IEICE TRANS. FUNDAMENTALS, VOL.E84–A, NO.5 MAY 2001

PAPER *Special Section on Discrete Mathematics and Its Applications*

On Detecting Digital Line Components in a Binary Image

Tetsuo ASANO[†a)], **Koji OBOKATA**[†], *and* **Takeshi TOKUYAMA**[††], ***Regular Members***

SUMMARY This paper addresses the problem of detecting digital line components in a given binary image consisting of n black dots arranged over $N \times N$ integer grids. The most popular method in computer vision for this purpose is the one called Hough Transform which transforms each black point to a sinusoidal curve to detect digital line components by voting on the dual plane. We start with a definition of a line component to be detected and present several different algorithms based on the definition. The one extreme is the conventional algorithm based on voting on the subdivided dual plane while the other is the one based on topological walk on an arrangement of sinusoidal curves defined by the Hough transform. Some intermediate algorithm based on half-planar range counting is also presented. Finally, we discuss how to incorporate several practical conditions associated with minimum density and restricted maximality.

key words: *algorithm, computer vision, computational geometry*

1. Introduction

One of the most fundamental tasks in pattern recognition on images is to detect lines in binary images resulting after edge detection. In fact, a great number of works have been proposed for the purpose. Among them, most popular is the one based on so-called Hough Transform. It originated in the patent application by Hough in 1962 [8]. His main idea was to parameterize a set of all possible lines passing through each edge point (a black pixel on a sharp edge) using two parameters and to detect digital line components by voting on the dual plane. Although Hough used the duality transform between points and lines, Duda and Hart [7] later claimed advantage of using another transformation of points into sinusoidal curves defined by angles and perpendicular distances to lines. Nowadays a group of these methods are generally referred to as Hough Transform [6], [9], [11], [13], [15].

There have been a number of approaches to improve the practical performance of Hough Transform. However, it does not seem that there is any essential improvement, mainly because the quality of output is usually evaluated based on human perception without giving explicit definition for the goal, digital line components. Another difficulty comes from their basic algorithmic technique, voting.

We start with a formal definition of a line component to be detected and present several different algorithms depending on how faithful to the definition. The most faithful way is to transform each black point to a pair of sinusoidal curves in the dual plane and examine all of the cells in the arrangement of such curves using Topological Walk. Then, we can find all possible line components satisfying the formal conditions. The other extreme is the conventional algorithm based on voting on the subdivided dual plane since outputs are optimized only on discrete values of angles and distances. Some intermediate algorithm based on half-planar range counting is also described.

Finally, we discuss how to put additional constraints that are necessary in practical line detection applications: First, we include density condition as a constraint, i.e., each line segment must be dense everywhere. In practice, it is very important to output only lines with dense segments; however, density of line segments has not been characterized well in the literature. Second, although Hough Transform detects only infinite lines, we sometimes want to output dense line segments with endpoints or the union of dense line segment components (as a point set). Finally, to avoid redundancy, we want to report only maximal components or *restricted maximal* components (see Sect. 6.5 for the definition).

Our algorithm can output all restricted maximal line components satisfying the threshold condition in $O(n^2)$ time and $O(n)$ working space. Note that the computational complexities do not depend on the image size N but only on the number n of edge points. We can attain the density condition by increasing the time complexity to $O(n^2 \log n)$. Our basic tools are Hough Transform (without voting) and topological walk on an arrangement of pseudolines together with an efficient dynamic data structure maintaining intervals.

There are some theoretical works [3], [4] based on the linear duality transformation, in which the width of a line component is defined by the L_∞ metric. This approach has a disadvantage that actual widths are different for small and large slopes of lines. Compared to that, our methods can consider the Euclidean width (thickness), which is independent on the slope

Manuscript received August 22, 2000.
Manuscript revised November 7, 2000.

[†]The authors are with the School of Information Science, Japan Advanced Institute of Science and Technology, Tatsunokuchi, Ishikawa-ken, 923-1292 Japan.
[††]The author is with the Graduate School of Information Sciences, Tohoku University, Sendai-shi, 980-8577 Japan.
a) E-mail: t-asano@jaist.ac.jp

and much more natural in computer vision applications. However, we have to pay for it by dealing with an arrangement of pseudo parabolas (i.e. each pair of curves intersect each other at most twice) instead of lines. To overcome this difficulty, we take advantage of the fact that output line components should be skinny; this is contrasted to the works utilizing fatness [1] of objects in computational geometry.

2. Digital Line Component

Let G be an $N \times N$ integer grid in the Euclidean plane $\mathbf{R^2}$, or G is defined to be a set of those $N \times N$ grid points. We consider as our input a binary image resulting after edge detection which contains n black points (pixels) among $N \times N$ points. Let P be the set of those n black points. Thus, $n \leq N^2$.

