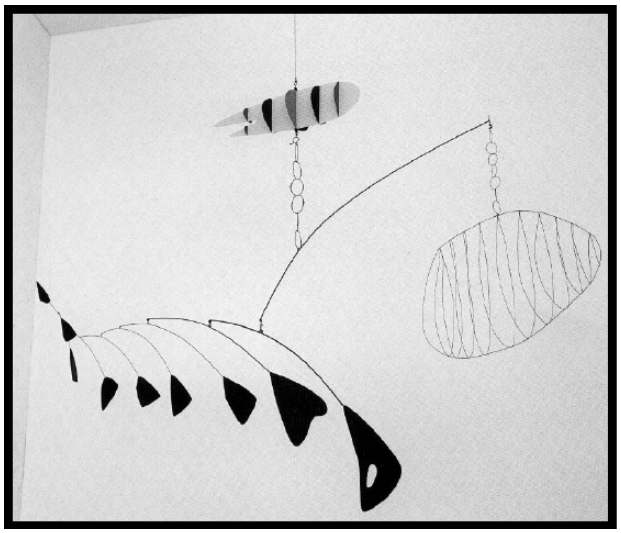


(c) Grey-coded distance to step edge manifold

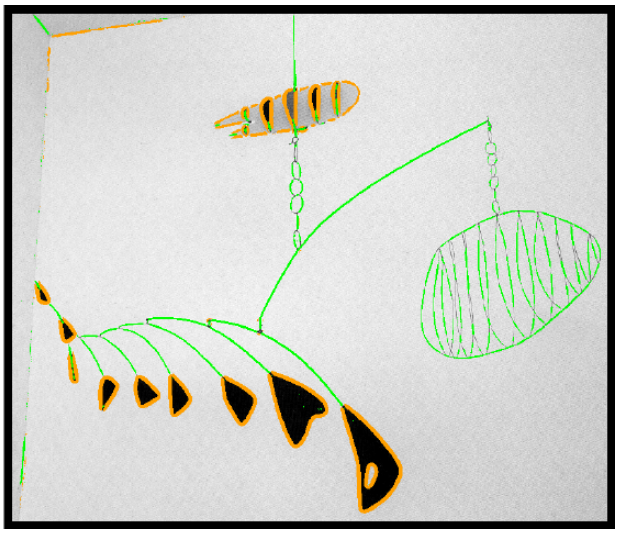


(d) Grey-coded distance to corner manifold

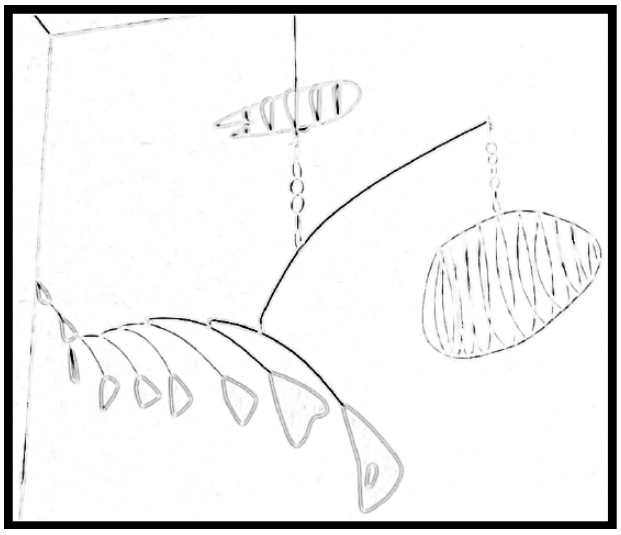
Figure 12: Results of step edge and corner detection for a 711 × 661 image of "Red and Blue," by *Gerrit Rietveld*, circa 1918. The raw (unthresholded) detector outputs in (c) and (d) reflect high accuracy in detection and localization as well as some similarly between the definition of corners and edges as the angle subtended by a corner nears 180^o. In (b), simple thresholds were used to find the dominant feature (if any) at each pixel. Some corners remain undetected due to their angles being less than the 40^o minimum which we imposed in our definition.



