Robust Image Corner Detection Through Curvature Scale Space

Farzin Mokhtarian and Riku Suomela

Abstract—This paper describes a novel method for image corner detection based on the curvature scale-space (CSS) representation. The first step is to extract edges from the original image using a Canny detector. The corner points of an image are defined as points where image edges have their maxima of absolute curvature. The corner points are detected at a high scale of the CSS and tracked through multiple lower scales to improve localization. This method is very robust to noise, and we believe that it performs better than the existing corner detectors. An improvement to Canny edge detector's response to 45^o and 135^o edges is also proposed. Furthermore, the CSS detector can provide additional point features (curvature zero-crossings of image edge contours) in addition to the traditional corners.

Index Terms—Low-level processing, feature extraction, corner detection, multiscale analysis, curvature scale space, Canny edge-detector.

1 Introduction

Corner detection is an important task in various computer vision and image-understanding systems. Applications include motion tracking, object recognition, and stereo matching. Corner detection should satisfy a number of important criteria:

- All the true corners should be detected.
- No false corners should be detected.
- Corner points should be well localized.
- Corner detector should be robust with respect to noise.
- Corner detector should be efficient.

This paper proposes a new corner detection method [16] based on the curvature scale-space (CSS) technique. The CSS technique is suitable for extraction of curvature features from an input contour at a continuum of scales. This corner-detection method requires image edge contours. In the implementation of the CSS detector, a Canny edge detector [3] was used. Note, however, that the Canny edge detector is not a crucial part of the technique: It can be replaced with another edge-detection algorithm. Nevertheless, with Canny's good edge detection, we believe our corner detector performs better than existing ones.

Much work has been carried out on corner detection, and Section 2 gives an overview. Section 3 briefly describes the Canny detector and the improvement made to its response on edges at 45^o or 135^o angles. Section 4 describes the CSS method in general, and Section 5 describes in detail the proposed corner detection method. The performance of a corner detector is best evaluated with real test images, and in Section 6, the results of the CSS detector are compared to three other corner detectors. Four different images with different properties are used in the experiments. The conclusions are presented in Section 7.

- *The authors are with the Centre for Vision, Speech, and Signal Processing, Department of Electronic and Electrical Engineering, University of Surrey, Guildford, England GU2 5XH, UK.*
  E-mail: F.Mokhtarian@ee.surrey.ac.uk.

Manuscript received 15 May 1997; revised 8 Sept. 1998. Recommended for acceptance by V. Nalwa.
For information on obtaining reprints of this article, please send e-mail to: tpami@computer.org, and reference IEEECS Log Number 107415.

2 Literature Survey

Considerable research has been carried out on corner detection in recent years. This section briefly reviews a number of proposed algorithms. Moravec [17] observed that the difference between the adjacent pixels of an edge or a uniform part of the image is small, but at the corner, the difference is significantly high in all directions. Harris [8] implemented a technique referred to as the Plessey algorithm. The technique was an improvement of the Moravec algorithm. Beaudet [2] proposed a determinant (DET) operator which has significant values only near corners. Dreschler and Nagel [6] used Beaudet's concepts in their detector. Kitchen and Rosenfeld [10] presented a few corner-detection methods. The work included methods based on gradient magnitude of gradient direction, change of direction along edge, angle between most similar neighbors, and turning of the fitted surface. Lai and Wu [12] considered edge-corner detection for defective images. Tsai [27] proposed a method for boundary-based corner detection using neural networks. Ji and Haralick [9] presented a technique for corner detection with covariance propagation. Lee and Bien [13] applied fuzzy logic to corner detection.

Fang and Huang [7] proposed a method which was an improvement on the gradient magnitude of the gradient-angle method by Kitchen and Rosenfeld. Chen and Rockett utilized Bayesian labeling of corners using a gray-level corner image model in [4]. Wu and Rosenfeld [29] proposed a technique which examines the slope discontinuities of the x and y projections of an image to find the possible corner candidates. Paler et al. [21] proposed a technique based on features extracted from the local distribution of gray-level values. Rangarajan et al. [22] proposed a detector which tries to find an analytical expression for an optimal function whose convolution with the windows of an image has significant values at corner points. Arrebola et al. [1] introduced corner detection by local histograms of contour chain code. Shilat et al. [23] worked on ridge's corner detection and correspondence. Nassif et al. [18] considered corner location measurement. Sohn et al. [25] proposed a mean field-annealing approach to corner detection.

