AN AFFINE RESILIENT CURVATURE SCALE-SPACE CORNER DETECTOR

Mohammad Awrangjeb, Guojun Lu, and Manzur Murshed

Gippsland School of Information Technology, Monash University, Churchill Vic 3842, Australia
{Mohammad.Awrangjeb, Guojun.Lu, Manzur.Murshed}@infotech.monash.edu.au

ABSTRACT

Curvature scale-space (CSS) corner detectors look for curvature maxima or inflection points on planar curves. They use arc-length parameterized curvature. Therefore, they are not robust to affine transformations since the arc-length of a curve is not preserved under affine transformations. However, the affine-length of a curve is relatively invariant to affine transformations. This paper presents an improved CSS corner detector by applying the affine-length parameterized curvature to the CSS corner detection technique. A thorough robustness study has been carried out on a large database considering a wide range of affine transformations.

Index Terms— Corner detection, curvature scale-space

1. INTRODUCTION

Corners which are visually significant features can be used for geometric distortion corrections in images [1], stereo matching, and object recognition. A large number of corner and interest point detectors have been proposed in the literature [2-6]. Mokhtarian and Mohanna [7] classified them into two groups: intensity-based [2-3] and contour-based [4-6] methods. Intensity-based methods estimate a measure which is intended to indicate the presence of a corner directly from the image grey values. In contrast, contour-based methods first recover planar curves using some edge detector and then search for the curvature maxima along those curves. In fact, both types of methods carry out some form of edge detection implicitly (intensity-based) or explicitly (contour-based). However, implicit edge detection affects the result of corner detection [5]. Corner detectors can also be divided into two: single-scale detectors [2] and multi-scale detectors [3-6]. Single-scale detectors work well if the image has similar size features, but ineffective otherwise; because either fine or coarse scale features are poorly detected, but images may contain both kinds of features. To improve the effectiveness of corner detection, multi-scale corner detectors have been proposed. They build the three-dimensional space through smoothing the image or edge contours using the Gaussian kernel.

This paper presents an improved *curvature scale-space* (CSS) corner detector which is affine resilient (ARCSS detector). The original CSS detector and its enhanced version [5-6] parameterize curves with the arc-length which is not affine invariant. Instead, the proposed corner detector parameterizes curves with the affine-length which is relatively invariant to affine transformations. Moreover, the existing CSS detectors do not use proper evaluation metrics for performance comparisons. We modify an evaluation metric for the purpose of fair comparisons. Experimental results show that the proposed ARCSS corner detector offers better performance in terms of both repeatability and localization.

In literature, affine-length has been used for affine invariant shape recognition [8]. The proposed corner detector is different from [8] due to two reasons. First, [8] uses higher order derivatives of up to third order. In digital implementation, precise approximation of higher order derivatives is a difficult task and it often causes instability and involves more errors. We will show by mathematical derivation that the affine-length parameterized curvature involves up to second-order derivatives. Second, in shape recognition, it is assumed that the object always maintains a closed contour even though its shape and size may be changed due to affine transformations. Consequently, they match shapes using the CSS image [8] and do not explicitly use corner positions. However, any corner detector explicitly obtains corner positions.

2. CURVATURE SCALE-SPACE CORNER DETECTORS

The CSS corner detectors, in general, extract planar curves from the image using some edge detector and parameterize each curve using the arc-length. Then they smooth each curve in CSS using the Gaussian kernel with different scales in order to remove noise. Thereafter, they calculate absolute curvature on each point of the curve at either all scales or one or more specific scales. Then they look for curvature maxima points as corners based on some constraints. If corners are detected at all scales, those which survive in most of the scales are finalized. If corners are detected at some specific scales, they are tracked down to the finest scale in order to improve localization.