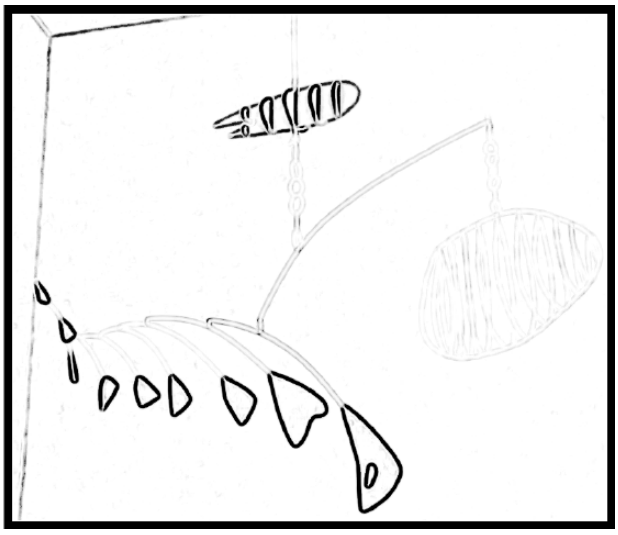
(a) Original image (796×679 pixels)



(b) Detected lines (green) and discs (orange)



(c) Grey-coded distance to line manifold



(d) Grey-coded distance to disc manifold

Figure 13: Results of line and disc detection for a 796×679 image of "Lobster Trap and Fish Tail," by *Alexander Calder,* 1939. 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.

puts of the line and disc detectors in Figure 13(b)–(d) are consistent with the structures of the images. The results in Figure 14(b)–(d) are included simply to convey the richness of information obtained by applying multiple detectors (step edge, line, and corner, in this case) to an image. The distance from manifold and the parameters produced by a detector at a pixel can be valuable in reinforcing or inhibiting the existence of the same or another feature at a neighboring pixel. It is argued, therefore, that multiple feature detection can improve the performance of individual feature detectors.

6 Discussion

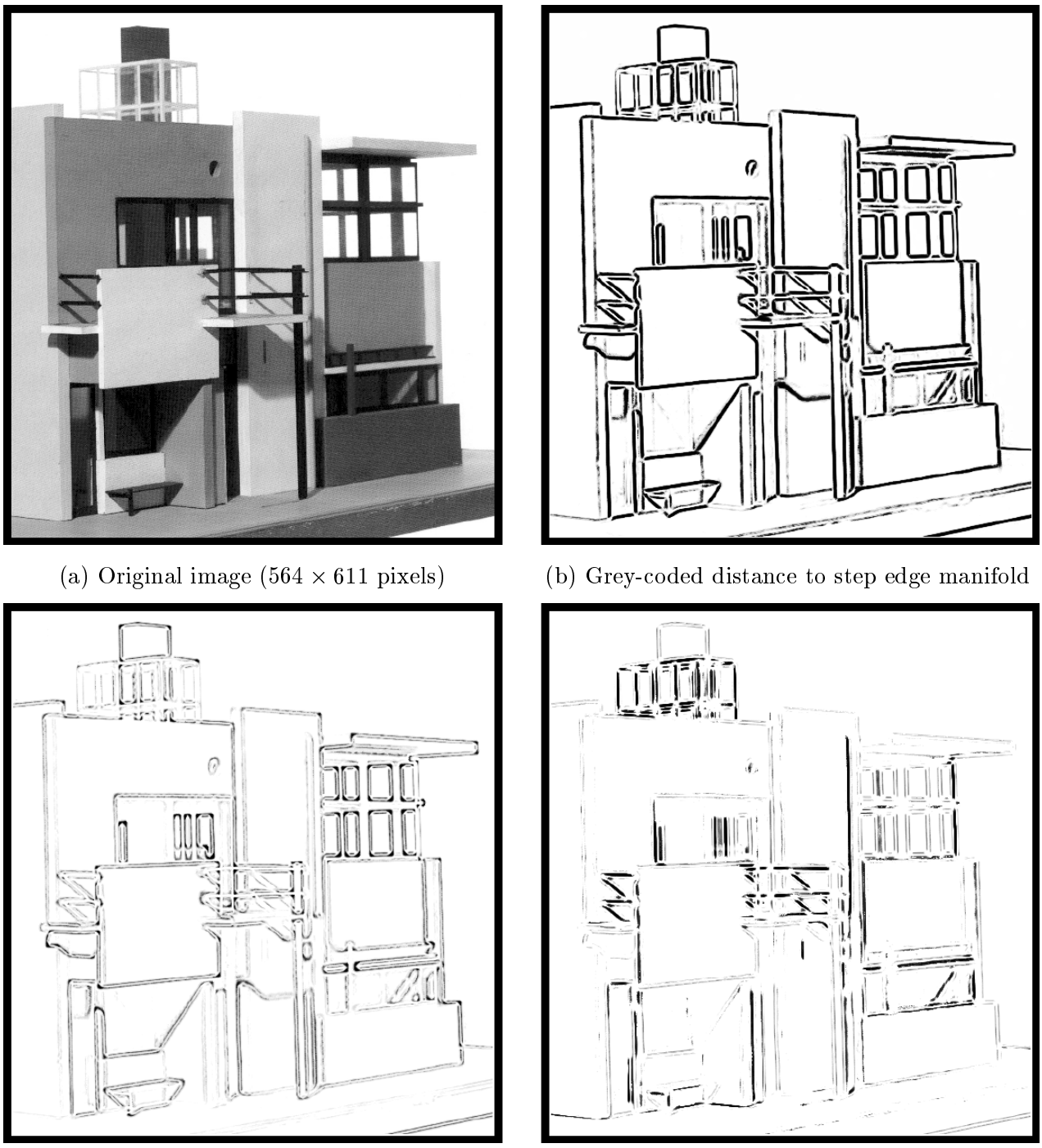
We have proposed an algorithm to generate detectors for arbitrary parametric features. We conclude with a few general observations related to the algorithm:

