An Improved Curvature Scale-Space Corner Detector and a Robust Corner Matching Approach for Transformed Image Identification

Mohammad Awrangjeb, *Member, IEEE*, and Guojun Lu, *Senior Member, IEEE*

***Abstract*—There are many applications, such as image copyright protection, where transformed images of a given test image need to be identified. The solution to this identification problem consists of two main stages. In stage one, certain representative features, such as corners, are detected in all images. In stage two, the representative features of the test image and the stored images are compared to identify the transformed images for the test image. *Curvature scale-space* (CSS) corner detectors look for curvature maxima or inflection points on planar curves. However, the arc-length used to parameterize the planar curves by the existing CSS detectors is not invariant to geometric transformations such as scaling. As a solution to stage one, this paper presents an improved CSS corner detector using the affine-length parameterization which is relatively invariant to affine transformations. We then present an improved corner matching technique as a solution to the stage two. Finally, we apply the proposed corner detection and matching techniques to identify the transformed images for a given image and report the promising results.**

***Index Terms*—Affine-length, corner detection, corner matching, curvature scale-space (CSS), transformed image identification (TII).**

I. Introduction

In many applications, such as image copyright protection [1], one common problem is to identify images which may have undergone unknown transformations. We can define this common problem as the *transformed image identification* (TII) where the goal is to identify geometric transformed and signal processed images for a given image. So the TII is different from conventional *content-based image retrieval* (CBIR) [2], where all images having the same or similar semantic feature, e.g., flower, are considered relevant to each other. The TII consists of two main stages: *feature detection* and *feature matching*. In the *feature detection* stage, a set of representative features, e.g., corners, are detected and represented for all images. In the *feature matching* stage, the representative features of the test image and the stored images are compared to identify the transformed images for the test image. We will focus on both of the stages of the TII in this paper. In addition, we will concentrate on the performance evaluation metrics.

Manuscript received October 30, 2007; revised August 18, 2008. Current version published November 12, 2008. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Ercan E. Kuruoglu.

The authors are with the Gippsland School of Information Technology, Monash University, Churchill, Victoria 3842, Australia (e-mail: mohammad.awrangjeb@infotech.monash.edu.au; guojun.lu@infotech.monash.edu.au).

Digital Object Identifier 10.1109/TIP.2008.2006441

A. Feature Detection

The most challenging problem in current research on image copyright protection is to devise a system that would verify the copyright information in the test image even when the image has undergone unknown geometric transformations. In image copyright protection, some pixels in the original image are used either to calculate the signature [1] or to embed the watermark [3]. However, geometric transformations change the location of those pixels. Consequently, the copyright verifier may fail to track the copyright information in the transformed (test) image. In such cases, if the locations of those pixels are defined with respect to some salient features, e.g., corners which are robust to geometric transformations, the verifier will be able to locate those pixels. Such salient features can be denoted as the reference points to those pixels.

Different types of image features, e.g., corners [3], wavelet maxima points [4], have been extensively used in the last decade in order to obtain robust watermarking-based copyright protection systems. Corners are visually distinguishable and well localized compared to other image features. The wavelet maxima-based watermarking system in [4] could survive only on small geometric distortions, e.g., 2° rotation. The reason is that the wavelet features are highly sensitive to geometric operations. In contrast, the corner-based watermarking system in [3] was shown robust to large geometric distortions.

The feature-points can also be used to devise blind signature-based copyright protection system [1]. A corner matching technique can be used to identify the transformed images for a given test image. The transformation matrix can also be estimated to undo the transformation if necessary. In the identified transformed images the signature area can be located using the reference points (corners).

A large number of corner and interest-point detectors have been proposed in the literature [5]–[10]. They can be broadly classified into three groups: intensity-based [5]–[7], contour-based [8]–[10], and model-based [11], [12] methods. Intensity-based methods estimate a measure which is intended to indicate the presence of an interest-point directly from the image pixel values. Contour-based methods first obtain planar curves using some edge detector and then search for the curvature maxima along those curves. Model or template-based methods find corners by fitting the image signal into a predefined model. Corner detectors can also be divided into two types: single-scale detectors [5] and multiscale detectors [6]–[10]. Single-scale detectors work well if the image has similar size features, but ineffective otherwise; because either fine or coarse scale feature is poorly detected, but images may contain both kinds of features. To improve the effectiveness of corner detection, multiscale corner detectors have been proposed. They are based on the classical scale-space theory [13]–[16].

In this paper, we will focus on the contour-based multiscale corner detectors which are commonly known as *curvature scale-space* (CSS) detectors. One of the main reasons why we choose the contour-based methods is that along with the detected corners, the contour-based methods make other information, like the curvature zero-crossing points and edges, available for later applications. A corner matching technique based on the available information from the corner detector can be used to identify the transformed images for a given test image. The transformation matrix can also be estimated to undo the transformation if necessary.

The CSS corner detectors look for curvature maxima or inflection points on planar curves. The original CSS detectors [8], [9] and the enhanced version [10] parameterize curves using the arc-length which is not invariant to geometric transformations. This paper presents an improved CSS corner detector which is affine-resilient (ARCSS detector). A preliminary version of the ARCSS corner detector was published in [17]. Here, we will detail each step of this corner detector (see Section II-B) and present the detailed experimental results (see Section V-B1). The proposed corner detector parameterizes curves with the affine-length which is *relatively invariant* to affine transformations. A function f is called *relatively invariant* to a transformation group G if whenever f is transformed to F by a transformation g in G, we obtain $F = z.f$, where z is a function of g alone. If $z \equiv 1$ for all g in G, f is called *absolutely invariant* [15]. For more details and rigorous discussions, see [18].

Note that the affine-length has been used for affine invariant shape recognitions in the literature [19], [20]. The corner detector and matching technique proposed in this paper are different from them in two aspects. First, existing techniques match shapes using the so called CSS image (see [19] and [20]) and do not explicitly use corner positions for matching. However, we explicitly obtain corner positions on planar curves and use them for matching. Second, their formula of curvature calculation uses up to third order derivatives which has the undesirable effects, since precise approximation of higher order derivatives is a difficult task and often causes instability and errors [19]. In this paper, we will simplify the formula so that only up to second-order derivatives are used (see Section II-B1).

B. Feature Matching

The feature (corner) matching technique is normally computationally intensive and difficult due to following reasons: first, additional corners may be detected in the test image due to noise or clutter; second, desirable corners may be missed in the test image due to noise or occlusion; and, finally, the corner strength represented as some numerical values may be changed by transformations [21]. Although a significant number of solutions have been proposed to this problem, none is general and reliable enough to cope with all possible cases [22].

Corner matching techniques can be broadly classified into two groups. In the first group, each corner is associated with its local neighbourhood and the matching procedure involves neighbourhood matching [6], [23]. These techniques are robust but computationally expensive and require large storage for descriptors. In the second group, each corner is associated with information like its curvature but do not use neighbourhood intensity values for matching [22], [24]–[29]. These techniques are computationally less expensive and require less storage for descriptors.

This paper presents a corner matching technique which belongs to the second category and is robust to geometric transformations, noising, and lossy compression. We first find the candidate corner matches between two images using absolute curvature values at corner points and affine-lengths between corners on the same curve. For each set of three candidate matches, if they are noncollinear on each image and the ratio of areas of corresponding triangles in both of the images is within a specific range, we estimate the transformation matrix between these triangles. Then we transform all corners in the database image using the estimated transformation matrix and match with the test corner set. In the rest of the paper, we will call the proposed matching technique as *affine-length and triangular area* (ALTA) matching technique. Note that a preliminary version of the ALTA matching technique with some initial experimental results was published in [30]. Here we improve it by proposing some speed up steps (see Section III-B1), discuss detailed matching results on a large database (see Section V-B2), and present the TII performance (see Section V-B3).

C. Performance Measurements

The existing CSS detectors [8]–[10], [31] did not use proper evaluation metrics for performance evaluation (see Section IV-A). They were evaluated based on the human judgement which is hard to establish. Moreover, existing work on matching techniques [22], [24]–[29] did not evaluate and compare their performance using large databases. For the evaluation of corner detection performance, we propose a modified corner matching metric (see *average repeatability* in Section IV-C1). We carry out a thorough robustness study on large databases considering a wide range of geometric transformations. We evaluate the corner detection performance using *average repeatability* and *localization error*. Corner matching performance is evaluated using *corner correspondence* and compared with the performance of the existing most promising Delaunay triangulations based matching technique [22]. Finally, the combined performance of the proposed ARCSS corner detector and the ALTA matching technique is evaluated when they are applied to TII. Their combined performance which we call the *identification performance* in rest of the paper is evaluated using the *precision-recall graph* and compared with the performance of the existing *grayscale histogram* (GSH) matching technique [2].

Note that corners and edges approximate multiple object shapes in an image, and, therefore, our corner matching technique using the affine-length between corners on the same curve is somewhat like a global feature matching technique. Many existing CBIR techniques [2] may not be applicable to TII, since the goal of the CBIR is different from that of TII. The GSH matching, which is a global feature (intensity) based CBIR technique, is highly robust to geometric transformations

[2] and, therefore, is chosen to be compared with the proposed corner matching technique. Moreover, the shape-based recognition techniques [19], [20] may not be applicable to TII, since they require automatic, efficient, and reliable image segmentation techniques to extract closed contours of multiple objects in the same image. However, it may be impossible since objects in images may be strongly noised, occluded, or cluttered.

D. Paper Organization

The rest of the paper is organized as follows. The existing CSS corner detectors along with the proposed improved corner detector are presented in Section II. Section III presents the existing corner matching techniques and the proposed ALTA matching technique. Section IV presents the performance evaluation metrics. Section V presents the performance study. Finally, Section VI concludes the paper.

II. Curvature Scale-Space Corner Detectors

The CSS corner detectors, in general, extract planar curves (open and close contours) from the image using the Canny edge detector [32] and parameterize each curve using the arc-length. They then smooth the parameterized curve and calculate the absolute curvature on each point of the smoothed curve at either all scales [8], one [9], or more specific scales [10]. Thereafter, 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 selected [8]. If corners are detected at some specific scales, they are tracked down to the finest scale to improve localization [9], [10].

Since the arc-length of a curve is not preserved under geometric transformations like scaling, the existing CSS detectors [8]–[10], [31] are vulnerable to geometric attacks. The proposed ARCSS corner detector uses the affine-length parameterization which is relatively invariant to affine transformations.

A. Existing Corner Detectors

The first CSS corner detector by Rattarangsi and Chin [8] analyzed the scale-space of the isolated single and double corners. It had two main drawbacks: high computational complexity because of involving a tree parsing technique and false corner detection due to quantization noise.

The next CSS corner detector by Mokhtarian and Suomela [9] detected corners at a high scale in the CSS using a single curvature-threshold and tracked them through multiple lower scales to the lowest scale in order to improve localization. When corners were detected in a very high scale, this method showed high robustness to noise but many true corners were smoothed away. On the other hand, if corners were detected in a low scale it detected many weak corners. Moreover, its performance was limited due to the use of the single threshold with the single smoothing scale, since the curvature value of a point may change when the image is geometrically transformed.

To alleviate these problems, Mokhtarian and Mohanna [10] later proposed the enhanced CSS corner detector where corners were detected in multiple (three) scales with different thresholds. In the remaining of this paper, we refer their former method [9] as the CSS detector and later one [10] as the ECSS detector. The ECSS detector, though performed better as they reported in [31], was found less robust than the CSS detector in our robustness tests (see Section V-B1). The corner detector in [33] did not use the scale-space, though the title seemed to be. In fact, it detected corners at a single low scale and did not track to the finest scale. Consequently, it might detect weak corners and suffer from localization problem.