Zhang and Zhao [30] considered a parallel algorithm for detecting dominant points on multiple digital curves. Kohlmann [11] applied the 2D Hilbert transform to corner detection. Mehrotra et al. [14] proposed two algorithms for edge and corner detection. The first is based on the first-directional derivative of the Gaussian, and the second is based on the second-directional derivative of the Gaussian. Davies [5] applied the generalized Hough transform to corner detection. Zuniga and Haralick [31] utilized the facet model for corner detection. Smith and Brady [24] used a circular mask for corner detection. No derivatives were used. Orange and Groen [20] proposed a model-based corner detector. Other corner detectors have been proposed in [26], [19], [28]. Our survey suggested that the Plessey corner detector, the Kitchen and Rosenfeld detector, and the SUSAN detector [24] have demonstrated good performance. These detectors were therefore chosen as our test detectors.

3 Canny Edge Detector

The CSS-based image corner detector uses the Canny [3] edge detector. During the implementation of the CSS corner detector it was found that Canny edge detector produced a thick edge when edge orientation was 45^o or 135^o.

The Canny edge detector uses a Gaussian function to compute the first derivatives from an image. The process produces two similar gradient values at either side of an edge if the areas at each side of the edge have a constant brightness level. Nonmaximum suppression is meant to ensure that the edge line is thinned and is only one pixel wide. Canny's nonmaximum suppression uses the direction of the gradient at an edge point to look at neighboring pixels. If the chosen neighboring pixels have larger gradient values

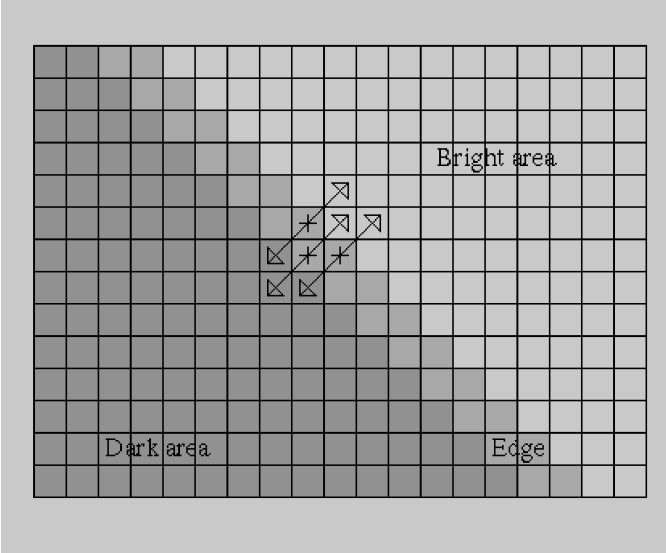


Fig. 1. Canny's nonmaximum suppression with 135^o edges.

than the examined point, the point is removed from the edge map. When there is a 45^o or 135^o edge with uniform areas on either side, the nonmaximum suppression produces a thick edge. This problem is caused by the fact that the gradient direction at the edge point points to nonedge pixels. This can be seen in Fig. 1. The edge points which are examined during the nonmaximum suppression do not see their neighboring edge pixels due to the 45^o or 135^o orientation.

This problem is solved with a small addition to the Canny edge detector algorithm. The final stage should be to compare each edge pixel which has an edge orientation of 45^o or 135^o to one of its horizontal or vertical neighbors. If the neighbor has the same orientation, the other point can be removed.

4 The Curvature Scale-Space Technique

The CSS technique is suitable for recovering invariant geometric features (curvature zero-crossing points and / or extrema) of a planar curve at multiple scales. To compute it, the curve Γ is first parameterized by the arc length parameter u:

$$\Gamma(u) = (x(u), y(u)).$$

An *evolved version* Γ_σ of Γ can then be computed. Γ_σ is defined by [15]:

$$\Gamma_\sigma = (X(u, \sigma), Y(u, \sigma)),$$

where

$$\mathcal{X}(u, \sigma) = x(u) \otimes g(u, \sigma) \qquad \mathcal{Y}(u, \sigma) = y(u) \otimes g(u, \sigma),$$

where $\otimes$ is the convolution operator and $g(u, \sigma)$ denotes a Gaussian of width σ. Note that σ is also referred to as the *scale* parameter. The process of generating evolved versions of Γ as σ increases from zero to infinity (∞) is referred to as the *evolution* of Γ. This technique is suitable for removing noise from and smoothing a planar curve as well as gradual simplification of its shape. In order to find curvature zero-crossings or extrema from evolved versions of the input curve, one needs to compute curvature accurately and directly on an evolved version Γ_σ. Curvature κ on Γ_σ is given by [15]:

$$\kappa(u, \sigma) = \frac{\mathcal{X}_u(u, \sigma)\mathcal{Y}_{uu}(u, \sigma) - \mathcal{X}_{uu}(u, \sigma)\mathcal{Y}_u(u, \sigma)}{\left(\mathcal{X}_u(u, \sigma)^2 + \mathcal{Y}_u(u, \sigma)^2\right)^{1.5}},$$

where

$$\mathcal{X}_u(u, \sigma) = x(u) \otimes g_u(u, \sigma) \qquad \mathcal{X}_{uu}(u, \sigma) = x(u) \otimes g_{uu}(u, \sigma)$$

$$\mathcal{Y}_u(u, \sigma) = y(u) \otimes g_u(u, \sigma) \qquad \mathcal{Y}_{uu}(u, \sigma) = y(u) \otimes g_{uu}(u, \sigma).$$

For examples of contour evolution, see [15].

