Curvature Scale Space Based Image Corner Detection

Farzin Mokhtarian and *Riku Suomela*
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

ABSTRACT

This paper describes a new 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 Canny detector sometimes leaves a gap in T-junctions so during edge extraction, the gaps are examined to locate the T-junction corner points. 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 the locations are tracked through multiple lower scales to improve localization. The final stage is to compare T-junction corners to CSS corners and remove duplicates. This method is very robust to noise and we believe that it performs better than the existing corner detectors.

1 Introduction

Corner detection is an important task in various machine vision and image processing systems. Applications include motion tracking, object recognition, and stereo matching. It is a fundamental problem and several different algorithms have been proposed. Corner detection should satisfy a number of criteria in order to perform satisfactorily. We propose the following criteria:

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

This paper proposes a new corner detection method 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 edge contours of the real image. In the implementation of the CSS detector a Canny edge detector was used [1]. The Canny detector was selected since we were satisfied with the quality of its results. Note however that the Canny detector is not a critical part of our corner detector: it can be replaced with another edge detector if the new edge detector is believed to perform better.

Section 2 gives a brief overview of the CSS method and section 3 describes the proposed corner detector. The performance of a corner detector is best evaluated with real test images and in section 4 the results of the CSS corner detector are presented. Four images with different properties are used in the experiments.

2 The curvature scale space technique

The curvature scale space 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 parametrized by the arc length parameter u:

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

An *evolved version* Γ_σ of Γ can then be computed [10]:

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

where

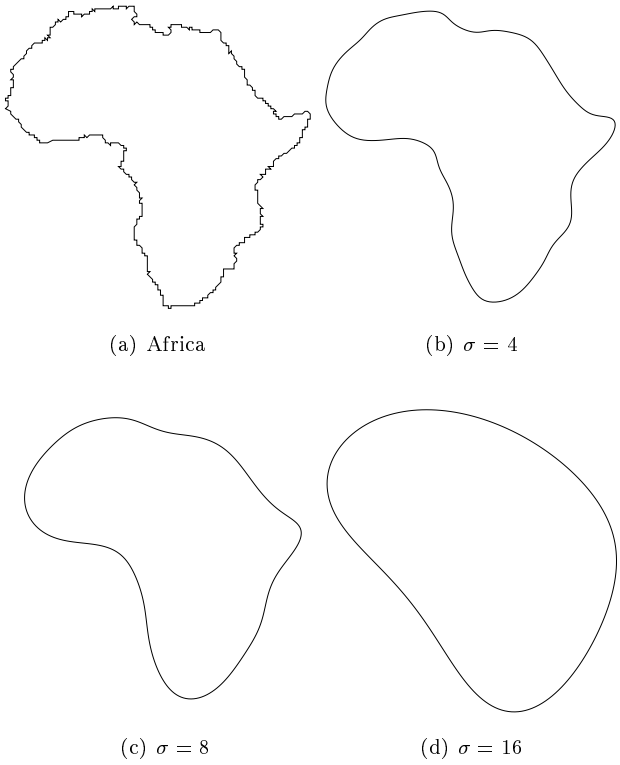
$$\mathcal{X}(u,\sigma) = x(u) \otimes g(u,\sigma)$$

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

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 0 to ∞ 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 [7, 9].

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 of that curve. Curvature κ on Γ_σ is given by:

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



(a) Africa (b) $\sigma = 4$

(c) $\sigma = 8$ (d) $\sigma = 16$

Figure 1: Evolution of Africa

where

$$\mathcal{X}_u(u,\sigma) = \frac{\partial}{\partial u}(x(u) \otimes g(u,\sigma)) = x(u) \otimes g_u(u,\sigma)$$

$$\mathcal{X}_{uu}(u,\sigma) = \frac{\partial^2}{\partial u^2}(x(u) \otimes g(u,\sigma)) = x(u) \otimes g_{uu}(u,\sigma)$$

$$\mathcal{Y}_u(u,\sigma) = y(u) \otimes g_u(u,\sigma)$$

and

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

The function defined implicitly by $\kappa(u,\sigma) = 0$ is the CSS image of Γ. Figure 1 shows an example of contour evolution. Figure 2 shows the CSS image of the contour shown in figure 1.