B. Proposed Improved Corner Detector

The arc-length was used to parameterize the planar curves by both the existing CSS and ECSS detectors [9], [10]. However, the arc-length is not invariant to affine transformations. In order to overcome this problem, the arc-length was replaced by the affine-length which is relatively invariant to affine transformations [19].

The main difference between the arc-length and the affine-length parameterizations of planar curves is in sampling. The arc-length parameterization used by both the existing CSS and ECSS detectors [9], [10] selects all the points on a curve when the sample-interval is either 1 (for vertical or horizontal neighbor points) or $\sqrt{2}$ (for diagonal neighbor points) pixels. However, as the arc-length of a curve is not invariant to geometric transformations, a substantially different set of curve points is selected from the transformed curve when the arc-length parameterization is used. In contrast, the affine-length parameterization selects only the important points of a curve. It selects more points on the curve segments where the curve makes significant direction changes (e.g., on and near corners) than on the straight-line like curve segments. Since the affine-length of a curve is relatively invariant to geometric transformations, it is expected that the affine-length parameterized curve is more stable and, therefore, leads to more stable curvature estimation than the arc-length parameterized curve.

Nevertheless, while the arc-length parameterized curvature involves up to second order derivatives [8]–[10], [31], the formula used by [19], [20] to calculate the affine-length parameterized curvature used third order derivatives. We will show below using simple mathematical derivation that the affine-length parameterized curvature can also be implemented using second order derivatives like the arc-length parameterized curvature. Therefore, the affine-length parameterized curvature can be exploited to extract robust corners with the same computational cost as the arc-length parameterized curvature requires, but with higher accuracy and stability.

1) Affine-Length Parameterized Curvature: In this section, we first present the affine-length parameterized curvature function which uses up to third order derivatives of original curve-point locations. We then simplify it to have only up to second order derivatives. Lower order derivatives help reducing the undesirable effects, e.g., instability and inaccuracy in curvature estimation, associated with higher order derivatives.

For a planar curve $\Gamma(t) = (x(t), y(t))$, where t is an arbitrary parameter, the affine-length τ between two points P_1 and P_2 is [17]

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

where $\dot{x}(t)$ and $\dot{y}(t)$ are first and $\ddot{x}(t)$ and $\ddot{y}(t)$ are second order derivatives with respect to t. We can easily show that the affine-length τ_a of an affine transformed curve is

$$\tau_a = \tau_0 \left[s_x s_y (1 - s_{hx} s_{hy})\right]^{1/3} \tag{2}$$

where τ_0 is the affine-length of the original curve, s_x and s_y are scaling factors and s_{hx} and s_{hy} are shearing factors along x and y directions respectively. The relation in (2) shows that the affine-length of a curve is absolutely invariant to rotation and translation, but relatively invariant to scaling and shearing. Such a relation does not exist for the arc-length of a curve. That means the arc-length is not invariant to scaling and shearing, though it is absolutely invariant to rotation and translation [19]. The Euclidean curvature is defined as [8]–[10], [31]

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

Using (3), the curvature is calculated at all the curve-points. However, when the curve is parameterized using the affine-length, the curvature is calculated at a subset of the curve-points. We select the subset such that the distance between successive points is affine-length λ. We will discuss more about this sampling (subset selection) procedure in Section II-B3b. After sampling, the curvature on the affine-length parameterized curve $\Gamma(\tau) = (x(\tau), y(\tau))$ can be calculated according to (3) using

$$\kappa(\tau) = \frac{\dot{x}(\tau)\ddot{y}(\tau) - \ddot{x}(\tau)\dot{y}(\tau)}{\left[\dot{x}^2(\tau) + \dot{y}^2(\tau)\right]^{3/2}}. \tag{4}$$

We can derive first and second order derivatives of $x(\tau)$ and $y(\tau)$ in (4) as [19] (Appendix I-A)

$$\dot{x}(\tau) = \frac{\dot{x}(t)}{\left[\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right]^{1/3}} \tag{5}$$

$$\ddot{x}(\tau) = \frac{3\ddot{x}(t)\left[\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right] - \dot{x}(t)\left[\dot{x}(t)\tilde{y}(t) - \tilde{x}(t)\dot{y}(t)\right]}{3\left[\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right]^{5/3}} \tag{6}$$

$$\dot{y}(\tau) = \frac{\dot{y}(t)}{\left[\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right]^{1/3}}, \quad \text{and} \tag{7}$$

$$\ddot{y}(\tau) = \frac{3\ddot{y}(t)\left[\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right] - \dot{y}(t)\left[\dot{x}(t)\tilde{y}(t) - \tilde{x}(t)\dot{y}(t)\right]}{3\left[\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right]^{5/3}} \tag{8}$$

where $\tilde{x}(t)$ and $\tilde{y}(t)$ are third order derivatives with respect to t.

We see that the second order derivatives of $x(\tau)$ and $y(\tau)$ involves up to third order derivatives of $x(t)$ and $y(t)$ as evident from (6) and (8) above. However, by using (5)–(8), we can show that

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

Hence, the simplified formula of the affine-length parameterized curvature from (4) using (9) is

$$\kappa(\tau) = \frac{1}{\left[\dot{x}^2(\tau) + \dot{y}^2(\tau)\right]^{3/2}} \tag{10}$$

which uses only first and second order derivatives of $x(t)$ and $y(t)$ as evident from (5) and (7) above. Thus, we avoid problems associated with curvature calculation using higher order derivatives.

2) Curve Smoothing: Curve smoothing prior to curvature measurement reduces the effect of noise [9]. To do that, each coordinate of the affine-length parameterized curve $\Gamma(\tau)$ is convolved with a Gaussian function g_σ of width σ, which is also known as the smoothing-scale or standard deviation of the Gaussian distribution. In the continuous form we have

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

where $*$ denotes the convolution. After smoothing we have the parameterized and smoothed curve $\Gamma(\tau,\sigma) = (\mathrm{X}(\tau,\sigma), \mathrm{Y}(\tau,\sigma))$. The Gaussian convolution property states that differentiation and convolution are commutative. So derivatives of the convolved function can be calculated as

$$\dot{\mathrm{X}}(\tau,\sigma) = x(\tau) * \dot{g}_\sigma \text{ and } \dot{\mathrm{Y}}(\tau,\sigma) = y(\tau) * \dot{g}_\sigma. \tag{12}$$

Since the exact form of $\dot{g}_\sigma$ is known, we can calculate the curvature on the smoothed affine-length parameterized curve according to (10) as

$$\kappa(\tau,\sigma) = \frac{1}{\left[\dot{\mathrm{X}}^2(\tau,\sigma) + \dot{\mathrm{Y}}^2(\tau,\sigma)\right]^{3/2}}. \tag{13}$$

Since the curvature extrema points describe the fundamental features of planar curves, the aim of the proposed ARCSS corner detector is to obtain inflection points using (13) as corners.

3) Corner Detection: The proposed ARCSS corner detector first uses the Canny edge detector [32] to extract edges (planar curves) from grayscale images. Each edge is parameterized using the affine-length and then smoothed using a medium smoothing-scale. On each smoothed curve, candidate corners are defined as the local maxima of the absolute curvature. Since the curvature of a strong corner is higher than that of a weak corner, we remove weak corners on an edge if their curvature values are lower than the predefined edge curvature-threshold. False corners are eliminated by comparing each curvature maximum with its two neighboring minima. If the curvature of a corner point is less than the double of the curvature of a neighboring minimum it is removed as a false corner [9]. The outline of the proposed corner detector is as follows.

- Find edge image using the Canny edge detector.
- Extract edges from the edge image: *i*) fill the gaps if they are within a range and select edges and *ii*) find T-junctions and mark them as T-corners.
- Parameterize each edge using its affine-length and smooth the parameterized curve using one of three smoothing-scales determined based on the number of points on it.
- Compute absolute curvature on each smoothed curve and find the candidate corners which are the absolute curvature maxima points.
- For each edge, determine corners by comparing the curvature maxima values to the edge curvature-threshold and the neighboring minima.

- Track corners down to the lowest scale considering a small neighborhood to improve localization.
- Compare T-corners with the tracked corners and add those T-corners which are far away from the detected corners.

We detail all the above steps in the following subsections. All the chosen parameter values that we will present below were decided either from the existing work or based on our empirical study presented later in Section V-B1a.

a) Edge extraction and selection: In the Canny edge detector [32] too many weak and noisy edges are detected when the edge strength threshold is set too low. Conversely, many legitimate edges are missed when the threshold is set too high. In addition, the edge strength may be changed due to geometric transformations. Therefore, in our implementation, we choose the threshold range between 0.2 and 0.7 after experimentation.

Note that the CSS corner detection technique [9] can detect corners on both closed and open curves. Therefore, we do not need to follow any postprocessing, like the level set method in [34], on the extracted curves to obtain close contours. We simply extract planar curves from the Canny edge detector output as follows.

The output of the Canny edge detector is a binary image where "1" represents edge-pixel and "0" represents nonedge-pixel. Only the edge-pixels are tracked to extract curves from the binary image. Initially, all the edge-pixels are kept unmarked. If an edge-pixel is found while scanning the binary image, it is marked and added to an initially empty curve (length 0). To extend the curve, an unmarked edge-pixel is searched in the 3×3 neighborhood of each end-point. If only one unmarked edge-pixel is found, it is selected. If more than one such unmarked pixels are found, the one having the lowest Euclidean distance from the curve-end is selected. The selected pixel is marked and added to the respective curve-end. This process continues until none of the ends can be extended anymore with the remaining unmarked edge-pixels. Once the construction of a curve is completed, the rest of the unmarked edge-pixels are tracked similarly to construct another curve.

Some of the curves obtained as above are either natural short edges or detected due to strong noises. It is difficult to smooth short curves due to the restriction of the smoothing window size (window size should be smaller than the curve length). Moreover, since the affine-length of a curve is much smaller than its arc-length, we need to select edges of reasonable lengths. Thus, in our implementation, curves are selected only when their lengths n_p meet the following condition:

$$n_p > (w + h)/\alpha \tag{14}$$

where w and h are width and height of the image and α is called the edge-length controller. If α is small, only long curves are selected; if it is large, short curves are also selected. In our experiments, we set $\alpha = 25$, and, therefore, for an image of size 512 $\times$ 512, the minimum n_p is 42. The above edge extraction setup also helps speeding up the later steps by avoiding a large number of weak and very short edges which may not contain strong corners. If an edge runs through any point within 2 pixels away from an end of another edge, we select that end as a T-junction and add to the T-corner set [9].

b) Affine-length parameterization and corner detection: We calculate the affine-length τ of each selected edge $\Gamma(t)$ and obtain $n_s = \lfloor \tau/\lambda \rfloor$ equally distant sample points which constitute the parameterized curve $\Gamma(\tau)$, where λ is the sample-interval in affine-length. Therefore, the distance between two successive points on $\Gamma(\tau)$ is affine-length λ on $\Gamma(t)$. The gap (in number of pixels) between the sample points mainly depends on the nature of the curve. It is large on almost straight-line like curve segments, but small on those segments where the curve makes significant direction changes (e.g., on and near corners). In order to reduce the gap, one may suggest using a small λ for not missing any important corners. We used λ =0.25, 0.5, 0.75, 1.0 in our experiments and found that all of them gave almost the same corner detection performance (see Section V-B1a).

Localization of the detected corners is better with smoothing using low sigma values. However, the stability (robustness) of the detected corners is better with smoothing using high sigma values. Moreover, a long curve requires higher smoothing-scale than a short curve [10]. In order to make a trade off, we smooth the parameterized curve at one of three medium smoothing-scales and corners detected at the smoothed curves are tracked down to the finest scale to improve the localization.

