Correspondence

Edge Detection and Linear Feature Extraction Using a 2-D Random Field Model

Y. T. ZHOU, V. VENKATESWAR, AND R. CHELLAPPA

***Abstract*—Edge detection and linear feature extraction are important components in an image understanding system. In this correspondence, we pose the edge detection problem as one of detecting step discontinuities in the observed correlated image, using directional derivatives estimated with a random field model. Specifically, the method consists of representing the pixels in a local window by a 2-D causal autoregressive (AR) model, whose parameters are adaptively estimated using a recursive least squares algorithm. Due to the modeling assumption, the directional derivatives are functions of parameter estimates. An edge is detected if the second derivative in the direction of the estimated maximum gradient is negatively sloped and the first directional derivative and a local estimate of variance satisfy some conditions. Although the ordered edge detector due to causal model representation has good directional properties, it may not detect well edges of all orientations. Hence, we scan the image in four different directions and take the union of the four edge images as the final output. The performance of the edge detector is illustrated using synthetic and real images. Comparisons to other edge detectors, such as the Marr–Hildreth operator, Canny edge detector, Nevatia–Babu compass operator, and facet model operator are given. Next, a linear feature extractor that operates on the edges produced by the AR model is presented. Comparison to the Nevatia–Babu line finder is given.**

***Index Terms*—Autoregressive models, edge detection, feature extraction, image models, low-level vision.**

I. Introduction

Edge detection and linear feature extraction are important components in an image understanding system. Over the last 20 years, several types of edge detectors have been developed. A summary of various edge detectors developed prior to 1982 may be found in [1] and [2]. Notable among the edge detectors developed over the last eight years are those that use statistical parametric and nonparametric tests [3]–[5], the Marr–Hildreth edge detector [6], the Nevatia–Babu line finder [7], the Canny edge detector [8], the regularization approach [9], and the facet-model-based approach [10].

Marr and Hildreth [6] propose using zero crossings of the operator $D^2G(x, y)$ on the given image where $G(x, y)$ is a 2-D Gaussian distribution and D^2 is the second derivative operator for detecting intensity changes in the image. For the sake of reducing computations, the operator D^2G is replaced by the rotation-invariant operator ∇^2G where ∇^2 is the Laplacian. By using ∇^2G with different widths, zero crossings of ∇^2G are obtained at different scales, and methods to combine these edge outputs for feature extraction are suggested. The Nevatia–Babu technique [7] consists of determining the edge magnitude and direction by convolving the image with a number of masks and thinning and thresholding these edge magnitudes.

In [10], Haralick also detects step edges using zero crossings of second directional derivatives. The required directional derivatives are obtained from an underlying facet model. The occurrence of a digital step edge is detected if the zero crossing of the second directional derivative in the direction of the estimated gradient is negatively sloped. It is postulated that the observed gray levels are generated by the facet model and an additive Gaussian noise variable. Since the directional derivatives are simple functions of model parameters, hypothesis testing procedures for the presence of an edge and confidence intervals for edge orientation have been developed using the methods of linear regression analysis.

In a recent work, Canny [8] defines certain desirable criteria for edge detection. These include good localization, good detection, and a single response to a single edge. These criteria are used in an optimization framework to derive optimal detectors for ridges, roof edges, and step edges. The optimal detector for step edges has an approximate implementation in which edge pixels are the maxima in the gradient field of a Gaussian-smoothed image. The step edge detector performance is further enhanced by extending the operator point spread function along the edge.

In most computer vision systems, the processing does not just stop with the identification of the edge pixels. In some of these systems, the edge pixels are linked to form lines. The lines serve as an intermediate representation, and the higher-level tasks usually utilize these types of intermediate representation.

A simple way to represent lines is to approximate them as piecewise linear segments. A linear segment is a natural representation for the boundaries of many man-made structures such as runways, roads, and buildings. In [11], such a description is used for matching a scene with a map which contains similar descriptions. Linear-feature-based description may also be used in matching stereo images [12] and in establishing correspondence between image frames prior to motion estimation [13].

Several methods are known in the literature [7], [14] for the extraction of straight lines from images. In the Nevatia–Babu method [7], edge pixels (after thinning and thresholding) are linked by marking the locations of the predecessor and the successor of each edge element in a predecessor and a successor file, respectively. The boundary segments are then computed from these files. Each boundary segment is then approximated by a series of piecewise linear segments using a variation of the well-known iterative end point fitting method.

In Burns *et al.* [14], edge pixels are first determined by convolution with two simple 2×2 masks. The pixels are then grouped into line-support regions of similar gradient orientation. Then, the intensity surface associated with each line-support region is approximated by a planar surface. Straight lines are then extracted by intersecting this fitted plane with a horizontal plane representing the average intensity of the region weighted by a local gradient magnitude.

In this correspondence, we pose the edge detection problem as one of identifying step discontinuities in correlated lattice data, the observed real image. The step discontinuities are identified using the magnitude of the first directional derivative and the sign of the second directional derivative along the direction of the maximum first directional derivative. Although the idea of edge detection based on the first and second directional derivatives is not new,

Manuscript received July 14, 1986; revised November 20, 1987. Recommended for acceptance by W. E. L. Grimson. This work was supported in part by the Joint Services Electronics Program at the University of Southern California through the Air Force Office of Scientific Research under Contract F 49620-85-C-0071.

The authors are with the Signal and Image Processing Institute, Department of Electrical Engineering Systems, University of Southern California, Los Angeles, CA.

IEEE Log Number 8823846.

what is new in our approach is the adaptive estimation of these directional derivatives using a 2-D autoregressive (AR) random field model and a recursive least squares (RLS) estimation algorithm. The mask size of our edge detector is fixed (3×3) due to the particular support region of the 2-D AR model, but the weights in the (3×3) mask used for estimating the directional derivatives are space varying. One could possibly choose any one of the possible random field models [15] for estimating the directional derivatives. We have chosen to use the simplest 2-D causal AR model. This model has been used for various applications, such as texture synthesis and coding, image enhancement and restoration, and image classification and segmentation, and as such is a well-researched and documented topic in the image processing/analysis literature. However, the use of the random field model for one of the basic operations in image analysis, viz. edge detection, has received only scattered attention in the literature. In [16], a Bayesian hypothesis testing approach is taken to detect edges in Gaussian images represented by a space-invariant, causal, separable AR model. This formulation leads to a complicated multiple compound hypothesis testing problem whose solution is computationally intensive.

Owing to the model assumption, the first and second directional derivatives are functions of AR parameter estimates. An edge with an orientation α, $0 \leq \alpha \leq (\pi/2)$, is detected if the second directional derivative in the direction of the estimated gradient is negatively sloped and the magnitude of the first directional derivative and a local estimate of sample variance are above some thresholds. To take into account the nonstationarities in the image, the AR parameters are adaptively estimated using an RLS algorithm. Thus, the directional derivatives are also estimated adaptively. In the sequel, we refer to this edge detector as the quarter plane (QP) edge detector.

Although the causal model results in an oriented edge detector with good directional selectivity properties, it may not detect well edges whose orientations are significantly different from that preferred by the edge detector. To circumvent this problem, we scan the image in four different directions and take the union of these outputs as the final edge output. In the sequel, we refer to this edge detector as the full plane (FP) edge detector.

The performance of the FP edge detector is illustrated using synthetic [10] and real images. Comparisons to outputs resulting from other edge detectors such as the Marr–Hildreth operator, Canny operator, Nevatia–Babu compass operator, and facet model operator are also given for one real image. Since it is difficult to evaluate analytically the edge detector output on real images, only visual inspection has been used to assess the quality of the edge detector output. It should be remarked that although we have used a space-variant random field, the resulting edge detector is computationally manageable. Furthermore, since the quality of the edge output is good, a simple algorithm (described in Section IV) is sufficient to extract linear features.

The process of linear feature extraction is broken into three steps. In the first step, each edge pixel is given a line label based on certain templates. The end result of this step is that edge pixels that lie on the same line get the same line label. Furthermore, corresponding to each line label, a complete description of the line in terms of its starting and ending coordinates, average first derivative magnitude, direction, and number of pixels in the line segment (referred to as support) is obtained. In the second step, these lines are further linked based on certain criteria. In the third step, we extend some of the lines to meet at corners. At the same time, we also obtain a description of the corners, which includes the coordinates of the corner and also the lines forming it. The performance of the linear feature extractor is illustrated using real images. A comparison to the Nevatia–Babu linear feature extractor is also given.

The organization of this correspondence is as follows. The design of the QP and FP edge detectors is given in Section II. Experimental results of using the FP detector on synthetic as well as real images are given in Section III. Linear feature extraction based on the AR model edge detector is given in Section IV. Extensions to blurred and noisy images are briefly discussed in Section V.

