Early Jump-Out Corner Detectors

James Cooper, Svetha Venkatesh, and Leslie Kitchen

Abstract—**We present two new corner detectors; one works by using dissimilarity along the contour direction to detect curves in the image contour and the other estimates image curvature along the contour direction. These operators are fast, robust to noise, and require no subjective thresholding.**

Index Terms—**Corner detection, image noise modeling, sequential analysis, similarity measures, thresholding.**

I. Introduction

Although primitive features such as edges and lines are vital visual clues, intersection of edges such as corners and junctions, which are commonly referred to as 2-D features, provide rich information for examining frame-to-frame displacement characteristics of images. Thus, in applications involving disparity analysis such as motion detection and depth from stereo, we need to identify these 2-D features. In general, edge detectors do not make good corner detectors because they give reduced output at corners, and because of this, considerable research has been specifically directed towards isolating corners and junction points.

Kitchen and Rosenfeld [1] measure cornerity as the rate of change of gradient direction along an edge multiplied by the gradient magnitude. Nagel [2] used a method based on minimizing the squared differences between a second-order Taylor series expansion of grey-level values from one frame to another. Noble [3] has shown how the Plessey corner detector estimates image curvature and has proposed an image representation that is based on the differential geometrical "topography" of the intensity surface. Spacek [4] relates cornerness to the difference in response between a directionally selective edge detector and a rotationally symmetric one.

In this correspondence, we propose a two-stage approach to corner detection in which first, the contour direction is measured, and then, image differences along the contour direction are computed. A knowledge of the noise characteristics is used to determine whether the image differences along the contour direction are sufficient to indicate a corner. The technique is less computationally expensive than existing corner detection techniques and is robust to noise.

The corner detection algorithms are described in Section II. A comparative study of the results obtained with our corner detectors and other corner detectors, specifically the Plessey, Kitchen–Rosenfeld, and Beaudet detectors, are carried out in Section III. The discussion follows in Section IV, and we draw conclusions in Section V.

Manuscript received February 7, 1991; revised January 29, 1992. This work was supported by the Department of Industry Technology and Commerce under the Artificial Intelligence Research Support Programme and by a CTEC Special Research Grant. Recommended for acceptance by Associate Editor Y. Aloimonos.

J. Cooper is with the Department of Computer Science, Edith Cowan University, Mt. Lawley, Australia.

S. Venkatesh is with the Department of Computer Science, Curtin University of Technology, Australia.

L. Kitchen is with the Department of Computer Science, University of Melbourne, Melbourne, Australia.

IEEE Log Number 9208015.

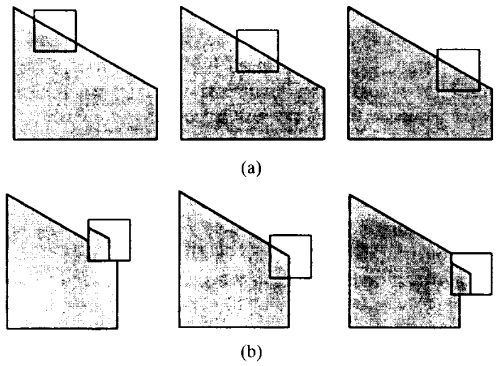


Fig. 1. (a) If we take an image patch belonging to a straight edge and compare it with others along the edge direction, we find that it is similar to them; (b) If we do a similar comparison along the local edge direction at a corner, we find that it is not similar to its neighbors.

II. Early Jump-out Corner Detectors

If two image patches along a straight edge segment are compared with each other, they will be very similar (Fig. 1(a)). If the edge segment is curved, as at a corner, the two image patches will be dissimilar (Fig. 1(b)). Thus, corners where the similarity between image patches along the edge direction is low can be detected.

This suggests a corner detection algorithm like the following:

```
for each pixel (x, y) in the image
    Obtain the components of the image gradient.
    If the gradient magnitude is low then
      The point (x, y) is not a corner.
    else
      Calculate positions (x_l, y_l) and (x_r, y_r) to left and right
      along the local contour direction. Use a similarity
      test to decide whether the image patches centered at
      (x_l, y_l) or at (x_r, y_r) differ from that at (x, y).
      if they differ then
      (x, y) is a corner
    else
      (x, y) is not a corner.
    endif
  endif
end
```

A. Image Self-Similarity and Early-Jump-out

The method makes extensive use of a similarity test that therefore must be fast. In this section, we outline a fast method using sequential analysis techniques [5] for determining similarity between pairs of subwindows. Barnea and Silverman [6] propose that a threshold sequence can be defined such that if at any stage in the computation of the difference value the partial result is greater than the corresponding value in the sequence, we can jump out of the similarity test. We term this the Early Jump-out technique. We have extended the Early Jump-out technique to use multiple threshold sequences, and the derivation of the sequences used for the dissimilarity corner detector is given below.

