A Fast Corner Detector Based on the Chord-to-Point Distance Accumulation Technique

Mohammad Awrangjeb∗, Guojun Lu†, Clive S. Fraser∗ and Mehdi Ravanbakhsh∗

∗Cooperative Research Center for Spatial Information, The University of Melbourne
723 Swanston St, Carlton Vic 3053, Australia
Email: {mawr, c.fraser, m.ravanbakhsh}@unimelb.edu.au
†Gippsland School of IT, Monash University, Churchill Vic 3842, Australia
Email: Guojun.Lu@infotech.monash.edu.au

Abstract—The previously proposed contour-based multi-scale corner detector based on the chord-to-point distance accumulation (CPDA) technique has proved its superior robustness over many other single- and multi-scale detectors. However, the original CPDA detector is computationally expensive since it calculates the CPDA discrete curvature on each point of the curve. The proposed improvement obtains a set of probable candidate points before the CPDA curvature estimation. The CPDA curvature is estimated on these chosen candidate points only. Consequently, the improved CPDA detector becomes faster, while retaining a similar robustness to the original CPDA detector.

Index Terms—fast; CPDA; multi-scale; corner detector.

I. INTRODUCTION

Image features, for example, corners and interest-points, have applications in various areas such as computer vision [1], content-based copyright protection [2], transformed image identification [3] and photogrammetry [4].

In the literature, a large number of corner and interest-point detectors [5]–[7] have been proposed. They can be broadly classified into three groups: contour-based, intensity-based and model-based detectors. Contour-based detectors [5] first obtain planar curves using an edge detector and then search for the curvature maxima-points along those curves. Intensity-based detectors [6] estimate a measure which is intended to indicate the presence of an interest-point directly from the image pixel values. Model or template-based detectors [7] find corners by fitting the image signal into a predefined model.

Like other detectors contour-based corner detectors can be divided into two types: single-scale detectors [8] and multi-scale detectors [3], [9]–[12]. Single-scale detectors use a single corner detection scale. Since selection of a single-scale for all curves is a difficult task, it is preferable to detect corners using a multi-scale analysis [13].

Multi-scale detectors using a full range of scales [9] are computationally expensive. Moreover, combining corners detected in multiple scales is also a problem. On the other hand, multi-scale detectors using a small set of smoothing-scales [3], [10]–[12] are computationally less expensive and more popular. They detect corners using one [10], two [11] or three [3], [12] smoothing-scales and may track the detected corners to a common small smoothing-scale to combine the detected corners and to improve their localization.

The curvature estimation technique used by most of the abovementioned multi-scale detectors [3], [10], [12] is based on the curvature scale-space (CSS) [14] that uses Gaussian smoothing (with scale σ). However, the CSS curvature estimation technique has two main drawbacks [5]. Firstly, it uses up to second order derivatives of curve-point locations. As a result, the estimated curvature varies significantly on a curve having high local variation and noise. Secondly, the CSS-based technique requires the selection of appropriate smoothing-scales (to overcome the first problem and to facilitate the corner detection), which is not an easy task. In a robustness study presented in [3], the scale selection based on the curve-length by the enhanced CSS detector [13] was proved to be less robust than the original CSS detector [10]. Though the adaptive smoothing technique [15], [16] improved the localization of the detected corners, such an iterative adaptation of σ is computationally demanding. Moreover, since the local variation and noise on the curve is unknown and curves of the same length may require different amounts of smoothing, one σ value may not be suitable for all curves of the same length.

The first multi-scale detector based on the chord-to-point distance accumulation (CPDA) technique [17] overcomes these problems [5]. The main advantage of the CPDA curvature estimation technique is that it does not use any derivatives. In addition, the selection of the chord-length L (used for CPDA curvature estimation) is less sensitive to local variation and noise on the curve. As a result, the estimated curvature values are more stable. Consequently, experimental results have shown that it offers better robustness than the existing CSS-based detectors [3], [8], [10], [12].

However, the CPDA curvature estimation technique [17] is an expensive operation and the CPDA detector [5] estimates the CPDA curvature value at each point of a given curve. Instead of estimating the CPDA curvature on all the points of the curve, a small subset of the points can be selected. This way the computational cost of the CPDA detector can be reduced considerably.