II. Design of the Edge Detector

A. QP Edge Detector

Let $g(x_0, y_0)$ be the gray level at position (x_0, y_0). We represent $g(x_0, y_0)$ as a 2-D space-varying AR model:

$$g(x_0, y_0) = a_1(x_0, y_0)\, g(x_0 - x, y_0) + a_2(x_0, y_0)\, g(x_0, y_0 - y) + a_3(x_0, y_0)\, g(x_0 - x, y_0 - y) + n(x_0, y_0) \qquad (1)$$

where $n(x_0, y_0)$ is a zero-mean mutually uncorrelated noise array. The space-invariant version of this model (where model parameters a_1, a_2, and a_3 are constants) has found many applications in problems such as image restoration [17], coding [18], texture synthesis [18], classification [19], and segmentation [20].

Assume for the present that the parameters $a_1(x_0, y_0)$, $a_2(x_0, y_0)$, and $a_3(x_0, y_0)$ are known. For simplicity of notation, we suppress arguments (x_0, y_0) in the coefficients $a_1(x_0, y_0)$, $a_2(x_0, y_0)$, and $a_3(x_0, y_0)$. Due to the modeling assumption, the first directional derivative.

$$\frac{\partial g(x_0, y_0)}{\partial \alpha} = \frac{\partial g}{\partial x} \cos \alpha + \frac{\partial g}{\partial y} \sin \alpha \qquad (2)$$

can be written in forms of the AR model in (1) as

$$\frac{\partial g(x_0, y_0)}{\partial \alpha} = \left[a_1 \frac{\partial g(x_0 - x, y_0)}{\partial x} + a_3 \frac{\partial g(x_0 - x, y_0 - y)}{\partial x} \right] \cdot \cos \alpha + \left[a_2 \frac{\partial g(x_0, y_0 - y)}{\partial y} + a_3 \frac{\partial g(x_0 - x, y_0 - y)}{\partial y} \right] \sin \alpha. \qquad (2')$$

The directional derivative is then approximated as

$$0.5\big[a_1[g(x_0 - 2, y_0) - g(x_0, y_0)] + a_3[g(x_0 - 2, y_0 - 1) - g(x_0, y_0 - 1)] \cos \alpha\big] + 0.5\big[a_2[g(x_0, y_0 - 2) - g(x_0, y_0)] + a_3[g(x_0 - 1, y_0 - 2) - g(x_0 - 1, y_0)] \sin \alpha\big] \qquad (2'')$$

where

$$\alpha = \arctan \left(\frac{\dfrac{\partial g}{\partial y}}{\dfrac{\partial g}{\partial x}} \right).$$

Likewise, the second directional derivative

$$\frac{\partial^2 g}{\partial \alpha^2} = \frac{\partial^2 g}{\partial x^2} \cos^2 \alpha + 2 \frac{\partial^2 g}{\partial x \partial y} \cos \alpha \sin \alpha + \frac{\partial^2 g}{\partial y} \sin^2 \alpha \qquad (3)$$

can be written in terms of the AR model in (1) as

$$\frac{\partial^2 g(x_0, y_0)}{\partial \alpha^2} = \left[a_1 \frac{\partial^2 g(x_0 - x, y_0)}{\partial x^2} + a_3 \frac{\partial^2 g(x_0 - x, y_0 - y)}{\partial x^2} \right] \cdot \cos^2 \alpha + 2\left[a_3 \frac{\partial^2 g(x_0 - x, y_0 - y)}{\partial x \partial y} \right] \cdot \cos \alpha \sin \alpha + \left[a_2 \frac{\partial^2 g(x_0, y_0 - y}{\partial y^2} + a_3 \frac{\partial^2 y(x_0 - x, y_0 - y)}{\partial y^2} \right] \sin^2 \alpha. \qquad (3')$$

The second directional derivative is then approximated as

$$\begin{aligned}
&\Big\{\alpha_1[g(x_0 - 2, y_0) - 2g(x_0 - 1, y_0) + g(x_0, y_0)] \\
&\quad + a_3\big[g(x_0 - 2, y_0 - 1) \\
&\quad - 2g(x_0 - 1, y_0 - 1) + g(x_0, y_0 - 1)\big]\Big\} \cos^2 \alpha \\
&\quad + \tfrac{1}{2}\Big\{a_3\big[g(x_0, y_0) - g(x_0 - 2, y_0) - g(x_0, y_0 - 2) \\
&\quad + g(x_0 - 2, y_0 - 2)\big]\Big\} \cos \alpha \sin \alpha \\
&\quad + \Big\{a_2[g(x_0, y_0 - 2) - 2g(x_0, y_0 - 1) + g(x_0, y_0)] \\
&\quad + a_3\big[g(x_0 - 1, y_0 - 2) \\
&\quad - 2g(x_0 - 1, y_0 - 1) + g(x_0 - 1, y_0)\big]\Big\} \sin^2 \alpha. \qquad (3'')
\end{aligned}$$

For the sake of completeness, the details of how the first and second directional derivatives are approximated as above are shown in the Appendix. We mark the pixel at (x_0, y_0) as an edge pixel if $|\partial g/\partial \alpha| > t_1$ (threshold to circumvent noise in the image), $(\partial^2 g/\partial \alpha^2) < 0$, $0 \le \alpha \le (\pi/2)$, and a local estimate of sample variance ρ computed in 3×3 window is greater than a threshold t_2. The thresholds t_1 and t_2 are preset. Fig. 1 illustrates the representation of images using (1). The QP edge detector concentrates on detecting edges whose orientations are orthogonal to the radius vector $r = [(\partial g/\partial x)^2 + (\partial g/\partial y)^2]^{1/2}$ at an angle α indicated in Fig. 1. The orientations of some typical edges detected by the QP detector are shown in Fig. 2, and Fig. 3 illustrates the orientations of some typical edges not detected well by the QP detector; this phenomenon is referred to as the "snake effect" in [21]. Our edge detector is "local" in that the computation of the first and second derivatives involves intensity values in the 3×3 mask $\{(x_0, y_0), (x_0 - 1, y_0), (x_0, y_0 - 1), (x_0 - 1, y_0 - 1), (x_0 - 2, y_0), (x_0 - 2, y_0 - 1), (x_0, y_0 - 2), (x_0 - 1, y_0 - 2), (x_0 - 2, y_0 - 2)\}$. However, these derivatives are also functions of estimates of parameters a_1, a_2, and a_3, which depend on all gray levels prior to position (x_0, y_0) in the raster scanning model, and thus include "global" information. The parameters a_1, a_2, and a_3 are estimated from the given image. Since real images are nonstationary, it is preferable to estimate these parameters adaptively. We use the well-known RLS approach [22] for this problem.

Let

$$\theta(x_0, y_0) = [a_1(x_0, y_0), a_2(x_0, y_0), a_3(x_0, y_0)]^T$$

$$\phi(x_0, y_0) = [g(x_0 - 1, y_0), g(x_0, y_0 - 1), g(x_0 - 1, y_0 - 1)]^T$$

The recursive algorithm to estimate $\theta(x_0, y_0)$ is,

$$\begin{aligned}
\tilde{g}(x_0, y_0) &= g(x_0, y_0) - \theta(x_0 - 1, y_0)^T \phi(x_0, y_0) \\
\theta(x_0, y_0) &= \theta(x_0 - 1, y_0) \\
&\quad + \frac{P(x_0 - 1, y_0)\, \phi(x_0, y_0)\, \tilde{g}(x_0, y_0)}{1 + \phi(x_0, y_0)^T P(x_0 - 1, y_0)\, \phi(x_0, y_0)} \\
P(x_0, y_0) &= P(x_0 - 1, y_0) \\
&\quad - \frac{P(x_0 - 1, y_0)\, \phi(x_0, y_0)\, \phi(x_0, y_0)^T P(x_0 - 1, y_0)}{1 + \phi(x_0, y_0)^T P(x_0 - 1, y_0)\, \phi(x_0, y_0)} \\
&\quad \text{for } x_0 = 2, \cdots, n \text{ and } y_0 = 2, \cdots, n. \qquad (6)
\end{aligned}$$

The initial conditions affect only the first few lines in an image. In our implementation, we chose a_1, a_2, and a_3 to be 0.33 each. Each diagonal element of the matrix P was set to 1.01×10^{-6}, and the nondiagonal elements were set to -0.5×10^{-6}. This ensures that the matrix P is positive definite. The relative importance of the initial values decays as more pixels are visited.

It should be remarked that the above estimation method does not

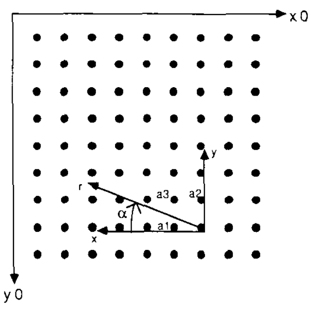


Fig. 1. Image representation using the AR model.