1) Early Jump-out High (EJH): At a strategically chosen se-

quence of locations, the absolute value of the difference in the grey levels of the corresponding pixels in the two windows is added to an accumulating sum. At each step of the Early Jump-out similarity test, we compare the accumulating value with the appropriate member of a threshold sequence to test the null hypothesis H_O:

"The two windows are similar so that the differences in pixel values can be attributed entirely to noise."

At any step, if the accumulating value is larger than the threshold, the process is terminated, and H_O is rejected. We will call the threshold sequence the Early Jump-out High (EJH) sequence.

a) Computing the threshold sequence: Barnea and Silverman use "a computer solution of an analytic system of recursive equations" to generate the threshold sequence so that if H_O was true, then at each step, the probability that the accumulated value will be greater than the current threshold is some small value q. In this section, we show a simple way in which the threshold sequence can be derived. We first describe the method to derive the first member of the threshold sequence and then show how subsequent members can be derived.

We model the image data at each pixel as an image signal with added Gaussian noise distributed as $N(0, \sigma)$. If the difference between the value of two pixels is due entirely to noise, then the difference is a random variable X distributed as $N(0, \sigma\sqrt{2})$ and whose probability distribution function (pdf) is given by

$$f_X(x) = \frac{1}{2\sigma\sqrt{\pi}} e^{-(1/4)(x/\sigma)^2}. \tag{1}$$

The absolute value of X is a random variable D_1 with pdf given by

$$f_{D_1}(d) = \begin{cases} \dfrac{1}{\sigma\sqrt{2}} e^{-(1/4)(x/\sigma)^2} & \text{if } d > 0 \\ \dfrac{1}{2\sigma\sqrt{2}} & \text{if } d = 0 \\ 0 & \text{else.} \end{cases} \tag{2}$$

We numerically integrate (2) to find the first member T_1 of the threshold sequence such that the probability that the first difference will be greater than this threshold is q. That is

$$\int_{T_1}^{\infty} f_{D_1}(x)dx = q. \tag{3}$$

Let R_N be the value of the accumulating result after N operations. At step N, only if $R_N < T_N$ is the absolute value of the difference between a new pair of pixels added to the accumulating result. The pdf of the survivors of each thresholding operation is therefore of interest. The value of R_N subject to $R_N < T_N$ is a random variable S_N whose pdf is given by

$$f_{S_N}(s) = \begin{cases} \dfrac{1}{(1-q)} f_{D_N}(s) & \text{if } s < T_N \\ 0 & \text{else.} \end{cases} \tag{4}$$

Since R_{N+1} is obtained by adding R_N to the absolute value of the difference between a new pair of pixels, we have $D_{N+1} = S_N + D_1$. Therefore, the pdf of D_{N+1} can be computed using

$$f_{D_{N+1}}(d) = f_{D_1}(d) * f_{S_N}(s) \tag{5}$$

where $*$ is the convolution operator. For each N, we can generate f_{D_N} and then use numerical integration to find a threshold T_N such that

$$\int_{T_N}^{\infty} f_{D_N}(x)dx = q. \tag{6}$$

In the preceding discussion, we have assumed that each pixel difference has the same pdf as the first. If this assumption was not valid, we could have obtained the pdf of each difference independently and used the new pdf in the convolution step. Alternatively, the EJH sequence could have been computed from distributions obtained directly from image data. The resulting comparison is shown in Section III.

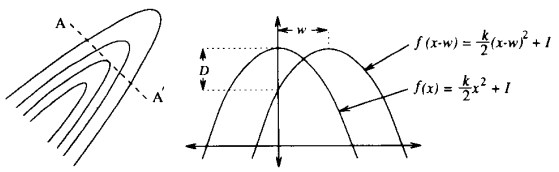


Fig. 2. If we take a cross section along the line AA' through the small image region whose contour map is shown in (a), we obtain a parabola $f(x)$ as in (b). We can displace this parabola a distance w and calculate $D(x, w) = f(x) - f(x - w)$, which is a function of the second derivative k.