In this paper, we obtain a set of probable candidate points before the CPDA curvature estimation as follows. We see that a curve-segment, with significant direction changes in the original curve, is more affected (shortened) by the Gaussian smoothing than a straight-line like curve-segment. As a result,
if \( d \) is the distance from a point on the original curve to its location in the smoothed curve, \( d \) is large on and near a corner location, but small if it does not have a corner nearby. Consequently, the maxima of the distance function consisting of point-to-point distances between the original and smoothed curves offer us the set of the probable candidate points. The CPDA curvature is estimated on these candidate points only. Through experiments on a large database we show that the improved CPDA detector runs much faster than the original CPDA detector but offers a similar robustness in terms of average repeatability and localization error.

The organization of this paper is as follows: Section II briefly describes the original CPDA detector \cite{5}; Section III presents the improvement for the fast CPDA detector; Section IV discusses the experimental results and, finally, Section V concludes the paper.

II. THE CPDA DETECTOR

The CPDA detector \cite{5} first extracts planar curves from the edge image detected by the Canny edge detector \cite{18}. Each curve is then smoothed with a small width Gaussian kernel in order to remove the quantization noise and trivial details. The CPDA technique \cite{17} is then used to estimate the curvature on the smoothed curves. In order to make the strong and weak corners more distinguishable, the CPDA detector first uses three different chord-lengths to estimate three normalized discrete curvature values on each point of the smoothed curve. It then multiplies the normalized curvature values to obtain the curvature product (a single estimated curvature) at each point. The maxima of the absolute curvature products along the smoothed curve are then marked as candidate corners. Finally, it follows a two-step refinement process that uses a curvature-threshold and an angle-threshold to remove any weak and false corners respectively.

We see that since the CPDA detector computes the expensive CPDA discrete curvature at each point of the curve, the detector becomes slow. In this section, we first briefly discuss present mathematical formula for each detection step and then analyze the detector’s computational complexity.

A. Steps of the CPDA Detector

The CPDA detector works in five major steps, briefly discussed in the following subsections. All the chosen parameter values used were decided either from existing work or based on our empirical study (see \cite{5} for details).

1) Edge extraction and selection: The CPDA detector extracts planar curves from the grey-scale image using the Canny edge detector \cite{18} with thresholds \( low = 0.2 \) and \( high = 0.7 \). To avoid having a large number of weak and short edges which may not contain strong corners, the detector selects those extracted curves whose lengths \( n \) meet the following condition:

\[
n > (w + h)/\alpha,
\]

where \( w \) and \( h \) are the width and height of the image and \( \alpha \) is the edge-length controller (\( \alpha = 25 \)).

2) Curve smoothing: The Gaussian convolution operation is used to smooth each selected curve \( \Gamma(t) = (x(t), y(t)) \), where \( t \) is an arbitrary parameter.

\[
\Gamma(t, \sigma) = \Gamma(t) \otimes G(t, \sigma) = (x(t) \otimes G(t, \sigma), y(t) \otimes G(t, \sigma)),
\]

where \( \Gamma(t, \sigma) \) is the smoothed curve, \( \otimes \) is the convolution operation and \( G(t, \sigma) \) is the Gaussian function defined as

\[
G(t, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{t^2}{2\sigma^2}\right),
\]

where \( \sigma \) is known as the smoothing-scale.

Since the CPDA detector does not follow any corner tracking step after corner detection, it uses a small scale Gaussian smoothing to keep the localization accurate. For curves of length \( n \leq 100 \) it selects scale \( \sigma = 1 \), for \( 100 < n \leq 200 \) it selects \( \sigma = 2 \) and for \( n > 200 \) it selects \( \sigma = 3 \).

3) Curvature estimation: In the CPDA discrete curvature estimation technique \cite{17}, a chord is moved along a curve and the perpendicular distances from each point \( P_i \) on the curve to the chord are summed to represent the curvature at that point (see Fig. 1).

To measure the CPDA curvature \( h_L(t) \) at \( P_i \) using a chord of length \( L \), we move the chord on each side of \( P_i \) at most \( L \) points while keeping \( P_i \) as an interior point.

\[
h_L(t) = \sum_{j=-L+1}^{t-1} d_{t,j},
\]

where \( j \) denotes the index of the first intersected point between chord and curve, every time the chord moves.