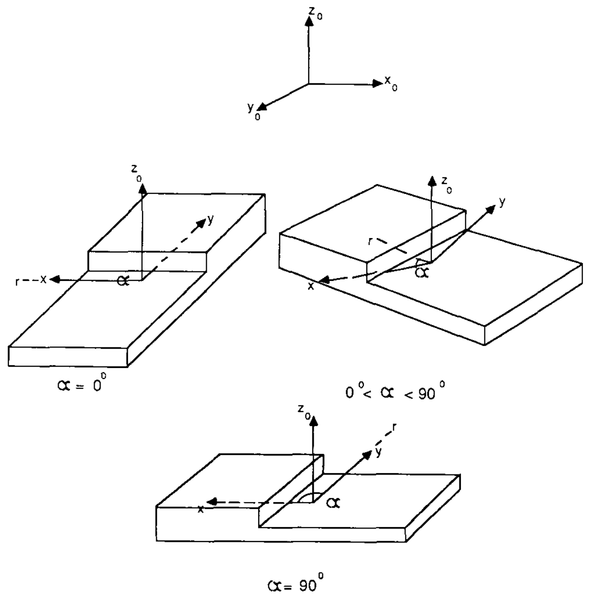


Fig. 2. Typical edges detected by the QP edge detector.

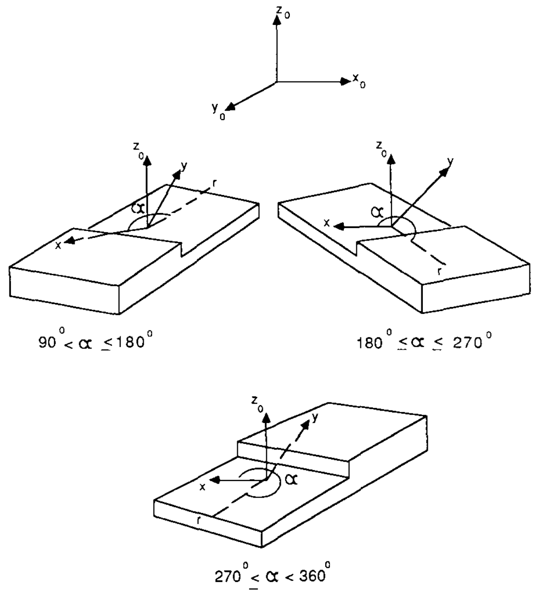


Fig. 3. Typical edges not detected well by the QP edge detector.

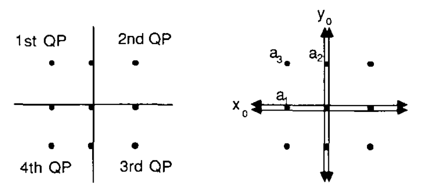


Fig. 4. Representation of the four QP edge detectors which combine to constitute an FP edge detector.

assume any specific probability distribution for the image pixels. However, if the image is Gaussian, one can show that the estimates given above possess several good properties [23].

The dependence of $\theta(x_0, y_0)$ on $\theta(x_0 - 1, y_0)$ appears to give the impression that the filtering is 1-D, which is incorrect. This dependence only indicates horizontal scanning. Also, the parameter vector $\theta(x_0, y_0)$ is dependent not only on $\theta(x_0 - 1, y_0)$, but on all the previous parameter vectors and the error covariances $P(\cdot)$ estimated as the image pixels are visited in a raster scan mode. Asymptotic properties of the RLS estimate are extensively discussed in [22].

B. FP Edge Detector

To detect well edges whose typical orientations are in Fig. 3, we need to implement four QP edge detectors and take the union of the edge detectors. The representations of the four QP edge detectors are in Fig. 4. The first QP edge detector is implemented by scanning the image left to right and top to bottom. The other three QP edge detectors are implemented by scanning top to bottom and right to left, right to left and top to bottom, and bottom to top and left to right.

III. Experimental Results

A. FP Edge Detector Results on Checkerboard Image

We illustrate the performance of the space-varying AR-model-based edge detector on the checkerboard image [10]. In this image, the true edge pixels are one pixel wide and are at the bottom of the step. The noise-free synthetic image used in the experiment and the result of applying the FP edge detector on this image are given in Fig. 5. It is evident that the FP edge detector detects all the edge points in the image. The three performance measures mentioned in [10] were evaluated for this example. They are the conditional probability of an assigned edge given a true edge, the conditional probability of a true edge given an assigned edge, and the error distance. The numerical values of these measures are 1.00, 1.00, and 0.0, respectively.

The results of applying the FP edge detector on a noisy checkerboard image are also shown in Fig. 5. The noisy checkerboard image was created exactly along the lines of [10]. Presmoothing using a recursive cross-shaped median filter was done, followed by edge detection. The three performance measures mentioned above computed for this case are 0.78, 0.78, and 1.68, respectively. Due to the differences in the definition of our true edges and Haralick's true edges, these performance measures are not directly comparable to the performance measures reported in [10], [24], [25]. It should be pointed out that the performance of a model-based edge detector is very much dependent on how well the model fits the given data. Since a structured image such as the checkerboard image is not fitted well by an AR model, the results in Fig. 5(d) are not as good as the results reported in [25]. However, the performance of the FP edge detector on real images is very satisfactory.

B. FP Edge Detector Results on Real Images

The performance of the FP edge detector on real images is illustrated in Figs. 6 and 7. In Fig. 6, the preprocessed pentagon image and the edge image are given. Preprocessing was done by filtering with a cross-shaped 3×3 median filter, followed by min-max replacement in a 3×3 neighborhood. A pixel was declared as an edge pixel if the estimated first derivative magnitude at that pixel was greater than a threshold of 4.0 (the threshold t_1) and a local estimate of the sample variance ρ computed in a 3×3 window was greater than a threshold of 70.0 (the threshold t_2). In Fig. 7, the preprocessed airport image and the edge image are given. For this image, the threshold t_1 was set to 3.0, and the threshold t_2 was set to 100.0. It should be mentioned that the edge images displayed in Figs. 6 and 7 are the direct outputs of the FP edge detector. No postprocessing, such as thinning, was done.

C. Comparisons to Other Edge Detectors

Comparison and evaluation of an edge detector output on real images is a difficult problem [24]. For the sake of completeness, we give the outputs of the Marr–Hildreth edge detector [26], Canny edge detector [8], facet model edge detector [10], and Nevatia–Babu line finder [7] applied to the preprocessed airport image in Fig. 7. The same preprocessed image was used as the input for all the edge detectors. The preprocessing is similar to the technique described in [10]. We used a cross-shaped median filter instead of a square-shaped median filter. The implementation details of these four edge detectors are briefly given below.

1) Marr–Hildreth Model: For the Marr–Hildreth edge operator, we used the implementation of the Institute for Robotics and Intelligent Systems (IRIS) at the University of Southern California (USC). Details of the implementation can be found in [26]. The resulting edge image is shown in Fig. 8. We used an σ of 2.0 and a filter size of 16×16. As the higher order σ's misplaced the edges, they were discarded.

2) Facet Model: In our implementation of the facet model, we used the first ten polynomials (cubic order or less), i.e., the discrete orthogonal polynomial set $\{1, r, c, r^2 - \mu_2/\mu_0, rc, c^2 - \mu_2/\mu_0, r^3 - (\mu_4/\mu_2)r, c(r^2 - \mu_2/\mu_0), r(c^2 - \mu_2/\mu_0), c^3 - (\mu_4/\mu_2)c\}$, where $\mu_k = \Sigma_{s \in R} s^k$ and R is the discrete integer index set that is symmetric, i.e., $r \in R$ implies $-r \in R$. For example, for a 5×5 filter, $R = \{-2, -1, 0, 1, 2\}$.

We applied the facet model to the preprocessed airport image. A pixel was declared as an edge pixel if along the direction of the maximum first derivative a negatively sloped zero crossing of the second directional derivative occurred within a distance of 0.9 pixels of the center pixel and the gradient magnitude was greater than 9. The resulting edge image is shown in Fig. 8.

3) Canny Operator: The software for the Canny operator was obtained directly from J. Canny. The procedure involves convolution with a first derivative of a Gaussian mask, estimation of noise energy, application of nonmaximum suppression to gradients, extension of contours using hysteresis, and thinning of the contours [8]. The edge image resulting from the Canny operator is shown in Fig. 8. We used an σ of 0.5. The image was thresholded with hysteresis [8], using a low threshold of 200 and a high threshold of 400. This is comparable to the edge output resulting from the AR-model-based approach.

4) Nevatia–Babu Compass Operator: The implementation of this method was done by using the linear feature extraction software of IRIS at USC. Edges are detected by first convolving an image with a number of edge masks and then thinning and thresholding these edge magnitudes. Details of this implementation can be found in [7]. We show the results of the edge detection process after thinning and thresholding at an edge magnitude level of 10 in Fig. 8.