For a given parametric vector equation of a planar curve $\Gamma(t) = (x(t), y(t))$, the curvature is defined in [5] as:

$$\kappa(t) = \frac{\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)}{(\dot{x}^2(t) + \dot{y}^2(t))^{3/2}}, \tag{1}$$

where $\dot{x}(t)$ and $\dot{y}(t)$ are first and $\ddot{x}(t)$ and $\ddot{y}(t)$ are second order derivatives. The arc-length s between two points P_1 and P_2 is

$$s = \int_{P_1}^{P_2} \sqrt{\dot{x}^2(t) + \dot{y}^2(t)}\,dt\,. \tag{2}$$

When the curve is parameterized by s, it is shown in [4] that

$$\dot{x}^2(s) + \dot{y}^2(s) = 1\,. \tag{3}$$

Therefore, the curvature on the arc-length parameterized curve in CSS is given by according to (1) as

$$\kappa(s,\sigma) = \dot{\mathrm{X}}(s,\sigma)\ddot{\mathrm{Y}}(s,\sigma) - \ddot{\mathrm{X}}(s,\sigma)\dot{\mathrm{Y}}(s,\sigma)\,, \tag{4}$$

where according to the property of the Gaussian convolution that states that the convolution and derivatives are commutative

$$\begin{aligned} &\dot{\mathrm{X}}(s,\sigma) = x(s) * \dot{g}(s,\sigma), \qquad \ddot{\mathrm{X}}(s,\sigma) = x(s) * \ddot{g}(s,\sigma), \\ &\dot{\mathrm{Y}}(s,\sigma) = y(s) * \dot{g}(s,\sigma), \text{ and } \ddot{\mathrm{Y}}(s,\sigma) = y(s) * \ddot{g}(s,\sigma), \end{aligned} \tag{5}$$

where $g(.,\sigma)$ denotes the Gaussian kernel of width (scale) σ and * denotes the convolution.

The first CSS corner detector [4] investigates curve behaviors in the scale-space and transforms it into a tree. Then a coarse-to-fine tree parsing technique, which is computationally demanding, is developed to detect the corners. However, it detects false corners, e.g., corners on a circle, because of quantization noise. Mokhtarian and Suomela [5] detects corners at a high scale in CSS and tracks them through multiple lower scales in order to improve localization (CSS detector). When corners are detected in a very high scale, this method shows high robustness to noise but loses many true corners. On the other hand, if corners are detected in a low scale it introduces false corners. Moreover, its performance is limited due to the use of the single curvature threshold [7]. The enhanced CSS corner detector (ECSS detector) detects corners in multiple (three) medium scales with different thresholds [6]. The ECSS detector, though performed better than the CSS detector as reported in [7], was found less robust than their former method in our robustness tests. The reason is that both of them use arc-length parameterization but while the CSS detector looks for corners at a high scale offering better noise immunity, the ECSS detector finds corners at three medium scales and, therefore, introduces many weak corners. Nevertheless, the ECSS detector requires less time for corner localization than the CSS detector.

3. PROPOSED CORNER DETECTOR

The arc-length, used to parameterize planar curves by the existing detectors, is not preserved under affine transformations [8]. Therefore, unlike the existing detectors [4-6], we parameterize planar curves using affine-length. To avoid choosing false corners and missing true corners, we detect corners in three consecutive medium scales like the ECSS detector and then track the detected corners down to the finest scale in two steps. Since a subset of true corners which are strong enough to survive in affine transformations should suffice, we tune the parameters at different steps so that weak corners, which are locally unstable and also known as round corners, are not detected.

3.1. Affine-length Parameterization

The affine-length τ between two points P_1 and P_2 is

$$\tau = \int_{P_1}^{P_2} (\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t))^{1/3} dt \,, \tag{6}$$

which is absolutely invariant to rotation, but relatively invariant to scale change [8], hence to affine transformations. Direct affine-length parameterization in (1) using (6) involves up to third order derivatives of $x(t)$ and $y(t)$ [8]. However, we show below that the numerator of (1), which incurs higher order derivatives, equals to one in the case of affine-length parameterization. Now by differentiating $x(\tau)$ and $y(\tau)$

$$\dot{x}(\tau) = \frac{\dot{x}(t)}{(\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t))^{1/3}}, \dot{y}(\tau) = \frac{\dot{y}(t)}{(\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t))^{1/3}} \,. \tag{7}$$