5 CSS Corner-Detection Method

5.1 Overview

The corners are defined as the local maxima of the absolute value of curvature. At a very fine scale, there exists many such maxima due to noise on the digital contour. As the scale is increased, the noise is smoothed away and only the maxima corresponding to the real corners remain. The CSS corner-detection method finds the corners at these local maxima.

As the contour evolves, the actual locations of the corners change. If the detection is achieved at a large scale the localization of the corners may be poor. To overcome this problem, tracking is introduced in the detection. The corners are located at a high scale σ_{high}, assuring that the corner detection is not affected by noise. σ is then reduced and the same corner points are examined at lower scales. As a result, location of corners may be updated. This is continued until the scale is very low and the operation is very local. This improves localization and the computational cost is low, as curvature values at scales lower than σ_{high} do not need to be computed at every contour point but only in a small neighborhood of the detected corners.

There are local maxima on the evolved contours due to rounded corners or noise. These can be removed by introducing a threshold value t. The curvature of a sharp corner is higher than that of a rounded corner. There is one final addition to the corner candidate declaration. Each local maximum of the curvature is compared to its two neighboring local minima. The curvature of a corner point should be double the curvature of a neighboring extremum. This is necessary since if the contour is continuous and round, the curvature values can be well above the threshold value t and false corners may be declared.

5.2 Outline

The process of CSS image corner detection is as follows:

- Utilize the Canny edge detector to extract edges from the original image.
- Extract the edge contours from the edge image:
  - —Fill the gaps in the edge contours.
  - —Find the T-junctions and mark them as T-corners.
- Compute the curvature at highest scale σ_{high} and determine the corner candidates by comparing the maxima of curvature to the threshold t and the neighboring minima.
- Track the corners to the lowest scale to improve localization.
- Compare the T-corners to the corners found using the curvature procedure and remove corners which are very close.

The following is an explanation of each stage of the CSS corner detector.

5.3 Canny Edge Detection

The first stage of the CSS corner-detection method is edge detection. A Canny edge detector was chosen for this implementation due to its good performance. A small σ was used for Canny to obtain good edge localization.

5.4 Filling the Gaps and T-Junctions

Canny detector can cause gaps at T-junctions and the corners might not be found with the CSS method. Canny can also cause gaps in otherwise continuous edges. When the edge-extraction method arrives at the endpoint of a contour, it performs two checks:

- If the endpoint is nearly connected to another endpoint, fill the gap and continue the extraction.

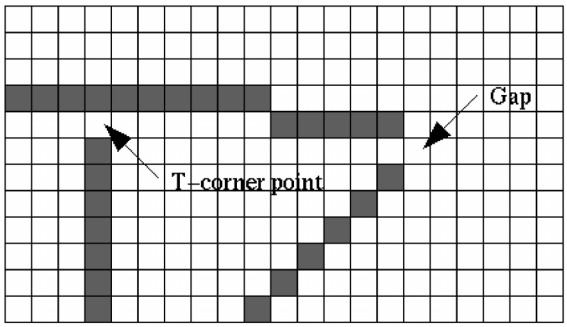


Fig. 2. The two cases of gaps in the edge contours.

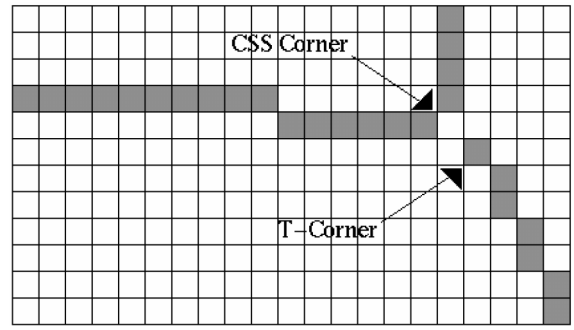


Fig. 3. Case where one corner is marked twice.

- If the endpoint is nearly connected to an edge contour, but not to another endpoint, mark this point as a T-junction corner.

In Fig. 2, both cases of gaps are shown. The T-junction gap is marked as a corner and the gap between two contour ends is filled.