D. Comments

As remarked before, the comparison of edge detectors on synthetic and real images is a very difficult problem. Although comparisons between the Marr–Hildreth and facet model approaches have been reported in [10], [24], [25] based on the three performance measures originally proposed in [10], any conclusions based on these comparisons of synthetic images have limited value. The

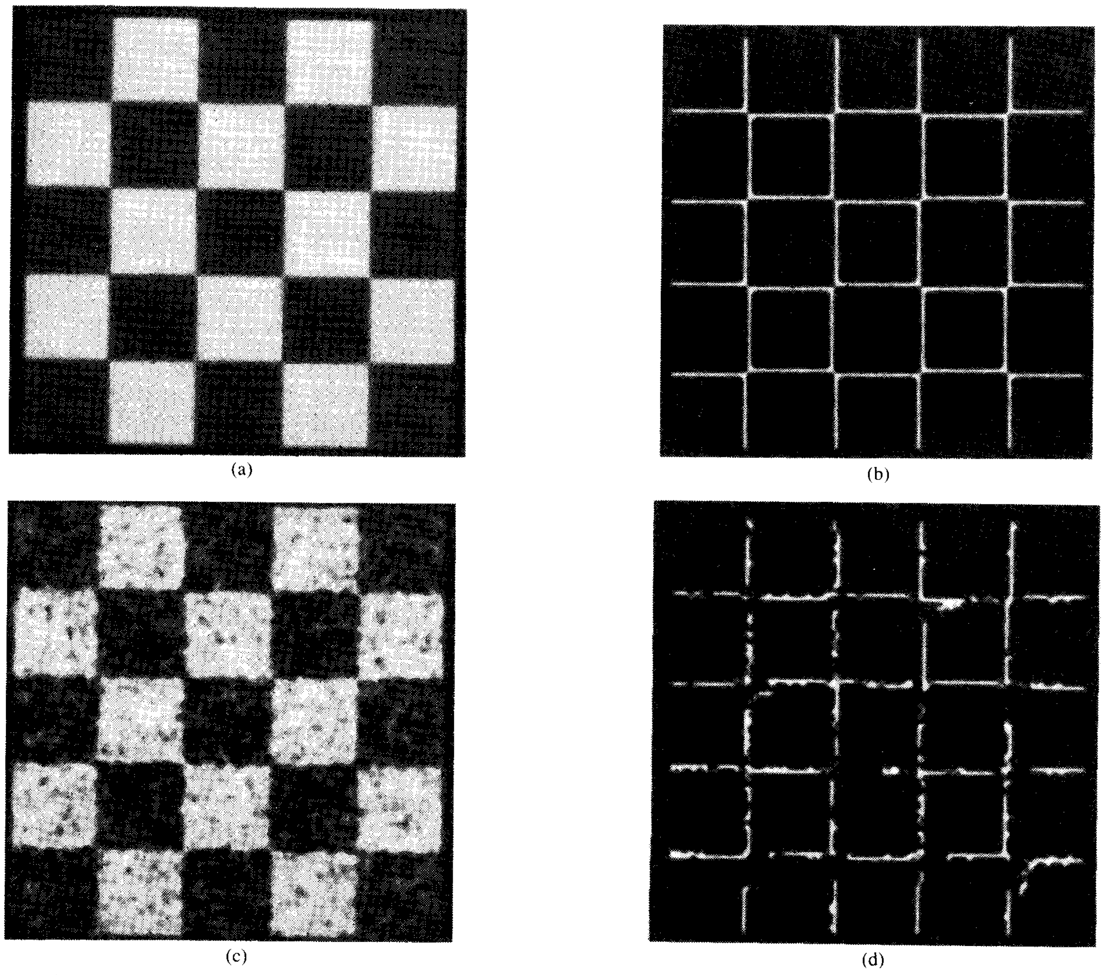


Fig. 5. Edge detection on a synthetic checkerboard image. (a) Noise-free image. (b) True edge image. The FP edge detector output is the same as (b). (c) Noisy image. (d) FP edge detector output for (c) after preprocessing.

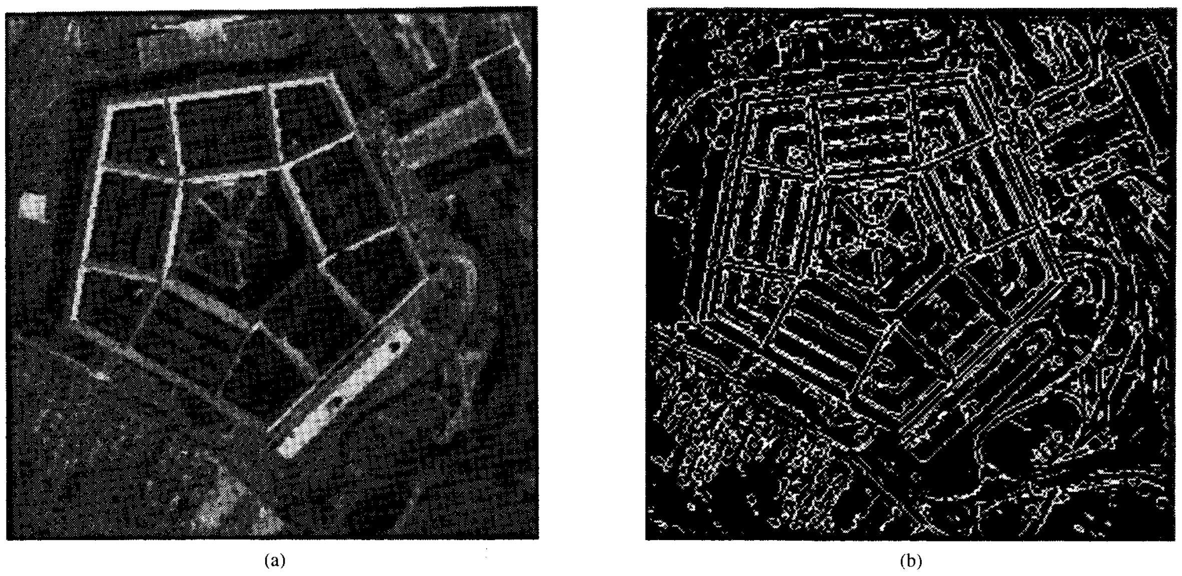
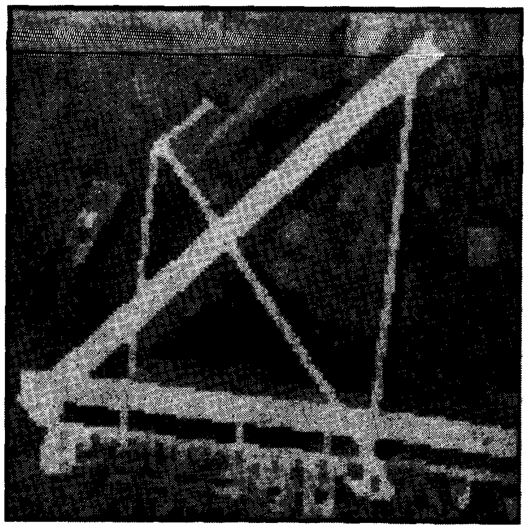
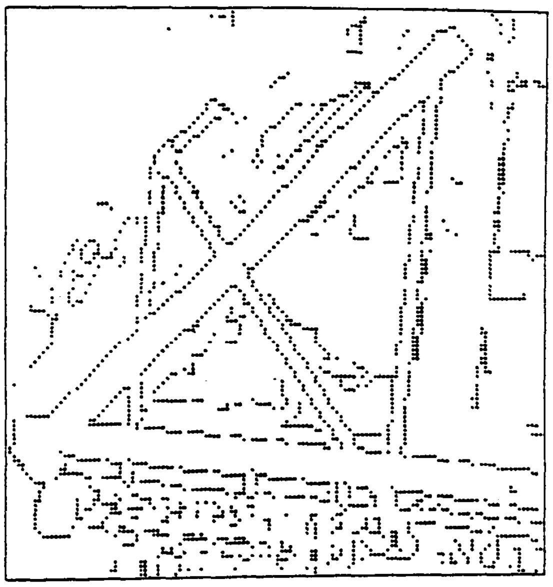


Fig. 6. Results of applying the FP edge detector on the preprocessed pentagon image. (a) The preprocessed image. (b) Edge output from the preprocessed image.



(a)



(b)

Fig. 7. Results of applying the FP edge detector on the preprocessed airport image. (a) The preprocessed image. (b) Edge output from the preprocessed image.

reason is that there is no simple extrapolation of conclusions based on synthetic images to real images! This only makes comparisons of real images even more difficult. One can compare visually the edge outputs produced by different methods. One can also compare based on the number of steps required to produce the final results. Thus, based on visual inspection of Fig. 8, we observe that the Canny operator and AR model edge detector produce comparable results, followed by the facet model, Nevatia-Babu, and Marr-Hildreth operators. When an improved version [29] of the facet model edge detector was tried, edges thinner than in Fig. 8 were obtained. However, all the edge detectors referred to above need one or two thresholds to be preset; we have observed in our experiments (we are certain that previous researchers have too) that results vary considerably based on the thresholds used. It is very difficult to derive quantitatively the threshold values from the real image. Given that the observed edge outputs are dependent on thresholds, one has to know *a priori* where the true edges are so that appropriate thresholds may be preset to detect them!

