Optik 127 (2016) 232–238

Contents lists available at ScienceDirect

Optik

journal homepage: www.elsevier.de/ijleo

An efficient curve detection algorithm

Lianyuan Jiang[a,b,c], Yongqiang Ye[a,*], Guili Xu[a]

[a] College of Automation Engineering, Nanjing University of Aeronautics and Astronautics, Nanjing 210016, China
[b] School of Computer, Guangxi University of Science and Technology, Liuzhou 545006, China
[c] Guangxi Experiment Center of Information Science, Guilin 541004, China

ARTICLE INFO

Article history:
Received 27 December 2014
Accepted 11 October 2015

Keywords:
Curve detection
Randomized Hough transform
Random sampling
Candidate curve
Evidence collecting

ABSTRACT

This paper presents a highly-efficient algorithm for the detection of lines, circles, and ellipses. The algorithm uses a novel method for the determination of candidate curves, and no accumulator is needed to save the information of the related parameters. We first define a curve parameter by random sampling and then calculate the amount of points on the curve after selecting several test points in the test area. If the amount of points on this curve is larger than a certain threshold, the curve is then considered as a candidate curve. Afterwards, the evidence-collecting process will be used to further affirm whether this candidate curve is true or not. A large number of experiments have shown that this new method has the advantage of a faster speed of about one to two orders of magnitude compared with the randomized Hough transform algorithm. Besides, it also has a higher detection accuracy and strong robustness.

1. Introduction

The detection of target curves from digital images is the key to various high level tasks in the field of computer vision and optical engineering [1]. The well-known Hough transform is a usual approach for detecting parametric curves like lines, circles, ellipses, etc from a predefined planar set of points [2–4]. This approach has its advantages of the insensitivity to noise, the convenience of realization in parallel computing and the high efficiency in straight line detection. The detection of curves which involves more parameters, however, has higher requirements for computing time and storage space. As a result, various improved versions of Hough transform were proposed [5–10]. Fischler and Bolles [5] proposed the RANSAC algorithm which is a generate-and-test method: n randomly selected points determine a possible curve parameter, and then it is tested by other points in the image. Kiryati et al. [6] proposed the Probabilistic Hough Transform in which a randomly selected subset from edge point set is used as an input for the Hough transform to improve the detection efficiency. Cheng et al. [8] presented a highly-efficient Hough transform through the particle swarm optimization technique.

To largely reduce the computing time and storage space in Hough transform, Xu et al. [11,12] proposed randomized Hough transform (RHT), which improves the efficiency of curve detection in computing time and memory requirements through random sampling, dynamic link-list data structure for the parameters, and many-to-one mapping, etc. However, the reliability of RHT would be highly reduced concerning complicated images or those with high noise ratio. Some improved RHT were then suggested to alleviate this problem [13–18]. Walsh and Raftery [14] identified curves in an easier way through the specifying of a target distribution and the weighing of the sampled parameters. To improve robustness and accuracy as well as to speed up the detection, Ji and Xie [15] made the analysis of the error propagation and the sampling of the points that are most probably located on the curve, leading to the curve estimation of highest accuracy. Jiang [18] optimized the approaches to the determination of sample points and candidate circles through probability sampling and feature points, largely improving the detection speed.

To avoid the computing time and memory space consumed in parameter accumulation in RHT, Chen and Chung [19] proposed the randomized circle detection algorithm (RCD) in which four points were randomly chosen from the set of edge points and the distance between any two was longer than T_d. A circle parameter can be determined by any three points. If the remaining point is proved to be on the same circle determined by the parameter, this circle can be affirmed for true by the evidence-collecting process. Chung and Huang [20] used a pruning-and-voting strategy to speed up the detection of curves. Chung et al. [21] speeded up the randomized approach with the strategy of multiple-evidence-based sampling and improved the detection accuracy with an efficient refinement strategy.

In this paper, we propose a highly-efficient curve detection algorithm which includes the detection of straight lines, circles, and

* Corresponding author. Tel.: +86 25 84892305 6020; fax: +86 25 84892368.
E-mail address: MelvinYe@nuaa.edu.cn (Y. Ye).

http://dx.doi.org/10.1016/j.ijleo.2015.10.063
0030-4026/© 2015 Elsevier GmbH. All rights reserved.

ellipses. In the algorithm, we obtain, by random sampling, a curve parameter which does not necessarily correspond to a true curve. However, we can fast rule out invalid curve parameters simply by the strategy of selecting several points from a particular area. Each one of the remaining parameters can determine a candidate curve which will be further affirmed for a true one through evidence-collecting process. The experimental results have shown that the proposed algorithm has such features as high speed, high accuracy, and strong robustness.

The rest of this paper is organized as follows. Section 2 provides a brief introduction of RHT curve detection. Section 3 displays the efficient method for curve detection proposed in this paper while in Section 4 experimental results are shown to prove this algorithm. Conclusions and discussions are drawn in Section 5.