- The algorithm offers a level of generality that is uncommon in the realm of feature detection. As far as we can ascertain, there is no single technique capable of detecting the five features (step edges, lines, corners, circular discs, and roof edges) we implemented. More importantly, the addition of a new feature to our implementation is simply a matter of writing a single C/C++ function that defines the feature model. Alternatively, features can be derived from experimentally obtained data sets by writing a defining function that transforms (e.g. interpolates, scales, rotates, shifts, and blurs) the stored data appropriately[5]. Further we note that although we present our results entirely in the context of visible-light images, the same approach is directly applicable to any sensing modality, including, X-ray, MR, infrared, ultrasound, and range.

- Most previous detectors have used relatively simple feature models with detection as the main goal and not parameter estimation. Such models do not entirely capture the properties of imaged features. The descriptive nature of our feature models and the incorporation of sensor and optical effects give the features an unusual level of realism. This serves to optimize the robustness of detection and accuracy of parameter estimation.

- The output of the detector consists of detected features, 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 manifold point. Combining the outputs of a number of such detectors,

[5]Recent work [Krumm 96] uses this approach to detect planar patterns with unknown rotation.



(a) Original image (564 × 611 pixels)

(b) Grey-coded distance to step edge manifold

(c) Grey-coded distance to corner manifold

(d) Grey-coded distance to line manifold

Figure 14: Results of step edge, corner, and line detection for a 564 × 611 image of "Schröder House," by *Gerrit Rietveld,* 1924. These results convey the richness of information is obtained from the application of multiple feature detectors, as well as similarities in feature definitions for extreme parameter values. The outputs (b), (c), and (d), together with parameter estimates, could serve as the basis for an effective relaxation scheme that produces a descriptive primal sketch.

each designed for a different feature, yields a large amount of information that would be valuable to a higher level algorithm such as relaxation [Rosenfeld et al. 76]. While existing relaxation algorithms assume a single feature in the image, often the step edge, powerful constraints result from the use of multiple feature detectors. For instance, a corner cannot exist in isolation, but instead must have edges in its vicinity. The incorporation of such constraints into a multi-feature relaxation algorithm should lead to much improved performance over single-feature algorithms.

A Recovering Normalized Parameters

The brightness normalization described in Section 2.4 was used to eliminate two of the parameters for each of our example features. We now describe how to recover the normalized parameters. The computation requires as input, the mean μ and norm 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, it is the peak intensity A and the intensity gradient M. Although we describe the recovery technique in terms of A and B, the same approach works for the roof edge by replacing B with M.