Given a binary image, we want to detect all line components. In this paper we define a line component to be a set of black points. Due to integrality of coordinates of pixels it rarely happens that many points lie exactly on a line. Thus, we should define a line component to have some width. More precisely, given a line l in $\mathbf{R}^2$, we consider the tube $tube_r(l) = \{x \in \mathbf{R}^2 : dist(x, l) \leq r\}$ around l of width $2r$; Here $dist(x, l)$ is the Euclidean distance from x to l. Then, a line component is a set of black points (a subset of P) that are contained in the tube of some line l.

If the distance from the origin to the line l is ρ, there is a unique angle θ such that the point $(\rho\cos\theta, \rho\sin\theta)$ is on the line; Moreover, l can be represented by an equation $\rho = x\cos\theta + y\sin\theta$. This equation can be regarded as a transformation from a point (x, y) in the (x, y)-plane into a sinusoidal curve in the (θ, ρ)-plane.

For technical simplicity, we only consider lines such that $0 \leq \theta \leq \pi/4$ since remaining seven cases are treated similarly or symmetrically. The slope of the line is $\tan(\theta - \pi/2)$, and hence less than -1. We sometimes further refine the angle interval $[0, \pi/4]$ into at most n *angle buckets*. We also assume that $\rho > r$, since we can always translate G so that this condition holds. Then, $tube_r(l)$ is the region in the (x, y)-plane bounded by two lines

$$\rho - r = x\cos\theta + y\sin\theta \quad \text{(called } \textit{far boundary}\text{) and}$$
$$\rho + r = x\cos\theta + y\sin\theta \quad \text{(called } \textit{near boundary}\text{).}$$

Refer to Fig. 1 for pictorial illustration.

A subset Q of P is a (*digital*) *line component* of *thickness* r if Q is a set of black points contained in the tube of a line l, that is, $Q = tube_r(l) \cap P$ for a line l. It is often required to extract (digital) line components satisfying some optimization conditions efficiently. As far as the authors know, the thickness r is usually assumed to be some constant, say 0.5 or 1. In this paper r is not required to be a constant. Theoretically r can

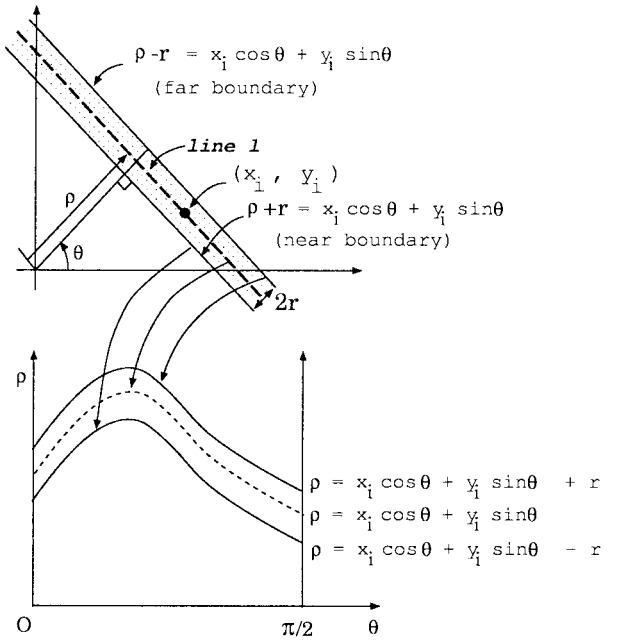


Fig. 1 Definition of $tube_r(l)$ for a line l passing through a point $p_i = (x_i, y_i)$ in the (x, y)-plane and that of ribbon $R(p_i)$ in the (θ, ρ)-plane.

be $O(n)$, but it is reasonable to assume $r = O(\sqrt{n})$ for most practical applications since the side length of an image containing n edge points is $\Omega(\sqrt{n})$. This assumption is in fact important to achieve computational complexities of the algorithms described in the latter half of the paper.

Important definitions and notations are summarized below for readability.
1: G**:** $N \times N$ integer grid or a set of those grid points.
2: P**:** a set of n black points.
3: $tube_r(l)$**:** the region bounded by two lines parallel to a line l separated by distance $2r$, or $tube_r(l) = \{x \in \mathbf{R}^2 : dist(x, l) \leq r\}$.
4: $\rho = x\cos\theta + y\sin\theta$**:** an equation of a line passing through (x, y) such that its normal line forms an angle θ with the positive x-axis and the distance from the origin to the line is ρ.
5: a line component: a set of black points contained in the tube of some line l.