5.5 Initial Corner Points

The edge contours are extracted from the edge image and the absolute value of curvature is computed at the initial scale σ_{high}. The local maxima of absolute curvature are the possible candidates for corner points. A local maximum is either a corner, the top value of a rounded corner or a peak due to noise. The latter two should not be detected as corners. The curvature of a real corner point has a higher value than that of a rounded corner or noise. The corner points are also compared to the two neighboring local minima. The curvature of a corner should be twice that of one of the neighboring local minima. This is because when the shape of the contour is very round, contour curvature can be above the threshold t. The threshold t depends on σ_{high} used and it is set according to it.

5.6 Tracking

After the initial corner points are located, tracking is introduced to the detection. As the corners were detected at scale σ_{high}, the corner localization might not be good. We compute curvature at a lower scale and examine the corner candidates in a small neighborhood of the previous corners. Corner locations are updated, if needed, in this neighborhood. Tracking is continued until scale is very low. This process gives very good localization. No thresholding is needed in the tracking. The number of corners is determined at the initial σ_{high}, and tracking only changes the localization, not the number of corners. Tracking improves the localization of the corners. Corners do not move dramatically during tracking and only a few other curvature values need to be computed.

5.7 Removing False Corners

As described before, corners are declared using two methods and, in some cases, the two methods mark the same corner. In Fig. 3, the case where one corner is marked twice is shown. The edge extraction algorithm examines a small neighborhood when it arrives at the end of a contour. The corner in Fig. 3 is a Y junction and it is marked twice. The CSS method finds a corner on the continuous contour and the edge extraction algorithm marks a T-corner at the end of the other contour as it is nearly connected to a continuous edge contour. The final part of the algorithm is to examine the points marked by the edge-extraction algorithm. These T-junction corners are compared to the corner points found with the CSS method and if they are very close to each other, the T-junction corners are removed. In the implementation, a 5×5 neighborhood was used.

6 Experimental Results and Discussion

The CSS corner detector was tested using four different images and the results are compared with the output of three other corner detectors: Kitchen and Rosenfeld, SUSAN, and Plessey corner detectors. Note that we attempted to obtain the best possible results for each corner detector tested by searching for parameter values that appeared to yield the best results. The first test image is an artificially created one with significant Gaussian noise added to the image. The second test image is a real image of blocks. Much texture and noise is present in the image. The third test image is an image of a house. This image has a lot of small details and texture in the brick wall. Finally, an image of a lab is used.

The results showed that CSS corner detector gave the best results in each of the four cases, and that it was robust to image noise.

The CSS detector performed very well on the noisy artificial image, but the other three other detectors did not perform well, as seen in Fig. 4. The real block image corner detection was a more difficult task for the detectors. Again the CSS corner detector gave the best results amongst the four. The results are seen in Fig. 5.

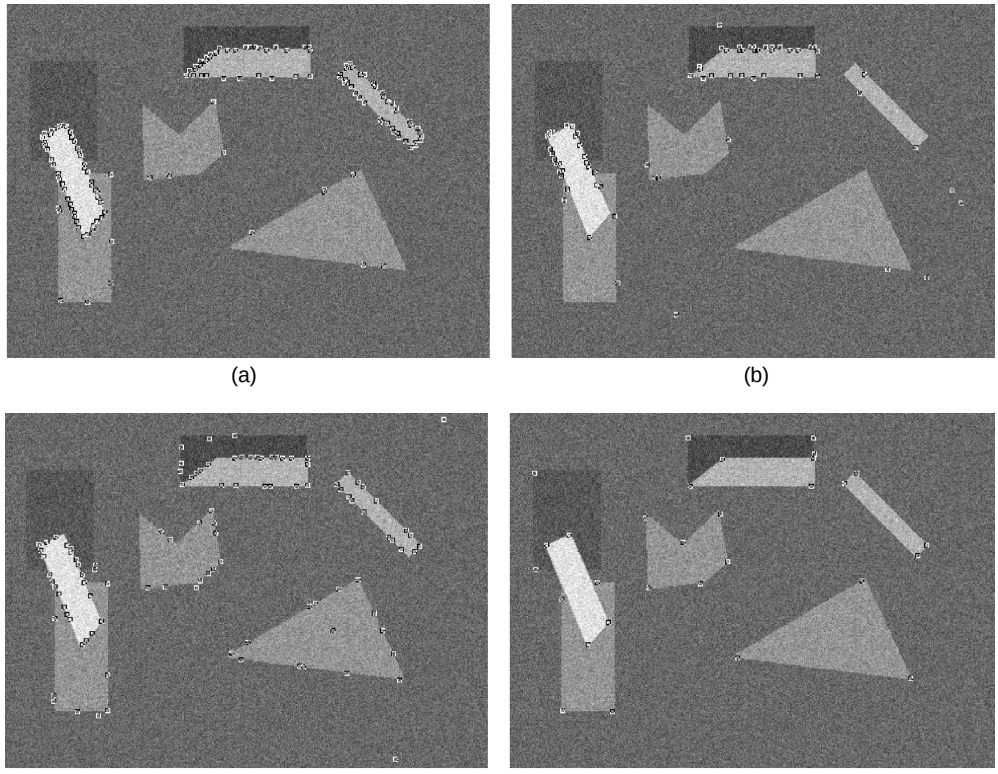
House image was a difficult task for all the detectors as the details are very varied. Overall, the CSS detector still performed better. Fig. 6 shows the results. Finally, the results on the lab image is shown in Fig. 7. The CSS detector performed very well with the image, but the three other detectors had serious problems with very obvious corners.