Again by differentiating $\dot{x}(\tau)$ and $\dot{y}(\tau)$ we have

$$\begin{aligned} \ddot{x}(\tau) &= \frac{3\ddot{x}(t)\left(\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right) - \dot{x}(t)\left(\dot{x}(t)\dddot{y}(t) - \dddot{x}(t)\dot{y}(t)\right)}{3\left(\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right)^{5/3}}, \\ \ddot{y}(\tau) &= \frac{3\ddot{y}(t)\left(\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right) - \dot{y}(t)\left(\dot{x}(t)\dddot{y}(t) - \dddot{x}(t)\dot{y}(t)\right)}{3\left(\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right)^{5/3}}, \end{aligned} \tag{8}$$

where $\dddot{x}(t)$ and $\dddot{y}(t)$ are third order derivatives. From (7) and (8)

$$\dot{x}(\tau)\ddot{y}(\tau) - \ddot{x}(\tau)\dot{y}(\tau) = 1 \,. \tag{9}$$

Therefore, the curvature on the affine-length parameterized curve in CSS is given by according to (1) as

$$\kappa(\tau,\sigma) = \frac{1}{\left(\dot{\mathrm{X}}^2(\tau,\sigma) + \dot{\mathrm{Y}}^2(\tau,\sigma)\right)^{3/2}}, \tag{10}$$

where according to the property of the Gaussian convolution

$$\dot{\mathrm{X}}(\tau,\sigma) = x(\tau) * \dot{g}(\tau,\sigma) \text{ and } \dot{\mathrm{Y}}(\tau,\sigma) = y(\tau) * \dot{g}(\tau,\sigma) \,. \tag{11}$$

The affine-length parameterized curvature in (10) incurs only up to second order derivatives of $x(t)$ and $y(t)$ like the arc-length parameterized curvature in (4). So, the affine-length parameterized curvature can be exploited to extract robust corners with the same computational cost as the arc-length parameterized curvature.

3.2. Corner Detection

We use Canny edge detector [9] to extract edges (planar curves) from gray-scale images. On each curve, corners are defined as the local maxima of the absolute curvature, defined in (10). We detect corners in three medium scales like the ECSS detector. We remove round corners by introducing three thresholds corresponding to three detection scales; since the curvature of a sharp corner is higher than that of a round corner. False corners are eliminated by comparing each curvature maximum with its two neighboring minima based on the assumption that the curvature of a corner point should be at least double the curvature of a neighboring minimum [5]. The outline of the proposed corner detector is:

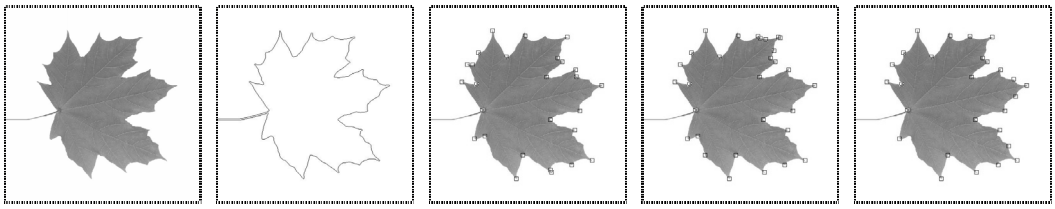
- Find edge image using the Canny edge detector.
- Extract edges from the edge image:
  a. fill gaps if they are within a range and select long edges,
  b. find T-junctions and mark them as T-corners.
- Parameterize each edge with its affine-length.
- For each parameterized edge, compute absolute curvature at an appropriate scale in $\{\sigma_m, \sigma_{m+1}, \sigma_{m+2}\}$ and determine corners by comparing the curvature maxima to the corresponding curvature threshold of edge in $\{t_m, t_{m+1}, t_{m+2}\}$ and the neighboring minima.
- Track the corners down to the lowest scale considering a small neighborhood in order to improve localization.
- Further track the corners on the original edge at the lowest scale considering the same neighborhood size.
- Remove multiple occurrences of same corners, compare T-corners with the tracked corners and add those T-corners which are far away from the detected corners.

