A ROBUST CSS CORNER DETECTOR BASED ON THE TURNING ANGLE CURVATURE OF IMAGE GRADIENTS

Cihan Topal[1], Kemal Özkan[2], Burak Benligiray[1], Cuneyt Akinlar[1]

[1]Anadolu University
Department of Computer Engineering
Eskisehir, Turkey

[2]Eskisehir Osmangazi University
Department of Computer Engineering
Eskisehir, Turkey

ABSTRACT

In this study, we present a new contour-based corner detection method based on the turning angle curvature computed from the contour gradients of the image. In general, curvature is computed with the pixel locations of the extracted image contours. In most contour extraction methods, the image gradient information is already computed. The proposed algorithm makes use of this available information to compute the curvature function and takes local extremums as potential corner candidates. Afterwards, the candidates are validated by a novel validation algorithm which tries to approximate the local geometric structure of the contour with an iterative least squares estimation algorithm. Thus, we not only eliminate the false detected corners; but also estimate the corner strength precisely in terms of degrees. The experiments show that the detected corners with gradient-based turning angle curvature are more durable to affine transformations according to the ACU and LE criterions.

Index Terms— Corner detection, curvature scale space (CSS), turning angle curvature, corner validation

1. INTRODUCTION

Corner detection has critical importance in computer vision research due to the substantial need for robust keypoint detection algorithms. Among many others, object recognition, motion tracking, pose estimation and image registration applications are active research problems that utilize corner detection as a critical step. For this reason, corner detection methodologies constitute a crowded set in the literature and can be classified into three major types according to detection strategies. First, intensity based corner detectors [1-7] try to locate the corners by directly using the intensity values of the pixels on the image. Second, contour based methods [8-16], first extract the planar curves from the image, and then process these planar curves to find the sharp changes, which correspond to corners. Third and last, model or template based methods [17-20] aim to locate the corners by fitting the image patterns onto a mathematical model.

There are several pros and cons for each of these strategies. Intensity based methods are very sensitive to fine details on the image; nevertheless, they are very sensitive to noise for the same reason. Contour based methods response to noise in a better way but efficient extraction of the planar curves from the real images is usually a tough problem. In contour based methods, lying on an edge segment becomes a prerequisite for an image point to be a corner. This situation provides the detection of reliable corners, which are also coherent to the human visual system.

In general, the contour based studies compute the curvature function with the coordinates of contour pixels with the formula in Eq. 1 [8-16]. In this study, we propose a new contour based corner detection algorithm based on the turning angle curvature of the contours. Another contribution of the study is the employment of the novel edge segment detection algorithm, Edge Drawing [21], to extract the planar curves efficiently. Furthermore, we propose a novel geometrical false corner removal method that is also able to estimate the strength of the corner precisely in terms of degrees.

2. PROPOSED METHOD

In geometry, curvature can be defined as the amount by which a curve deviates from being straight and the most common method to compute the curvature is:

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

where *x(t)* and *y(t)* denote the *x* and *y* coordinates of the curve points and the dots denote the first and second derivations with respect to time.

Another way to compute the curvature is by taking the difference between tangents of consecutive points, called the turning angle curvature [22]. Computing the turning angle curvature is usually easier because a typical contour detector computes the gradient magnitudes at each pixel of an image during edge detection [21, 23]. Therefore, it becomes possible obtain the tangents of the contour points in an effortless manner by using the horizontal (G_x) and vertical (G_y) gradient responses as shown in Fig. 1. Thus angle values (α) for every pixel location on the image are obtained easily and they correspond to the angle of the gradient magnitude vector with respect to the *x* axis.

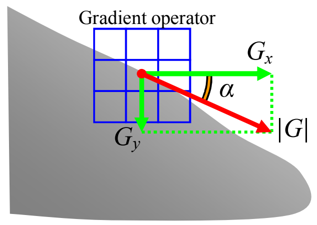


Figure 1. Tangent estimation using the gradient responses.

To obtain the curvature response of the extracted angle values, we simply need to compute the relative angle variations for consecutive gradient vectors. If the tangent estimation for consecutive contour elements is considered to be a function of time denoted by *α(t)*, then the angle changes along the contour with respect to time. The illustration in Fig. 2 shows two consecutive gradient vectors (G_i and G_{i+1}) and the relative angulation (θ) in between them.