Another point to note is that computer programs that implement edge detectors usually include some heuristic steps (not discussed in the original papers). It is a reality that these heuristic steps improve the quality of edge outputs. The improvements are also due to continuous refinements of the software (probably tuned to specific images), which are not part of the underlying theory behind the development of this edge detector. Such progressive refinements can be thought of for all the edge detectors included in the comparison. We wish to point out that the edge outputs produced by the AR model and the facet model given in Figs. 7 and 8 are the direct results of steps described in this paper and in Section III-C2), respectively.

IV. Linear Feature Extraction

The block diagram of the implementation is shown Fig. 9. The preprocessing was done by using a cross-shaped median filter followed by min-max replacement (i.e., a pixel value was replaced by the maximum or minimum in the neighborhood, whichever was closer in value to the original value) in a 3×3 neighborhood.

A. Labeling the Edge Pixels

The full plane edge detector can detect the gradient direction in a continuum of $-180°$ to $+180°$. But to simplify the linking procedure, we quantize the gradient direction to one of eight bins (as in Fig. 10). We scan the edge image left to right and top to bottom. To each edge pixel, we assign a line label based on certain templates. We have one set of templates for each direction bin. The templates for four of the direction bins are shown in Fig. 11. These templates essentially check if the current edge pixel is on the same line as a neighboring edge pixel (note that this neighborhood is a subset of the area previously scanned). If it is, then the current edge pixel is assigned the same line label as the neighboring pixel. If none of the templates matches, then the current pixel is probably the starting of a new line, so it is assigned a new line label. A line label is just an integer number. The first new line label is 1, the second is 2, and so on.

We also maintain an array of *structures*, called the line structures array. The array subscript is a line label. When we assign a new line label for an edge pixel, we also activate the structure corresponding to this line label. The line structure has various fields that give a complete description of the line to which it corresponds. The Start_Row and Start_Col fields are assigned the coordinates of the edge pixel that activates the structure. The direction field is initialized to the edge pixel direction. The support field (which is the number of pixels assigned the line label) is initialized to 1. The magnitude field (which is the average first-derivative magnitude along the line segment) is initialized to the edge strength of this pixel. The End_Row and End_Col fields are also initialized to be the coordinates of this edge pixel. When another pixel gets the same line label, then all the fields, except for the Start_Row and Start_Col fields, are appropriately updated.

The end result of this labeling process is the production of two descriptions for the lines. At the pixel level, each edge pixel is associated with a line label. At the line level, there corresponds to this line label a data structure that gives a complete description of the line it represents.

B. Linking the Lines

The lines formed in the labeling process of step1 are now to be linked. For each line, we examine a small neighborhood near both the ends. The neighborhoods examined depend on the direction of the line. These neighborhoods were chosen after careful examination of a substantial number of test cases. The neighborhoods for two typical cases are shown in Fig. 12.

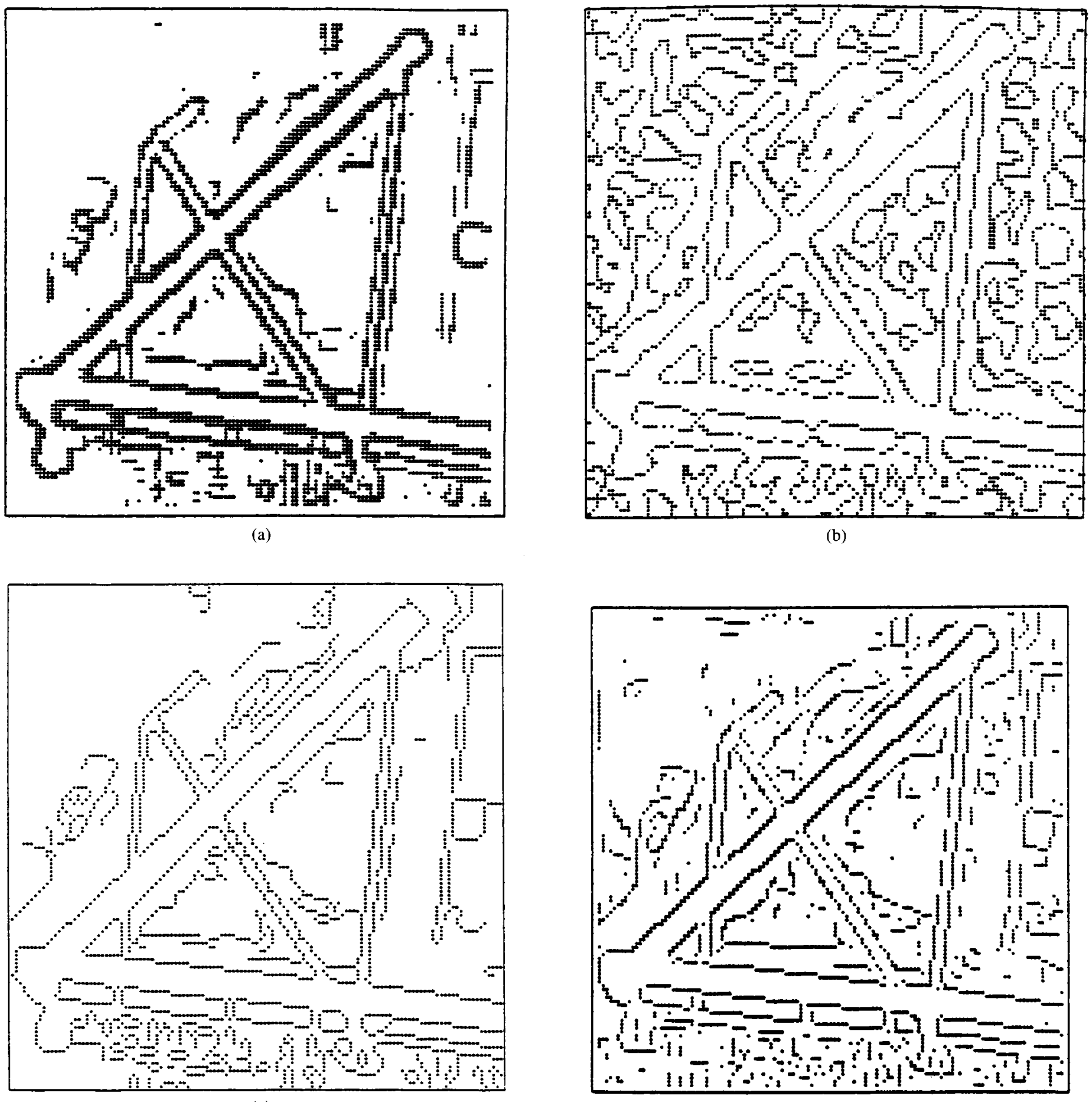


Fig. 8. Comparison to other edge detectors. (a) Result of applying the facet model edge detector. (b) Result of applying the Marr–Hildreth operator. (c) Result of applying the Canny edge operator. (d) Result of applying the Nevatia–Babu compass operator. The Canny and Nevatia–Babu operators include a thinning procedure.

In the neighborhood corresponding to the starting point of a line (say line1) in the label image, we search for another line label and check to see if it corresponds to the end point of some other line. Suppose such a line (say line2) exists in the neighborhood. We then check to see if its direction is approximately the same as that of line1 and its magnitude is at least half or at most twice that of line1. If both these conditions are satisfied, then the two lines are merged. The linking procedure is performed twice. In the first pass, the directions of the lines to be linked should not differ by more than 8°. In the second pass, these directions should not be more than 13° apart.

In the merging process, the Start_Row and Start_Col fields of line 1 are updated to be those of line2. The support field of line2 is made zero to signify that it is being merged, and the magnitude field is made equal to the label of line2. The fields of line1 are appropriately updated. In the future when we encounter the line2

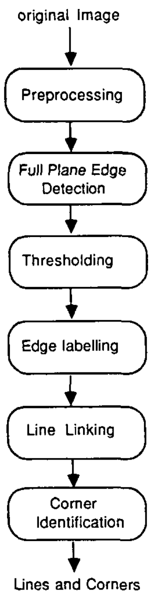


Fig. 9. Block diagram of the entire procedure of linear feature extraction.

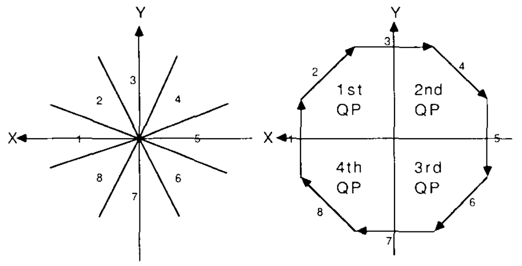


Fig. 10. Edge directions.