2) Early Jump-out Low (EJL): The dissimilarity corner detector compares image patches along an edge. If the edge is straight, a high degree of similarity can be expected, the accumulating value will never exceed the EJH sequence, and the cost of the EJH similarity test is then $O(M^2)$. We use another sequence—the Early Jump-out Low (EJL) sequence [7]—to allow the process of accumulating differences to be terminated early if the image patches are similar.

At each step of the Early Jump-out technique, the accumulated value is compared with the appropriate member of a threshold sequence in order to test a new null hypothesis H_O:

"The two windows are dissimilar to each other."

This time, the sequence is precomputed so that if H_O is true, then at each step, the probability that the accumulating difference will be less than the current threshold is some small value p.

a) Computing the EJL threshold sequence: As before, we first derive the first member of the EJL sequence and then show how subsequent members of the sequence can be derived.

Consider contours of equal grey level at a noise-free corner as shown in Fig. 2(a). If line segment AA' is parallel to the edge direction, then a profile taken along AA' can be approximated by a second-order Taylor series expansion as a parabola

$$f(x) = f(0) + f'(0)x + \frac{f''(0)x^2}{2} \tag{7}$$

where f'' is the second directional derivative in the direction orthogonal to the gradient, f' is the first derivative, and $f(0)$ is the image intensity. As AA' is parallel to the edge direction, f' is zero. If noise is taken into account

$$f(x) = f(0) + \frac{f''(0)x^2}{2} + e(x) \tag{8}$$

where $e(x)$ is the error due to image noise (with standard deviation σ) at x.

If the parabola is displaced by a small distance w as shown in Fig. 2, then we can compute the difference function D as

$$\begin{aligned} D(x, w) &= |f(x) - f(x - w)| \\ &= \left| kxw - \frac{kw^2}{2} + e(x) - e(x + w) \right| \end{aligned} \tag{9}$$

where $k = f''(0)$.

If k is large and $x \neq w/2$, then D has a normal distribution with mean $\left| kxw - \frac{kw^2}{2} \right|$ and standard deviation equal to $\sigma\sqrt{2}$. Further, the threshold T_1 is the first value of the EJL sequence, and if the first difference D_1 is less than T_1, we jump out of the similarity test with the conclusion that k is small and the pixel is not a corner pixel. For particular values of x and w, we can choose values $k_{\min}$ and T_1 such that the probability of D_1 being less than T_1 is some small value p. Then, $k_{\min}$ becomes the smallest curvature that can be reliably detected for the particular values of x, w, and T_1.

For $x = 0$ and $w = 2$, D_1 distributed as $N(2k, \sigma\sqrt{2})$. The pdf of D_1 is given by

$$f_{D_1}(d) = \frac{1}{2\sigma\sqrt{\pi}} e^{(1/4)[(d-2k)/\sigma]^2}. \quad (10)$$

We numerically integrate (10) to find the first member T_1 of the threshold sequence such that

$$\int_{\infty}^{T_1} f_{D_1}(x)dx = p. \quad (11)$$

As with the EJH threshold sequence, let R_N be the value of the accumulating result after N operations. At step N, only if $R_N > T_N$ is the absolute value of the difference between a new pair of pixels added to the accumulating result. The value of R_N subject to $R_N > T_N$ is a random variable S_N, whose pdf is given by

$$f_{S_N}(s) = \begin{cases} \frac{1}{(1-p)} f_{D_N}(s) & \text{if } s > T_N \\ 0 & \text{else.} \end{cases} \quad (12)$$

As before, for $N > 1$, the pdf of D_{N+1} can be computed recursively from the pdf of S_N and D_1 using

$$f_{D_{N+1}}(d) = f_{D_1}(d) * f_{S_N}(s). \quad (13)$$

For each N, we can generate $f_{D_{N+1}}(d)$ using the above distribution and then use numerical integration to find T_N such that

$$\int_{0}^{T_N} f_{D_N}(x)dx = p. \quad (14)$$

By specifying x, w, and $k_{\min}$, we thus determine the entire EJL threshold sequence T_N.