If we work in a continuous domain and fix the parameters $\mathbf{q} - \{A, B\}$ then the mean μ and norm ν are linear functions of A and B:

$$\mu = c_{11} \cdot A + c_{12} \cdot B \tag{16}$$

$$\nu = c_{21} \cdot A + c_{22} \cdot B \tag{17}$$

where $c_{ij} = c_{ij}(\mathbf{q} - \{A, B\})$ are coefficients that depend upon the unnormalized parameters. If we incorporate the discretization of the sensor, the relationships may not hold exactly, however the deviation should be very small. Hence, we use equations (16) and (17) to recover the normalized parameters. If we know c_{ij} we can easily invert equations (16) and (17):

$$A = \frac{c_{22}}{\Delta} \cdot \mu - \frac{c_{12}}{\Delta} \cdot \nu \tag{18}$$

$$B = -\frac{c_{21}}{\Delta} \cdot \mu + \frac{c_{11}}{\Delta} \cdot \nu \tag{19}$$

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 precomputed during the construction of the manifold. For each vector of unnormalized parameters $\mathbf{q} - \{A, B\}$ used to sample the manifold, we evaluate the feature for $A = 0$, $B = 1$ and then normalize as in Section 2.4 to give mean μ_1 and norm ν_1. Repeating for $A = 1$, $B = 1$, we obtain mean μ_2 and norm ν_2 and can then compute:

$$c_{11} = \mu_2 - \mu_1 \tag{20}$$

$$c_{12} = \mu_1 \tag{21}$$

$$c_{21} = \nu_2 - \nu_1 \tag{22}$$

$$c_{22} = \nu_1 \tag{23}$$

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 and then A and B can be recovered from the mean μ the magnitude ν using equations (18) and (19).

We tested the accuracy of normalization inversion for each of our 5 features. After computing the coefficients, c_{ij}, for every manifold sample point using the method described above, we then randomly generated a sequence of 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, we generated the feature according to our feature and sensor models. After normalizing the feature, we then use equations (18) and (19) to recover A and B. The normalization inversion works almost completely without error. The worst performance across all 5 features and over every single feature instance we generated gave an error of 0.02%. The average reconstruction error was an order of magnitude lower.

Acknowledgements

We thank Vic Nalwa for providing an implementation of the Nalwa-Binford edge detector, and Sergio Cuniolo, Michael Heath, Tony Lindenberg, Jose-Maria Montiel, and Geoff West for providing pointers to implementations of Canny edge detectors. Thanks also go to the reviewers who provided a number of very useful comments. This research was supported in parts by ARPA Contract DACA-76-92-C-007, DOD/ONR MURI Grant N00014-95-1-0601,

and an NSF National Young Investigator Award. Hiroshi Murase was supported by the NTT Basic Research Laboratory.

References

[Abdou and Pratt 79] I.E. Abdou and W.K. Pratt, "Quantitative Design and Evaluation of Enhancement/Thresholding Edge Detectors," *Proceedings of the IEEE,* 67:753–763, 1979.

[Baker and Nayar 96] S. Baker and S.K. Nayar, "Pattern Rejection," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* San Francisco, 1996.

[Barbe 80] D.F. Barbe, editor, *Charge-Coupled Devices,* Springer-Verlag, 1980.

[Berzins 84] V. Berzins, "Accuracy of Laplacian Edge Detectors," *Computer Vision, Graphics, and Image Processing,* 27:195–210, 1984.

[Born and Wolf 65] M. Born and E. Wolf, *Principles of Optics,* Permagon Press, 1965.

[Bracewell 78] R.N. Bracewell, *The Fourier Transform and Its Applications,* Second Edition, McGraw-Hill Book Co., 1978.

[Canny 86] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:679–698, 1986.

[Deriche and Giraudon 93] R. Deriche and G. Giraudon, "A Computational Approach for Corner and Vertex Detection," *International Journal of Computer Vision,* 10:101–124, 1993.

[Deutsch and Fram 78] E.S. Deutsch and J.R. Fram, "A Quantitative Study of the Orientation Bias of Some Edge Detector Schemes," *IEEE Transactions on Computers,* 27:205–213, 1978.

[Fram and Deutsch 75] 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,* 24:616–628, 1975.

[Fukunaga 90] K. Fukunaga, *Introduction to Statistical Pattern Recognition,* Academic Press, 1990.

[O'Gorman 78] F. O'Gorman, "Edge Detection Using Walsh Functions," *Artificial Intelligence,* 10:215–233, 1978.

[Haralick 84] R.M. Haralick, "Digital Step Edges from Zero Crossing of Second Directional Derivatives," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 6:58–68, 1984.

