Computational Comparison of Voting-based and Arragement-based Schema for Digital Line Detection

Tetsuo Asano [1] **Yasuyuki Kawamura**

School of Information Science,
Japan Advanced Institute of Science and Technology
Asahidai, Tatsunokuchi, 923-1292 Japan
{t-asano,kawamura}@jaist.ac.jp

1 Introduction

The problem of detecting all possible digital line components contained in a given binary edge image is one of the most fundamental problems in pattern recognition and a great number of algorithms have been proposed under the name of Hough Transform [3, 4]. Most of them are based on the voting technique on a subdivided parameter space.

Our basic idea is as follows: When we parameterize a line passing through an edge point by the angle θ and the distance ρ to the line from the origin, the edge point is mapped to a sinusoidal curve in the (ρ, θ) parameter space. The intersection between two such curves corresponds to the line passing through the edge points. Thus, what we have to do is to compute intersections where a number of curves intersect. However, Since edge points are located on integer grids, it rarely happens that many edge points lie exactly on a line. We want to detect a set of edge points which lie roughly on a line. For this purpose we partition the parameter plane into small regions called buckets by axis-parallel lines, and for each such bucket we count the number of lines which pass through the bucket region. If the number of such lines exceeds a predetermined threshold, we report a digital line component corresponding to the bucket region.

In computational geometry the dual transform between points and lines is more common than the above-described transform. It transforms a point (a, b) into a line $y = ax + b$ and a line $y = -cx + d$ into a point (c, d). Again, an intersection between two lines for two edge points corresponds to a line which passes through these two edge points. Thus, if we enumerate all intersections at which many lines intersect, we can detect all digital line components.

This voting technique is easy to be programmed and also easy to be tuned for practical use. This is why the Hough transform is commonly used. However, there are many problems to be resolved. For example, an optimal way of partitioning the parameter plane is somewhat strange and not good for efficient computation.

In this paper we analyze the voting technique based on the Hough transform from a standpoint of computational complexity following a mathematical definition of a digital line component to be detected. We also present a new algorithm for detecting line segments with two endpoints considering their point density.

[1] This work was partially supported by Grant in Aid for Scientific Research of the Ministry of Education, Science and Cultures of Japan.

2 Standard Hough Transform

We first describe a standard algorithm based on the Hough transform [3, 4]. For an edge point (x_i, y_i), let ρ be the distance from the point to a line of angle θ passing through it. Then, the equation for the line is given by

$$\rho = x_i \cos\theta + y_i \sin\theta. \tag{1}$$

In the Hough transform, the above equation is considered as a mapping from an edge point (x_i, y_i) to a sinusoidal curve $\rho = x_i \cos\theta + y_i \sin\theta$ on the $\rho\theta-$parameter plane. An important property is that two parameters ρ and θ determining the line passing through two edge points is given by the two parameters (coordinate values) of the intersection between two curves for the corresponding points. Based on the property, the problem of finding digital line components in an image plane can be reduced to that of detecting intersections of a number of lines.

If a number of edge points lie exactly on a line, then those corresponding lines meet exactly at one point in the dual plane. However, since they are on integer grids, their intersections are scattered in a region. A common technique to absorb this effect is to partition the dual plane into small regions called buckets and to use voting along lines on them. An advantage of the Hough transform is that the angle and distance are both limited to intervals of constant-length, $[0, \pi]$ and $[0, \sqrt{2}N]$, respectively.

Usually, the rectangular region in the dual plane is partitioned into $O(N \times N)$ buckets by axis-parallel lines, where N is the length (number of pixels) of one side of an input image. Then, for each edge point (x_i, y_i) we follow the curve $\rho = x_i \cos\theta + y_i \sin\theta$ to put a vote into buckets intersected by the curve at variable θ values. When we complete the voting process for every edge point, we enumerate all the buckets whose counts of votes exceed a predetermined threshold and then report those lines corresponding to those buckets. This is the basic idea of the Hough transform.

The original form of the Hough transform described above has several problems for practical use. For example, it is not known how to choose a threshold value. To remove redundancy it is desirable not to report non-maximal digital line components. The Hough transform is not designed to find line segments with endpoints but infinite lines. Then, we need some definition of digital line segments probably based on another definition of density of edge points along lines.

3 Definition and Characterization of Digital Lines

We start with the definition of our objects, digital line components. Throughout the paper we assume that lines have slopes between -1 and 1 for simplicity. It is easy to adapt the following discussions for lines of slopes greater than 1 or smaller than -1.

[Definition: Digital Line Components]

A common algorithm to draw a line $y = ax + b$ on a screen puts white dots of distance to the line being at most 0.5. Such a set of grid points is called a digital representation of a line $y = ax + b$ and it is denoted by $G(a, b)$. Then, a set P of edge points for which there exist a and b such that $P \subseteq G(a, b)$ is a (digital) line component. A line component is maximal if addition of any edge point violates the condition.

Now, the problem of detecting line components is described as follows:

[Problem of Detecting Line Components:]

Given a binary edge image of size $O(N \times N)$ containing n edge points and a threshold t, report all maximal line components of size greater than t in the image.

Let P be a set of edge points. If P is a line component, there exist constants a and b such that

$$-\frac{1}{2} \leq y_i - ax_i - b \leq \frac{1}{2} \tag{2}$$

holds for each point $p_i = (x_i, y_i)$ in P. Here remark that our objects are lines with slope between -1 and 1 and thus it suffices to consider the vertical distance to the line.