If we let g be a second-order approximation of the local image function with $g' = (g_x, g_y)^{\mathsf{T}}$ (the image surface gradient) and $g'' = \begin{pmatrix} g_{xx} g_{xy} \\ g_{yx} g_{yy} \end{pmatrix}$ be the second derivative at $(0,0)^{\mathsf{T}}$, then the value of k is given by Torre and Poggio [8] as

$$k = \frac{g_{xx}g_y^2 + g_{yy}g_x^2 + 2g_{xy}g_x g_y}{g_x^2 + g_y^2}. \quad (15)$$

This is the quantity measured by the Kitchen–Rosenfeld corner operator [1] that Kitchen and Rosenfeld interpert as the curvature of the contour multiplied by the edge magnitude. Our interpretation allows a simple error analysis and renders the operator amenable to sequential analysis.

3) The Meaning of the Early Jump-out Sequences Fig. 3 shows typical EJH and EJL threshold sequences, which divide "Early Jump-out space." A difference sum starts in the undecided region. If the difference sum outgrows the EJH sequence, we conclude that the two samples are not similar ($k > 0$). If the difference sum gets below the EJL sequence, we conclude that either the samples are similar or the value of k for the pixel in question is too low for a corner to be detected ($k < k_{\min}$). If the difference sequence gets into the fish tail region shown in Fig. 3, it indicates that both of the above conclusions could be accepted, which means that the image surfaces are not similar, but the strength of the feature is not sufficient ($0 < k < k_{\min}$)—there is a corner there, but the value of k is so low that EJL would reject it as a corner.

If we let $T_1 = 0$, then we can detect the reliably detectable corner with the smallest value of k. In practice, we use a higher value for T_1 than zero. This raises both the starting point and the slope of the EJL threshold sequence curve and reduces the time spent in the undecided region of Early Jump-out space by difference sums associated with weak corners. The fish-tail region in Fig. 3 then becomes important as difference sums can now get into this region when there is a detectable corner.

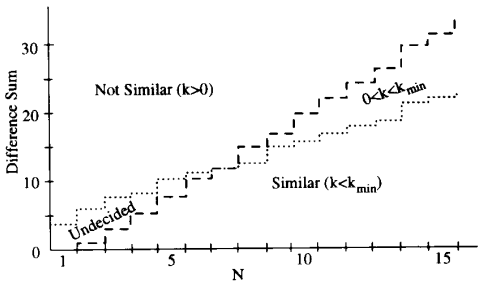


Fig. 3. Division of Early Jump-out space by EJH (dots) and EJL (dashes) threshold sequences calculated with $\sigma = 1.15$ and $T_1 = 0$. An accumulating difference sum starts in the undecided region between the two threshold sequences. If it crosses the EJH threshold sequence, we can conclude that the two image patches are dissimilar, and if it is crosses the EJL threshold sequence we can conclude that either the patches are similar or k is small (see text). The difference sum can get into the fish-tail region, in which case, we could conclude that the two image patches are not similar, but k is smaller to that of the minimal confidently detectable corner.