The CPDA detector uses \( 3 \) chords of lengths \( L_1 = 10, L_2 = 20 \) and \( L_3 = 30 \) and calculates \( 3 \) corresponding curvature functions \( h_1(t), h_2(t) \) and \( h_3(t) \). Then it normalizes them as

\[
h_j^1(t) = \frac{h_j(t)}{\max\{ \text{abs}(h_j) \}}, \quad \text{for } 1 \leq t \leq n \text{ and } 1 \leq j \leq 3.
\]

Consequently, the discrete curvature values become in the range \([0, 1]\). Finally, the three normalized curvature functions \( h_j^1(t) \) are multiplied together to find a single curvature value

\[
H(t) = h_1^1(t)h_2^1(t)h_3^1(t), \quad \text{for } 1 \leq t \leq n.
\]

The use of the above curvature product has two advantages. First, as the normalized curvature value of a strong corner
should be higher than that of a weak corner on the same curve, by multiplying three estimated curvatures at each point, the strong corners become more distinguishable from the weak corners. Second, a single curvature-threshold is sufficient to remove the weak corners.

4) Candidate corner set refinement: The CPDA detector gathers the local maxima points on the function $H(t)$ from all curves in the candidate corner set. A local maximum can either be a strong, weak (also known as ‘round’ in the literature [10]), or false corner. The latter two should not be detected as corners.

If a local maximum value is less than the curvature-threshold $T_h$ ($T_h = 0.2$), it is removed as a false corner. If the angle at a candidate corner (with its two neighboring candidate corners or curve-ends when enough neighboring candidate corners are unavailable) is greater than the angle-threshold $\delta$ ($\delta = 157^\circ$), it is removed as a false corner.

5) Final corner set: The final corner set is obtained after removing the weak and false corners from the candidate corner set. There may be a corner between the ends of a loop curve. To detect such a corner, the angle at one end of the curve is estimated and if it is less than $\delta$, it is added to the final corner set. At last, a T-corner is added to the final corner set if it is far from the already detected corners. In both the above cases, the CPDA detector considers the area outside a $5 \times 5$ neighborhood as ‘far’ [10].

Fig. 2 shows corner detection examples on the ‘Lena’ image. We see that without applying the (curvature and angle) thresholds, the detector detects many weak and false corners. However, after applying the thresholds, it finds no such corners.

B. Analyzing Computational Complexity

In this section, we analyze the computational complexity of the CPDA detector and compute the cost of corner detection on a single curve $\Gamma(t)$.

The first step (edge extraction and selection) is common to

1‘Lena’ image is from Fabien Petitcolas’s Photo Database at http://www.petitcolas.net/fabien/watermarking/image_database/index.html, courtesy of the signal and image processing institute at the University of Southern California. Permission is provided for research and non-commercial uses at the above web site.
all detectors and therefore is not included in our complexity analysis.

The Gaussian convolution (defined in (2)) is used for smoothing. Algebraically, convolution is the same operation as multiplying the polynomials whose coefficients are the elements of $G(t, \sigma)$ and $\Gamma(t)$. Therefore,\(^2\) the complexity of curve-smoothing is $O(mn)$, where $n$ is the curve-length and $m$ is the window size for $G(t, \sigma)$ which increases with the increase of $\sigma$. Note $G(t, \sigma)$, defined in (3), can be precomputed for a given $\sigma$ value.

A total of 7 operations (multiplication, divisions, squares and square roots) in each Euclidean distance computation ($d_{i,j}$ in (4)) is required and there are $L-1$ such distance computations for each point $P_t$. The CPDA detector uses 3 chords and on average there are 20 chord movements, i.e. $L = 20$, for $P_t$. So there is a total of $21n(L-1)$ computations in (4).

For each chord, there are $2n$ computations for finding maxima and normalization (to compute $h'_i(t)$ in (5)). An additional $2n$ multiplications is required for computing $H(t)$ in (6). As a result, $h'_i(t)$ and $H(t)$ together take $8n$ computations.

Finding candidate corners (curvature maxima points) also takes $O(n)$ operations. Let $n_c$ be the number of candidate corners. Since in practice $n_c << n$, the number of computations in refining and finalizing the candidate corner set (last two steps of the CPDA detector) is negligible.

Consequently, the number of total operations of the CPDA detector is $mn + 21n(L-1) + 8n + n = mn + 408n$, since $L = 20$ on average. For small values of $\sigma$, the value of $m$ is a small number too (for example, for $\sigma = 3$, $m$ is approximately 25). Therefore, the majority of time is passed during the CPDA curvature computation.