3. Conventional Algorithms for Line Detection

A naive algorithm for detecting line components in a binary image is to check $O(n^2)$ lines passing through pairs of black points. This algorithm is not only slow but also incomplete in the sense that some line components may be missed since the center line of a tube defining a line component is not always defined by a line passing through two black points.

The first algorithm for line detection was invented by Hough [8] with two algorithmic ideas. One is the duality transform between points and lines and the

other is voting technique on the dual plane. Later, Duda and Hart [7] claimed advantage of using another transformation of a point (x_i, y_i) into a sinusoidal curve $\rho = x_i \cos\theta + y_i \sin\theta$, where θ is an angle of the normal vector to a line passing through the point and ρ is the distance from the origin to the line. Nowadays the latter transform is generally referred to as Hough Transform.

In both cases an intersection of two lines or curves in the dual plane corresponds to the line passing through the two points. Due to this property our task is to find busy intersections in the dual plane. Unfortunately, due to integrality of coordinates many points rarely lie on a line. In fact, a line we can recognize in a binary image is a sequence of black points (on grids) which lie sufficiently close to a line. This leads to an idea to subdivide the dual plane into small cells (rectangular regions) to find those cells intersected by many lines or curves. This subdivision corresponds to discretization of ρ and θ values.

Recall that our goal here was to detect line components of thickness $2r$. Obviously the height (length in ρ) of each cell must be at most $2r$. If the height is at most $2r/k$, points on a line of width $2r$ are scattered over k different cells. Thus, if we don't want to miss any line component, we need the sum of counts in k consecutive cells of the same θ value. This is an easy task by linear scan.

4. Algorithm without Discretization of ρ

As described above, the conventional method is characterized by discretization of both of ρ and θ. We could devise an algorithm without discretization of ρ.

Let $(0 = \theta_0, \theta_1, \ldots, \theta_M = \pi/4)$ be a sequence of discretized θ values (see Fig. 2). For each θ_j we compute $\rho_i^{(-)} = x_i \cos\theta_j + y_i \sin\theta_j - r$ and $\rho_i^{(+)} = x_i \cos\theta_j + y_i \sin\theta_j + r$ to define an interval $[\rho_i^{(-)}, \rho_i^{(+)}]$. Then, the problem is to find intervals that are maximal and are intersected by more than some predetermined number of intervals. Each such interval must be of length at most $2r$. So, if we take the center point of such an interval, its dual gives us an equation of a corresponding line.

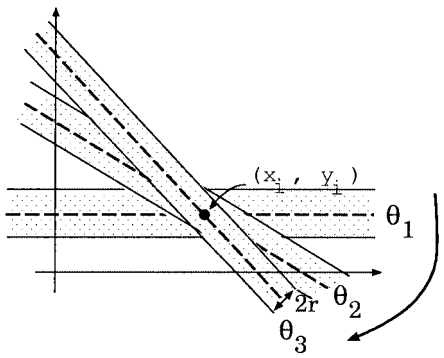


Fig. 2 Discretization of θ values.

Fixed a θ value, this is done in $O(n \log n)$ time by sorting $2n$ values $\rho_1^{(-)}, \rho_1^{(+)}, \ldots, \rho_n^{(-)}, \rho_n^{(+)}$ computed. If we assume that M, the number of different θ values is $O(\sqrt{n})$, then the overall time complexity is $O(n^{1.5} \log n)$. Essentially the same algorithm was proposed as "sorting verification" in [15].

The algorithm above could be improved by randomization. First of all we choose n_r black points randomly out of n black points. Then, we implement the same procedure as above with $M = O(\sqrt{n_r})$. Note that the counts obtained depend on random samples. For safety we should check candidate intervals found in this manner using original point data. Range counting scheme serves as a basic tool for efficient check in this direction. That is, for each candidate interval of length $2r$, we check how many black points are contained in the associated region bounded by two parallel lines defined by endpoints of the interval. Such counting is done in $O(\sqrt{n})$ time using linear space if we use hierarchical cutting by Matoušek [12]. If we choose the size n_r of random samples to be $n^{2/3}$ and the discretization size of θ values to be $O(\sqrt{n_r})$, then candidate enumeration can be done in time $O(n_r^{1.5}) = O(n)$. So, if the number of candidate intervals is $O(\sqrt{n})$, then the overall time complexity is $O(n)$, except the construction time of the range counting data structure, which is $O(n^{1+\delta})$ for small δ.