[Hartley 85] R. Hartley, "A Guassian-Weighted Multiresolution Edge Detector," *Computer Vision, Graphics, Image Processing,* 30:70–83, 1985.

[Horn 86] B. K. P. Horn, *Robot Vision*, McGraw Hill, 1986.

[Hueckel 71] M.H. Hueckel, "An Operator Which Locates Edges in Digitized Pictures," *Journal of the Association for Computing Machinery,* 18:113–125, 1971.

[Hueckel 73] M.H. Hueckel, "A Local Visual Operator Which Recognizes Edges and Lines," *Journal of the Association for Computing Machinery,* 20:634–647, 1973.

[Hummel 79] R.A. Hummel, "Feature Detection Using Basis Functions," *Computer Graphics and Image Processing,* 9:40–55, 1979.

[Knuth 81] D.E. Knuth, *The Art of Computer Programming, Volume II: Seminumerical Algorithms*, Addison-Wesley, 1981.

[Koenderink 84] J.J. Koenderink, "The Structure of Images," *Biological Cybernetics,*' 50:363–370, 1984.

[Krumm 96] J. Krumm, "Eigenfeatures for Planar Pose Measurement of Partially Occluded Objects," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* San Francisco, 1996.

[Lenz 87] R. Lenz, "Optimal Filters for the Detection of Linear Patterns in 2-D and Higher Dimensional Images," *Pattern Recognition,* 20:163–172, 1987.

[Marr and Hildreth 80] D. Marr and E. Hildreth, "Theory of edge detection," *Proceedings of the Royal Society of London, Series B,* 207:187–217, 1980.

[MOMA 84] *The Museum of Modern Art New York: The History and the Collection,* Harry N. Abrams, Inc., 1984.

[Moravec 77] H.P. Moravec, "Towards Automatic Visual Obstacle Avoidance," In *Proceedings of the Fifth Internation Joint Conference on Artificial Intelligence,* 1977.

[Murase and Nayar 95] H. Murase and S.K. Nayar, "Visual Learning and Recognition of 3-D Objects from Appearance," *International Journal of Computer Vision,* 14:5–24, 1995

[Nalwa 93] V.S. Nalwa, *A Guided Tour of Computer Vision*, Addison-Wesley, 1993.

[Nalwa and Binford 86] V.S. Nalwa and T.O. Binford, "On Detecting Edges," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 8:699–714, 1986.

[Nayar et al. 96] S. K. Nayar, S. Baker, and H. Murase, "Parametric Feature Detection," In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition*, San Fransisco, June 1996.

[Nandy et al. 96] D. Nandy, Z. Wang, J. Ben-Arie, K.R. Rao, N. Jojic, "A Generalized Feature Extractor using Expansion Matching and the Karhunen-Loeve Transform," In *Proceedings of the ARPA Image Understanding Workshop,* Palm Springs, February 1996.

[Nayar et al. 96] S.K. Nayar, S. Baker, and H. Murase, "Parametric Feature Detection," In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition,* San Francisco, 1996.

[Nobel 88] J.A. Nobel, "Finding corners," *Image and Vision Computing,* 6:121–127, 1988.

[Norton 82] H.N. Norton, *Sensor and Analyzer Handbook,* Prentice-Hall, 1982.

[Oja 83] E. Oja, *Subspace Methods of Pattern Recognition*, Research Studies Press, 1983.

[Pratt 90] W.K. Pratt, *Digital Image Processing,* John Wiley & Sons, 1990.

[Prewitt 70] J.M. Prewitt, "Object enhancement and extraction," In *Picture Processing and Psychopictorics,* B.S. Lipkin and A. Rosenfeld, Eds, Academic Press, 1970.

[Rohr 92] K. Rohr, "Recognizing Corners by Fitting Parametric Models," *International Journal of Computer Vision,* 9:213–230, 1992.

[Rosenfeld et al. 76] 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.

[Yianilos 93] P.N. Yianilos, "Data Structures and Algorithms for Nearest Neighbor Search in General Metric Spaces," *Proceedings of the ACM-SIAM Symposium on Discrete Algorithms,* 1993.

[Zucker and Hummel 81] S.W. Zucker and R.A. Hummel, "A Three-Dimensional Edge Operator," *IEEE Transactions on Pattern Analysis and Machine Intelligence,* 3:324–331, 1981.