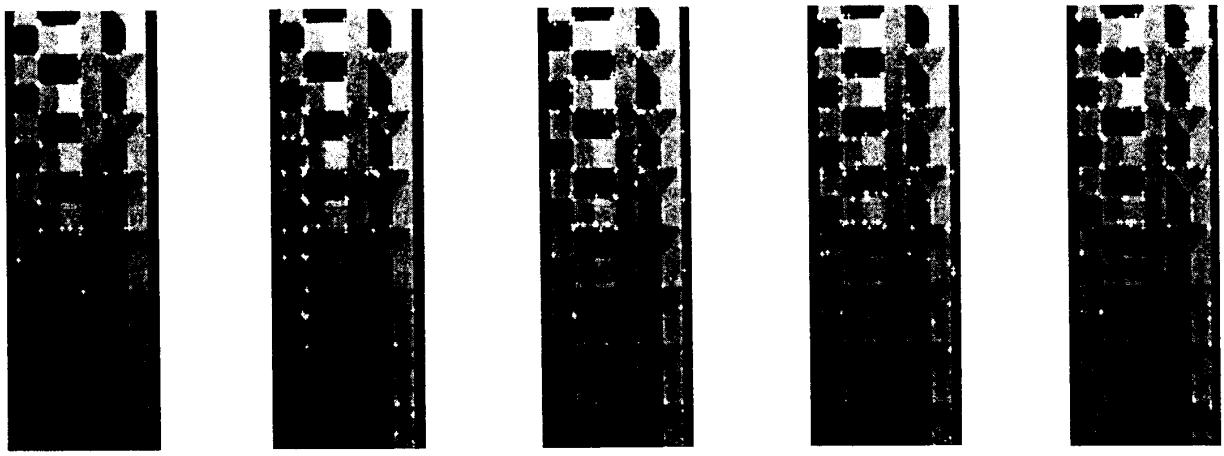
B. Compensation for Errors

In our implementation, the dissimilarity corner detector uses the output of the Sobel operator to calculate the positions to the right and left along the contour of the current pixel. However, even on a straight edge, spatial digitization leads to errors in the computed contour direction, and the predicted position of a contour is in general displaced by a subpixel distance from the center of the nearest pixel. The apparent image grey-level difference caused by the resulting displacement error in the predicted position is proportional to the product of the displacement error and the image gradient magnitude, and we use this to modify the threshold sequences to take the digitization noise into account.

C. Direct Estimation of k (DEK)

This discussion in Section II-A-2 suggests another algorithm. When the dissimilarity corner detector is used with Early Jump-out, a *first* difference in grey level is estimated to the left and the right along the contour direction, and if this value if large in both directions, a corner is detected. From (9) though, we see that this is actually an estimation of k since by using (9), we could calculate k from the difference values that we obtain. These one-sided estimates of the image curvature involve compensating for the contour position error. We could estimate the value of k at the test pixel directly by adding together the estimates on either side. If we do this, the errors for which we compensated in Section II-B cancel out. This should make the corner detector more robust to noise. The algorithm this discussion motivates is shown below:

```
for each pixel (x, y) in the image
    Obtain the components of the image gradient.
    if the gradient magnitude is low then
      The point (x, y) is not a corner.
    else
      Calculate the contour direction and use an Early Jump-
      Out test to determine whether the absolute value of
      the second derivative (k) along the contour direction
      is greater than zero.
      If k is greater than zero then
        (x, y) is a corner
      else
        (x, y) is not a corner.
      endif
    endif
end
```



(a) (b) (c) (d) (e)

Fig. 4. Comparison of outputs of the operators when applied to the synthetic image with added Gaussian noise ($\sigma = 15$): (a) DEK; (b) dissimilarity operator; (c) Plessey operator; (d) Kitchen–Rosenfeld operator; (e) Beaudet operator.

The value of k is now estimated via a calculation of the form $k = 2a - b - c$, where a, b, and c are the values of three different pixels. If the standard deviation of the value of each pixel is σ, then the standard deviation of k is $\sigma\sqrt{6}$. The same procedures for generating the Early Jump-out threshold sequences can be used here, except for the different value of the standard deviation.

III. Experimental Results

The results obtained by our corner detectors are compared with other methods both in terms of the quality of output and the speed of performance. The values $x = 0$ and $w = 2$ from (9) were used in the DEK, and if the accumulating difference sum went to more than five terms on any test, the test was terminated with the conclusion that the sum was growing too slowly. This effectively means that the largest region of support for any corner was 9×9.

We demonstrate that the Early Jump-out corner detectors are computationally less expensive than the Kitchen–Rosenfeld detector and the Plessey corner detectors.

A. Measuring Image Noise

We measure the image pixel noise by repeatedly reading the value of a pixel and computing the standard deviation of the values. With this method, we estimated the pixel noise at 1.92 grey levels. The standard deviation of the noise levels for all the pixels in the 256×256 image was 0.16 grey levels. The noise level was almost invariant to grey level as the measured pixel noise remained constant over a grey-level range from 10 to 220.

B. The Corner Detectors

Three images are used:

1) Synthetic image with added gaussian noise ($\sigma = 6$)
2) synthetic image with gaussian noise ($\sigma = 15$)
3) real image.

First, a synthetic corner image composed of L, T, Y, and $+$ junctions was created. The contrast of the junctions was reduced progressively from top to bottom in the image, and this gave rise to a decreasing signal strength from top to bottom in the synthetic image. The image was then smoothed by convolution with a gaussian ($\sigma = 0.6$) to simulate the blurring introduced by the imaging system.

TABLE I
Error Performance of Operators. (The performance of the five operators is shown in terms of False Positives (FP) and False Negatives (FN) for medium and high noise images.)

image noise	DEK		Sim.		Plessey		K-R		Beaudet	
	FN	FP	FN	FP	FN	FP	FN	FP	FN	FP
$\sigma = 6$	18	6	17	18	18	44	17	33	18	140
$\sigma = 15$	44	25	43	15	45	56	43	52	41	42

Finally, Gaussian noise of appropriate standard deviation was added. The result was a synthetic image with corners of decreasing signal-to-noise ratio from top to bottom.