2. RHT algorithm for curve detection

Major problems of the standard Hough transform are its long computation time and large storage space requirements. The former is caused by the fact that more than a few curves are calculated from any single point in the image and the accumulator array needs to store the unit information related to these curves. Whereas the latter is a greater problem as the size of the accumulator array can grow exponentially with the number of parameters [14]. These problems can be addressed through RHT method using random sampling of edge points, and many-to-one mapping from the image space to the parameter space.

For a binary image after edge detection, we use the term "points" to specifically refer to edge pixels (black) in all the examples presented here. The RHT algorithm is outlined as follows for the general case of detecting all instances of a particular curve with n parameters in a binary image.

Step 1. Construct edge point set D, initialize $P = null$ and $f = 0$ (P is the link-list for parameters and f is the sampling times).

Step 2. Randomly sample n different points $d_1, \ldots, d_n$ from D, and calculate the curve parameter p determined by those points.

Step 3. Search among P for such an element p_s that satisfy $p_s \approx p$. If p_s is found, switch to step 5; otherwise, go to step 4.

Step 4. Insert p into P as a new element, set its score as 1 and go to step 6.

Step 5. Increase the score of p_s by one, and check whether the resultant score is smaller than the specified threshold n_t (which is a very small number, like 2 or 3). If yes, go to step 6; if not, switch to step 7.

Step 6. Make $f = f + 1$. Check whether f is larger than the number of the maximum allowable consecutive sampling failures. If yes, stop the process; Otherwise, turn back to step 2.

Step 7. Consider p_s as the parameter for the candidate curve and calculate the number of points on this curve. If it is larger than the threshold $m_{\min}$, go to step 8. Otherwise, p_s is assumed as the parameter of a false curve and needs to be removed from P; make $f = f + 1$ and go back to step 2.

Step 8. A true curve represented by p_s has been detected; delete the points on this true curve from D. Reset $P = null$, $f = 0$ and go back to step 2.

For more details about the guidelines of stopping on the RHT algorithm, please refer to Ref. [12].

3. The proposed curve detection algorithm

3.1. The accumulator for related parameters in RHT

Suppose that n different points from each sampling can just determine one curve parameter. In RHT, the curve parameter is determined by n different points from random sampling, which means that those n points in the image space are mapped to be one point in the parameter space. These calculated parameters need to be accumulated in the accumulator which is actually a link-list data structure. Take lines for example, the structure of the link-list for the parameters is shown in Fig. 1. If the linear equation is $y = ax + b$, the parameter link-list P will consist of a serial of elements as shown in Fig. 2 with one line parameter (a, b) and its score (*score*) in every element. When creating P, elements are inserted in order of the size of parameters for quick searching.

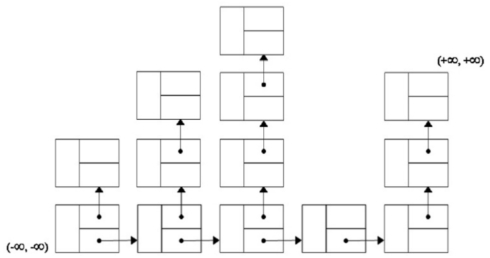

Fig. 1. A particular example of the link-list data structure P.

The specific accumulation process of parameters in the link-list is as follows: search first in the link-list for a parameter element which is approximately equal to the newly-generated parameter. Increase its score by one if it can be found; if not, insert it as a new element into the link-list in a proper place. When the score of some parameter in the link-list reaches n_t, a candidate curve can be determined by this element.

The introducing of the link-list in RHT is to find candidate curves. The aimless random sampling, however, generates so many elements that more storage space would be required for the increased link-list. Moreover, one searching in the link-list is necessary for each sampling which costs a large amount of computing time.

3.2. Definitions of the test area and test points

Definitions of the test area and test points are going to be specified in this section for illustrations of the proposed algorithm. In this paper, the test area refers to a rectangle or the difference operation of two rectangles for fast detection. When determining whether a point is located in the test area, we only need to make simple comparison between the coordinate values of this point and the boundaries of the rectangle. The test areas of different curves are not the same. The following is going to introduce the test areas of three kinds of curves (lines, circles, and ellipses) as well as the methods about determining whether a point is located in a test area.

3.2.1. The test area and test points in line detection

Between the line, determined by any two points, and the boundaries of an image, there are always two intersection points, by whose coordinates, a unique rectangle or line segment (when the abscissas or ordinates of these two points are the same) can be determined. Two cases can be partitioned by judging whether these two intersection points are on the adjacent boundaries.

First, as is shown in Fig. 3, points A and C are the two intersection points between line l and the image boundaries, θ is an angle formed

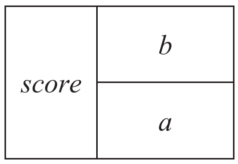

Fig. 2. The structure of element.