Very Fast Ellipse Detection for Embedded Vision Applications

Michele Fornaciari  
SOFTECH-ICT  
University of Modena and Reggio Emilia  
Via Vignolese, 905 - 41125 Modena, Italy  
Email: michele.fornaciari@unimore.it

Andrea Prati  
Dept. of Design and Planning of Complex Environments  
University IUAV of Venice  
Santa Croce 1957 - 30135 Venezia, Italy  
Email:andrea.prati@iuav.it

Abstract—Real-time ellipse detection is an important yet challenging task, since the estimation of the five parameters of an ellipse requires heavy computation. This task is even more challenging when the processing must be done on a mobile device with limited computational power. The typical trade-off between accuracy, efficiency and limited resources of embedded vision programming must be accounted. In this paper we present a novel strategy for edge point selection, which allows to drastically reduce the number of edge points to be evaluated for parameters estimation, making embedded mobile vision applications feasible. Extensive results show the increased efficiency of the proposed method over state-of-the-art ellipse detectors, in synthetic and challenging real images, and in a live mobile application.

I. INTRODUCTION

With the advent of powerful mobile technologies for everyone and the spreading of new generations of smart-phones, the request of new applications running on these devices increased enormously. These devices can now be considered in full as (very) smart cameras which are (or can at least be) distributed as well. In fact, despite their limitations, mobile devices are now powerful enough to permit on-board processing of complex data, including images and videos. By joining processing with location-based information, provided by embedded GPS sensors as well as by GSM-call-based localization and other sensors (accelerometer, gyroscope, etc.), sophisticated tools for entertainment, tourism, monitoring, etc. can be provided.

As a consequence, the scientific community has recently found large interest to image/video processing on smartphones, often called embedded or mobile vision [1]. In addition to the application side, the implementation of computer vision and image processing algorithms on smart-phones carries a scientific value because it poses several challenges [1]: limited battery life (which calls for energy-efficient design of algorithms), limited computational resources (in particular in the case of large-scale visual tasks), limited storage capabilities (retrieval applications which require large amount of stored data must rely on cloud-based or distributed solutions), and how to leverage the rich contextual information to build robust mobile vision systems. As well, mobile applications often require a certain degree of network connectivity, thus the best trade-off between local (on-board) and remote processing must be determined. Given these premises, image analysis algorithms should be re-designed to increase efficiency for mobile vision and to keep as much of the computation as possible on the device, as typical of smart cameras.

In this paper we address the problem of fast ellipse detection on mobile devices. This is a starting point for many computer vision applications, since ellipse shape is very common in nature and in hand-made objects. For instance, ellipse detection has been used in detecting wheels [2], road sign detection and classification [3], object segmentation for industrial applications [4], automatic segmentation of cells from microscope imagery [4], pupil/eye tracking [5], etc..

Given these premises, this paper focuses on the description of a real-time implementation of ellipse detection capable to find very fast the ellipses in images captured by a mobile device. The excellent trade-off between efficiency (in the order of 7 ms per image) and accuracy makes this approach a superb candidate for implementation on mobile devices, leaving to successive tasks enough time to be computed in real time.

II. RELATED WORKS

Many proposals [2]–[4], [6]–[9] for efficient ellipse detection have been published. The basic approach shared by most of the existing methods employs Hough transform (HT) on the edge points. HT applied to ellipses calls for a 5-dimensional parameter space and most of the previous works propose techniques for either reducing the number of votes in the accumulator or decomposing the parameter space to speed up the process [6]. Proposals belonging to the first class can be further divided in those which use probabilistic HT [10], those exploiting symmetry-based detection [6], [7], [11] and those which use random sampling/RANSAC [5], [8], [12], [13].

In parameter space decomposition, the calculations are split into multiple stages processed in series [14], [15]. The decomposition is achieved by exploiting the geometric constraints of the ellipse. For instance, Ito et al. [7] decomposed the parameter space of HT for each parameter (ellipse center, slope and axes), allowing to reduce complexity from $O(N^5)$ to $O(N^2 + NE)$, where $N$ is the size of each dimension in the parameter space and $E$ is the number of edge points. A very common approach (also thanks to its Matlab available implementation) is that proposed by Xie and Ji [15], which employs one-dimensional accumulator array for the minor axis length, exploiting simple geometric constraints. Its complexity
is $O(E^3)$, where $E$ is the number of edge points, but it can be further reduced to $O\left(\frac{E^2}{C}\right)$ (where $C$ is a constant much smaller than $E$), by randomly selecting a subset of the edge points [13].