To compare the four detectors in a nonsubjective manner, we identify two errors as follows: We associate a 5×5 *detection area* centered at a true corner. Should a corner be detected in this area by a corner detector, the corner is correctly detected. Any corner detected outside this region is termed a *false positive* (FP), and failure to detect a corner in the detection area is termed *false negative* (FN). The corner detectors were tuned to give similar numbers of false negatives. Table I shows the false negatives and positives obtained for the synthetic images when the four different corner detectors are applied to the synthetic images ($\sigma = 6$) and ($\sigma = 15$).

Images corresponding to the performance statistics with $\sigma = 15$ are shown in Fig. 4. At medium noise ($\sigma = 6$), both the Kitchen–Rosenfeld and Plessey operators gave higher numbers of false positives than either of the Early Jump-out corner detectors. This is true at high noise as well (Fig. 4). The outputs of the four operators when applied to a real image are shown in Fig. 5, which indicates the similarity of results at this level of real camera noise (estimated $\sigma = 0.88$).

C. Number of Operations

One of our contributions to the technique of image patch similarity testing is the addition of the EJL sequence. It is the use of both the EJH and EJL sequences that leads to the speed of the corner detector. Table II compares the operators in terms of the number of operations performed. In the computation of the number of operations, the cost of one derivative operation is taken to be $2n^2$, where n is the size of the neighborhood of the difference operation used. For the

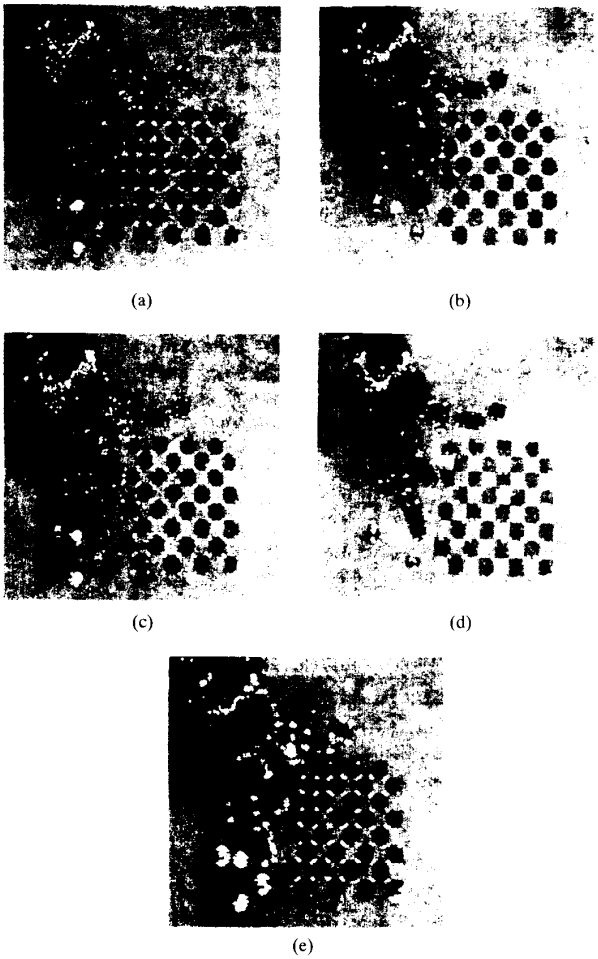


Fig. 5. Comparison of outputs of the operators when applied to the real image: (a) DEK; (b) dissimilarity operator; (c) Plessey operator; (d) Kitchen–Rosenfeld operator; (e) Beaudet operator.

TABLE II
NUMBER OF OPERATIONS USED BY DIFFERENT OPERATORS

DEK	Dissimilarity	Plessey	Kitchen–Rosenfeld	Beaudet
$10+5$	$10+10$	$6n2+m2+7$	$10n2+11$	$6n2+3$

dissimilarity-based operator, ten operations are performed to compute the position and error and ten to perform the similarity test. For DEK, ten operations are performed to compute the position and error and five operations to perform the similarity test. The Plessey operator computes the $n \times n$ derivatives, which it smoothes with an $m \times m$ Gaussian. Thus, the cost of this operator is $6n^2+m^2+7$. The costs for the Beaudet and the Kitchen–Rosenfeld operators are as computed in [1]. It should be noted that these operation counts are implementation specific. For instance, the convolution we used in the Plessey operator is separable, and its convolution could therefore be reduced to $2m^2$.