The speed of the corner detectors was measured on a Sun SPARCstation 5. The Kitchen and Rosenfeld detector was the fastest of these detectors, but the rest of the detectors had quite similar speeds. All the detectors are implemented in C++. Note that the source code for the CSS corner detector can be obtained at:

http://www.ee.surrey.ac.uk/Personal/F.Mokhtarian/corners/source.html.

Over 80 percent of the time used by the CSS detector is spent in edge detection.

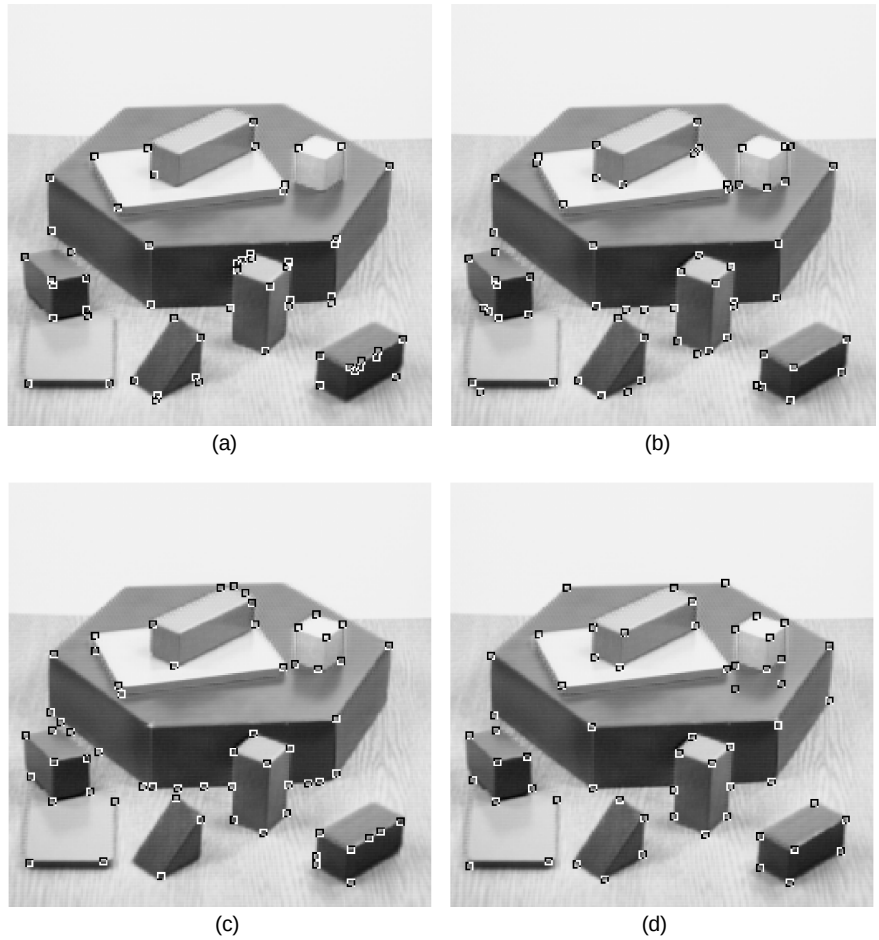
The CSS corner detector uses only two important parameters. Experiments showed that $\sigma_{high} = 4$ gave good results with almost all images. The threshold t depends on the value of σ_{high} and with $\sigma_{high} = 4$, the threshold can be set to 0.03. Other values of σ_{high} are also possible and for a very noisy image $\sigma_{high} = 8$ and threshold $t = 0.02$ can be used. Starting with $\sigma_{high} = 4$, tracking can be accomplished at $\sigma = 2$, $\sigma = 1$, and $\sigma_{final} = 0.7$. The final scale σ_{final} should be as local as possible to ensure good localization. It was found that the results were not sensitive to the exact values of the parameters, and that the same values worked well for the different test images used except for one that was very noisy by intention. Note, however, that the *detection* of corners can be carried out at multiple scales. As a result, by adjusting the scale, the number of corner points recovered can increase or decrease, depending on the requirements of later processes. For example, in a motion tracking system, object detail is not needed when tracking in a noncluttered scene, and a small number of corners will be sufficient. However, when part of the object becomes occluded, a larger number of corners will be required.