In order to reduce the complexity, in [9] the edge directional properties are integrated to decompose the parameter space and point pair selection is exploited for computational efficiency. This paper makes use of Absolute Difference Mask (ADM) algorithm [16] to efficiently detect edges, their strengths and directions. Moreover, one crucial contribution of this paper is the removal of spurious edge points by analyzing the curvature convexity. However, this method suffers mainly from error propagation from stage to stage [6].

Other developments have been proposed to improve Randomized HT (RHT). For instance, Lu and Tan [17] proposed an iterative algorithm which limits the influence of noise on the accuracy of RHT. Unfortunately, iterative procedures are typically time consuming and thus unsuitable for real-time implementations on mobile devices.

Finally, several approaches proposed specific rules for speeding up the search of ellipses by efficient edge selection strategies. Cooke [2] proposed solutions based on the progressive merge of line segments to form first circular arcs and then elliptical arcs. Finally, ellipse fitting is applied to the resulting arcs. A similar approach is followed in [3] for traffic sign detection: the different edge fragments are classified based on their possible position within the ellipse (left and right, up and down) and grouped to form potential ellipse by means of genetic algorithms. Even if these approaches share some concepts with our proposal, the reported results in terms of computational efficiency are unacceptable for our scopes, being in the order of seconds per image.

Our approach exploits edge convexity and mutual edge position to discard at an early stage a large amount of points to be evaluated to estimate the ellipse parameters. A selection strategy is applied to edge segments (i.e. connected edge points with the same convexity) rather than single points (as in [9]), thus reducing the number of required comparisons. For the parameters estimation we require 3 points, but instead of computing all possible triplets as in [15] we use only those points whose edges are likely to belong to the same ellipse, drastically reducing the number of false positives and the overall computational time.

III. REAL-TIME ELLIPSE DETECTION ON MOBILE DEVICES

The proposed ellipse detection method finds the ellipses that best fit the edge points computed on the input image. First of all, a pre-processing step is required in order to form edge segments, that are connected edge points sharing the same coarse convexity direction. A selection strategy is applied to identify those edge segments that belong to the same ellipse, and only their points are used for estimating the parameters of that ellipse. In the post-processing phase, multiple detections of the same ellipse are merged, and other clustering criteria may be applied.

The advantages of the proposed method are:
- only combinations of edge segments are checked, in contrast with combinations of single points (as done in previous proposals);
- the selection strategy, detailed in Section III-B, relies on the property of the midpoints of parallel chords of an ellipse, which does not require accurate information of the direction of the edges and is thus well suited to real world images;
- the data collected for the selection strategy are exploited for estimating the center of the ellipse, avoiding two-dimensional accumulators and the problems related to the search for local maxima;
- the actual parameter estimation is computed only on those edges that satisfy the selection strategy constraints; indeed, only 3 one-dimensional accumulators are required;
- the selection strategy does not rely on constraints on axis dimensions, eccentricity, or maximum distance between edges, resulting in a more general-purpose algorithm.

As a result, the proposed algorithm runs faster than all the others state-of-the-art methods, allowing for real time performances even on mobile devices, obtaining also a very high accuracy in ellipse detection.

In the following the proposed method is further detailed. We recall that the ellipse parameters are: $(x_c, y_c)$ as coordinates of the center; $A$ and $B$ as the major and minor semi-axis length, respectively; $\rho$ as the angle between the major semi-axis and the $x$–axis, i.e. the ellipse orientation.

A. Pre-Processing

The grayscale input image is smoothed using a Gaussian filter, and then the edges are extracted using the Canny edge detector, which outputs also the Sobel derivatives $dX, dY$. With reference to Fig. 1(a), the edge points are divided in two main classes: $\{1, 3\}$ and $\{2, 4\}$, given the sign of the slope $(dY/dX)$ of the edge direction.

![Fig. 1. (a) Classes of edge segments representing an ellipse. (b) Simple method to determine the convexity direction: if the area of the region A is bigger than the area of the region B, then the convexity is upward, else is downward.](image)

For each main class, the 8–connected points are labeled together, obtaining edge segments. In order to remove all those edge segments that are not useful for further processing, very short edge segment (noise) and straight edge segments (not belonging to the curved edge of an ellipse) are discarded. Straight edge segments are detected thresholding the axes–ratio of the corresponding oriented bounding box. Within the
same main class, we can discriminate between edges with convexity upward (classes 3, 4) or downward (classes 1, 2). The convexity direction can be easily computed comparing the areas of the regions A and B, as depicted in Fig. 1(b).