3.2.1. Edge Extraction and Selection

In default parameter setup, the Canny edge detector extracts too many edges of different lengths, many of which do not contain strong corners. Furthermore, the number and lengths of the extracted edges may vary significantly in affine transformations. In order to extract strong edges, we set the low and high thresholds of the Canny edge detector at 0.2 and 0.7, respectively. While extracting edges from the edge image we allow a gap size of 1 pixel. We select the extracted edges of length n_p (in pixels) if

$$n_p > (w+h)/\alpha \,, \tag{12}$$

where w and h are width and height of the image and α is a length control parameter. If α is small only long edges are selected, if α is



(a) original image (b) edge map of (a) (c) CSS detector (d) ECSS detector (e) ARCSS detector

Fig. 1: Corner detection in the original maple-leaf image by different curvature scale-space corner detectors

large short edges are also selected. In our experiments, we set $\alpha = 15$. The above edge extraction set up avoids selecting large number of weak and very short edges, thereby helps speeding up the later steps. If an edge runs through any point which is 2 pixels away from an end of another edge, we select that end as a T-junction and add to the set T-corners.

3.2.2. Affine-length Parameterization and Corner Detection

We then parameterize each selected edge $\Gamma(t)$ using its affine-length τ and select $n = floor(\tau)$ equally distant points on the parameterized curve $\Gamma(\tau)$. Therefore, the distance between two successive points on $\Gamma(\tau)$ is affine-length 1 on $\Gamma(t)$. This point selection strategy reduces the possible division by zero effect that may be caused in the case of long horizontal or vertical edges during curvature calculation. We set corner detection scales $\sigma_m = 3$, $\sigma_{m+1} = 4$, and $\sigma_{m+2} = 5$ and corresponding thresholds $t_m = 0.04$, $t_{m+1} = 0.035$, and $t_{m+2} = 0.03$ for short, medium, and long edges respectively. The short, medium, and long definitions of affine parameterized edges are defined by the constraints $n \leq 100$, $100 < n \leq 300$, and $n > 300$, respectively. We compute absolute curvature at all points of $\Gamma(\tau)$. A local maximum is either a corner, the top of a round corner, or a peak due to noise. The later two should not be detected as true corners [5]. The curvature of a true corner should be greater than that of a round corner. If a curvature maximum is greater than the corresponding edge threshold, the maximum point is selected as a corner candidate, and thus eliminating round corners. In the smoothed curve, there may be points (due to noise or false corners) whose curvature values are greater than the edge threshold. A corner candidate is added to the corner set if the curvature at this point is at least double of its two neighboring minima, and hence eliminating false corners.

3.2.3. Corner Tracking and Adding T-corners

As corners are detected at coarse scales, their localization may not be good. The improvement is done in two steps. In first step, we track the corners on $\Gamma(\tau)$ down to the lowest scale by considering a neighborhood of size 3 on each side of each corner. For example, if a corner is detected at the p-th point on $\Gamma(\tau)$ at $\sigma = 5$, first we track this corner at $\sigma = 4$ and consider 7 consecutive points, where the p-th point is the midpoint. The point having the maximum absolute curvature is the tracked position of that corner at $\sigma = 4$ and the successive tracking is executed at $\sigma = 3$, $\sigma = 2$, and $\sigma = 1$. In second step, each tracked corner is further tracked on the original edge $\Gamma(t)$ at $\sigma = 1$ using the same neighborhood size. Note that no threshold is used in the tracking system which only changes the corner positions, not the number of corners. However, corners do not move dramatically during tracking and only a few other (neighbors) curvature values need to be computed during tracking [5]. As a consequence, corner localization is improved without increasing the computational cost since the number of curvature computations is almost the same as the ECSS detector. During corner tracking, there may be some corners that merge at the same point with the same curvature value. Therefore, once we have the tracked corner set, we refine the set so that there is only one existent of any corner. At last, a T-corner is added to the final corner set if it is far away from the detected corners considering a 10×10 neighborhood.

4. PERFORMANCE EVALUATION

Most of the corner detectors use only a few of images and test their performance using the *ground truth* or *visual inspection* based on human intuition [4-7]. This evaluation criterion is useless for proper robustness tests. First, it is very hard to point out all corner locations in natural images by human intuition. Second, human eyes are unable to measure the corner strength. Third, the volume of works with a large image database for ground truth collection prohibits its adoption. Finally, there is no standard procedure to collect ground truth, e.g., how many people should be involved and how to assess their subjective decisions.