3 CSS corner detection method

3.1 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 peaks corresponding to the real corners remain. The CSS corner detection method finds the corners at these local maxima. The problem is to find the right scale where the corners are to be detected.

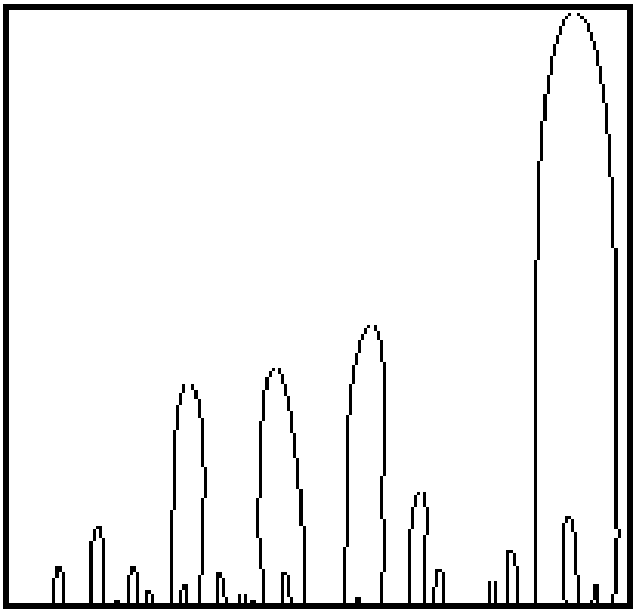


Figure 2: Curvature Scale Space image of Africa

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 will be poor. To overcome this problem tracking is introduced in the detection. The corners are located at a high scale σ_{high}, this assures that the corner detection is not affected by noise. Then sigma is 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 not high, as curvature values at scales lower than σ_{high} do not need to be computed at every contour point but only in a small neighbourhood of the detected corners.

There are local maxima on the evolved contours due to rounded corners. These can be removed by introducing a threshold value. 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 curvature is compared to its two neighbouring local minima. The curvature of a corner point has to be 2 times higher than the curvature of a neighboring extremum. This is necessary since if the contour is continuous and round, the curvature values are well above the threshold value and false corners would be declared.

3.2 Outline

The process of CSS image corner detection is as follows:

- Utilize the Canny edge detector to extract edges from the original image

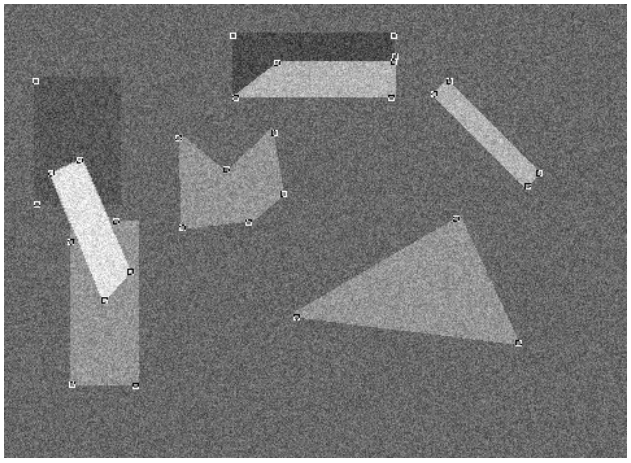


Figure 3: Artificial test image with noise

- Extract the edge contours from the edge image
  - Fill the gaps in the edge contour
  - Find 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 local 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 too close.

4 Experimental results and discussion

The CSS corner detector was tested using four different images and the results were compared to the output of three other corner detectors: Kitchen and Rosenfeld [5], Susan [15] and Plessey [4] corner detectors. Other corner detectors [3, 17, 13, 2, 11, 14, 8, 6, 12, 16] were also considered. 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 test image with well-defined geometric shapes. It was obtained by adding Gaussian noise with zero mean and standard deviation 20. 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 with small details and texture in the brick wall. Finally the fourth test image is a rotated lab image.

The results show that the CSS corner detector gives the best results in each of the four cases. As the noise in the images increases, the performance of the CSS image corner detector was not affected much. Due to shortage of space, only the results for the CSS detector are shown here (see figures 3-6).