B. Edge Segment Selection and Ellipse Detection

For our scopes, an ellipse is found if there is at least a triplet of edge segments \( (e_i, e_j, e_k) \) belonging to the same ellipse. To avoid the computational cost to estimate the parameters for each of the \( \binom{n_S}{3} \) possible triplets (with \( n_S \) representing the number of edge segments), which will also lead to a high number of false positive, a selection strategy is applied.

First, there are obvious correlations between an edge segment of class \( i \), and the classes \( j \) and \( k \) of its following and preceding edge segments. Assuming counterclockwise order and with reference to Fig. 1(a), the classes \( (i, j, k) \) can only be one of \( \{(1, 2, 4), (2, 3, 1), (3, 4, 2), (4, 1, 3)\} \).

Second, recalling that all edge segments of a given ellipse have the convexity toward the center, a simple constraint based on the mutual position of the segments can be applied, as summarized in Table I, which reports the mutual position of a class on a given row with respect to a class on a given column.

<table>
<thead>
<tr>
<th></th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>left</td>
<td>right</td>
<td>above</td>
<td>above</td>
</tr>
<tr>
<td>2</td>
<td>left</td>
<td>right</td>
<td>above</td>
<td>right</td>
</tr>
<tr>
<td>3</td>
<td>below</td>
<td>below</td>
<td>right</td>
<td>left</td>
</tr>
<tr>
<td>4</td>
<td>below</td>
<td>below</td>
<td>right</td>
<td>left</td>
</tr>
</tbody>
</table>

Table I: Mutual positions of classes of edges. The edge of class (ROW) must be (…) the edge of class (COLUMN).

Then, given a triplet \( (e_i, e_j, e_k) \) that satisfies the first two constraints, the center of the ellipse is estimated for the pair of edge segments \( (e_i, e_j) \), and for the pair \( (e_i, e_k) \). If the two resulting ellipse centers \( C_{ij} \) and \( C_{ik} \) are close enough, then the three edge segments are likely to belong to the same ellipse, and the parameters estimation can start. Otherwise, the triplet is discarded.

In order to estimate the ellipse center for a given pair of edge segments the following ellipse property is exploited: the line crossing the midpoints of parallel chords passes through the center of the ellipse, as shown in Fig. 2(a).

Given a pair of edges \( (e_a, e_b) \), with \( (a, b) \in \{(1, 2), (2, 3), (3, 4), (4, 1)\} \), the coordinates of the center \( C_{ab} \) can be computed as follows (see Fig. 2(b):

\[
C_{ab,x} = \frac{M_{b,y} - t_bM_{b,x} - M_{a,y} + t_a + M_{a,x}}{t_b - t_a} \\
C_{ab,y} = \frac{t_aM_{b,y} - t_bM_{a,y} + t_a + t_b(M_{a,x} - M_{b,x})}{t_b - t_a}
\]

where \( P_a, P_b \) and \( H_a, H_b \) are, respectively, the extremes and the middle points of the two considered edge segments, and \( ph_a \) and \( ph_b \) the lines connecting them. Taking the midpoints of all the parallel chords to \( ph_a \) and \( ph_b \) (dotted lines in Fig. 2(b)) we can estimate the lines converging towards the ellipse center. Unfortunately, these midpoints are affected by noise and do not lie on a perfectly-straight line. The slopes \( t_a \) and \( t_b \) of these lines are thus estimated with a variant of the Theil-Sen estimator (Alg. 1), which resulted to be fast and robust. Due to the noise, also exact points \( (M_a \) and \( M_b \) on which these lines pass through are estimated as the points whose \( x, y \) coordinates are the medians of the coordinates of the midpoints of the chords parallel respectively to \( ph_a \) and \( ph_b \).