(a) (b)

(c) (d)

Fig. 4. Artificial test image with noise. (a) Plessey. (b) Kitchen/Rosenfeld. (c) SUSAN. (d) CSS.



(a) (b)

(c) (d)

Fig. 5. Blocks image. (a) Plessey. (b) Kitchen/Rosenfeld. (c) SUSAN. (d) CSS.

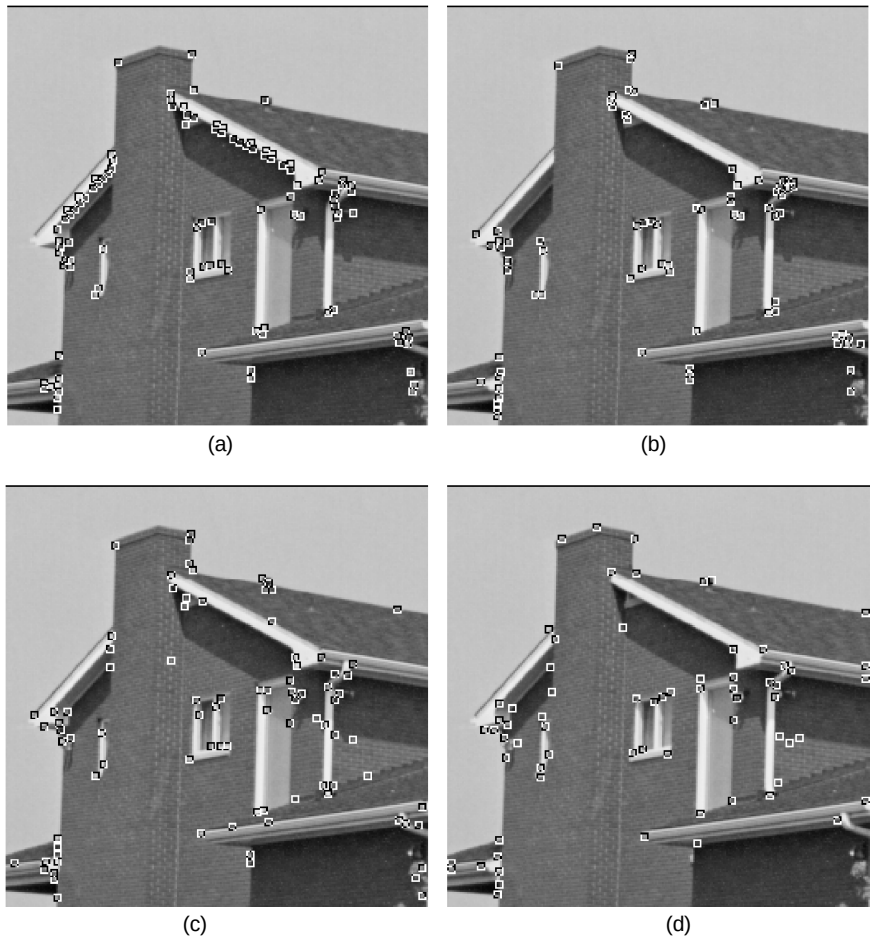


Fig. 6. House image. (a) Plessey. (b) Kitchen/Rosenfeld. (c) SUSAN. (d) CSS.

It has been argued that corner detectors that perform directly on images may be preferable, since they do not depend on the results of an earlier stage (such as edge detection). It should be pointed out that most corner detectors carry out some form of edge detection either implicitly or explicitly. As a result, even when they appear to be directly applicable to the input image, the results are affected by the implicit edge detection. The CSS detector simply makes the process explicit.

The CSS detector makes both image edges and image corners available for later processes. It can also provide additional point features as well as the traditional corners. The new features are the curvature zero-crossings or inflection points of the image edge contours recovered in a similar way as the corners. They can complement the traditional corners when used by later processes. For example, they can be utilized by motion-tracking systems in an area of the image where there is a lack of corner features.

7 Conclusions

This paper proposed a new corner-detection method based on the curvature scale-space technique. The edges of a real image were extracted using the Canny edge detector. The gaps between two close contours were examined in order to find T-junction corners or to fill the gap to form a continuous contour. Curvature maxima were extracted at a high scale and the corner locations were tracked at multiple lower scales to improve localization. Finally, the T-junction corners were compared to the CSS corners in order to remove corners marked twice. The CSS image corner-detection method was very robust with respect to noise and performed better than the other detectors it was compared to.