We smooth each parameterized curve using one of three medium smoothing-scales determined based on the number of sample points (n_s) on it. We set the three smoothing-scales as $\sigma_m = 3$, $\sigma_{m+1} = 4$, and $\sigma_{m+2} = 5$ for short, medium, and long edges respectively and edge curvature-threshold $t = 0.03$ for all types of curves. The short, medium, and long edge definitions of parameterized curves are defined by the constraints $n_s \leq 60$, $60 < n_s \leq 120$, and $n_s > 120$, respectively. We compute curvature at all points on the smoothed $\Gamma(\tau)$ and absolute curvature maxima points are taken into the candidate corner set. A local maximum point of the absolute curvature function is likely to be a strong corner, but it could also be a weak corner (also called 'round' corners in the literature [9]) or a false corner [9] as explained below.

The strong corners are very sharp in nature and visually prominent features on curves. Their localization is very good and curvature values are usually very high. In contrast, the weak corners are flat in nature and visually less significant features on curves. Their localization is very poor since it may be very difficult to point out their exact locations. Furthermore, their curvature values are usually smaller than the strong corners. The false corners are due to noise or high local variation on the curve. They are usually detected on those curves where there may be no visually prominent corners, for example, on circles. Note that although the notion of corner seems to be intuitively clear, no generally accepted mathematical definition exists, at least for digital curves [35]. Therefore, it is hard to give formal definitions for strong, weak, and false corners and we define them using their apparent characteristics as discussed above.

In the candidate corner set, we may have all three types of corners—strong, weak, and false. The later two should be removed from the candidate corner set. We follow the same technique as the CSS detector [9] to remove weak and false corners. Since the curvature of a strong corner should be greater than that of a weak corner, a candidate corner is removed from the candidate corner set if its curvature value is lower than t. To remove false corners, the curvature value of each candidate corner is compared with two of its neighboring minima on the same curve. If the curvature value of a candidate corner is not at least twice of its two neighboring minima, this candidate corner is removed from the candidate corner set [9].

Note corners detected on the smoothed curve obviously suffer from some absolute localization error with respect to the actual corner locations on the nonsmoothed curve. This error can be mitigated in two ways. First, the smoothing-scale should be as low as possible. Since, the affine-length of a curve is usually small, the ARCSS detector selects the low sigma value $(\mathrm{sigma} = 3)$ for most of the curves. Second, a corner tracking step, which tracks the detected corners in the lower scales, is followed to improve the error. Like other CSS detectors, we also follow a corner tracking step as discussed below.

c) Corner tracking and adding T-corners: As corners are detected at coarse scales, their localization may not be good. We track the corners on $\Gamma(t)$ 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 pth point on $\Gamma(\tau)$ at $\sigma = 3$, first we track this corner at $\sigma = 2$ considering 7 consecutive points, where the pth point is the midpoint. The point having the maximum absolute curvature among these 7 points is the tracked position of that corner at $\sigma = 2$ and the next tracking is executed at $\sigma = 1$.

In the above corner tracking 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 six (neighbors) curvature values around each of the detected corners need to be computed during tracking [9]. As a consequence, corner localization is improved without increasing the computational cost.

Note that in the above multiscale corner tracking, we follow a linear decreasing of smoothing-scale. Because we observed that when corners are detected using a small sigma value, the nonlinear decreasing made the procedure expensive by increasing the number of iterations, but did not give better localization performance than the linear decreasing which requires less number of iterations. However, if one uses a large smoothing-scale for corner detection, the linear decreasing of sigma in multiscale corner tracking may cause a high localization error.

Finally, we add T-corners to the final corner set when there are intersections of curves. There may be an already detected corner on any of the intersecting curves (i.e., at or near a T-corner point). A T-corner is added to the final corner set if there is not an already detected corner in the 5×5 pixels neighborhood around it [9]. Such an use of the Euclidean neighborhood should not be a problem because in practice there may be no or only a few of such points in real world images. Ideally, T-corners should be detected in the affine space. But this is not possible for two reasons. First, the affine-length cannot be calculated between points on different curves. Second, at the corner detection stage we need to detect T-corners regardless of whether the image has been transformed or not.

III. Corner Matching

The successful application of a corner detector in many applications depends on how to match corners between images. The proposed matching technique is robust to both geometric transformations and signal processing attacks. It can be used with contour-based corner detectors. Particularly, here we will use it with our proposed ARCSS corner detector which provides useful information such as curvatures and affine-lengths between corners.

A. Existing Matching Techniques

In the second group of corner matching techniques (see Section I-B), the matching problem was first formulated as the subgraph isomorphism problem to find the maximal clique [26]. Nevertheless, finding the maximal clique is an NP-complete problem [21]. The problem was later formulated as an optimization problem and either a relaxation technique [29] or a binary Hopfield network [27] was applied for cost minimization. A similar approach based on the fuzzy inference procedure was proposed in [28]. Optimization techniques were robust to rotation, translation, and partial occlusion of the object; however, were vulnerable to small scale change and computationally expensive. The solution in [25] used the Hausdorff distance to determine the degree of mismatch between two sets of feature points. However, the Hausdorff distance is too sensitive to noises or outliers and is not invariant to geometric transformations. The method in [24] used a similarity measure based on the corner strength. It first matched local feature groups and then estimated the affine transformation by global matching. But it was vulnerable to down-scaling and computationally expensive.

Zhou *et al.* [22] proposed a novel corner matching technique based on the *Delaunay triangulations* (DT) which were formed among the detected corners. The corner correspondence was established based on the observation that the interior angles of the Delaunay triangles completely and uniquely characterized the corners and their values were not affected much by rotation, uniform scale change, and translation. At the matching stage, the most similar triangle pairs were obtained and then their edges were extended circularly until all matching corners were triangulated and mismatching corners were discarded. The DT matching was very much promising; however, it would fail under aspect-ratio change (nonuniform scaling) and other geometric transformations such as shearing where interior angles of the Delaunay triangles are affected.

B. Proposed Matching Technique

We observed that the absolute curvatures of corners may change due to geometric transformations; however, for many corners they only change slightly (see Fig. 2 and explanation in Section III-B1b). Moreover, the affine-length between two corners on the same curve [see (2) in Section II-B1] and the triangular area consisting of any three corners as vertices [see (32) in Appendix I-C] are relatively invariant to affine transformations. Therefore, in addition to corner positions, the proposed matching procedure uses the affine-length and triangular area invariances and curvature value to establish corner correspondences between two images.

Fig. 1 shows the block diagram for the proposed ALTA matching technique. For two given corner sets A and B of images I_A and I_B respectively, we first obtain candidate corner matches using curvature values and affine-lengths. For each

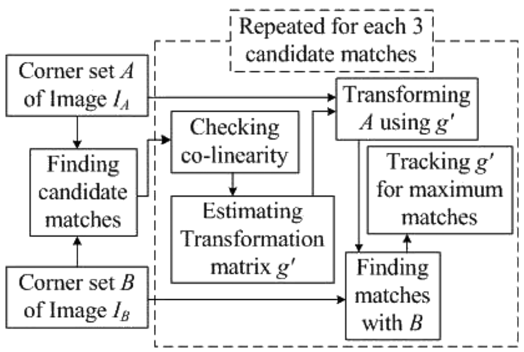

Repeated for each 3 candidate matches
Corner set A of Image I_A
Finding candidate matches
Corner set B of Image I_B
Checking co-linearity
Estimating Transformation matrix g'
Transforming A using g'
Tracking g' for maximum matches
Finding matches with B

Fig. 1. Block diagram for the proposed matching technique.

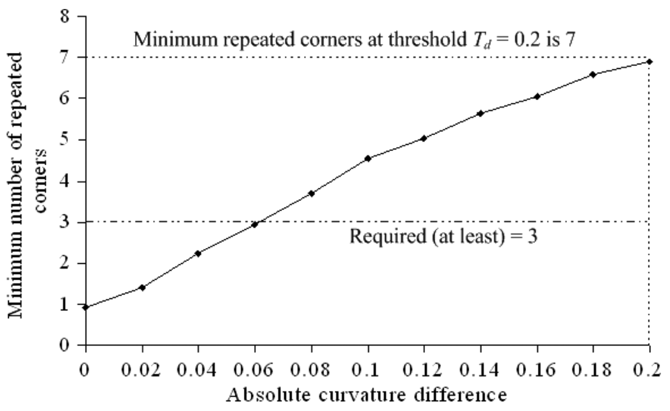


Fig. 2. Minimum number of repeated corners by the ARCSS (affine-resilient curvature scale-space) detector in geometric transformations (using the image database mentioned in Section V-A1).

combination of any three candidate matches, if the corners are noncollinear on the respective image plane, we have two triangles—one in each image plane. If the areas of these two triangles are similar (the difference is within the certain threshold), we estimate the transformation matrix g' that transforms the triangle in I_A into the triangle in I_B. Finally, we transform the corners in A using g' and find matches within B. We keep track of g' over all iterations. The g' which gives the highest number of corner matches corresponds to the transformation matrix g we are looking for. The corner matching performance is calculated for this best matching case.

In the above procedure, the candidate corner matches are obtained in two steps. In the first step, we obtain candidate matches (P_A, P_B) within a specific curvature difference. In the second step, if there are a total of two or more detected corners on both of the curves where P_A and P_B are detected, we add more candidate corner matches (Q_A, Q_B) from the same pair of curves by matching the affine-lengths between corners. This means that the affine-length between corners P_A and Q_A on a curve in I_A is compared with the affine-length between corners P_B and Q_B on a curve in I_B. Notice that P_A and Q_A denote corners in A and P_B and Q_B denote corners in B. Further notice that both of the similarity measurements (affine-length using (2) and triangular area using (32)) depend on the considered ranges of scaling and shearing transformation factors.

We need only three true matching corners to estimate the original transformation matrix g. However, finding true corner matches is the aim of the matching procedure and is not straightforward. The worst case is to consider every possible combination of three corner pairs between I_A and I_B with the complexity of $O(m^3n^3)$, where $m = |A|$ and $n = |B|$ (see Appendix I-B). Therefore, we apply some strategies to reduce the computational cost.

1) Speeding Up the Matching Procedure: If the two images I_A and I_B are originated from the same image due to some geometric transformations or signal processing attacks, the above matching procedure finds the maximum number of true corner matches. The estimated transformation matrix g' that causes the highest number of corner matches is an approximation of the original transformation matrix g. This process will be expensive in TII where one test image is compared with all database images to find possible transformed images. We apply the following strategies to reduce the cost.

a) Limiting the number of candidate corner matches: A maximum of three candidate corners are used from each pair of curves between two images. Because we need only three true corner matches to estimate g. If there are two or more corners on the same curve, we obtain maximum two more candidate matches (Q_A, Q_B) using the affine-length, for each candidate match (P_A, P_B) determined previously using the curvature value. The reason is if (P_A, P_B) is a true match, then we need only two more true matches to determine the transformation; but if it is a false match, we do not want to obtain more false matches based on it.

b) Setting the curvature difference threshold: We need to obtain at least three repeated corners to estimate g'. The absolute curvature value of a corner should not change significantly under geometric transformations. For the transformations considered in Section V-A1, the ARCSS detector offered minimum 3 repeated corners when the curvature difference was greater than 0.06 (see Fig. 2). However, in order to provide a safe margin, we set a threshold $T_d = 0.2$ for the curvature difference when we have minimum 7 repeated corners. Consequently, two corners will be considered a candidate match only when the difference between their curvature values is within 0.2.

c) Using hash tables: We do not need to obtain a candidate corner match more than once. But the addition of candidate matches based on the affine-length may produce duplicates. We use a hash table to store the candidate corner matches obtained so far. If a new candidate corner match is found and if it is not in the hash table, we store it in the hash table. Moreover, in a later iteration we do not need to consider those combinations of (three) candidate corner matches that we have considered in an earlier iteration to estimate g'. That means while estimating g' in each iteration, at least one of three considered candidate matches should be new. To ensure it, we store the iteration number in which a candidate match is obtained and entered into the hash table. We use the division method [36] for implementing the hash table.

d) Selecting possible transformed images: The number of detected corners between the original and transformed images may differ, but it does not change dramatically. For the database in Section V-A1, the maximum difference in number of detected corners by the ARCSS detector was 31. Consequently, when we apply the matching procedure to TII, we only compare images if the difference in corner numbers between the query and database images is within 40.

e) Selecting top corners: If all the detected corners are considered, it may make the procedure too expensive depending on the number of corners in I_A and I_B. Considering that the corners with the highest curvature values should be the most important and robust, we can speed up the matching process without compromising accuracy by using only a few corners with the highest curvature values. In fact, by selecting only the top few corners based on curvature values, it not only speeds up the iteration but also offers better performance, as discussed later in Section V-B3.