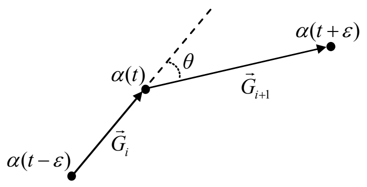


Figure 2. Geometric representation of turning angle curvature.

The θ in Fig. 2 denotes the local variation of the curve from being flat for current location t. Therefore, the turning angle curvature as a function of t can be computed for two consecutive gradient vectors of different lengths as follows [22]:

$$\kappa_\alpha(t) = \frac{\alpha(t) - \alpha(t-\varepsilon)}{\|t-(t-\varepsilon)\|} \tag{2}$$

Since the ε values of consecutive elements (i.e. pixels) of the contour are fixed and equal to 1 in our case; ε becomes irrelevant. Thus the angular curvature can be obtained by a simple derivation of *α(t)* function with respect to time;

$$\kappa_\alpha(t) = \alpha(t) - \alpha(t-1) \tag{3}$$

The curvature shown in Eq. 3 denotes the turning angle curvature [22]. To obtain the curvature at a specific Gaussian scale σ, i.e. $\kappa_\alpha(t,\sigma)$, we need to convolve the κ_α with a Gaussian kernel. To do this, we need to re-parameterize the consecutive angle values with a simple normalization process. In this normalization, we compute cumulative gradient series (α_c) of $\alpha(t)$ by adding up the acute angle values between consecutive gradient directions;

$$\alpha_c(t) = \begin{cases} \alpha_c(t-1) + \theta, & \theta < 90 \\ \alpha_c(t-1) + 180 - \theta, & \theta \geq 90 \end{cases} \tag{4}$$

where $\theta = \alpha(t) - \alpha(t-1)$. After obtaining α_c, we convolve it with $\dot{g}(t,\sigma)$, which is the derivative of the Gaussian kernel in scale σ;

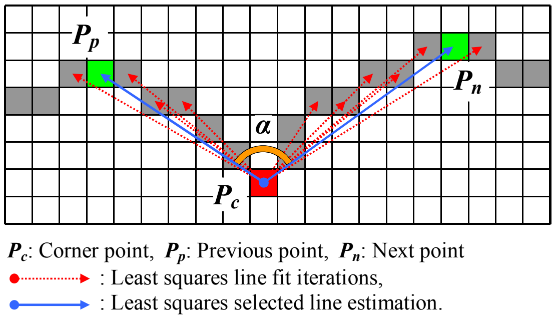
$$\kappa_\alpha(t,\sigma) = \alpha_c(t) * \dot{g}(t,\sigma) \tag{5}$$

Once the turning angle curvature is computed, we detect all corner candidates as the local maximas of the curvature which are bigger than a certain global threshold. Finally, we validate the detected corner candidates with the method explained in sec. 2.1.

2.1. Corner Validation Method

In real images, the local variations on contours are very common and curvature responses of corners and non-corners can be very difficult to distinguish. Moreover, there is no mathematical definition of corners that even barely corresponds to the human perception. For this reason, corner detection algorithms first aim to detect all appropriate corner candidates, and then try to eliminate the false ones with validation methods. Most of the corner validation algorithms are based on the curvature function; however, curvature function can easily be corrupt due to noise and Gaussian scale convolution process. Mokhtarian et al. [9] try to find reliable corners by tracking them in an intra-scale manner. In [11], He and Yung try to solve this problem by applying adaptive local threshold in a dynamic region of support (ROS), which is determined with the curvature function. In [16], the authors use a two-step refinement process, which is mainly based on the curvature information. The common problem for these methods is; they only use the curvature information for validation in an isolated manner regardless of what the spatial structure is.

In this paper, we propose a very simple and efficient corner validation method that enables the precise estimation of corner strength in terms of angles. Instead of using the curvature information again, the proposed method validates the corners with respect to the geometric structure of the contour. After we detect all corner candidates, we try to approximate the angle of each corner by fitting two line segments with an iterative least square estimation procedure. In Fig. 3, the validation procedure is illustrated. We determine a least squares error threshold and try to fit lines to the neighbour pixels of the corner candidate until the fitting error exceeds the threshold. Thus, all neighbour pixels are taken into account and a 2-dimensional geometrical region of support is established.