References

[1] F. Arrebola, A. Bandera, P. Camacho, and F. Sandoval, "Corner Detection by Local Histograms of Contour Chain Code," *Electronics Letters*, vol. 33, no. 21, pp. 1,769-1,771, 1997.

[2] P.R. Beaudet, "Rotationally Invariant Image Operators," *Int'l Joint Conf. Pattern Recognition*, pp. 579-583, 1978.

[3] J.F. Canny, "A Computational Approach to Edge Detection," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679-698, Nov. 1986.

[4] W.C. Chen and P. Rockett, "Bayesian Labelling of Corners Using a Grey-Level Corner Image Model," *IEEE Int'l Conf. Image Processing*, vol. 1, pp. 687-690, 1997.

[5] E.R. Davies, "Application of the Generalized Hough Transform to Corner Detection," *IEE Proc.*, vol. 135, pp. 49-54, 1988.

[6] L. Dreschler and H.H. Nagel, "Volumetric Model and 3D Trajectory of a Moving Car Derived From Monocular TV Frame Sequences of a Street Scene," *Int'l Joint Conf. Artificial Intelligence*, pp. 692-697, 1981.

[7] J.Q. Fang and T.S. Huang, "A Corner Finding Algorithm for Image Analysis and Registration," *Proc. AAAI Conf.*, pp. 46-49, 1982.

[8] C.G. Harris, "Determination of Ego-Motion From Matched Points," *Proc. Alvey Vision Conf.*, Cambridge, UK, 1987.

[9] Q. Ji and R.M. Haralick, "Corner Detection With Covariance Propagation," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 362-367, 1997.

[10] L. Kitchen and A. Rosenfeld, "Gray Level Corner Detection," *Pattern Recognition Letters*, pp. 95-102, 1982.

[11] K. Kohlmann, "Corner Detection in Natural Images Based on the 2-D Hilbert Transform," *Signal Processing*, vol. 48, no. 3, pp. 225-234, 1996.

[12] K.K. Lai and P.S.Y. Wu, "Effective Edge-Corner Detection Method for Defected Images," *Proc. Int'l Conf. Signal Processing*, vol. 2, pp. 1,151-1,154, 1996.