IV. Performance Evaluation Metrics

In this section, we will briefly describe some performance evaluation metrics used by the existing corner detectors and matching techniques along with their merits and demerits. Then we will present the metrics that we have used for fair performance comparison. We will also present the metrics which were used for evaluating the TII performance.

A. Existing CSS Corner Detector Evaluation

The performance evaluation of existing CSS corner detectors [8]–[10], [31] is based on the human judgement. The measurement of *consistency of corner numbers* and *accuracy* proposed in [31] were flawed. Moreover, they used only a few of test images, some of which were artificial block images. As a result, their performance measurement may not be suitable for large databases consisting of real world images.

1) Human Visual Inspection: In [8]–[10], [31], both the false positive and false negative probabilities were evaluated by differentiating the set of detected corners from the set of corners pointed out by *human visual inspection* (HVI). This kind of evaluation is not very much useful for proper robustness tests due to several reasons. First, it is very hard to point out all corner locations by human intuition in natural images containing many corners. Second, human visual system is unable to measure the strength of corners. Consequently, it fails to distinguish between different corners and to sort them out according to their strength. Thirdly, the volume of work with a large image database for HVI prohibits its adoption. Finally, there is no standard procedure e.g., how many people should be involved and how to assess their decisions.

Mokhtarian and Mohanna [31] calculated *accuracy* by matching corners identified by human eyes to those detected by a corner detector. As *accuracy* usually indicates the localization of the detected corners, such a metric based on the number of the detected corners is flawed. In order to measure *accuracy*, the locations of the detected corners should be considered instead [37] (see Section IV-B2).

2) Consistency of Corner Numbers: Consistency of corner numbers (CCN) measures the similarity in corner numbers between the original and transformed images. Mokhtarian and Mohanna [31] defined CCN as

$$CCN = 100 \times 1.1^{-|N_t - N_0|} \tag{15}$$

where N_0 and N_t are numbers of detected corners in original and transformed images respectively. This metric is severely flawed for the robustness test because if a corner detector detects N points in the original image as corners and completely different N points in the transformed image, then $N_0 = N_t = N$ and CCN will be 100%. Moreover, it does not consider whether the detected corners are repeated or not. As a result, CCN could not be accepted as a fair and unambiguous metric for the robustness evaluation.

Mokhtarian and Mohanna [31] used both the *accuracy* and CCN to compare the performance of their CSS and ECSS corner detectors [9], [10]. Therefore, though they showed that the ECSS corner detector performed better than the CSS detector, we found the reverse in our performance study (see Section V-B1b).

3) Preferred Metrics: In fact, the performance evaluation of corner detectors should be automated and without using any human intuition. For example, they could be evaluated in terms of *repeatability* and *localization error* which were previously used for evaluating other types of feature detectors and matching techniques [37]–[39]. We present them in the following section. However, one of them had problems and we propose to modify it to eliminate problems (see Section IV-C1).

B. Existing Corner Matching Evaluation

Many existing corner or interest-point matching techniques used the repeatability and localization error to evaluate their performance.

1) Repeatability: Repeatability indicates how stable the detected corners are under different geometric transformations and signal processing attacks. It is defined as the percentage of the total observed corners repeated between the original and transformed (and/or signal processed) images. Trajkovic and Hedley [38] estimated repeatability as the ratio R_0 of the number of corner matches (between the original and transformed images) N_m to the number of corners in the original image N_0. Schmid *et al.* [39] used the ratio R_t of N_m to the number of detected corners in the transformed image, N_t. Both R_0 and R_t are shown in (16)

$$R_0 = \frac{N_m}{N_0} \quad \text{and} \quad R_t = \frac{N_m}{N_t}. \tag{16}$$

The potential problem with R_0 and R_t alone is that they are highly sensitive to false corners. If a corner detector detects all points in the original image as corners, then $N_t = N_m$ and R_t will be 100%. In contrast, if a corner detector detects all points in the transformed image as corners, then $N_0 = N_m$ and R_0 will be 100%. Consequently, the performance evaluation based on R_0 and R_t alone could be ambiguous.

2) Localization Error: Due to the error introduced by the edge detection, corner detection, and pixel value interpolation during transformations, a particular location (corner-point) in an original image may not be accurately detected in the transformed image. Therefore, we need to allow some error while estimating the repeatability.

Localization error shows how accurately a detected corner is localized by the detector. Lower localization error indicates the better accuracy. It was measured using the *root-mean-square-*

error (RMSE) of corresponding positions of matching corners in the original and transformed images [6], [37]

$$L_e = \sqrt{\frac{1}{N_m}\sum_{i=1}^{N_m}\left[(x_{0i}-x_{ti})^2+(y_{0i}-y_{ti})^2\right]} \tag{17}$$

where (x_{0i}, y_{0i}) and (x_{ti}, y_{ti}) are the positions of ith matching corner in original and transformed images respectively. An RMSE value of maximum 3 pixels was allowed to find the corner matches [37]. If less than 3 pixels error is allowed, then some true corners may be missed leading to low repeatability. However, allowing a high error may consider some neighboring corners as repeated corners.

Note the localization error using (17) measures the relative error (between the detected corners in the original and transformed images), rather than the absolute localization error which is measured between the actual corner location (obtained using the ground truth from humans) and the detected corner location in an image. However, as discussed in Section II-B3b if a corner detector uses a low smoothing-scale and follows a corner tracking step after the corner detection on the smoothed curve, the relative error as well as the absolute error can be minimized.

C. Proposed Evaluation Metrics

We want to measure three types of performance: corner detection performance, corner matching performance, and TII performance. For evaluating detection performance, we modify existing repeatability metric and use localization error metric directly. For evaluating matching performance, we use the similar metrics as for the detection performance. For evaluating identification performance, we use the precision-recall graph.

1) Corner Detection Performance Measurement: The performance of a corner detector is evaluated using repeatability and localization error [37], [39]. For many applications, e.g., image copyright protection, once we have the required number of repeated corners, localization of the repeated corners is very important. The reason is, with the lower localization error the copyright verifier will be able to better localize the target pixel positions by using the repeated corners as reference points. Moreover, in the case of TII, the lower localization error helps estimating the geometric transformation matrix more accurately resulting in better identification performance.

To determine the corner detection performance, we first detect corners in each pair of original and test (geometric transformed and signal processed) images using a corner detector. We know the original transformation matrix. We transform the original corners using the known transformation matrix, prior to finding their repetitions in the test corner set. An RMSE value of maximum 3 pixels was allowed to find a repetition. In Section IV-A, we presented the problems with the performance evaluation metrics of existing CSS corner detectors [8]–[10], [31]. For fair and unambiguous comparison we employed the average of R_0 and R_t as the *average repeatability* R_{avg} defined as

$$R_{\text{avg}} = \frac{R_0 + R_t}{2} = \frac{N_r}{2}\left(\frac{1}{N_0}+\frac{1}{N_t}\right). \tag{18}$$

where N_0 and N_t are the number of corners in the original and test images, respectively, and N_r is the number of repeated corners between them. Note that R_{avg} is free from drawbacks associated with R_0 and R_t alone. We used the same evaluation metric, defined in (17) where $N_m = N_r$, for evaluating the localization error L_e of the repeated corners.

2) Corner Matching Performance Measurement: To determine the corner matching performance, we first detect corners in each pair of the original and test (geometric transformed and signal processed) images using the proposed ARCSS detector. The original transformation matrix is not known in this case. However, the proposed corner matching technique provides us an estimated transformation matrix. We transform the original corners using the estimated transformation matrix prior to finding their matches in the test corner set. Again, an RMSE value of maximum 3 pixels was allowed to find a match. We used the same metric like the average repeatability as *corner correspondence* (CC) between original and test images

$$\text{CC} = \frac{N_m}{2}\left(\frac{1}{N_0}+\frac{1}{N_t}\right) \tag{19}$$

where N_m is the number of corner matches between them. For a pair of images when one is a transformed version of other, $N_r \geq N_m$, i.e., $R_{\text{avg}} \geq$ CC. We also evaluate the localization error L_e of the matched corners using the metric in (17).

3) Transformed Image Identification Performance Measurement: In TII there are many groups of images and each group contains only geometric transformed and signal processed images of a particular image. Only images belonging to the same group are considered relevant (true positive match) to each other. In contrast, in the CBIR system all the images having the same or similar semantic features, e.g., red car, are considered relevant to each other and reside in the same group. We use *precision* and *recall* [2] collectively to measure the *identification performance*. *Recall* measures the system capacity to retrieve the relevant images from the database. It is defined as the ratio between number of retrieved relevant images R and total number of relevant images R_t in the database

$$\text{Recall} = \frac{R}{R_t}. \tag{20}$$

Precision measures the retrieval accuracy. It is defined as the ratio between R and number of retrieved images R_n

$$\text{Precision} = \frac{R}{R_n}. \tag{21}$$

In practice, the performance of an information retrieval system is presented using the *precision-recall graph*, where the higher the precision and recall values the better the performance is considered.

V. Performance Study

We carried out the following three performance studies and in each study we compare the proposed technique with one or more existing techniques by using different evaluation metrics discussed in Section IV-C.

- For evaluating *corner detection performance* we implemented the proposed ARCSS and existing CSS and ECSS

(which were set at their default parameters mentioned in [9], [10], and [31]) corner detectors and compare their performance using *average repeatability* and *localization error*.
- For evaluating *corner matching performance*, we implemented proposed ALTA and existing DT [22] matching techniques with the proposed ARCSS corner detector and compare their performance using *corner correspondence*.
- For evaluating *identification performance* we apply the proposed ARCSS detector and ALTA matching technique for TII. Their identification performance is compared with that of the existing GSH matching technique using the *precision-recall graph*.

A. Databases

We had two datasets of 23 and 100 original images respectively [40]. The first database was built using the first dataset and used for evaluating corner detection and matching performance. The second database was built using the second dataset and their transformed images and used for evaluating the TII performance. The use of two different databases of different sizes eliminated the bias of the proposed corner detection and matching technique to a particular database.

1) Corner Detection and Matching Database: There were a total of 23 different original 512 $\times$ 512 grayscale images [40], including "Lab," "Block," and "House" images used by the CSS and ECSS detectors, in this database. We obtained a total of 8694 transformed images of seven categories as test images:
- rotation at 18 different angles θ in $[-90^\circ, +90^\circ]$ at 10° apart, excluding 0°;
- uniform (U) scaling factors $s_x = s_y$ in [0.5,2.0] at 0.1 apart, excluding 1.0;
- nonuniform (NU) scaling factors s_x in [0.7,1.3] and s_y in [0.5,1.8], at 0.1 apart, excluding the cases when $s_x = s_y$;
- combined transformations (rot.-scale): θ in $[-30^\circ, +30^\circ]$ at 10° apart, excluding 0°, followed by uniform or nonuniform scaling factors s_x and s_y in [0.8,1.2] at 0.1 apart;
- JPEG lossy compression at 20 quality factors in [5,100], at 5 apart;
- zero mean white Gaussian noise at ten variances in [0.005,0.05] at 0.005 apart;
- shearing factors s_{hx} and s_{hy} in [0,0.012] at 0.002 apart, excluding the one when $s_{hx} = s_{hy} = 0.0$.