P_c: Corner point, P_p: Previous point, P_n: Next point
: Least squares line fit iterations,
: Least squares selected line estimation.

Figure 3. Illustration of corner validation algorithm.

Once we determine the previous and next points (P_p and P_n), we obtain two vectors V_1 and V_2 to compute the angle α:

$$V_1 = P_c - P_p, \quad V_2 = P_c - P_n \tag{6}$$

$$V_1 \cdot V_2 = \|V_1\| . \|V_2\| . \cos\alpha \tag{7}$$

$$\alpha = \tan^{-1}\left(\frac{V_1 \cdot V_2}{\|V_1\| . \|V_2\|}\right) \tag{8}$$

Finally, the true corners become more distinguishable among the candidates and the ones having an angle wider than T_{angle} are eliminated. Thus a pure geometric validation is achieved in order to keep the corners which address the human perception in a better way. The steps of the proposed algorithm can now be summarized as follows:

1. Detect the edge segments with Edge Drawing (ED) [21],
2. Compute the turning angle curvature with image gradients at a single scale σ,
3. Detect the corner candidates with respect to the curvature function's local maximas that are greater than a certain threshold,
4. Eliminate the corners which fail the validation test.

3. EXPERIMENTAL RESULTS

In this section we measure the performance of the proposed algorithm by comparing its results with two well-known intensity-based corner detection algorithms; Harris [2] and Shi & Tomasi [3], and four contour-based algorithms; CSS [9], ECSS [10], adaptive threshold CSS (ATCSS) [11] and chord-to-point distance accumulation technique (CPDA) [15]. In the experiments, we all use the original implementations of the algorithms acquired from the authors (for Harris and Shi-Tomasi, OpenCV implementations are substituted). We also assess the efficiency of employed detectors with two evaluation metrics, i.e. accuracy (ACU) (Eq. 9) and localization error (LE) [24]. We substitute a 4-pixel maximum displacement (D_{max}) in ACU calculations for all detectors.

$$ACU = 100 \times (N_{true} / N_{detected} + N_{true} / N_{real}) / 2 \qquad (9)$$

To compute the accuracy, having test images with the ground truths is necessary. Since "what is a corner?" is an ill posed question for real images, synthetic images are employed in most studies. But also, using only synthetic images lacks from scientific methodology. For this reason, we prepared a simple dataset with real images such that they have *mostly* definite corners. We make the dataset available online in our webpage also with the coordinates of ground truths, and image results of proposed method on the prepared dataset [25].

Eventually, we perform our experiments with 15 images (7 synthetic and 9 real) images. Each test image is subjected to rotation (-80° to 80°, 16 steps), uniform scaling (0.5 to 2, 15 steps), x scaling (0.5 to 2, 15 steps), y scaling (0.5 to 2, 15), and Gaussian noise addition (σ = 0 to 0.045, 5 steps) experiments. Thus, each algorithm runs 82 times for each test image, hence, total 16 x 82 = 1312 experiments are performed. We set the algorithms to their default parameters indicated in the publications (see Table 1). In Fig. 4, we present the experiment results in terms of ACU and LE. We also provide the overall results for all experiments in Fig. 5. Furthermore, we also provide an online demo in our webpage that interested users can test the proposed method with any image [25].

Table 1: Employed algorithms and their parameter settings.

Algorithm	Parameters
Harris (OpenCV)	block size: 3, aperture size: 3, k: 0.04
Shi-Tomasi (OpenCV)	block size: 3, aperture size: 3, k: 0.04
CSS (MATLAB)	sigma: 3, high threshold: 0.35, low threshold: 0, gap size: 1
ECSS (MATLAB)	sigma: 3, high threshold: 0.35, low threshold: 0, gap size: 1
ATCSS (MATLAB)	C: 1.5, T_{angle}: 162, sigma: 3, end points: true, gap size: 1
CPDA (MATLAB)	high threshold: 0.7, low threshold: 0.2, gap size: 1, end points: true
Proposed (C/C++)	sigma: 2, curvature threshold: 4, angle threshold: 160, line fit thr.: 0.25