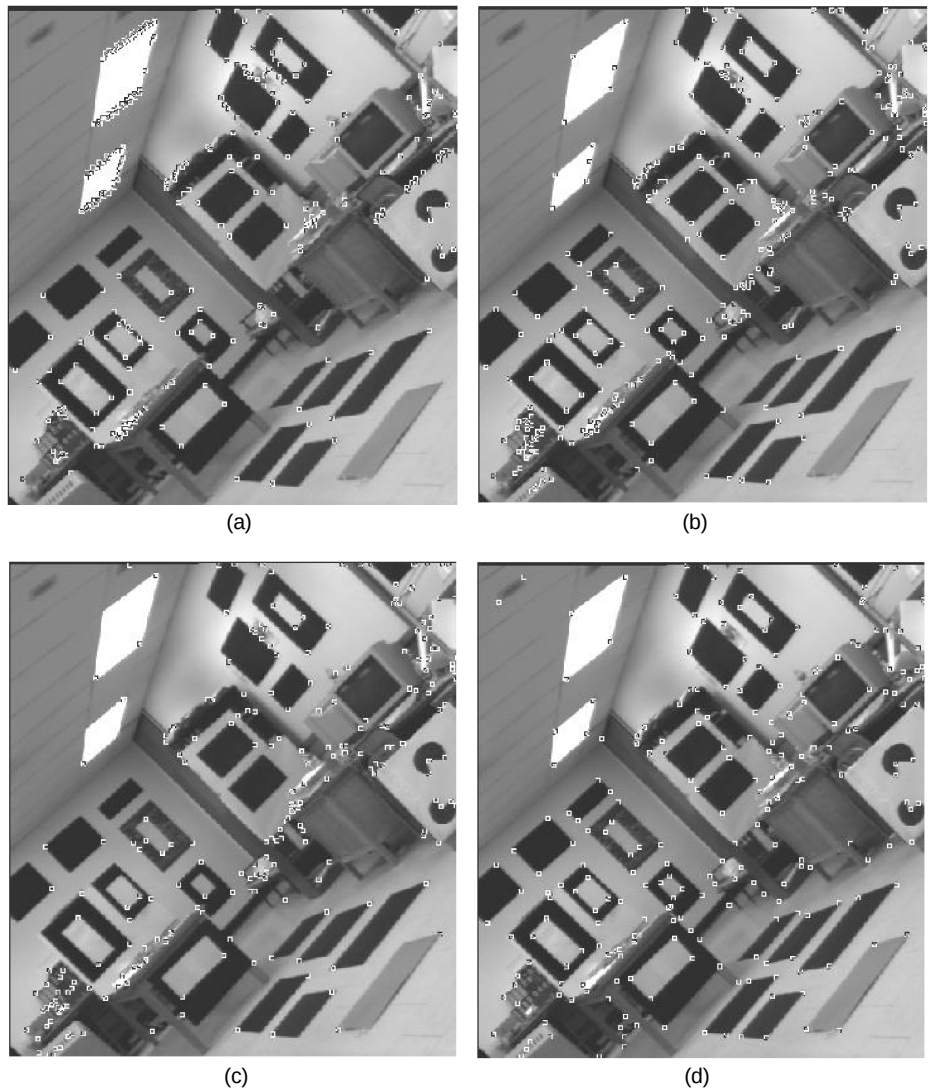


Fig. 7. Lab image. (a) Plessey. (b) Kitchen/Rosenfeld. (c) SUSAN. (d) CSS.

[13] K.J. Lee and Z. Bien, "Grey-Level Corner Detector Using Fuzzy Logic," *Pattern Recognition Letters*, vol. 17, no. 9, pp. 939-950, 1996.
[14] R. Mehrotra, S. Nichani, and N. Ranganathan, "Corner Detection," *Pattern Recognition*, vol. 23, no. 11, pp. 1,223-1,233, 1990.
[15] F. Mokhtarian and A.K. Mackworth, "A Theory of Multi-Scale, Curvature-Based Shape Representation for Planar Curves," *IEEE Trans. Pattern Analysis and Machine Intelligence*, vol. 14, no. 8, pp. 789-805, Aug. 1992.
[16] F. Mokhtarian and R. Suomela, "Curvature Scale Space for Robust Image Corner Detection," *Int'l Conf. Pattern Recognition*, Brisbane, Australia, 1998.
[17] H.P. Moravec, "Towards Automatic Visual Obstacle Avoidance," *Proc. Int'l Joint Conf. Artificial Intelligence*, p. 584, 1977.
[18] S. Nassif, D. Capson, and A. Vaz, "Robust Real-Time Corner Location Measurement," *Proc. IEEE Conf. Instrumentation and Measurement Technology*, pp. 106-111, 1997.
[19] A. Noble, "Finding Corners," *Image and Vision Computing*, vol. 6, pp. 121-128, 1988.
[20] C.M. Orange and F.C.A. Groen, "Model Based Corner Detection," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, 1993.
[21] K. Paler, J. Foglein, J. Illingworth, and J. Kittler, "Local Ordered Grey Levels as an Aid to Corner Detection," *Pattern Recognition*, vol. 17, no. 5, pp. 535-543, 1984.
[22] K. Rangarajan, M. Shah, and D. Van Brackle, "Optimal Corner Detector," *Computer Vision, Graphics, and Image Processing*, vol. 48, pp. 230-245, 1989.
[23] E. Shilat, M. Werman, and Y. Gdalyahu, "Ridge's Corner Detection and Correspondence," *Proc. IEEE Conf. Computer Vision and Pattern Recognition*, pp. 976-981, 1997.
[24] S.M. Smith and J.M. Brady, "SUSAN—A New Approach to Low Level Image Processing," Defence Research Agency, Technical Report no. TR95SMS1, Farnborough, England, 1994.
[25] K. Sohn, J.H. Kim, and W.E. Alexander, "Mean Field Annealing Approach to Robust Corner Detection," *IEEE Trans. Systems, Man, and Cybernetics*, vol. 28B, no. 1, pp. 82-90, 1998.
[26] M. Trajkovic and M. Hedley, "Fast Corner Detection," *Image and Vision Computing*, vol. 16, no. 2, pp. 75-87, 1998.
[27] D.M. Tsai, "Boundary Based Corner Detection Using Neural Networks," *Pattern Reognition*, vol. 30, no. 1, pp. 85-97, 1997.
[28] H. Wang and M. Brady, "A Practical Solution to Corner Detection," *Proc. Int'l Conf. Image Processing*, vol. 1, 1994.
[29] Z.O. Wu and A. Rosenfeld, "Filtered Projections as an Aid to Corner Detection," *Pattern Recognition*, vol. 16, no. 31, 1983.
[30] X. Zhang and D. Zhao, "Parallel Algorithm for Detecting Dominant Points on Multiple Digital Curves," *Pattern Recognition*, vol. 30, no. 2, pp. 239-244, 1997.
[31] O.A. Zuniga and R.M. Haralick, "Corner Detection Using the Facet Model," *Proc. Conf. Pattern Recognition and Image Processing*, pp. 30-37, 1983.