Therefore, we had a total of 414 rotated, 345 uniform scaled, 2691 nonuniform scaled, 3450 rotated and scaled images. We also had 460 JPEG compressed, 230 Gaussian noised, and 1104 sheared images. Note that transformations comprising rotations were also followed by cropping such that the outer black parts were discarded. Consequently, many detected corners in the original images were cropped off in the transformed images for the transformations involving rotations.

2) TII Database: In this database, there were a total of 100 different original 512 $\times$ 512 grayscale images [40] and their 1600 transformed images of five categories:
- rotation at four different angles θ in [5°,20°] at 5° apart;
- uniform scale factors $s_x = s_y$ in [0.85,1.15] at 0.05 apart, excluding 1.0;
- combined transformations: $[\theta, s_x, s_y] = [5^\circ, 0.8, 1.2]$ and [10°,0.7,1.1];
- JPEG lossy compression at quality factors 20 and 25;
- zero mean white Gaussian noise with noise variances 0.005 and 0.01.

We had five queries for each original image: rotation 10°, uniform scale factor 1.05, combined transformation [5°,0.8,1.2], lossy JPEG compression with quality factor 20, and Gaussian noise with variance 0.005. Therefore, we had a total of 100 groups of images in this database, each group consists of 17 relevant images (16 transformed and 1 original) for a corresponding query, and a total of 1700 images in the database. Like the previous database we also cropped the rotated images so that the outer black parts were disappeared. This diminishes the bias of large black regions in the rotated images for GSH matching technique. Note that the transformations we used are similar to those provided by the stirMark benchmark [41], which is commonly used for identifying copyright infringements.

B. Results and Discussions

We present results and discussions of the three performance studies in this section: corner detection, corner matching, and transformed image identification. Due to space limitation we will present the average results for each of the performance study over the whole respective database. Please see the detailed results on each type of attacks in [40].

1) Corner Detection Performance: In this section, we first summarize the experimental results of some parameter setting for the proposed ARCSS detector. We then present the comparative results between the proposed ARCSS and existing CSS [9] and ECSS [10] detectors. To present the performance comparison, we first show an example and then present the average results on the whole database discussed in Section V-A1.

a) Summary of ARCSS parameter setting: Many of the parameters used by the ARCSS detector had been originally set by the existing detectors, e.g., 2 pixel gap for detecting T-junctions by the CSS detector [9]. We have carried out experiments for setting the rest of the parameter values, e.g., Canny edge detection thresholds, edge-length controller, smoothing scale, sample-interval, and curvature-threshold. We summarize and discuss the results below. Note that these thresholds can be used for different image sets and need not be changed once determined.

Fig. 3 shows the effect of Canny edge detection thresholds changes on the ARCSS corner detector. When both the *low* and *high* thresholds were set small, the average number of detected corners by the ARCSS detector was quite high, but its robustness was low (low average repeatability and high localization error). The reason is, at lower thresholds the Canny edge detector obtains too many edges, many of which may be weak and noisy. In contrast, when the thresholds were increased, only strong edges were detected; therefore, the robustness of the ARCSS detector increased. However, when the thresholds were increased above $low = 0.2$ and $high = 0.7$, the robustness (average repeatability) of the ARCSS detector decreased. Moreover, when higher threshold values were used it missed some legitimate edges (hence, corners). Therefore, we have chosen the Canny

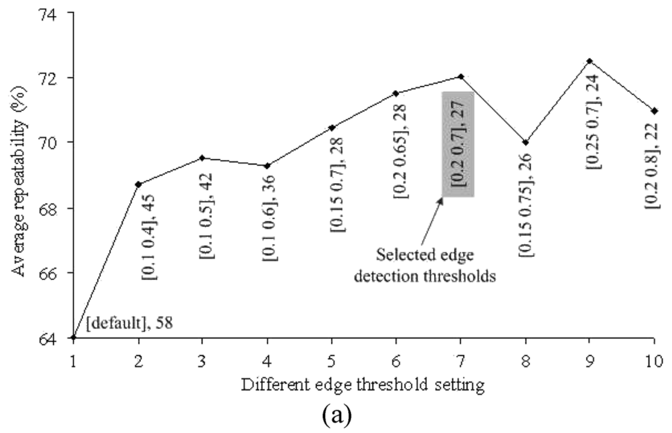
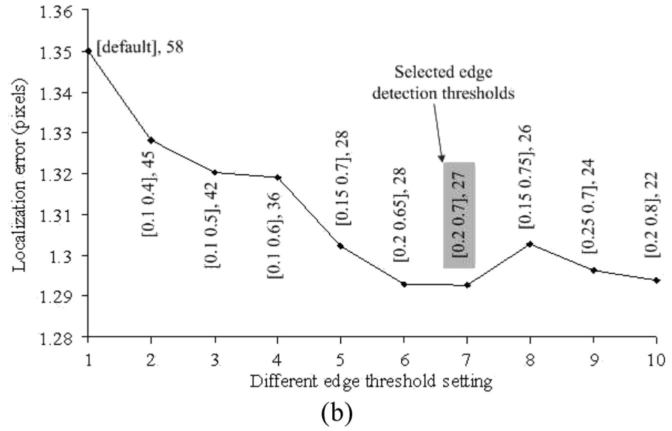



Fig. 3. Effects of Canny edge detection thresholds changes on the ARCSS (affine-resilient curvature scale-space) corner detector: (a) average repeatability and (b) localization error. At each data label the values within the square brackets are *low* and *high* thresholds of Canny edge detector and the value outside the bracket is the average number of detected corners by the ARCSS detector. The selected edge detection thresholds are shown using the gray background.

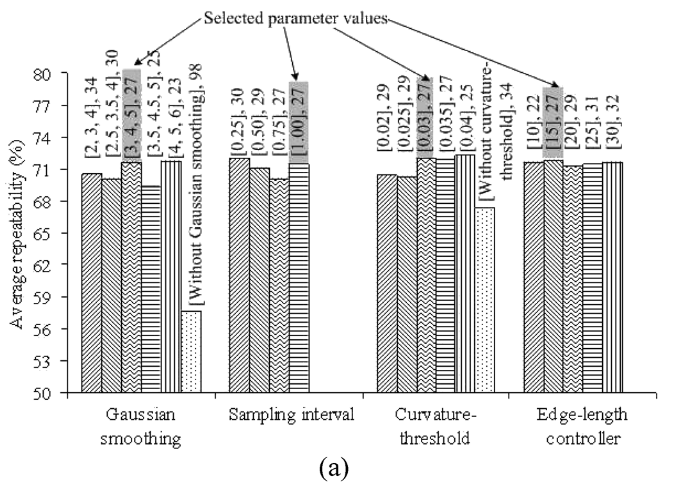
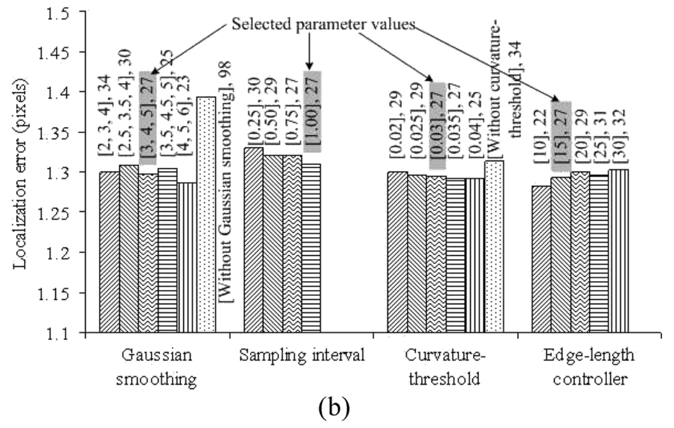



Fig. 4. Effects of different parameter changes (edge-length controller, Gaussian smoothing-scale, sampling interval, and curvature-threshold) on the ARCSS (affine-resilient curvature scale-space) corner detector. For each parameter, we had 4 or 5 different sets of values and we showed them in the square brackets above the corresponding bar. The value outside the square bracket above each bar is the average number of detected corners by the ARCSS detector. The selected values for each parameter are shown using the gray background.

edge detection thresholds at $low = 0.2$ and $high = 0.7$ as default for the ARCSS detector.

Fig. 4 shows the effect of different parameter changes (edge-length controller α, Gaussian smoothing-scale σ, sampling interval λ, and curvature-threshold t) on the ARCSS corner detector. We considered four or five different sets of values for each parameter.

We observed that the localization error increased with the increase of the value of α. Because many short and comparatively weak edges were detected when high values of α were used. Since the average repeatability was the highest at $\alpha = 15$, we selected this value as default.

When the value of σ was increased, the robustness of the ARCSS detector slightly increased (higher average repeatability and lower localization error) slightly, as shown in Fig. 4. However, at high σ value it missed some strong corners. Therefore, we have selected the set of σ values 3, 4, and 5, where most of the curves are smoothed using $\sigma = 3$.

The affine-length parameterization used by the ARCSS corner detector selects more points on the curve segments where the curve makes significant direction changes (e.g., on and near corners) than on the straight-line like segments. A high sampling interval λ selects less number of points on a curve than a low λ value does. To investigate whether λ affects the corner detection performance, we used four different λ values: 0.25, 0.50, 0.75, and 1.00. Fig. 4 shows the average repeatability and localization error in above four cases. Though the average repeatability was slightly increased (better) in the lowest λ value, the localization error also increased slightly (worse). Since, all the four cases offered almost the same average repeatability and $\lambda = 1$ offered the lowest localization error, the rest of the experimental results presented hereafter is with $\lambda = 1.0$.

The robustness of the ARCSS detector also slightly increased with the increase of curvature-threshold t, as shown in Fig. 4, since only the stronger corners are detected with higher t. Since at high t value it missed some strong corners, we have selected $t = 0.03$ as a trade off.

Fig. 4 also shows that the average repeatability was quite low and the localization error was quite high without using different Gaussian smoothing-scale and curvature-threshold. Thus, these results show the importance of curve smoothing before corner detection and of using the curvature-threshold to remove the weak corners.