Lemma 3.1 For any line component P, there is such (a, b) among those parameter pairs characterizing P that a and b are both rational numbers of the forms $\frac{q}{p}$ and $b = y - \frac{q}{p}x \pm \frac{1}{2}$, respectively, where $0 \leq q \leq p \leq N-1$ and $0 \leq x, y \leq N-1$.

4 Voting Technique and Its Limit

Lemma 3.1 says that it suffices to consider $O(N^2)$ different slopes. For y-intercepts, there are $O(N^4)$ possibilities since x and y can take $O(N)$ different values. However, for a fixed slope $O(N^2)$ intercepts are enough. This means that if we make different partitions for each slope then we can save space complexity to $O(N^2 \times N^2)$. Now we have a collection of one-dimensional arrays one for each slope instead of a uniform 2-dimensional array. Using this data structure, we do the voting. That is, for each edge point (x_i, y_i) we compute the value $b = -ax_i + y_i$ for each slope a to put a vote at the bucket defined by (a, b). If we exchange the order of iteration so that the iteration on each slope encloses that on edge points, then we do not need to maintain the whole arrays. When we finish the voting process for a slope, then after reporting maximal buckets exceeding a threshold we can move to the computation for the next slope. Finding maximal buckets is done in $O(N)$ time although we do not go in detail due to space limit.

If our goal is to find all maximal line components, the number of votes required for each slope is not one but many since as many buckets as 1/(vertical width of a bucket) satisfy the inequality (1). So, in general it is required to put votes into consecutive buckets. If we vote one for each bucket, it takes time proportional to the number of votes. For efficient implementation it is desirable to put votes as an interval. It is possible using a data structure dealing with intervals, such as segment trees or interval trees, in which insertion of an interval is done in $O(\log N)$ time. Thus, the overall running time is improved to $O(nN^2 \log N + N^4)$. Although this is a great improvement over the previous method which takes $O(nN^4)$ time, it looks pessimistic to have further improvement without a revolutionary change of idea.

5 An Algorithm Based on Arrangement of Lines

There is a totally different algorithm based on an arrangement of lines in the dual plane. A basic idea is presented by one of the authors[2]. In this paper we refine it.

A basic idea is as follows: An edge point $p = (x, y)$ is contained in a line component $G(a, b)$ when the following inequality hods:

$$-\frac{1}{2} \leq y - ax - b \leq \frac{1}{2}. \tag{3}$$

Rewriting it, we have

$$-ax + y - \frac{1}{2} \leq b \leq -ax + y + \frac{1}{2}. \tag{4}$$

A set of points in the dual plane which satisfy the inequalities above corresponds to a stripe bounded by two parallel lines. In other words, an edge point is mapped to a stripe. When

stripes associated with two edge points have non-empty intersection, for any point (a, b) in the intersection there is a line component for the line $y = ax + b$ which contains these two edge points. Thus, our job is to compute intersections where many such stripes meet.

Drawing the $2n$ lines bounding the stripes in the dual plane, the dual plane is partitioned into small regions called cells. Here note that the cells in the arrangement are exactly the equivalence classes define earlier. That is, for any two point in one cell, their corresponding line components are exactly the same. Thus, it suffices to check all the cells. A naive method to enumerate all the cells takes $O(n^3)$ time and $O(n^2)$ space, but Topological Walk by Asano, Guibas and Tokuyama[1] can visits all of them in $O(n^2)$ time and $O(n)$ space. An advantage of the Topological Walk is that it can visit cells continuously. In other words, the next cell to be visited is one of the cells adjacent to the current cell by an edge. Therefore, if we distinguish upper and lower lines bounding stripes, it is not so hard to check the maximality of a cell.

6 Detecting Line Segments Considering Point Density

One of the disadvantages of the methods described so far is that it is hard to extract line segments instead of infinite lines, because their basic informations are the sizes of line components over the entire image and thus it is not so easy to incorporate informations about endpoints or point density. In the following we show how to detect endpoints of line segments based on point density without increasing the computational complexity.

In the voting schema based on voting, edge points are processed in order. But it was not described in what order they are processed. A basic idea to take point density and endpoints into accounts comes from an appropriate order of edge points. In our method we first determine a sequence of slopes and then for each slope a_i, $-1 \leq a_i \leq 1$ we try to detect line components. For this purpose, we compute the value $v_j = y_j - a_i x_j$ for each edge point $p_j = (x_j, y_j)$ and put a vote into buckets associated with it. If we process edge points in the order of their x-coordinates, then each bucket receives edge points in the same order. So, at each bucket we can easily maintain appropriate informations concerning point density and endpoints of line segments. The details will be included in the final version.

Using those informations we can find a component of a line segment for which the point density is greater than a threshold at any interval on the line segment and gaps between consecutive edge points are within some threshold value. If there is any violation on the point density or gaps, we should report a set of edge points enumerated so far at the bucket if the count is greater than a threshold. In this way we can detect components of line segments with two endpoints without any increase the computational complexity of the algorithm.

References

[1] T. Asano, L. Guibas, and T. Tokuyama, "Walking in an Arrangement Topologically," Int. J. of Comput. Geom. and Appl., 4, pp.123-151, 1994.

[2] T. Asano and N. Katoh: "Variants for the Hough Transform for Line Detection," Computational Geometry: Theory and Applications, 6, pp.231-252, 1996.

[3] R. O. Duda and P.E. Hart: "Use of the Hough Transformation to Detect Lines and Curves in Pictures", Comm. of the ACM, 15, January 1972, pp.11-15.

[4] P. V. C. Hough: "Method and Means for Recognizing Complex Patterns", U.S. Patent 3069654, December 18, 1962.