In terms of running time, the proposed algorithm takes a mere 2.56 msec for 256x256 and 10.47 msec for 512x512 images on a 2 GHz Intel CPU on the average. The running times include both the edge detection and the corner detection by the proposed algorithm. Since we already have the image gradients from the edge/contour extraction step, a very slight computation burden remains for the corner detection scheme. Due to the discrepancies of platforms which the algorithms are implemented on (see Table 1); we cannot make a fair comparison in terms of running time. Nevertheless, the timing results for the proposed algorithm are very competitive compared to the ones in the literature to the best of our knowledge.

4. CONCLUSION

In this study, we proposed a novel approach for CSS corner detection paradigm, which produces promising results and runs in real-time. Contrary to the majority of the related studies, we try to compute the curvature from the image gradient information instead of the pixel locations. Since the gradient information is already available from the edge/contour extraction scheme, the algorithm runs very fast.

From the experiments, we see that the proposed curvature computation method is more durable against affine transformations, i.e. rotation, uniform and non-uniform scaling, etc (Fig. 4). The only experiment that our method falls behind is the Gaussian noise addition (see Fig. 4.e). We explain this situation as a drawback of our parameter selection. To provide better corner localization and faster computation; we prefer to convolve the gradient series with lower scales of Gaussian functions in narrower apertures, (i.e. 1x7 kernel, σ = 2). Therefore, our method suffers from ACU results as the noise is increasing due to the insufficient noise cancellation; nevertheless, it is still doing fine in terms of LE criterion (Fig. 4.k). Another drawback of our algorithm is; it does not check for the T-junctions with its current version, however, we are planning to improve it by incorporating an additional scheme.

Besides using the gradient information for curvature computation, our method also differentiates itself from the rest by employing the new edge detection method, i.e. ED, algorithm for contour extraction. With high quality contiguous edges and fast execution, ED eases the successful completion of the task.

As a final contribution, we proposed a novel corner validation algorithm which aims to approximate the spatial geometrical structure of the edge contour. In other words, we do not use the curvature scale space information to validate the contours.

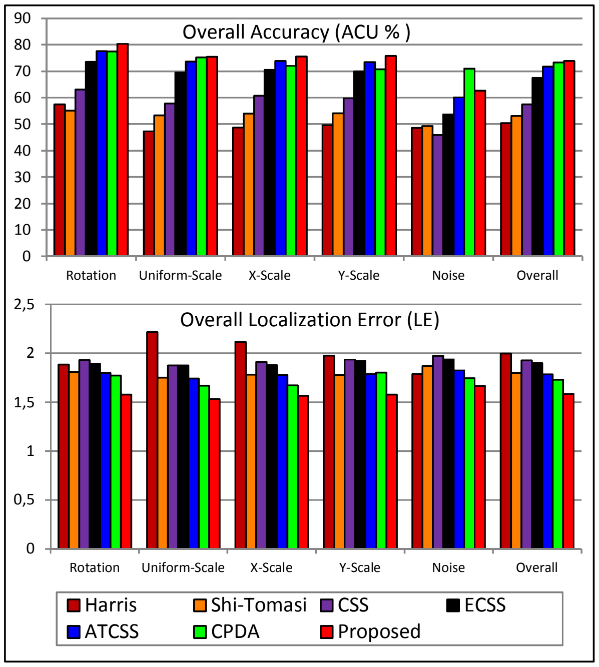


Figure 5. Overall results for ACU and LE criterions.

ACKNOWLEDGEMENTS

We would like to thank Xiao Chen He and Mohammed Awrangjeb for being generous on making their work available for comparison.

This work is supported by TÜBİTAK (grant no. 111E053) and Anadolu University BAP (grant no. 1207F113).