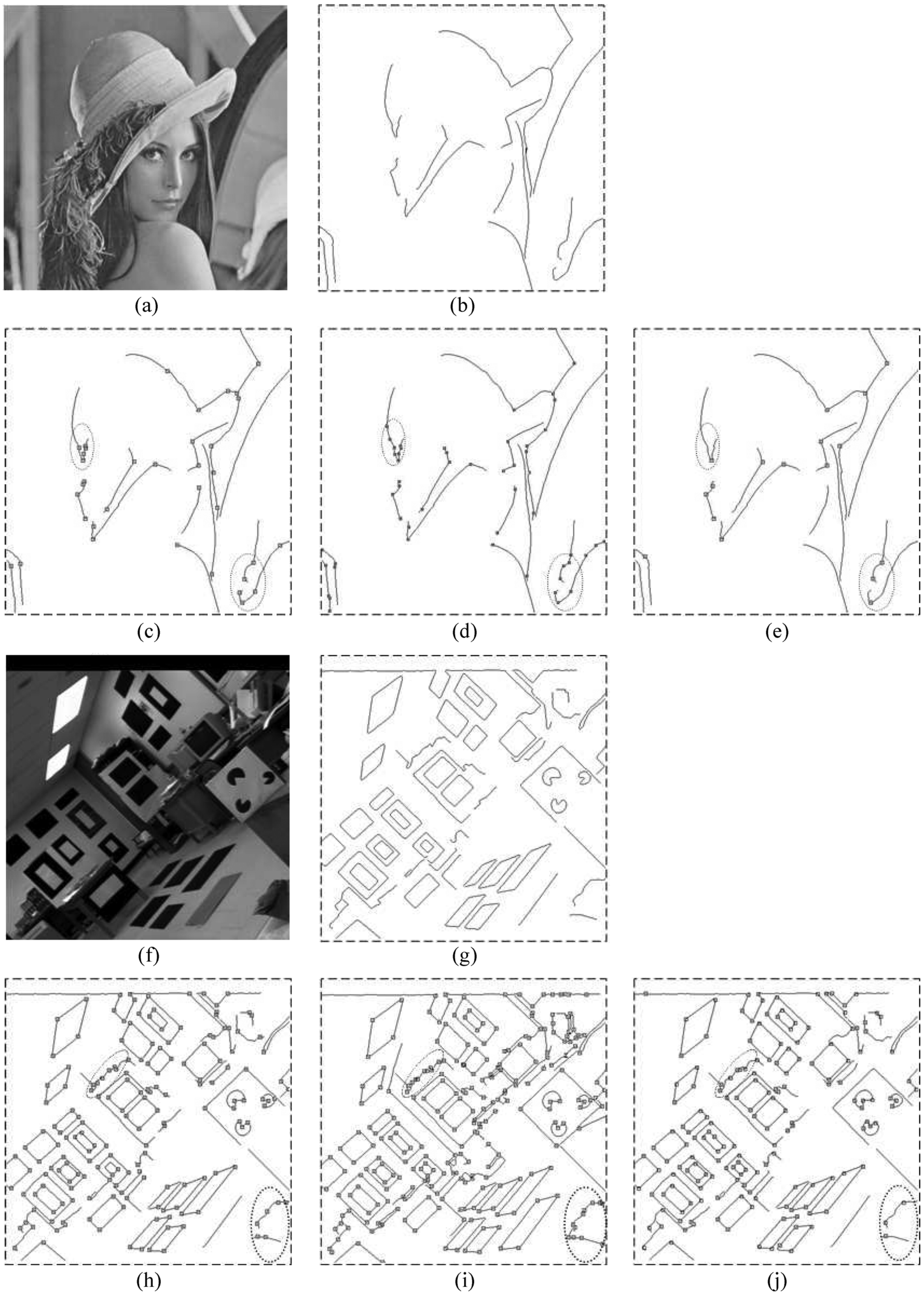


Fig. 5. Corner detection in original "Lena" and "Lab" images by different corner detectors. Edges were detected using Canny edge detector where how and high edge-detection thresholds were set at 0.2 and 0.7, respectively, and choosing those extracted edges whose length were at least 68 pixels. CSS = curvature scale-space, ECSS = enhanced CSS, ARCSS = affine-resilient CSS. (a) Original Lena (512 $\times$ 512); (b) edge map from (a); (c) CSS detector; (d) ECSS detector; (e) ARCSS detector (proposed); (f) original Lab (512 $\times$ 512); (g) edge map from (f); (h) CSS detector; (i) ECSS detector; (j) ARCSS detector (proposed).

b) Comparative results: Fig. 5 shows corner detection examples by the existing CSS and ECSS and the proposed ARCSS detectors in "Lena" and "Lab" images. On the curves inside the dotted ellipses in Fig. 5, we see that the ARCSS detector detected fewer weak corners than both of the CSS and the ECSS detectors.

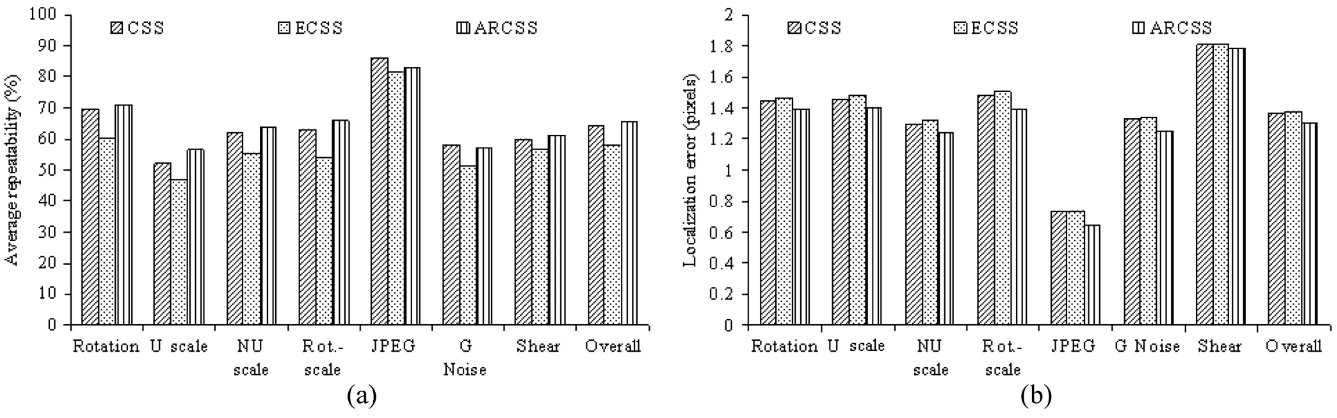


Fig. 6. Overall performance comparison of different corner detectors. CSS = curvature scale-space, ECSS = enhanced CSS, ARCSS = affine-resilient CSS, U = uniform scaling, NU = nonuniform scaling, G = Gaussian noising. (a) Average repeatability; (b) localization error.

Fig. 6 shows the average repeatability and localization error under geometric transformations, JPEG compression, and Gaussian noise. 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, both of 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 of the CSS and ECSS detectors in JPEG compression and noising. The possible reason is that the CSS detector obtains corners at a higher scale than the other two and, therefore, gains higher noise immunity to signal processing attacks. But as a consequence, it may miss some true corners. In contrast, since the ECSS and ARCSS detectors detect corners at one of the three medium smoothing-scales determined based on the parameterized curve length, they may detect some weak corners. However, the ARCSS detector detects fewer weak corners than the ECSS detector, since the former uses the affine-length parameterization.

Although the performance improvement in terms of localization-error (Fig. 6) by the proposed ARCSS corner detector is not great, it can be considered important in real applications such as copyright protection. The bar chart in Fig. 6(b) shows the average localization error. Some error could be zero pixels (minimum), some could be 3 pixels (maximum), and some could be in between the minimum and maximum. The lower average localization error indicates there were fewer corners with large errors (close to 3 pixels). In a copyright protection system, lower localization error helps minimizing the effect of synchronization error. The lower localization error also leads to more precise estimation of the transformation matrix during corner matching.

Note that in our experiments we found the CSS detector [9] to be more robust than the ECSS detector [10]. However, the opposite was found by Mokhtarian and Mohanna [31] and He and Yung [33]. The reasons of this conflict are as follows. First, the use of higher smoothing-scale by the CSS detector makes it more robust than the ECSS detector. While the former uses $\sigma = 4$ for all the curves, the ECSS detector uses $\sigma = 3$ for most of the curves. Second, the smoothing-scale selection according to the arc-length by the ECSS detector makes it more sensitive to the geometric transformations under which the arc-length of a curve is not preserved [19]. Third, Mokhtarian and Mohanna [31] used the flawed metrics (*accuracy* and CCN) as we discussed in Sections IV-A1 and IV-A2. Fourth, to measure *accuracy* they [31] used the ground truth from humans and did not consider the geometric transformed images. Finally, He and Yung [33] also used the ground truth from humans and did not consider the image transformations at all.

2) Corner Matching Performance: In this section, we first present the matching time with and without the speed up procedures for the proposed ALTA matching technique. We then present the detailed comparative results between the ALTA and DT matching techniques.

We observed that the implementation of the speed up procedures significantly improved the running time of the proposed ALTA matching technique (using Intel Pentium 3-GHz Processor, 1-GB RAM, Windows XP). Without applying the speed up procedures, the running time was 2.016 s for each pair of original and transformed images. By applying all the proposed speed up procedures, other than the selection of the top 15 corners, the matching time was 0.327 s. When we applied all the speed up procedures including the selection of top 15 corners the running time was 0.046 s only. Note that this gain in matching speed was achieved without compromising the matching accuracy.

Fig. 7 shows the overall corner correspondence by the ALTA and DT matching techniques under different geometric transformations and signal processing attacks. The ALTA method offered significantly higher corner correspondence than the DT method. The maximum corner correspondence by a matching technique will be the average repeatability of the involved detector as discussed in Section IV-C2. By comparing the ARCSS repeatability in Fig. 6 and the corner correspondence in Fig. 7, we see that by estimating the transformation in advance the corner correspondence by the ALTA matching is very close to the average repeatability of the ARCSS detector. In contrast, the DT matching offers a very limited performance.

The significant performance difference in terms of corner correspondence between ALTA and DT methods is due to the following reasons. First, the ALTA matching estimates the affine transformation matrix and transforms the corners of the data-

TABLE I
ESTIMATED TRANSFORMATION PARAMETERS BY THE PROPOSED ALTA (AFFINE-LENGTH AND TRIANGULAR AREA) MATCHING TECHNIQUE ON "LENA"

Attacks	**sizes**	$\theta(^\circ)$	s_x	s_y	t_x	t_y
Rotation-crop ($\theta = 10^\circ$)	442×442	10.29	1.00	1.01	0.41	0.42
Uniform scale ($s_x = s_y = 0.7$)	359×359	00.36	0.71	0.71	-0.31	-0.09
Non-uniform scale ($s_x = 0.9, s_y = 1.1$)	461×564	-0.10	0.89	1.10	-0.01	0.78
Rotation-scale ($\theta = 10^\circ, s_x = 0.9, s_y = 0.9$)	398×398	09.87	0.88	0.87	1.02	-0.82
Rot.-scale-trans. ($\theta = 10^\circ, s_x = s_y = 0.9, t_x = 10, t_y = -5$)	388×393	09.87	0.88	0.88	3.52	-5.82

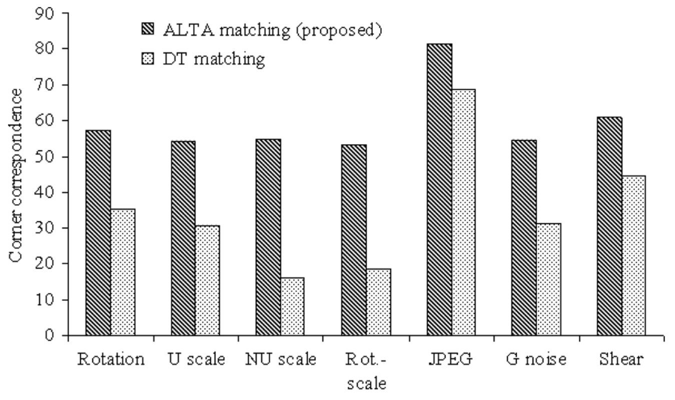


Fig. 7. Overall corner correspondence. ALTA = affine-length and triangular area, DT = Delaunay triangulations, U = uniform scaling, NU = nonuniform scaling, G = Gaussian noising.

base image prior to matching with the test corner set. Table I shows the estimated transformation parameters by the proposed ALTA matching technique under some geometric transformations on "Lena" image. Second, the Delaunay triangulation is very sensitive to outliers. It is unique for a given set of corners but a corner detector often offers a different set of corners under geometric transformations as well as in noising and lossy compression. Therefore, the sets of triangles between the original and test images differ considerably. Third, the interior angles of Delaunay triangles, used for the similarity measurement by the DT method, change according to the strength of the involved geometric transformation, especially when the image is nonuniformly scaled and sheared (see NU scale, Rot.-scale, Shear in Fig. 7). Fourth, true triangle matches may be missed in DT matching if they are not adjacent to any of the already obtained triangle matches. Finally, the DT matching would fail if the initial triangle match is false.