Repeatability is defined as the ratio R_o of the number of corner repetitions (between original and test images) N_r to the number of corners in the original image N_o. The ratio R_t of N_r to the number of detected corners in the test image N_t is also used [3]. The potential problem with R_o and R_t is that they are highly sensitive to false corners. If a corner detector detects all points in the original image as corners, then $N_t = N_r$ and R_t will be 100%. In contrast, if a corner detector detects all points in the test image as corners, then $N_o = N_r$ and R_o will be 100%. For fair and unambiguous comparisons we propose the average of R_o and R_t as the *average repeatability* R_{avg} defined as

$$R_{avg} = \frac{R_o + R_t}{2} = \frac{N_r}{2}\left(\frac{1}{N_o} + \frac{1}{N_t}\right). \quad (13)$$

Localization error is defined as the amount of pixel deviation of a repeated corner. It is measured using the *root-mean-square-error* (RMSE) of corresponding positions of repeated corners in the original image and in the transformed image [3]:

$$L_e = \sqrt{\frac{1}{N_r}\sum_{i=1}^{N_r}(x_{oi} - x_{ti})^2 + (y_{oi} - y_{ti})^2}, \quad (14)$$

where (x_{oi}, y_{oi}) and (x_{ti}, y_{ti}) are the positions of i-th repeated corner in original and transformed images respectively.

5. PERFORMANCE COMPARISON

We used MATLAB 7.0 to implement the proposed ARCSS detector and compare it with existing CSS and ECSS detectors. We had total 23 different original 512×512 gray-scale images [10] and total 6394 test images of six categories. We rotated each original