Table III indicates the effectiveness of the Early Jump-out operations in reducing the amount of work for DEK. The number of difference operations that preceded threshold (EJH or EJL) jump-out is close to the number of jump-outs of either type. This means that the number of difference operations performed at each potential corner point was just more than, one and the region of support for corners was usually 5×5. The largest possible region of support (9×9) was rarely used.

TABLE III
THE NUMBER OF DIFFERENCE OPERATIONS THAT PRECEDED JUMP-OUT VIA EITHER OF THE THRESHOLD SEQUENCES IN THE DEK ALGORITHM. (There where a total of 20159 test pixels left after thresholding on edge magnitude and approximately 1.12 operations were performed per test pixel.)

	EJH	EJL	No Jump Out
Number of jump-outs	1050	19109	9
Number of operations before	1358	19344	45

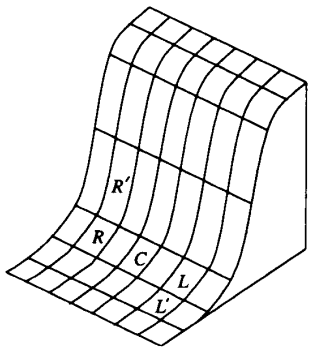


Fig. 6. 7×7 image surface patch centered on pixel C, which is situated near a straight edge. As the gradient magnitude is small at C, the influence of noise on the estimated gradient direction will be large (consider the effect of noise on $\arctan(y/x)$ when x and y are small). This can cause L' and R' to be the predicted pixels on the same contour instead of L and R. DEK would then report C to be a corner as $|R'+L'-2C|$ is large, but the dissimilarity corner detector would not report a corner as $|L'-C|$ is small.

IV. DISCUSSION

The robustness to noise of the early jump-out methods can be attributed to the fact that we use an accurate image model that incorporates image noise.

One would expect DEK to be more robust to noise than the dissimilarity detector because DEK effectively computes the two one-sided estimates of k (see Section II-C), but the results in Table I indicate that this is not the case. The reason can be seen in Fig. 6. Future work will be directed towards modifying the DEK algorithm to incorporate a correction for this problem. One possible approach would be to reassess detected corner points where the edge strength is low by using wider edge detection masks to get more accurate estimates of edge direction and then retesting with the new direction information.

V. CONCLUSION

We present two new corner detectors: one works by using dissimilarity tests along the contour direction to detect curves in the image contour and the other estimates image curvature along the contour direction. These operators are fast and robust to noise.

The speed of the operators can be mainly attributed to the fast Early Jump-out tests. The use of the EJL sequence allows these tests to be terminated early when the test pixels are destined for rejection as corners. Further, the members of the EJL sequence can be chosen so that on average, one image operation is performed per

test. Explicit modeling of image noise contributed to the robustness of these operators to noise.

The corner detectors presented here require no subjective thresholding. The standard deviation of the image noise must be specified, but this value can be easily measured.

We also present a new interpretation of the Kitchen–Rosenfeld corner operator in which we show that this operator can also be viewed as the second derivative of the image function along the edge direction.

[1] L. Kitchen and A. Rosenfeld, "Gray level corner detection," *Patt. Recogn. Lett.*, vol. 1, pp. 95–102, Dec. 1982.
[2] H.-H. Nagel, "Displacement vectors derived from second-order intensity variations in image sequences," *Comput. Vision Graphics Image Processing*, vol. 21, pp. 85–117, 1983.
[3] A. J. Nobel, "Finding corners," *Image Vision Comput.*, vol. 6, pp. 121–128, May 1988.
[4] L. Spacek, "The detection of contours and their visual motion," Ph.D. thesis, Univ. of Essex, 1985.
[5] A. Wald, *Sequential Analaysis*. London: Chapman and Hall, 1947.
[6] D. I. Barnea and H. F. Silverman, "A class of algorithms for fast digital image registration," *IEEE Trans. Comput.*, vol. C-21, no. 2, pp. 179–186, 1972.
[7] J. Cooper, S. Venkatesh, and L. Kitchen, "Early jump-out corner detectors," Tech. Rep. 90/14, Univ. of Western Australia, Comput. Sci. Dept., 1990.
[8] V. Torre and T. A. Poggio, "On edge detection," *IEEE Trans. Patt. Anal. Machine Intell.*, vol. PAMI-8, pp. 147–163, Mar. 1986.