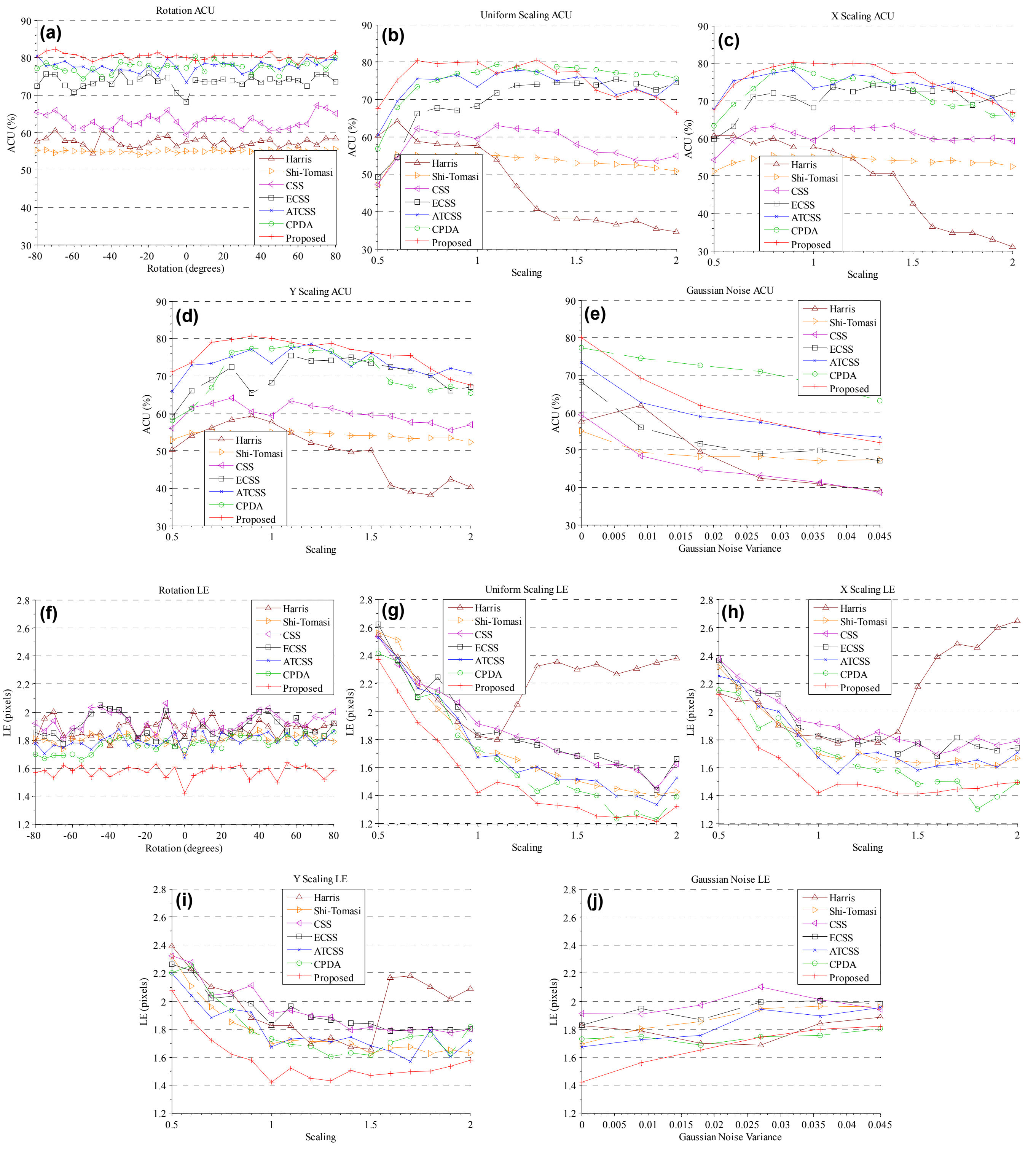


Figure 4. Detailed results for the experiments. a - e) ACU plots; f - j) LE plots for rotation, uniform scaling, x scaling, y scaling and Gaussian noise, respectively.

12. REFERENCES

[1] L. Kitchen and A. Rosenfeld, “Gray Level Corner Detection,” Pattern Recognition Letters, pp. 95-102, 1982.

[2] C. J. Harris and M. Stephens, “A Combined Corner and Edge Detector,” in Proc. Alvey Vis. Conf., pp. 147–151, 1988.