Algorithm 1 Get the slope of the line best fitting the midpoints of parallel chords.

```java
function GETSLOPE(Point midpoints[])
    middle ← midpoints.length() / 2
    for i ← 0, middle do
        x1 ← midpoints[i].x
        y1 ← midpoints[i].y
        x2 ← midpoints[middle + 1 + i].x
        y2 ← midpoints[middle + 1 + i].y
        slope ← (x2 − x1) / (y2 − y1)
        S[i] ← slope
    end for
    return MEDIAN(S)
end function
```

Given the triplet \( (e_i, e_j, e_k) \) satisfying the above three constraints, the following parameters are now available for the pair \( (e_a, e_b) \):

- \( C_{ab} \): intersection of the lines \( l_a, l_b \);
- \( S_a, S_b \): vectors of slopes computed into the Theil-Sen estimator (see Alg. 1);
- \( M_a, M_b \): median points of the midpoints of the chords parallel to \( ph_a, ph_b \);
- \( t_a, t_b \): slopes of the lines \( l_a, l_b \);
- \( r_a, r_b \): slopes of the lines \( ph_a, ph_b \);

C. Parameters Estimation

As ellipse center coordinates \( x_c \) and \( y_c \) we consider the mean of the computed centers:

\[
x_c = (C_{ij}x + C_{ik}x) / 2 \\
y_c = (C_{ij}y + C_{ik}y) / 2
\]
The value of minor semi–axis 

\[ q = \frac{S_a}{s_y} \]  

and taking the highest value. The value of the 

\[ \rho = \frac{K + \tan(\alpha)}{1/N_+} \]  

where:

\[ \alpha = q_1 q_2 - q_3 q_4 \]  

\[ \beta = (q_3 q_4 + 1)(q_1 + q_2) - (q_1 q_2 + 1)(q_3 + q_4) \]  

\[ K_+ = \frac{-\beta + \sqrt{\beta^2 + 4\alpha^2}}{2\alpha} \]  

\[ N_+ = \sqrt{\frac{(q_1 - K_+)(q_2 - K_+)}{(1 + q_1 K_+)(1 + q_2 K_+)}} \]  

Assigning to \( q_1, q_2, q_3, q_4 \) the values reported in Tab II, the values of \( N \) and \( \rho \) can be estimated in two one–dimensional accumulators and by taking the highest values.

Once \( N \) and \( \rho \) are estimated, the value of the major semi–axis \( A \) is given by:

\[ A = A_x / \cos \rho \]  

where:

\[ x_0 = \frac{(x_i - x_c) + (y_i - y_c)K}{\sqrt{K^2 + 1}} \]  

\[ y_0 = \frac{-(x_i - x_c)K + (y_i - y_c)}{\sqrt{K^2 + 1}} \]  

\[ A_x = \sqrt{x_0^2 N^2 + y_0^2 N^2 (K^2 + 1)} \]  

The value of \( A \) can be estimated in a one–dimensional accumulator considering as \((x_i, y_i)\) every point of the three edge segments, and taking the highest value. The value of the minor semi–axis \( B \) is then:

\[ B = A \cdot N \]  

Once all the ellipse parameters have been estimated, it is possible to assign a measure of accuracy to the detection in the interval \((0, 1)\). This score summarizes how well the points of the edge segments fit the contour of the actual ellipse:

\[ \text{score} = \frac{\# \text{points lying on the actual contour}}{|e_1| + |e_j| + |e_k|} \]  

D. Post–Processing

If an ellipse has more than 3 edge segments which satisfy our constraints, then there will be multiple detections with slightly different parameters of the same ellipse which need to be merged. Therefore, all the detected ellipses with high scores are clustered adopting the technique presented in [19], which allows to assess the similarity of two ellipses comparing the distances between centers, axes length, eccentricity and rotation angle separately, and thus permitting the customization of the weights of each distance in the final response.

IV. Experiments

To prove its efficiency and accuracy, our method is compared against three methods for fast ellipse detection, namely the methods of Xie and Ji [15], its randomized version of Basca [13], and Zhang [9]. These methods have been selected due to their diffusion, declared efficiency and available source code implementations (or clear explanation of the algorithm to allow easy re-implementation). Since the methods in [15] and [13] do not provide a pre–processing step, in order to guarantee the fairness of the comparison, all methods start from the same edge mask, and, for [9] and our method only, the same angle of the gradient of the edge points.