This provided the motivation us to investigate whether the CPDA curvature could be calculated on a small number of curve-points, thus improving the complexity of the detector. The proposed improvement, discussed in the following section, finds a small number of candidate points out of a total of $n$ points on the curve.

### III. PROPOSED IMPROVEMENT

The aim of the proposed improvement is to obtain a set of probable candidate points before the CPDA curvature estimation, which is then carried out on the candidate points obtained only following the technique discussed in Section II-A3.

The basic idea of choosing the candidate points is as follows: a curve-segment with significant direction changes in the original curve is more affected (shortened) by the Gaussian smoothing (with scale $\sigma$) than a straight-line like curve-segment. As a result, the distance from a point on the original curve to its location in the smoothed curve is high on and near a corner location but small if it does not have a corner nearby. Consequently, the maxima of the distance function consisting of point-to-point distances between the original and smoothed curves offer us the set of the candidate points.

Since the original curve may have noise and a large number of local variations that may not contain true corners, it should be smoothed using a small $\sigma$ before calculating the distance function. So we use two $\sigma$s. The original curve is first smoothed using $\sigma_1$ and $\sigma_2$, where $\sigma_1 < \sigma_2$. Then the distance function is obtained by calculating the distance between the two smoothed curves.

While a small value of $\sigma_1$ ensures a good localization of the candidate points obtained, a large value of $\sigma_2$ indicates their robustness. In the following section, we first derive the distance function and then discuss the difficulty of choosing values for $\sigma_1$ and $\sigma_2$.

#### A. Distance Function

According to (2) we have

$$\Gamma(t, \sigma_1) = \Gamma(t) \otimes G(t, \sigma_1) \quad \text{and} \quad \Gamma(t, \sigma_2) = \Gamma(t) \otimes G(t, \sigma_2)$$

and the distance function $F_d$ is calculated as the square Euclidean distance between the corresponding points on $\Gamma(t, \sigma_1)$ and $\Gamma(t, \sigma_2)$:

$$F_d(t, \sigma_1, \sigma_2) = [x(t) \otimes G(t, \sigma_2) - x(t) \otimes G(t, \sigma_1)]^2 + [y(t) \otimes G(t, \sigma_2) - y(t) \otimes G(t, \sigma_1)]^2$$

$$= [x(t) \otimes (G(t, \sigma_2) - G(t, \sigma_1))]^2 + [y(t) \otimes (G(t, \sigma_2) - G(t, \sigma_1))]^2$$

(by Distributive law of convolution)

$$= [x(t) \otimes G_d(t, \sigma_1, \sigma_2)]^2 + [y(t) \otimes G_d(t, \sigma_1, \sigma_2)]^2$$

where $G_d(t, \sigma_1, \sigma_2)$ is the distance function of two Gaussian functions. A similar distance function was used in [11]. The maxima of $F_d$ are simply passed as the candidate points.

#### B. Selection of $\sigma_1$ and $\sigma_2$

The selection of $\sigma_1$ and $\sigma_2$ is quite critical. While $\sigma_1$ ensures good localization, $\sigma_2$ confirms the robustness of the candidate points. This means a large value for $\sigma_2 - \sigma_1$ could be a good choice. However, if $\sigma_1$ is too small the candidate points may still have noise and therefore will be ill positioned. On the other hand, if $\sigma_2$ is too large some true candidate points may be missed. In addition, a large value for any of these $\sigma$s causes high computational cost during smoothing.

Fig. 3 depicts this scenario. On the original curve, points 1 and 5 are considered as false candidate points because they indicate high local variations or noise. Smoothing with $\sigma_1 = 1$ and $\sigma_2 = 4$ though removes points 1 and 5, point 3 on $F_d$ is confused since $\sigma_1 = 1$ could not sufficiently remove the local variation and noise (see Fig. 3(b)). In Fig. 3(c), $F_d$ obtained all four candidate points unambiguously with $\sigma_1 = 3$ and $\sigma_2 = 4$. However, if $\sigma_2$ is increased, some candidate points are merged as shown in Fig. 3(e).

In order to have a better trade off, we need to have medium values for $\sigma_1$ and $\sigma_2$. In an empirical study, presented in

\(^2\)The big-oh notation ‘$O$’ denotes the asymptotic upper bound (the worst case running cost) of a function. See [19] for a rigorous definition.
Section IV, we found that \( \sigma_1 = 3.0 \) and \( \sigma_2 = 4.0 \) are a good compromise.