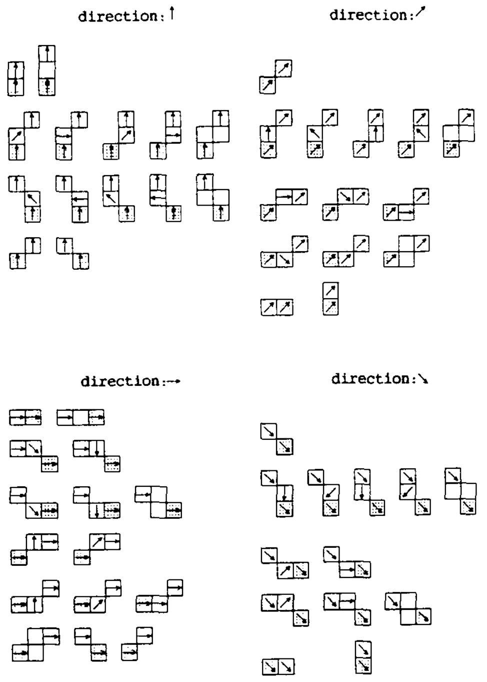


Fig. 11. Templates for four directions. The squares correspond to pixels. The dotted square represents the current edge pixel. The arrows show the direction of the maximum first derivative.

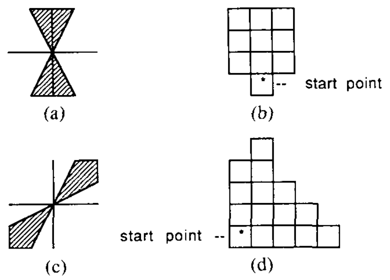


Fig. 12. Neighborhoods for linking. Lines with a direction in the shaded region shown in (a) have the neighborhood for the start point shown in (b). Lines with a direction in the shaded direction shown in (c) have the neighborhood for the start point shown in (d).

label in a neighborhood, we can identify that it has been merged to line1 and get its characteristics from various fields of line1.

At the end of the linking procedure, we have a description of the lines. But what we do not have is information on the corners. By extending some of the lines, we identify most of the corners. Each line structure has two fields, which correspond to corners at the ends of a line. Each corner is assigned a number. So the numbers in the corner fields correspond to individual corners. The corner fields are initialized to 0. The corner count is initialized to 1.

We now extend the lines formed in step2 if there is a corner near any of the ends. For this, we examine a small neighborhood near the end of each line. The neighborhood examined is the same as that in step2. We check if there is any line in this neighborhood that can intersect with the current line. Instead of being merged (as in step2), the two lines are extended to meet at a corner. At the same time, the appropriate corner fields of the two lines are assigned a corner number. The corner number is the same as the corner count. The corner count is then incremented.

Just as with an array of line structures, we have an array of corner structures. The array subscript corresponds to a corner number. Each array element has as its fields the row and column coordinates and also the labels of the two lines intersecting to form that corner. As new corners are discovered, new array elements in the corner structure array are initialized.

It should be mentioned that the corner description is not complete because some corners can be formed by more than two lines. However, only two of these lines would be included in the corner description. Since our main purpose is to identify the corners and since most of the corners are formed by two lines, the above representation is quite adequate.

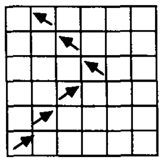
Fig. 13 illustrates the complete data structure.

The final result is a rich description of the lines in the image. At the pixel level, we have line labels associated with each edge pixel. At the line level, we have a description of the line in terms of its various characteristics. Also derived is a list of corners. Given a line, we can find the corner labels associated with the ends of the lines. Given the corner labels, we can determine both the lines forming the corner.

This description will be extremely useful in further processing. For example, if we wanted to find antiparallels, we could search at

0	8	0	0	0	0
0	0	9	0	0	0
0	0	0	7	0	0
0	0	3	0	0	0
0	4	0	0	0	0
5	0	0	0	0	0

(a)



(b)

0	A	0	0	0	0
0	0	A	0	0	0
0	0	0	A	0	0
0	0	B	0	0	0
0	B	0	0	0	0
B	0	0	0	0	0

(c)

(d)

```
Line [A] = {
          Start_Row 0;
          Start_Col 1;
          End_Row 2;
          End_Col 3;
          Direction -45 ;
          Magnitude 8;
          Start_Corner 0;
          End_Corner C1;
}

Line [B] = {
          Start_Row 3;
          Start_Col 2;
          End_Row 5;
          End_Col 0;
          Direction 45;
          Magnitude 4;
          Start_Corner C1;
          End_Corner 0;
}
```

note: Magnitude is the average first derivative magnitude along the line segment

(e)

```
Corner [C1] = {
          Row 2;
          Col 3;
          Line1 A;
          Line2 B;
}
```

Fig. 13. Illustration of the data structure for a simple 6 × 6 image. (a) The edge magnitudes of the image. (b) The edge directions. (c) The line labels. (d) The line structure. (e) The corner structure.

the pixel level for a line label and then get the full characteristics of this line from the array element corresponding to this line label.

We illustrate the results of linear feature extraction on the airport image in Fig. 14 and the pentagon image in Fig. 15. For the sake of comparison, we also present the results of applying the Nevatia-Babu line finder on these images in Fig. 16.

V. Conclusions and Extensions

In this correspondence, we have presented a simple edge detector based on a space-variant random field model. We have illustrated the performance of this edge detector using synthetic and real images. The edge detector adapts itself to take into account nonstationarities in the image by virtue of the space-varying representation and adaptive estimation method.

We have recently extended our edge detector to handle noisy and blurred images. In this work [27], we assumed that the original image obeys a model such as (1), but we observe only

$$f(x_0, y_0) = h(x_0, y_0) * g(x_0, y_0) + w(x_0, y_0) \tag{7}$$

where $w(x_0, y_0)$ is an additive noise array of unknown variance γ and $h(x_0, y_0)$ is a known space-invariant point spread function. The proposed approach involves using a reduced update Kalman filter [28] to estimate simultaneously the parameters a_1, a_2, a_3, and γ and a local state vector made of the pixels in the mask $\{(x_0, y_0), (x_0 - 2, y_0), (x_0, y_0 - 2), (x_0 - 2, y_0 - 2)\}$. The algorithm is designed such that the parameters needed for edge detection at pixel $(x_0 - 1, y_0 - 1)$ would have been estimated by the reduced update Kalman filter at pixel (x_0, y_0). Details of this algorithm are presented in [27].

We are also looking into generalizations of this edge detector to multiresolution images and for images corrupted by heavy tailed noise or outliers leading to so-called robust edge detectors.

Appendix

In this Appendix, we show how to approximate the first and second directional derivatives of the gray level $g(x_0, y_0)$ using the AR model (1). It is well known that the first and second derivatives of a function $g(x, y)$ can be approximated by the symmetric differences

$$\frac{\partial g(x, y)}{\partial x} \cong \frac{1}{2}[g(x+1, y) - g(x-1, y)]$$

$$\frac{\partial^2 g(x, y)}{\partial x^2} \cong g(x+1, y) - 2g(x, y) + g(x-1, y)$$

$$\frac{\partial^2 g(x, y)}{\partial x \partial y} \cong \frac{1}{4}[g(x+1, y+1) - g(x-1, y+1) - g(x+1, y-1) + g(x-1, y-1)].$$

In the above equations, the spacing between the picture cell centers has been assumed to be 1.

If the coordinates x and y are substituted with $-x$ and $-y$, respectively, then the derivatives become

$$\frac{\partial g(-x, -y)}{\partial x} \cong \frac{1}{2}[g(-x-1, y) - g(-x+1, y)]$$

$$\frac{\partial^2 g(-x, -y)}{\partial x^2} \cong g(-x-1, y) - 2g(-x, y) + g(-x+1, y)$$

$$\frac{\partial^2 g(-x, -y)}{\partial x \partial y} \cong \frac{1}{4}[g(-x-1, -y-1) - g(-x+1, -y-1) - g(-x-1, -y+1) + g(-x+1, -y+1)].$$

Replacing $-x$ and $-y$ by $x_0 - x$ and $y_0 - y$, respectively, and rearranging terms, we can write

$$\frac{\partial g(x_0 - x, y_0 - y)}{\partial x} \cong \frac{1}{2}[g(x_0 - x - 1, y_0 - y) - g(x_0 - x + 1, y_0 - y)] \tag{8}$$

$$\frac{\partial^2 g(x_0 - x, y_0 - y)}{\partial x^2} \cong g(x_0 - x + 1, y_0 - y) - 2g(x_0 - x, y_0 - y) + g(x_0 - x - 1, y_0 - y) \tag{9}$$

$$\frac{\partial^2 g(x_0 - x, y_0 - y)}{\partial x \partial y} \cong \frac{1}{4}[g(x_0 - x + 1, y_0 - y + 1) - g(x_0 - x - 1, y_0 - y + 1) - g(x_0 - x + 1, y_0 - y - 1) + g(x_0 - x - 1, y_0 - y - 1)]. \tag{10}$$

By setting $y = 0$ in (8) and (9), we have

$$\frac{\partial g(x_0 - x, y_0)}{\partial x} \cong \frac{1}{2}[g(x_0 - x - 1, y_0) - g(x_0 - x + 1, y_0)] \tag{11}$$

$$\frac{\partial^2 g(x_0 - x, y_0)}{\partial x^2} \cong g(x_0 - x + 1, y_0) - 2g(x_0 - x, y_0) + g(x_0 - x - 1, y_0). \tag{12}$$