For example, Fig. 8 shows the corner matching examples by the proposed ALTA and existing DT [22] matching techniques. A total of 17 and 12 corners were detected in the original [Fig. 5(a) and (e)] and transformed [10° rotation, 80% scaling, 1.2% shearing, and cropping to 348 by 356 pixels, Fig. 8(a) and (c) (right)] images respectively by the ARCSS detector. The ALTA matching technique estimated the transformation matrix very precisely and found all the 8 true corner matches, as shown in Fig. 8(c). However, the first five most similar triangle matches by the DT matching technique were false, as shown in Fig. 8(d). The sixth most similar triangle match was true, but the DT matching could not find other true matches [as indicated by m in Fig. 8(d)] adjacent to this true match. Consequently, the DT matching technique obtained only three true corner matches (vertices of true triangle match 6).

Notice that we also estimated *localization error* of matching corners. Both of the ALTA and DT matching techniques had almost the same localization error since they were executed with the same corner detector.

3) Identification Performance: In this study, we investigated the TII performance of the proposed corner detector and matching technique. We compared their identification performance with that of the existing GSH matching technique [2] using the precision-recall graph.

Therefore, in TII for each query image we need to compare with all of the images in the database. As discussed in Section III-B1e, we can reduce the computational cost by only using the top few corners according to the curvature values. In our experiments, we considered two cases—when all the detected corners were considered and when only top 15 corners with the highest curvature values were considered. We selected top 15 corners based on two observations. First, if more than 15, say 20 or more, were selected, the procedure became more expensive. Second, if less than 15, say 10 or less, were selected, almost the same number of corner matches were found for both relevant and irrelevant images in the database.

Fig. 9 presents the average identification performance on five different queries (mentioned in Section V-A2) by the proposed ALTA matching and the existing GSH matching [2] techniques. While the ALTA matching used the corner correspondence between query and database images to rank the retrieved images, the histogram matching used the normalized L-1 distance [2].

We observed that the ALTA matching procedure offered better performance than the GSH matching technique in most of the cases. Nevertheless, for lower recall values, the GSH matching outperformed the ALTA matching, especially in queries comprising scaling. However, for higher recall values while the precision of the ALTA matching decreased slightly, that of GSH matching dropped significantly in all queries. This might be due to two reasons. First, scaling may preserve the ratio of grayscale intensities but in higher recall values different images may have the similar histogram. Second, when the number of corners was high, there might be many false matches between the irrelevant images.

Instead of considering all the detected corners, when we considered only the top 15 corners the identification performance increased considerably. Fig. 9 shows that for top 15 corner matching the precision of the ALTA matching procedure is above 90% which is almost the same for all the recall values but that of GSH matching dropped to 40% at 100% recall.

VI. CONCLUSION

The arc-length parameterized curvature [31] used by the existing CSS [9] and ECSS [10] detectors uses second-order

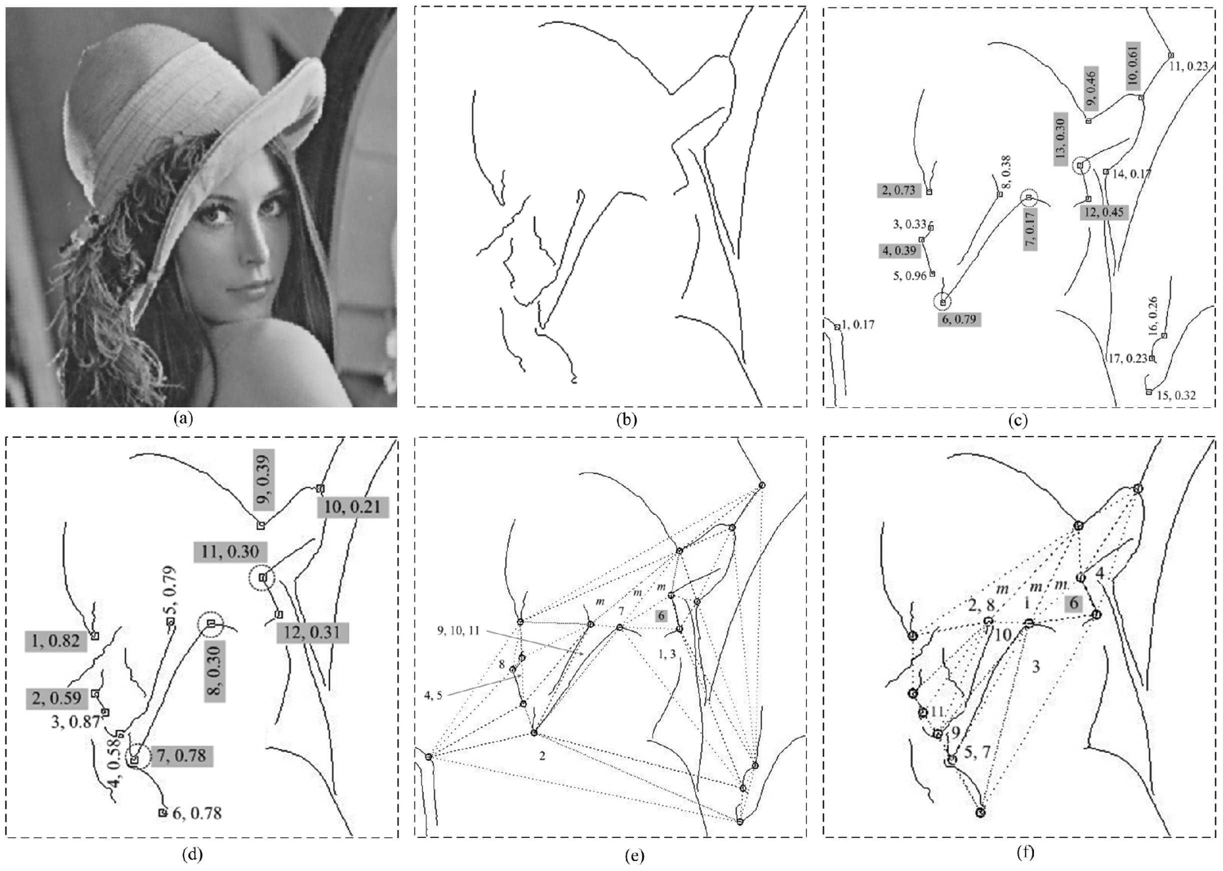


Fig. 8. Corner matching by proposed ALTA (affine-length and triangular area) and existing DT (Delaunay triangulations) matching techniques. ARCSS = affine-resilient curvature scale-space. (a) Transformed "Lena": rotation 10°, scaling factor 0.8, shearing factors 0.012, and cropping (348 × 356); (b) edge map from (a); (c)–(d) ALTA corner matching (proposed) between original (left) and transformed (right) images. The original "Lena" image, its edge map, and detected corers by the proposed ARCSS detector were shown in Fig. 3. The left image here also shows corners detected in original image with curvature values. The corners detected in the transformed image are shown in the right image along with their curvature values. Two numbers with each corner denote corner number and absolute curvature value, respectively. Two true candidate corner matches were found between original (corner numbers 6 and 13) and transformed (corner numbers 7 and 11) images within curvature difference 0.01. Two more true candidate corner matches (corner numbers 7 and 12 in original and 8 and 12 in transformed images) were found using affine-length between corners on the same curves. Three (shown in circles) of total four of these true candidate matches were used to estimate the transformation parameters. All eight true corner correspondences (whose numbers are shown using gray background) were obtained after transforming the original corner set with the estimated parameters; (e)–(f) DT corner matching between original (left) and transformed (right) images. Total 11 triangles were initially matched out of which 10 were false matches and only 1 was true match (triangle number 6 which is shown using gray background). The matched triangles (both false and true) are shown with numbers 1 to 11 with decreasing similarity. That means triangle number 1 is the most similar triangle between original and transformed images. The true triangle (number 6) match was used to find more true triangle matches. Though there were three more true triangle matches (shown indicated by m), the DT matching technique did not find any of them using the true match (number 6). Consequently, it obtained only 3 (vertices of triangle 6) true corner matches out of total 8 [shown using gray background in (c)–(d) above].

derivatives of curve-point locations.In contrast, the proposed ARCSS corner detector used the affine-length parameterized curvature which usually uses third order derivatives. The higher order derivatives involve more computational cost and cause unstable curvature estimation [19]. However, by mathematical derivations we have shown that the affine-length parameterized curvature can also be implemented using second-order derivatives. Therefore, the ARCSS detector extracts more robust corners with the similar computational cost as the existing detectors. It outperforms the existing detectors in terms of both average repeatability and localization under geometric transformations. It also possesses better localization than the existing detectors in JPEG compression and Gaussian noise.

The proposed ALTA corner matching technique uses information like curvature values and affine-length between corners on the same curve to obtain candidate corner matches. Then it reduces the search-space by matching the triangular areas of each combination of any three candidate corner matches before estimating the possible geometric transformation matrix. Finally, it transforms all the corners in one image using the estimated matrix before finding their matches in the other image. We have compared the proposed method with one of the most promising corner matching methods based on the Delaunay triangulation [22] using a wide range of transformed images. In experiments, the ALTA matching offered significantly higher corner correspondence than the DT matching and reached the highest performance of the involved ARCSS corner detector.

The TII performance of the proposed ARCSS corner detector and ALTA matching procedure was compared with that of the GSH matching technique [2] using the precision-recall graphs. The average precision (see Fig. 9) by the proposed corner detector and matching technique was always above 90% under all recall values. However, we observed that in spite of taking measures discussed in Section III-B1, the proposed ALTA matching

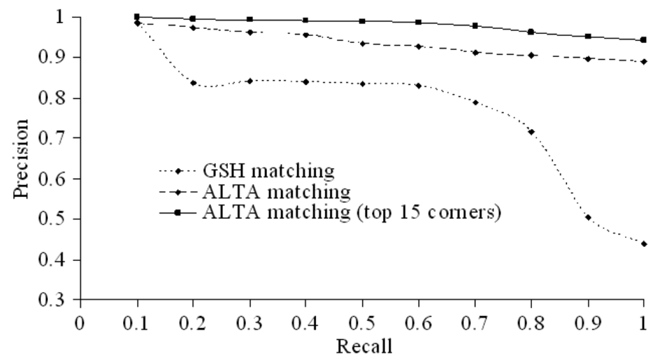


Fig. 9. TII (transformed image identification) performance (average on five queries mentioned in Section V-A2). ALTA = affine-length and triangular area, GSH = grayscale histogram.

technique required more time than the existing GSH technique. Future work will investigate more time-efficient matching technique and apply both the detector and matching technique along with some copyright protection technique (watermarking [3] or signature-based [1]) for ownership establishment.