Optimally Small Operator Supports for Fully Parallel Thinning Algorithms

Richard W. Hall

***Abstract*—Requirements on the support size and shape are investigated for the class of all adequate fully parallel thinning operators. Eleven pixel supports are shown to be the smallest possible supports, and the possible positions of the support pixels are shown to be well constrained. Constraints on support positions are also demonstrated for operators with supports that are larger than optimal, and a sufficient test for connectivity preservation is reviewed. These results allow algorithm designers looking for small support operators to focus on a relatively small set of acceptable supports.**

***Index Terms*—Connectivity preservation, image processing, optimally small operator supports, parallel thinning, reduction operators, support constraints.**

I. Introduction

Thinning is a fundamental early processing step in image processing [17], and parallel thinning algorithms are of particular interest as these allow for the efficient use of parallel computers [3], [4], [13]. A *fully parallel* thinning algorithm applies the same thinning operator over the entire image in each iteration of the parallel algorithm. The support of a given operator applied to pixel p is the set of pixels in the image space defined with reference to p, which are required to determine the operator's result. Supports confined to a fixed region around p independent of image space size are referred to as *local supports*. Operators with small local support are highly desirable in parallel implementations since operators with larger supports require either higher time cost or higher interconnection complexity. In fully parallel application, operators with only 3×3 support have been recognized as unable to preserve connectivity while providing adequate parallel thinning [6], [16], [17]. Investigators have partially serialized their approaches to enable use of 3×3 thinning supports [1], [6], [13], [16], [18]. Fully parallel approaches have also been developed around operators with larger support [2], [7], [11]. It is of some interest to know what the optimally small support size could be for a fully parallel thinning operator and to know something about the possible positions for these supports and constraints on support positions when support size is larger than optimal. With such knowledge, algorithm designers looking for operators with small support can efficiently direct their attention to appropriate forms of support.

Operators with 11-pixel support are known [2], [7], illustrating that optimally small supports include ≤ 11 pixels. In this paper, preliminary arguments that relate thinning requirements to specific expectations on special images are given. These observations allow one to claim that 3×3 operator supports are inadequate and that all of the 3×3 neighborhood is required in any support. Then, optimally small supports are shown to require 11 pixels, and the possible locations for the 11 pixels are shown to be well constrained. Finally, support position constraints are illustrated for support sizes greater than 11, and a sufficient test for connectivity preservation is illustrated based on results in [9] and [15].

Manuscript received February 11, 1991; revised May 8, 1992. This work was supported by the Defense Advanced Research Projects Agency under Grant AFOSR90-0310 and monitored by the Air Force Office of Scientific Research. Recommended for acceptance by Associate Editor E. Delp.

The author is with the Department of Electrical Engineering, University of Pittsburgh, Pittsburgh, PA 15261.

IEEE Log Number 9208442.

II. Image Space and Parallel Operator Notions

A rectangularly tessellated binary image space is considered where pixels take values 0 or 1. In practice, image spaces are bounded, but it is assumed here that image space size is sufficiently large to hold any desired image. Test images that are imagined to be of arbitrary size in some arguments are used, but typical realistic image space sizes are sufficiently large to make the results compelling in practical situations. The 1-valued pixels (**ones**) form a set S representing *objects* (connected components) to be thinned, and 0-valued pixels (**zeros**) form a set $\backslash S$ representing the background of S and holes in objects of S. We define S and $\backslash S$ with 8-connectivity and 4-connectivity, respectively, which is denoted an 8–4 space [12]. The 8-*neighborhood* $N_8^*(p)$ of a pixel p is defined as the set of 8-adjacent neighbors of p. $N_8(p)$ is defined as $N_8^*(p) \cup p$. Any use of the term *distance* in the image space may be interpreted as "city block" distance [17]. $A(p)$ is defined as the number of distinct 4-connected components of **one**'s in $N_8^*(p)$, $B(p)$ is defined as the number of **one**'s in $N_8^*(p)$, and $C(p)$ is defined as the number of distinct 8-connected components of **one**'s in $N_8^*(p)$. When illustrating regions in the image space, the notation $< e >$ refers to the set of all pixels labeled e (sometimes the labeling is inferred) or e_i for i an arbitrary subscript. Any other notation used is generally consistent with [12].

In a binary image space, the image is transformed by operators that either take a **one** to a **zero**, which is referred to as *reduction*