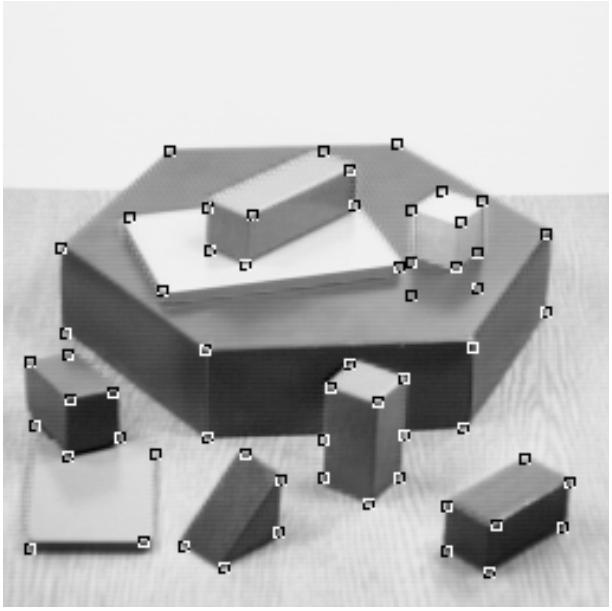


Figure 4: Blocks image

All the detectors were implemented in C++, and had rather similar speeds. Tests showed that about 85% of the time used by the CSS detector is spent in the edge detection stage.

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 non-cluttered 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.

Figure 5: House image

Figure 6: Lab image

It has been argued that corner detectors that perform directly on images may be preferrable 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.

References

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

[2] E. R. Davies. Application of the generalized hough transform to corner detection. *IEE Proceedings*, 135:49–54, 1988.

[3] J. Q. Fang and T. S. Huang. A corner finding algorithm for image analysis and registration. In *Proc. AAAI Conf*, pages 46–49, 1982.

[4] C. G. Harris. Determination of ego-motion from matched points. In *Proc. Alvey Vision Conf.*, Cambridge, UK, 1987.

[5] L. Kitchen and A. Rosenfeld. Gray level corner detection. *Pattern Recognition Letters*, pages 95–102, 1982.

[6] H. J. Lee and H. C. Deng. Three-frame corner matching and moving object extraction in a sequence of images. *Computer Vision, Graphics, and Image Processing*, 52:210–238, 1990.

[7] A. K. Mackworth and F. Mokhtarian. Scale-based description of planar curves. In *Proc Canadian Society for Computational Studies of Intelligence*, pages 114–119, London, Ontario, 1984.

[8] R. Mehrotra, S. Nichani, and N. Ranganathan. Corner detection. *Pattern Recognition*, 23(11):1223–1233, 1990.

[9] F. Mokhtarian and A. K. Mackworth. Scale-based description and recognition of planar curves and two-dimensional shapes. *IEEE Trans Pattern Analysis and Machine Intelligence*, 8(1):34–43, 1986.

[10] F. Mokhtarian and A. K. Mackworth. A theory of multiscale, curvature-based shape representation for planar curves. *IEEE Trans Pattern Analysis and Machine Intelligence*, 14(8):789–805, 1992.

[11] A. Noble. Finding corners. *Image and Vision Computing*, 6:121–128, 1988.

[12] C. M. Orange and F. C. A. Groen. Model based corner detection. In *Proc IEEE Conf on Computer Vision and Pattern Recognition*, 1993.

[13] K. Paler, J. Foglein, J. Illingworth, and J. Kittler. Local ordered grey levels as an aid to corner detection. *Pattern Recognition*, 17(5):535–543, 1984.

[14] K. Rangarajan, M. Shah, and D. V. Brackle. Optimal corner detector. *Computer Vision, Graphics and Image Processing*, 48:230–245, 1989.

[15] S. M. Smith. A new class of corner finder. In *Proc British Machine Vision Conference*, 1992.

[16] H. Wang and M. Brady. A practical solution to corner detection. In *Proc International Conference on Image Processing*, volume 1, 1994.

[17] O. A. Zuniga and R. M. Haralick. Corner detection using the facet model. In *Proc Conference on Pattern Recognition and Image Processing*, pages 30–37, 1983.