C. Complexity of the Improved CPDA Detector

The candidate points selection technique described costs \( mn + n \) operations, since smoothing using (8) costs \( mn \) operations and finding maxima of \( \Gamma_d(t, \sigma_1, \sigma_2) \) costs additional \( n \) operations.

As a result, the total number of operations of the improved CPDA detector is \( 2mn + n + 408n_p \), where \( n_p \) is the number of obtained candidate points. For the original CPDA detector it was \( mn + 408n \) (see Section II-B). As \( n_p < n \) in practice and \( m \) is a small number for a small \( \sigma \), the total number of operations for the improved CPDA detector is significantly reduced.

IV. PERFORMANCE STUDY

In this section we empirically show that the improved CPDA detector is much faster than the original CPDA detector [5] without adversely compromising the robustness. It also runs faster than many existing CSS-based detectors.

In our performance study, we follow the automatic evaluation system introduced in [3], rather than the manual system in [13], which involves human judgement of corner detection and thereby is not suitable for the performance evaluation when using a large data set.

In the automatic system, corners detected in the original images by a detector are considered as reference corners and corners detected by the same detector in the test images which were signal processed and geometrically transformed, are taken as test corners. Then, reference and test corners are compared to evaluate the performance of that detector.

We evaluated seven detectors namely CSS [10], ARCSS [3], He & Yung [8], MSCP [12], Zhang et al. [11], CPDA [5] and the proposed improved CPDA detector. All the detectors, except the CPDA detector, were set at the default parameter settings indicated by their authors.\(^3\) We observed that setting different \( \sigma \) values based on the curve-length is rather impractical, since a different set of edges may be extracted from the test images and the extracted edges often vary in lengths. Therefore, we set \( \sigma = 3 \) for all curves for the CPDA detector.

We use three evaluation metrics: average repeatability, localization error and running time and a large database to compare performance.

A. Evaluation Metrics

We use average repeatability (\( R_{avg} \)) and localization error (\( L_e \)) from [3] for measuring the robustness and accuracy of the detected corners.

\[
R_{avg} = \frac{N_r}{2} \left( \frac{1}{N_0} + \frac{1}{N_t} \right) \tag{9}
\]

where \( N_0 \) and \( N_t \) are the number of detected corners in the original and test images, respectively and \( N_r \) is the number of repeated corners between them (within maximum 3 pixels error [3]).

\[
L_e = \sqrt{\frac{1}{N_r} \sum_{i=1}^{N_r} [(x_{oi} - x_{ti})^2 + (y_{oi} - y_{ti})^2]}, \tag{10}
\]

where \( (x_{oi}, y_{oi}) \) and \( (x_{ti}, y_{ti}) \) are the original (reference) and test positions of the \( i \)-th repeated corner.

We also compare the efficiency of the detectors by measuring running time [20] which is usually averaged for a corner detector over a large number of executions on different images on the same machine. The running time presented in Section IV-C2 indicates the average corner detection time per image, excluding the time for edge extraction and selection step (presented in Section II-A1).

B. Test Database

The database had a total of 25 different original \( 512 \times 512 \) grey-scale images including some artificial images like ‘Block’ and real world images like ‘Lena’, ‘Leaf’, ‘House’ and ‘Lab’. Many of the above original images were collected from standard databases [21], [22] and can be found with their detected corners in [23]. The database had a total of 9450 transformed (test) images, which were obtained by applying the following seven approaches of attacks on each original image:

- **Rotation**: 18 different angles \( \theta \) in \([-90^\circ, +90^\circ]\) at 10° intervals, excluding 0°;
- **Uniform (U) scale**: scale factors \( s_x = s_y \) in \([0.5, 2.0]\) at 0.1 intervals, excluding 1.0;

\(^3\)The authors of this paper are grateful to the authors of He & Yung [8] and Zhang et al. [11] for providing their code.
Non-uniform (NU) scale: scale factors $s_x$ in $[0.7, 1.5]$ and $s_y$ in $[0.5, 1.8]$ at 0.1 intervals, excluding the cases when $s_x = s_y$.

Combined transformations (rot.-scale): $\theta$ in $[-30^\circ, +30^\circ]$ at 10° intervals, excluding 0°, followed by uniform or non-uniform scale factors $s_x$ and $s_y$ in $[0.8, 1.2]$ at 0.1 intervals;

Lossy JPEG compression: compression at 20 quality factors in $[5, 100]$ at 5 intervals;

Gaussian (G) noise: white Gaussian (G) noise with a mean 0 at 10 variances in $[0.005, 0.05]$ at 0.005 intervals; and

Shearing: shear factors $sh_x$ and $sh_y$ in $[0, 0.012]$ at 0.002 intervals, excluding $sh_x = sh_y = 0$.