[3] J. Shi, C. Tomasi, "Good features to track," in Proc. 9th IEEE Conf. on Computer Vision and Pattern Recognition, 1994.

[4] S. M. Smith and J. M. Brady, “SUSAN - A New Approach To Low Level Image Processing,” Int. J. Comp. Vis., vol. 23, no. 1, pp. 45–78, 1997.

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

[6] E. Rosten and T. Drummond, "Machine learning for high-speed corner detection," in Proc. European Conference on Computer Vision, pp.430-443, 2006.

[7] X. Gao, F. Sattar, and R. Venkateswarlu, "Multiscale Corner Detection of Gray Level Images Based on Log-Gabor Wavelet Transform," in Proc. IEEE ICASSP, pp.1253-1256, 2007.

[8] A. Rattarangsi and R. T. Chin, “Scale-based detection of corners of planar Curves”, IEEE Trans on Pattern Analysis and Machine Intelligence, 14(4), 430-449, 1992.

[9] F. Mokhtarian and R. Suomela. “Robust image corner detection through curvature scale space”. IEEE Trans. Pattern Analysis and Machine Intelligence, 20(12): 1376-1381, 1998.

[10] F. Mokhtarian and F. Mohanna, “Enhancing the curvature scale space corner detector”, Proc. Scandinavian Conf. on Image Analysis, pp. 145-152, Bergen, Norway, 2001.

[11] X.C. He, N.H.C. Yung, “Curvature scale space corner detector with adaptive threshold and dynamic region of support,” in Proc. 17th Int'l. Conf. Pattern Recognition (ICPR). pp. 791–794, 2004.

[12] X.C. He and N.H.C. Yung, “Corner detector based on global and local curvature properties,” Optical Eng. 47(5), May 2008.

[13] B. Zhong and W. Liao, “Direct curvature scale space: Theory and corner detection,” IEEE Trans. Pattern Anal. Machine Intell., vol. 29, no. 3, pp. 508–512, Mar. 2007.

[14] M. Awrangjeb, G. Lu, and M. Murshed, “An affine resilient curvature scale-space corner detector,” in Proc. IEEE Int. Conf. Acoustics, Speech, and Signal Processing, (ICASSP), pp. 1233–1236, 2007.

[15] M. Awrangjeb and G. Lu, “Robust Image Corner Detection Based on the Chord-to-Point Distance Accumulation Technique,” IEEE Trans. Multimedia, vol.10, no.6, Oct 2008.

[16] X.C. He, N.H.C. Yung, Corner detector based on global and local curvature properties, Optical Engineering vol. 47, issue 5, May 2008.

[17] K. Rangarajan, M. Shah, and D. V. Brackle, “Optimal corner detector,” Comp. Vis., Graph., Image Process., vol. 48, pp. 230–245, 1989.

[18] A. Singh and M. Shneier, “Gray level corner detection a generalization and a robust real time implementation,” Comp. Vis., Graph., Image Process., vol. 51, pp. 54–69, 1990.

[19] R. Laganiere and T. Vincent, “Wedge-based corner model for widelyseparated view matching,” in Proc. Int. Conf. Pattern Recognit., Aug. 2002, vol. 3, pp. 672–675.

[20] E. D. Sinzinger, “A model-based approach to junction detection using radial energy,” Pattern Recognit., vol. 41, no. 2, pp. 494–505, Feb. 2008.

[21] C. Topal and C. Akinlar, “Edge Drawing: A Combined Real-Time Edge and Segment Detector,” Journal of Visual Communication and Image Representation, 23(6), 862-872, Aug. 2012.

[22] Anne Driemel, Multiscale Curvature Matching for Smooth Polylines, Diploma Thesis, Freie Universität Berlin, 2009.

[23] C. Topal, C. Akinlar, Y. Genc, Edge Drawing: A Heuristic Approach to Robust Real-time Edge Detection, in Proc. ICPR, pp. 2424–2427, 2010.

[24] F. Mohanna and F. Mokhtarian, Performance Evaluation of Corner Detection Algorithms under Similarity and Affine Transforms, BMVC, pp. 353–362, 2005.

[25] http://ceng.anadolu.edu.tr/cv/CornerDetection, Accessed on Nov. 30th 2012.