All the methods have been tested on both a synthetic dataset and two challenging real datasets. The synthetic dataset consists of a total of 1000 images of resolution 800x600, containing 10 sets of 100 images each with an increasing number (from 1 to 10) of randomly generated (i.e. with different size, orientation and position) ellipses, entirely contained in the image and possibly overlapped. The first real dataset consists in 43 images freely downloaded from Internet (mainly Google Images - with a resolution ranging from 194x259 to 1600x1200) and from a mobile phone camera with resolution 800x600 representing traffic signs, indoor and outdoor scenes, wheels, coins etc.. These images have been manually annotated for evaluation purposes. The second real dataset is taken from one of the datasets used in [5] and consists in 150 annotated images of size 620x460. Because of the different score strategies, it was impossible to provide a meaningful score threshold \( th_s \) for all the methods. Therefore, only the best result varying \( th_s \) for each image is considered for each method. For evaluation purposes, first the ellipses scoring above \( th_s \) are clustered with an approach similar to [19], keeping as cluster center the highest score ellipse; then, each cluster center is compared against every Ground Truth ellipse, given an Hit (True Positive) if both are in the same cluster, or a Miss (False Positive) otherwise. The result of this evaluation is the F–measure:

\[ \text{F–measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \]
The accuracy on the synthetic dataset is depicted in Fig. 3, and the computational time for each method is given in Fig. 4(a), with a zoomed view in Fig. 4(b) to better appreciate the difference. Computational time has been computed on a PC with 6GB of RAM and an Intel Core i7–950 processor.

As expected, the exhaustive search of Xie and Ji [15] guarantees good average accuracy (Fig. 3(d)) (which however worsens as the number of ellipses per image increase) at the cost of a very high computational time (see Fig. 4(a)). Its randomized version (Basca et al. [13]) is much faster (on average, less than 1.7 sec wrt about 82 sec per image), but loses in accuracy (Fig. 3(c)). The points selection strategy in [9] speeds up the process (less than 1 sec per image, but still not enough for our purposes), but some weaknesses in the method (like the high number of constraints among the points and the search for local maxima in 2–D accumulators) affect the overall accuracy (Fig. 3(b)). Our method achieves the best detection accuracy (Fig. 3(a), average F measure of 93.64% wrt to Xie and Ji’s method with 85.54%), with a considerably faster computational time (in the order of few milliseconds).

In fact, the edge segments selection strategy allow to avoid computation for the edge segments which do not belong to an ellipse, reducing the number of false positives and increasing the run–time performance.

<table>
<thead>
<tr>
<th>First dataset</th>
<th>Second dataset</th>
</tr>
</thead>
<tbody>
<tr>
<td>F-measure</td>
<td>Time</td>
</tr>
<tr>
<td>Ours</td>
<td>68.37% 60.24 msec</td>
</tr>
<tr>
<td>[9]</td>
<td>21.25% 768.37 msec</td>
</tr>
<tr>
<td>[13]</td>
<td>20.25% 820.26 msec</td>
</tr>
<tr>
<td>[15]</td>
<td>22.85% 640651.91 msec</td>
</tr>
</tbody>
</table>

| TABLE III | RESULTS IN TERMS OF ACCURACY AND COMPUTATIONAL TIME ON THE TWO REAL DATASETS. |

The same evaluation has been performed on the real datasets and the results are shown in Table III. Due to the high variability in resolution the average computational time on the first dataset is higher than in the synthetic dataset, but our approach still achieves the fastest results and significantly outperforms the other methods in terms of accuracy. In the case of the dataset for eye tracking, the pupil is only a very small portion of the image and the resulting edge points are very few. This is the reason of much lower computational times for all the methods. In this case, due to the generally-simpler images which contains only one ellipse, the exhaustive method proposed in [15] achieves higher accuracy, even though with a 300x time wrt our approach. Some results on the first real dataset are shown in Fig. 5.
This paper proposes a very fast ellipse detection method, able to run in real time thanks to an accurate selection of the points to be evaluated for the estimation of the ellipse parameters. The comparison with other state-of-the-art methods shows that our proposal has the highest accuracy, comparable only with [15] on images with a few ellipses, and is much faster. These features allow to run the proposed method on a mobile device for real-time ellipse detection.

V. CONCLUSIONS

Finally, our method has been also implemented in C++ on an Samsung Galaxy S2 equipped with Android 2.3, and called via Java Native Interface calls. The computational time (averaged on 3 video sequences of 1000 frames of size 800x440 each), comprising JNI call, ellipse detection and final clustering is 112,23 msecs, which still allows the processing of about 10 frames per second on the mobile device.

REFERENCES