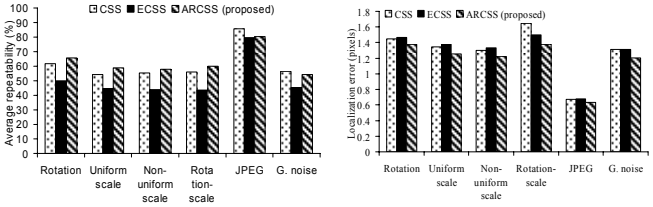


Fig. 2: Overall performance comparisons of corner detectors

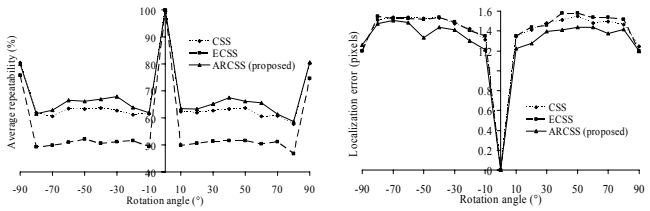


Fig. 3: Repeatability and localization error under rotation

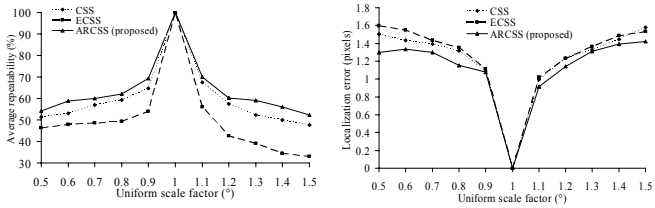


Fig. 4: Repeatability and localization error under uniform scale

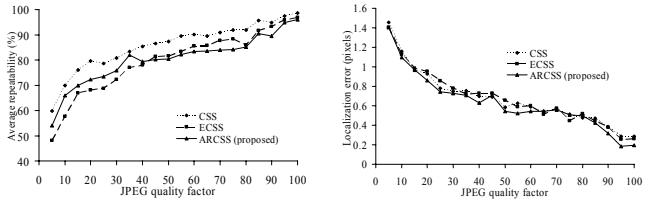


Fig. 5: Repeatability and localization error under JPEG

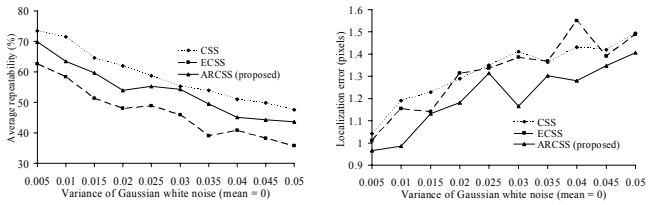


Fig. 6: Repeatability and localization error under Gaussian noise

image at 18 different angles from –90° to +90° at 10° apart, excluding 0°. We resized each original image using uniform scale factors from 0.5 to 1.5 at 0.1 apart, excluding 1.0. We also resized each original image using non-uniform scale factors from 0.7 to 1.3 along the *x*-axis and from 0.5 to 1.5 along the *y*-axis at 0.1 apart, excluding the cases when scaling factors were same along both the axis. We further used combined transformations: rotation from –30° to +30° at 10° apart, excluding 0°, followed by uniform or non-uniform scale factors from 0.8 to 1.2 at 0.1 apart. We compressed each original image using JPEG at 20 quality factors from 5 to 100 at 5 apart. We also added zero mean white Gaussian noise at 10 variances from 0.005 to 0.05 at 0.005 apart. Therefore, we had total 414 rotated, 230 uniform scaled, 1610 non-uniform scaled, 3450 rotated and scaled transformed images. We also had 460 JPEG compressed and 230 Gaussian noised images. For calculating average repeatability R_{avg}, we transformed the original corners using the original transformation parameters, if necessary, prior to finding their repetitions in the test corner set and an RMSE value of maximum 3 pixels was allowed to find a repetition.

Fig. 1 shows the corners detected by the existing CSS and ECSS and the proposed ARCSS detectors in maple-leaf image. While the CSS detector missed some true corners and the ECSS detector introduced some weak corners, the ARCSS detector missed less true corners and introduced less weak corners. Fig. 2 shows average repeatability and localization error under geometric transformations, JPEG compression, and Gaussian noising. Figs. 3 to 6 show the details under different attacks. It was observed that the ARCSS detector performed the best among the three in terms of both average repeatability and localization in geometric attacks. This is because during corner detection while the ARCSS detector uses the affine-length parameterization, the CSS and ECSS detectors use the arc-length parameterization. Nevertheless, it showed lower average repeatability than the CSS detector but better localization than both the CSS and ECSS detectors in JPEG compression and noising. The reason is the CSS detector obtains corners at a higher scale in CSS than the other two and, therefore, gains higher noise immunity to signal processing attacks. But as a consequence it misses some true corners.

6. CONCLUSIONS

Unlike existing CSS and ECSS corner detectors, the proposed ARCSS corner detector parameterizes planar curves with the affine-length. By mathematical derivation we show that the affine-length parameterized curvature involves up to second-order derivatives like the arc-length parameterized curvature. Therefore, the ARCSS detector extracts more robust corners with the same computational cost as the existing detectors. It outperforms the existing detectors in terms of both average repeatability and localization under affine transformations. It also possesses better localization than the existing detectors in lossy compression and noising. So, it could be exploited in many computer vision fields.

7. REFERENCES

[1] Awrangjeb, M., M. Murshed, and G. Lu, "Global geometric distortion correction in images," in *Proc. IEEE Int. Work. on Mult. Sig. Process.*, pp. 435–440, Victoria, BC, Canada, Oct. 2006.

[2] Harris, C. J. and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vis. Conf.*, 147–151, 1988.

[3] Mikolajczyk, K. and C. Schmid, "Scale & affine invariant interest point detectors," *J of Comp. Vis.*, 60(1), 63–86, Oct. 2004.

[4] Rattarangsi, A. and R. T. Chin, "Scale-based detection of corners of planar curves," *IEEE Trans. on Pat. Anal. and Mach. Intel.*, 14(4), 430–49, Apr. 1992.

[5] Mokhtarian, F. and R. Suomela, "Robust image corner detection through curvature scale space," *IEEE Trans. on Pat. Anal. and Mach. Intel.*, 20(12), 1376–1381, Dec. 1998.

[6] Mokhtarian, F. and F. Mohanna, "Enhancing the curvature scale space corner detector," in *Proc. Scandinavian Conf. on Image Anal.*, 145–52, 2001.

[7] Mokhtarian, F. and F. Mohanna, "Performance evaluation of corner detectors using consistency and accuracy measures," *Comp. Vis. and Image Understanding*, 102(1), 81–94, Apr. 2006.

[8] Mokhtarian, F. and S. Abbasi, "Affine curvature scale space with affine length parametrisation," *Pat. Anal. and Appl.*, 4(1), 1–8, 2001.

[9] Canny, J., "A Computational Approach to Edge Detection," *IEEE Trans. on Pat. Anal. and Mach. Intel.*, 8(6), 679–698, 1986.

[10] Acclaim images, http://www.acclaimimages.com.