Using symmetry arguments, we can derive $\partial g(x_0, y_0 - y)/\partial y$,

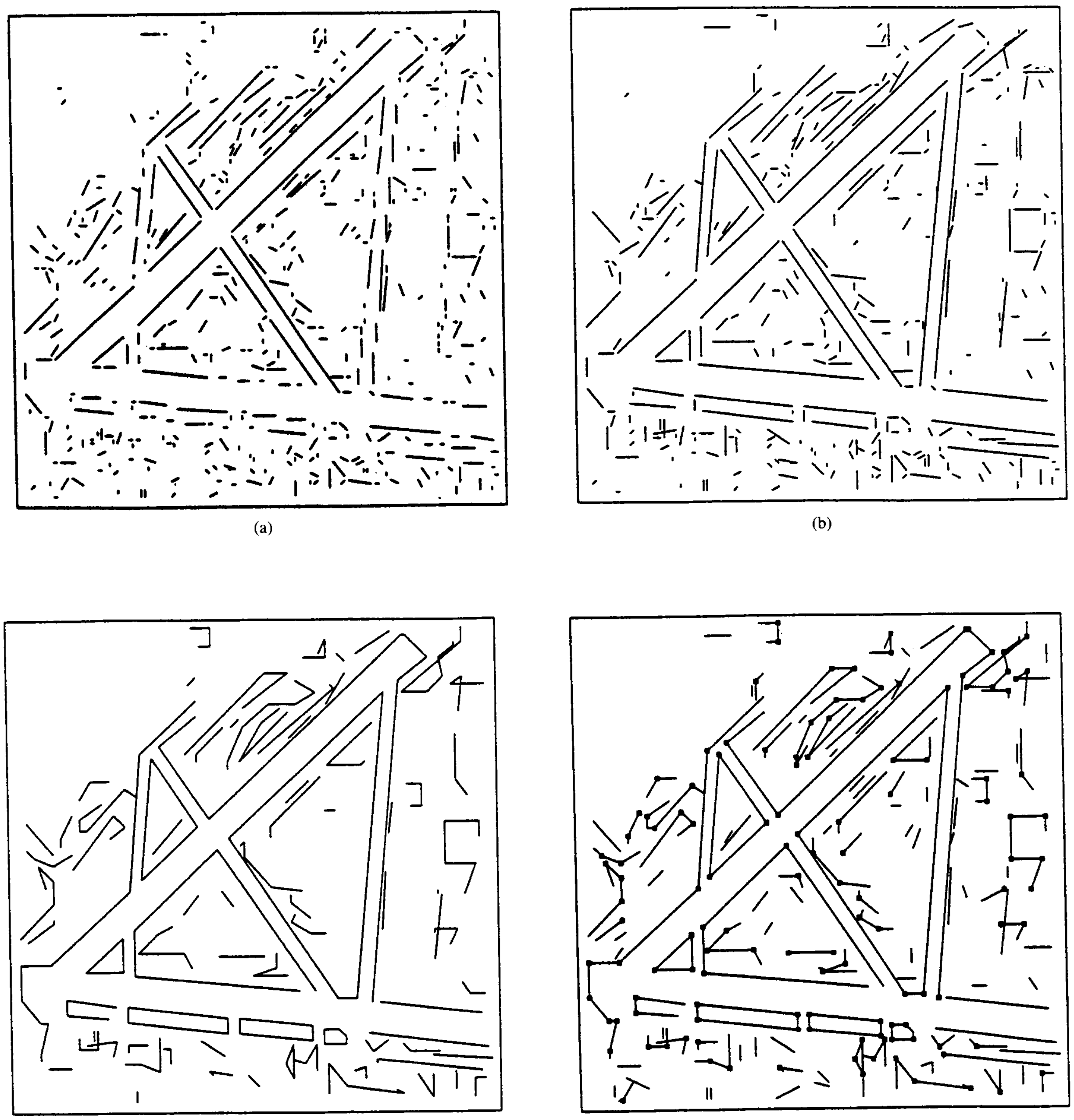
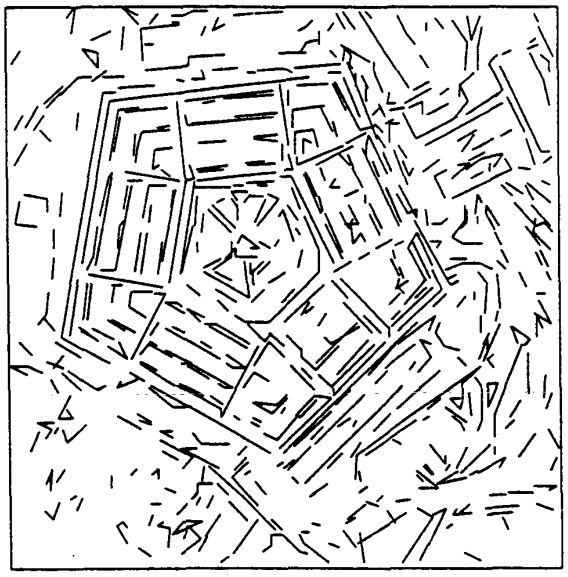


Fig. 14. Illustration of linear feature extraction on the airport image. (a) Result after labeling. (b) Result after linking. (c) Result after extending the lines. (d) The corners identified in the image are shown as small squares in this figure.


(a)

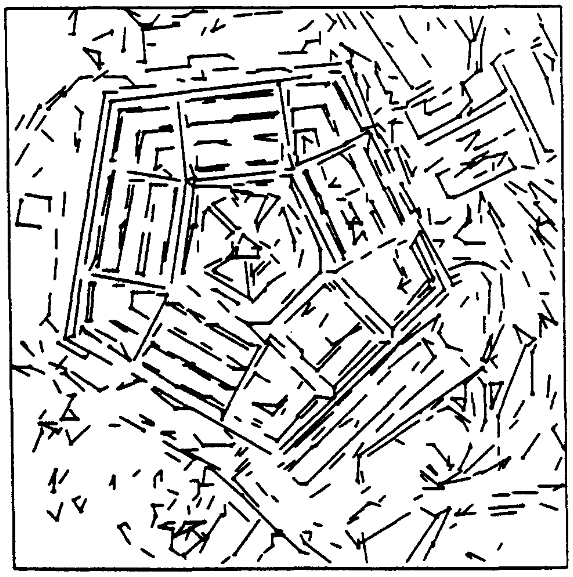

(b)

Fig. 15. Illustration of linear feature extraction on the pentagon image. (a) The final result. (b) The corners identified in the image are shown as small squares in this figure.

$\partial^2 g(x_0, y_0 - y)/\partial y^2$, $\partial g(x_0 - x, y_0 - y)/\partial y$, and $\partial^2 g(x_0 - x, y_0 - y)/\partial y^2$.

The derivatives in (2′) and (3′) can be evaluated using (8)–(12) and (1) to yield (2″) and (3″).

Acknowledgment

The authors are thankful to Mr. A. Huertas for providing the pentagon and airport image data and software for linear feature extraction and the Marr–Hildreth operator. They also wish to thank Dr. J. Canny for providing the software for the Canny edge detector.

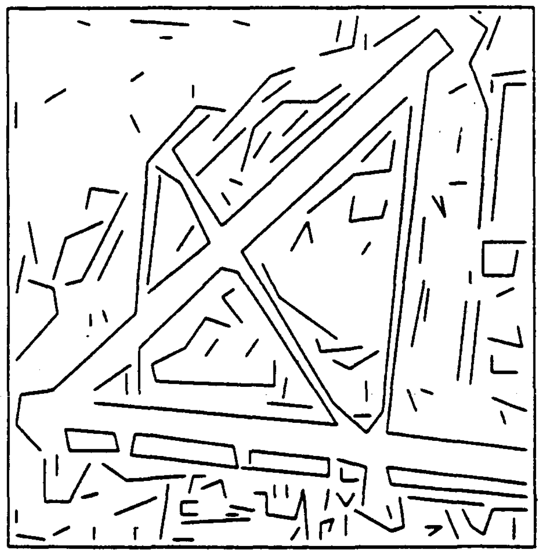


Fig. 16. Comparison to the Nevatia–Babu method. Linear features of the airport image using the Nevatia–Babu method.

References

[1] L. S. Davis, "A survey of edge detection techniques," *Comput. Graphics Image Processing*, vol. 4, pp. 248–270, 1975.

[2] T. Peli and D. Malch, "A study of edge detection algorithms," *Comput. Graphics Image Processing*, vol. 20, pp. 1–21, Jan. 1982.

[3] A. C. Bovik, T. S. Huang, and D. C. Munson, Jr., "Nonparametric edge detection with an assumption on minimum edge height," in *Proc. IEEE Conf. Comput. Vision Pattern Recognition.*, Washington, DC, June 1983, pp. 142–143.