Therefore, the database had a total of 450 rotated, 375 uniform scaled, 2925 non-uniform scaled, 3750 rotated and scaled transformed images. It also had 500 JPEG compressed, 250 Gaussian noise induced and 1200 sheared images. Note that transformations comprising rotations were also followed by cropping that removed the outer black parts. Consequently, many detected corners in the original images were cropped off in the test images for the transformations involving rotations.

C. Experimental Results and Discussion

This section describes two main parts of the experimental results. It first summarizes the experimental results for setting values for $\sigma_1$ and $\sigma_2$. The comparative results between the proposed improved CPDA and other detectors are then presented.

1) Summary of selecting $\sigma_1$ and $\sigma_2$: We set low to medium values (1.0 to 3.0 at 0.5 intervals) for $\sigma_1$ and medium to high values ($\sigma_1 + 1$ to 5.0 at 0.5 intervals) for $\sigma_2$.

Fig. 4 shows the average repeatability and localization error for the improved CPDA detector with different combinations of $\sigma_1$ and $\sigma_2$. We see that average repeatability increases and localization error decreases with the increase of $\sigma_2$. However, at high $\sigma_2$ values the detector missed some true corners. Considering better average repeatability and localization error, we selected $\sigma_1 = 3.0$ and $\sigma_2 = 4.0$.

Note that while $\sigma_1$ provides good localization error, $\sigma_2$ ensures the expected robustness level for a particular application.

2) Comparative results: In this section, we first compare the robustness of different detectors and then compare their efficiency.

Robustness: Fig. 5 shows the comparative results. The CPDA detector performed the best among all the detectors in terms of both average repeatability rate and localization error. The same performance was observed in [5]. The proposed improved CPDA detector (fast CPDA) offered slightly lower average repeatability and higher localization than the original CPDA detector. The reason is that the search space for corners is significantly reduced by the proposed improved detector, which still offered better robustness than all the existing CSS-based detectors.

Efficiency: Table I compares the running time of different detectors on a Windows XP machine with 3.00GHz of Intel(R)

<table>
<thead>
<tr>
<th>Detectors</th>
<th>Running time (sec.)</th>
</tr>
</thead>
<tbody>
<tr>
<td>CSS [10]</td>
<td>0.0084</td>
</tr>
<tr>
<td>ARCSS [3]</td>
<td>0.0156</td>
</tr>
<tr>
<td>He &amp; Yung [8]</td>
<td>0.0090</td>
</tr>
<tr>
<td>MSCP [12]</td>
<td>0.0056</td>
</tr>
<tr>
<td>Zhang [11]</td>
<td>0.0146</td>
</tr>
<tr>
<td>CPDA [3]</td>
<td>0.0131</td>
</tr>
<tr>
<td>Fast-CPDA</td>
<td>0.0048</td>
</tr>
</tbody>
</table>

Fig. 4. Average repeatability (in percentage) and localization error (in pixels) on different $\sigma_1$ and $\sigma_2$ ($\sigma_2 \geq \sigma_1 + 1$) values for the improved CPDA detector.
In this paper, we have proposed an improvement to the original CPDA detector [5]. The proposed improvement obtains a set of probable candidate points before the CPDA curvature estimation. The maxima of the distance function of two Gaussian smoothed versions of the curve are chosen as the candidate points. The CPDA curvature is estimated on these chosen candidate points only.

We have experimentally shown that the proposed fast-CPDA detector not only runs significantly faster than the original CPDA detector, but also offers a similar robustness in terms of average repeatability and localization error. In addition, the fast-CPDA detector is also faster than all the popular detectors chosen in our performance study.

We believe such a fast and robust corner detector can be exploited in various applications including copyright protection [2], image matching [24], photogrammetry [4] and transformed image identification [25].

V. CONCLUSION

Core(TM)2 Duo CPU and 3.23GB of RAM. We see that the proposed improved CPDA-based detector is faster than any of the other detectors used in the comparison and it is more than 2.7 times faster than the original CPDA detector [5]. On the other hand, the ARCSS detector is the slowest detector, since unlike the others it required significant time for affine-length parameterization of the curve.

REFERENCES