Appendix I Some Mathematical Derivations

A. Derivatives of $x(\tau)$ and $y(\tau)$

By differentiating (1) we have

$$\frac{d\tau}{dt} = \left[\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right]^{1/3}. \tag{22}$$

So, we have

$$\frac{dt}{d\tau} = \frac{1}{\left[\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right]^{1/3}} \tag{23}$$

from which we can derive the first order derivative of $x(\tau)$ as

$$\dot{x}(\tau) = \frac{dx(\tau)}{dt} = \dot{x}(t)\frac{1}{\left[\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right]^{1/3}}. \tag{24}$$

Again, differentiating (23) with respect to τ

$$\frac{d^2t}{d\tau^2} = -\frac{1}{3}\frac{\dot{x}(t)\tilde{y}(t) - \tilde{x}(t)\dot{y}(t)}{\left[\dot{x}(t)\ddot{y}(t) - \ddot{x}(t)\dot{y}(t)\right]^{5/3}} \tag{25}$$

where $\tilde{x}(t)$ and $\tilde{y}(t)$ are third-order derivatives with respect to t. Therefore, by differentiating (24) with respect to τ and using (23) and (25), the second order derivative of $x(\tau)$ is

$$\ddot{x}(\tau)=\frac{3\ddot{x}(t)\left[\dot{x}(t)\ddot{y}(t)-\ddot{x}(t)\dot{y}(t)\right]-\dot{x}(t)\left[\dot{x}(t)\tilde{y}(t)-\tilde{x}(t)\dot{y}(t)\right]}{3\left[\dot{x}(t)\ddot{y}(t)-\ddot{x}(t)\dot{y}(t)\right]^{5/3}}. \tag{26}$$

Similarly, we have first and second order derivatives of $y(\tau)$ with respect to τ as

$$\dot{y}(\tau)=\dot{y}(t)\frac{1}{\left[\dot{x}(t)\ddot{y}(t)-\ddot{x}(t)\dot{y}(t)\right]^{1/3}} \text{ and} \tag{27}$$

$$\ddot{y}(\tau)=\frac{3\ddot{y}(t)\left[\dot{x}(t)\ddot{y}(t)-\ddot{x}(t)\dot{y}(t)\right]-\dot{y}(t)\left[\dot{x}(t)\tilde{y}(t)-\tilde{x}(t)\dot{y}(t)\right]}{3\left[\dot{x}(t)\ddot{y}(t)-\ddot{x}(t)\dot{y}(t)\right]^{5/3}}. \tag{28}$$

Using (24) and (26)–(28), we can easily derive

$$\dot{x}(\tau)\ddot{y}(\tau) - \ddot{x}(\tau)\dot{y}(\tau) = 1. \tag{29}$$

B. Estimating g' and the Worst Case Complexity of ALTA Matching

To estimate the transformation matrix g' between two given corner sets A and B of cardinalities $m = |A|$ and $n = |B|$, we need to solve the following equation:

$$\begin{bmatrix} x_B \\ y_B \end{bmatrix} = \begin{bmatrix} a & b \\ c & d \end{bmatrix}\begin{bmatrix} x_A \\ y_A \end{bmatrix} + \begin{bmatrix} e \\ f \end{bmatrix} \tag{30}$$

where (x_A, y_A) is a point in A and (x_B, y_B) is a point in B. As g' comprises six unknowns [transformations parameters a, b, c, d, e, and f as shown in (30)], we need at least three corners. In the worst case, we need to consider all possible cases, and, therefore, the complexity is ${}^mC_3 \times {}^nC_3 \approx O(m^3n^3)$.

C. Affine Invariance of Triangular Area

The area of a triangle with vertices $v_1 = (x_1, y_1)$, $v_2 = (x_2, y_2)$, and $v_3 = (x_3, y_3)$, when v_1 is shifted to the origin, is

$$\Delta(v_1,v_2,v_3)=\frac{1}{2}\left\|(x_2-x_1)(y_3-y_1)-(x_3-x_1)(y_2-y_1)\right\|. \tag{31}$$

Similar to (2), we can obtain the area of an affine transformed triangle as

$$\Delta(v_{1a}, v_{2a}, v_{3a}) = \Delta(v_1, v_2, v_3)\left[s_x s_y(1 - s_{hx}s_{hy})\right] \tag{32}$$

which implies that the triangle area is also absolutely invariant to rotation and translation, but relatively invariant to scaling and shearing.

References

[1] M. Awrangjeb and M. Murshed, "Robust signature-based geometric invariant copyright protection," in *Proc. Int. Conf. Image Processing*, Oct. 2006, pp. 1961–1964.
[2] G. Lu, *Multimedia Database Management Systems*. Norwood, MA: Artech House, 1999.
[3] J. S. Seo and C. D. Yoo, "Image watermarking based on invariant regions of scale-space representation," *IEEE Trans. Signal Process.*, vol. 54, no. 4, pp. 1537–1549, Apr. 2006.
[4] M. Alghoniemy and A. H. Tewfik, "Progressive quantized projection approach to data hiding," *IEEE Trans. Image Process.*, vol. 15, no. 2, pp. 459–472, Feb. 2006.
[5] C. J. Harris and M. Stephens, "A combined corner and edge detector," in *Proc. Alvey Vis. Conf.*, Aug.–Sep. 1988, pp. 147–151.
[6] K. Mikolajczyk and C. Schmid, "Scale & affine invariant interest point detectors," *Int. J. Comput. Vis.*, vol. 60, no. 1, pp. 63–86, Oct. 2004.
[7] L. Alvarez and F. Morales, "Affine morphological multi-scale analysis of corners and multiple junctions," *Int. J. Comput. Vis.*, vol. 25, no. 2, pp. 95–107, 1997.
[8] A. Rattarangsi and R. T. Chin, "Scale-based detection of corners of planar curves," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 14, no. 4, pp. 430–449, Apr. 1992.
[9] F. Mokhtarian and R. Suomela, "Robust image corner detection through curvature scale space," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 20, no. 12, pp. 1376–1381, Dec. 1998.
[10] F. Mokhtarian and F. Mohanna, "Enhancing the curvature scale space corner detector," in *Proc. Scand. Conf. Image Analysis*, Jun. 2001, pp. 145–152.

[11] R. Laganiere and T. Vincent, "Wedge-based corner model for widely separated view matching," in *Proc. Int. Conf. Pattern Recognit.*, Aug. 2002, vol. 3, pp. 672–675.

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

[13] L. Alvarez, F. Guichard, P.-L. Lions, and J.-M. Morel, "Axioms and fundamental equations in image processing," *Arch. Rational Mech. Anal.*, vol. 123, pp. 199–257, 1993.

[14] F. Mokhtarian and A. K. Mackworth, "Scale-based description and recognition of planar curves and two-dimensional shapes," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 1, pp. 34–43, Jan. 1986.

[15] G. Sapiro and A. Tannenbaum, "Affine invariant scale-space," *Int. J. Comput. Vis.*, vol. 11, no. 1, pp. 25–44, Aug. 1993.

[16] A. P. Witkin, "Scale-space filtering," in *Proc. Int. Joint Conf. Artifical Intelligence*, Aug. 1983, pp. 1019–1021.

[17] M. Awrangjeb, G. Lu, and M. Murshed, "An affine resilient curvature scale-space corner detector," in *Proc. Int. Conf. Acoustics, Speech, Signal Processing*, Apr. 2007, vol. 1, pp. 1233–1236.

[18] F. Klein, *Elementary Mathematics From an Advanced Standpoint: Geometry*. New York: Macmillan, 1939.

[19] F. Mokhtarian and S. Abbasi, "Affine curvature scale space with affine length parametrisation," *Pattern Anal. Appl.*, vol. 4, no. 1, pp. 1–8, 2001.

[20] M. Daoudi and S. Matusiak, "New multiscale planar shape invariant representation under a general affine transformations," in *Proc. Int. Conf. Pattern Recognit.*, Sep. 2000, vol. 3, pp. 786–789.

[21] E. Davies, *Machine Vision: Theory, Algorithms, Practicalities*. London, U.K.: Academic, 1990.

[22] D. Zhou, G. Li, and Y. Liu, "Effective corner matching based on delaunay triangulation," in *Proc. Int. Conf. Robotics and Automation*, Apr.–May 2004, vol. 3, pp. 2730–2733.

[23] F. Mohanna and F. Mokhtarian, "Robust corner tracking for unconstrained motions," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, Apr. 2003, vol. 5, pp. 804–807.

[24] I. Jung and S. Lacroix, "A robust interest points matching algorithm," in *Proc. Int. Conf. Computer Vision*, Jul. 2001, vol. 2, pp. 538–543.

[25] J. You, E. Pissaloux, and H. Cohen, "A hierarchical image matching scheme based on the dynamic detection of interesting points," in *Proc. Int. Conf. Acoustics, Speech, Signal Process.*, May 1995, vol. 4, pp. 2467–2470.

[26] R. Horaud and T. Skordas, "Stereo correspondence through feature grouping and maximal cliques," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 11, pp. 1168–1180, Nov. 1989.

[27] N. Nasrabadi and W. Li, "Object recognition by a hopfield neural network," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 6, pp. 1523–1535, Nov.–Dec. 1991.

[28] K. Lee, Y. Kim, H. Myung, J. Kim, and Z. Bien, "A corner matching algorithm with uncertainty handling capability," in *Proc. Int. Conf. Fuzzy Systems*, Jul. 1997, vol. 3, pp. 1469–1474.

[29] W. Rutkowski, "Recognition of occluded shapes using relaxation," *Comput. Graph. Image Process.*, vol. 19, no. 2, pp. 111–128, Jun. 1982.

[30] M. Awrangjeb and G. Lu, "A robust corner matching technique," in *Proc. Int. Conf. Multimedia Expo.*, Jul. 2007, pp. 1483–1486.

[31] F. Mokhtarian and F. Mohanna, "Performance evaluation of corner detectors using consistency and accuracy measures," *Comput. Vis. Image Understand.*, vol. 102, no. 1, pp. 81–94, Apr. 2006.

[32] J. Canny, "A computational approach to edge detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 8, no. 6, pp. 679–698, Jun. 1986.

[33] X. C. He and N. H. C. Yung, "Curvature scale space corner detector with adaptive threshold and dynamic region of support," in *Proc. Int. Conf. Pattern Recognition*, Aug. 2004, vol. 2, pp. 791–794.

[34] J. A. Sethian, *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge, U.K.: Cambridge Univ. Press, 1999.

[35] X. Gao, W. Zhang, F. Sattar, R. Venkateswarlu, and E. Sung, "Scale-space based corner detection of gray level images using plessey operator," presented at the Int. Conf. Inf., Commun., Signal Procesing, Dec. 2005.

[36] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to Algorithms*. Cambridge, MA: MIT Press, 1999.

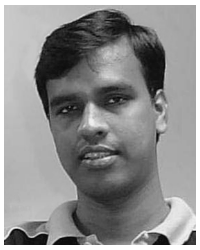
[37] Y. Abdeljaoued and T. Ebrahimi, "Feature point extraction using scale-space representation," in *Proc. Int. Conf. Image Processing*, Oct. 2004, vol. 5, pp. 3053–3056.

[38] M. Trajkovic and M. Hedley, "Fast corner detection," *Image Vis. Comput.*, vol. 16, no. 2, pp. 75–87, Feb. 1998.

[39] C. Schmid, R. Mohr, and C. Bauckhage, "Evaluation of interest point detectors," *Int. J. Comput. Vis.*, vol. 37, no. 2, pp. 151–172, 2000.

[40] M. Awrangjeb, 2007, Database [Online]. Available: http://www.personal.gscit.monash.edu.au/~awran/pr.html

[41] F. A. P. Petitcolas, 2007, Watermarking Database [Online]. Available: http://www.petitcolas.net/fabien/watermarking/image_database/index.html



Mohammad Awrangjeb (M'03) received the B.Sc.Engg. degree in computer science and engineering from the Bangladesh University of Engineering and Technology (BUET) in 2002 and the M.Sc. degree in computer science from the National University of Singapore (NUS) in 2004. He has submitted his Ph.D. dissertation with Monash University, Australia, where he has been fully funded.

He joined NUS as a Research Scholar. Before joining NUS, he was with the University of Asia-Pacific, Dhaka, as a Lecturer in the Computer Science and Engineering Department, and after finishing the M.Sc. degree, he joined the American International University—Bangladesh as a Lecturer in the Computer Science Department. He is currently working as a Research Fellow with the University of Melbourne, Australia. His research interest includes feature detection and matching, digital watermarking, digital photogrammetry multimedia security, biometrics, and network security. He has already published three journal papers and ten conference papers in these areas.

Guojun Lu (SM'96) received the B.Eng. degree from the Nanjing Institute of Technology (now South East University) in 1984 and the Ph.D. degree from Loughborough University.

He is currently a Professor at the Gippsland School of Information Technology, Monash University, Australia. He has held positions at Loughborough University, the National University of Singapore, and Deakin University. His main research interests are in multimedia communications and multimedia information indexing and retrieval. He has published over 120 refereed journal and conference papers in these areas and authored the books *Communication and Computing for Distributed Multimedia Systems* (Artech House, 1996) and *Multimedia Database Management Systems* (Artech House, 1999).