[4] M. Basseville, "Edge detection using sequential methods for change in level—Part 2: Sequential detection of changes in mean," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-29, pp. 32–50, Feb. 1981.

[5] P. Eichel and E. Delp, "Sequential edge detection in correlated random fields," in *Proc. Conf. Comput. Vision Pattern Recognition*, San Francisco, CA, June 1985, pp. 14–21.

[6] D. Marr and E. C. Hildreth, "Theory of edge detection," *Proc. Roy. Soc. London*, vol. B-207, pp. 187–217, Feb. 1980.

[7] R. Nevatia and K. R. Babu, "Linear feature extraction and description," *Comput. Graphics and Image Processing*, vol. 13, pp. 257–269, 1980.

[8] J. F. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, pp. 679–698, Nov. 1986.

[9] V. Torre and T. Poggio, "On edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, pp. 147–163, Mar. 1986.

[10] R. M. Haralick, "Digital step edges from zero crossings of second directional derivatives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-6, pp. 58–68, Jan. 1984.

[11] R. Nevatia and K. E. Price, "Locating structures in aerial images," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-4, pp. 476–484, Sept. 1982.

[12] G. Medioni and R. Nevatia, "Segment-based stereo matching," *Comput. Vision, Graphics, Image Processing*, vol. 31, pp. 2–18, July 1985.

[13] B. L. Yen and T. S. Huang, "Determining 3-D motion and structure of a rigid body using straight line correspondence," in *Image Sequence Processing and Dynamic Scene Analysis*. New York: Springer-Verlag, 1983, pp. 365–394.

[14] J. B. Burns, A. R. Hanson, and E. M. Riseman, "Extracting straight lines," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, pp. 425–455, July 1986.

[15] R. L. Kashyap, "Analysis and synthesis of image patterns by spatial interaction models," in *Progress in Pattern Recognition, Vol. 1*, L. N. Kanal and A. Rosenfeld, Eds. Amsterdam, the Netherlands: Elsevier Science, 1981.

[16] N. D. A. Mascarehas and L. O. C. Prado, "A Bayesian approach to edge detection in images," *IEEE Trans. Automat. Contr.*, vol. AC-25, pp. 36-43, 1980.
[17] D. P. Panda and A. C. Kak, "Recursive least squares smoothing of noise in images," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. ASSP-25, pp. 520-524, Dec. 1977.
[18] E. J. Delp *et al.*, "Image data compression using autoregressive time series models," *Pattern Recognition*, vol. 11, pp. 313-323, Dec. 1979.
[19] W. K. Pratt *et al.*, "Visual discrimination of stochastic texture fields," *IEEE Trans. Syst., Man, Cybernet.*, vol. SMC-8, pp. 796-814, Nov. 1978.
[20] C. W. Therrien, "An estimation-theoretic approach to terrain image segmentation," *Comput. Vision, Graphics, Image Processing*, vol. 22, pp. 313-326, 1983.
[21] W. E. L. Grimson, *From Images to Surfaces*. Cambridge, MA: M.I.T. Press, 1981.
[22] L. Ljung and T. Soderstorm, *Theory and Practice of Recursive Identification*. Cambridge, MA; M.I.T. Press, 1983.
[23] B. D. O. Anderson and J. B. Moore, *Optimal Filtering*. Englewood Cliffs, NJ: Prentice-Hall, 1979.
[24] R. M. Haralick, "Authors reply," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, pp. 127-129, Jan. 1985.
[25] W. E. L. Grimson and E. C. Hildreth, "Digital step edges from zero crossings of second directional derivatives," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-1, pp. 121-127, 1985.
[26] A. Huertas and G. Medioni, "Detection of intensity changes with subpixel accuracy using Laplacian-Gaussian masks," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. PAMI-8, pp. 651-664, Sept. 1986.
[27] Y. T. Zhou, A. Rangarajan, and R. Chellappa, "Simultaneous filter/detector for edge detection in noisy images," in *Proc. Int. Symp. Inform. Theory*, Ann Arbor, MI, Oct. 1986.
[28] H. Kaufman, J. W. Woods, S. Dravida, and A. Murattekalp, "Estimation and identification of two-dimensional images," *IEEE Trans. Automat. Contr.*, vol. AC-28, pp. 745-756, July 1983.
[29] R. M. Haralick, personal communication, 1987.

Tree Searched Chain Coding for Subpixel Reconstruction of Planar Curves

RAGHAVACHARI SRIRAMAN, JACK KOPLOWITZ, AND SESHADRI MOHAN

Abstract—**We investigate coding schemes for the quantization of line drawings that outperform basic chain codes. First, subpixel accuracy reconstruction is obtained by a simple linear filtering of the chain code points, which achieves a factor of three to four reduction in average reconstruction distortion for smooth curves. Second, the basic chain encoding schemes are generalized to a multipath tree searched encoding scheme. A variation of the (M, L)-algorithm is used to maintain M contending chain code paths in storage and choose the "best" path from these. Over a wide variety of source curves, tree searched chain coding results in nearly an order of magnitude reduction in average reconstruction distortion over smoothed chain codes. The performance improvement for curves is obtained with only a slight increase in bit rate over basic chain codes.**

Index Terms—**Chain coding, linear filtering, (M, L)-algorithm, subpixel reconstruction, tree searching.**

Manuscript received October 18, 1985; revised February 5, 1988. Recommended for acceptance by C. Y. Suen. This work was supported in part by the National Science Foundation under Grant. ECS-8206609.

R. Sriraman is with North West Instrument Systems, Inc., Bldg. D4, Suite 200, 19545 North West Von Neumann Dr., Beaverton, OR 97006.
J. Koplowitz and S. Mohan are with the Department of Electrical and Computer Engineering, Clarkson University, Potsdam, NY 13676.

IEEE Log Number 8824650.

I. Introduction

Line drawings are an important class of pictorial data which occur in many different areas including cartography, graphics, computer vision, engineering drawings, and handwritten characters. It is necessary to digitize these curves without significant distortion for digital processing, storage or transmission. Traditionally, chain encoding has been used to quantize line drawings due to its simplicity and ease of implementation.

Chain codes digitize a given planar curve by superimposing on it a grid of fineness T and then choosing a sequence of links, or displacements, from an allowed set. In one chain coding scheme, illustrated in Fig. 1(a), a square of side T is superimposed around each grid node. A grid node is a point of the chain sequence if the source curve intersects any part of the square region around it. The source curve is then represented by a sequence of links joining the chain points. Such codes, usually referred to as square quantized chain codes, have a set of four possible links. Thus each link can be encoded into two bits. The accuracy of reconstruction is of the order of T, denoted $O(T)$, where T is the grid size or pixel distance.

Other chain coding schemes with different sets of possible links are available. An important example is grid intersect quantization which selects the closest node to each grid crossing as shown in Fig. 1(b). This scheme is 8-directional allowing diagonal links and hence uses three bits to encode a link. Different quantization schemes have the same order of performance [1], differing only by a constant factor. Four directional codes are considered here. They are the most basic among chain coding schemes and also occur as boundary contours for two-tone digitized images on a square array of pixels. However, the multipath tree searching procedure presented here has a straightforward extension to 8-directional codes.

Subpixel accuracy refers to a reconstruction or estimation of the original curve or edge with an average error of less than one-half T (the grid or pixel size). Such accuracy is often important for edge detection in image analysis. In digitized line drawings this is necessary if a curve is to be dispalyed with a greater accuracy or enlarged without increasing the coarseness. Subpixel accuracy can be accomplished in a number of ways for chain coded curves with low curvature. One method is to apply a simple linear filtering, or smoothing operation at the decoder. Basic chain coding algorithms are purely local in nature, choosing successive links to minimize the distance from the curve to the chain code point without regard to future behavior of the curve. Low pass filtering, however, introduces significant reconstruction distortion in curves with large curvature. The reconstruction distortion of such curves can be reduced by *looking ahead* and choosing link sequences other than the chain code sequence which, when filtered, obtain greater accuracies. There exist several such look ahead algorithms in the literature that belong to the class of *multipath tree searched* encoding schemes. One such algorithm is the well known (M, L)-algorithm. Here we use a variation of the (M, L)-algorithm to choose code sequences that are globally better than the chain code sequence according to some predefined criterion. Previous applications of tree searching to source coding have been directed at reducing the bit rate of the encoder. Here, the bit rate of the encoder is not reduced; instead, the redundancy inherent in the encoded source is used to reduce the quantization error at the decoder when filtering is present. The concept of filtering and tree searched chain coding is illustrated with the following example.

Consider the simple filter which replaces each chain point with an average of itself and its two immediate neighbors, i.e., each of the three points is weighted by $1/3$. Fig. 2(a) shows a straight line segment, its square quantized chain code and the filtered or smoothed point of the chain sequence. Smoothing reduces the reconstruction distortion as measured by the distance of the points to the original curve. However, by choosing a different 4-directional