The size of random samples has been chosen so that the overall time complexity becomes almost linear. However, the size $O(n^{2/3})$ is quite reasonable from an experimental view. It may be easily understood that $O(\sqrt{n})$ random samples are not good enough since it means only constant number of samples are taken in each column or row. With $O(n^{2/3})$ samples we can still recognize lines.

The arguments above can be summarized as follows: There is an algorithm for detecting digital line components in almost linear time based on random sampling and range counting technique in computational geometry. Unfortunately this algorithm does not guarantee the completeness of the algorithm in the sense that every line component that meets the condition is not detected. This incompleteness may cause no problem in practice for most of practical cases. In this paper we are more interested in theoretical aspect of the problem and so we insist on the completeness.

5. Algorithm without Any Discretization

Let P be a set of black points in a given binary image. For a black point $p_i = (x_i, y_i) \in P$, the set of lines passing through p_i is parameterized by ρ and θ to satisfy that $\rho = x_i \cos\theta + y_i \sin\theta$. Hence, in the (θ, ρ) plane, it becomes a sinusoidal curve f_i. Those sinusoidal curves $f_1, \ldots, f_n$ corresponding to n black

points can be treated just as lines since each pair of them intersect at most once in the range $0 \leq \theta \leq \pi/4$ (it follows from the fact that the line through two points is uniquely determined). Such a curve is called a **pseudoline**. In fact, an arrangement of pseudolines can be treated just in the same manner as that of lines, i.e., topological walk algorithm [2] can be applied to search in an arrangement in the same computational complexity as that for lines.

The discussion above implies that the framework for an arrangement of lines applies to pseudoline arrangement unless the width of a line component is taken into consideration. Then, the next natural question is whether it also applies to line components of some width. To take the width into accounts, for each black point $p_i \in P$ we consider a set of lines the distance from p_i to which is at most r. Fixed an angle θ of lines there are two such extreme lines, the far side and near side with respect to the origin, which are defined as (far side:) $f_i^{(0)}: \rho = x_i \cos\theta + y_i \sin\theta + r$ and (near side:) $f_i^{(1)}: \rho = x_i \cos\theta + y_i \sin\theta - r$. These lines in the (x, y)-plane are sinusoidal curves in the (θ, ρ)-plane. These equations represent lines in (x, y)-plane and sinusoidal curves in the (θ, ρ)-plane. As stated before, points and lines in the (x, y)-plane are mapped into sinusoidal curves and points in the (θ, ρ)-plane, respectively. Thus, a black point $p_i = (x_i, y_i)$ is contained in the tube of a line l characterized by (θ, ρ) if and only if the point (θ, ρ) in the (θ, ρ)-plane corresponding to the line l lies between $f_i^{(0)}$ and $f_i^{(1)}$. This observation leads to the following lemma.

Lemma 5.1: Define for a point p_i a ribbon region $R(p_i)$ bounded by two sinusoidal curves $f_i^{(0)}$ and $f_i^{(1)}$ as above. Then, a set B of black points forms a line component, in other words, there is a line l such that $B \subseteq tube_r(l)$ if and only if the intersection $\bigcap_{p_i \in B} R(p_i)$ of the ribbon regions is not empty (contains l).

The lemma 5.1 suggests that a set of black points forming a line component corresponds to a region bounded by the sinusoidal curves defined above. That is, if a region is included in a ribbon region $R(p_i)$ then the corresponding set of black points includes the point p_i.

By the definition of upper and lower curves we have the following lemma.

Lemma 5.2: A set of black points associated with a cell is maximal if its upper boundary consists only of upper curves and its lower one does of lower curves.

We can enumerate all possible line components by an exhaustive search in the arrangement of those curves. Since we have $2n$ curves and any pair of curves can intersect each other at most constant times, the complexity of the arrangement is $\Omega(n^2)$. Our goal here is to implement an exhaustive search on the arrangement in $\Theta(n^2)$ time and $O(n)$ space. A basic idea is Topological Walk algorithm [2] for searching in an arrangement of lines (or pseudolines) in optimal time and space.

The curves $f_i^{(0)}$s $(i = 1, 2, .., n)$ are called *upper curves*, while $f_i^{(1)}$s are called *lower curves*. Now we have $2n$ sinusoidal curves in the (θ, ρ)-plane forming an arrangement. Topological Walk starts from a cell c_0 and finds a set of black points associated with c_0, which is denoted by $S(c_0)$. Then, it iteratively visits adjacent cells until all the cells are exhausted. To move from a cell c_1 to c_2 adjacent to c_1 it crosses an edge. If it is a part of an upper curve $f_i^{(0)}$, its corresponding point p_i is included into a set $S(c_2)$. On the other hand, if it is a part of a lower curve $f_i^{(1)}$, its corresponding point p_i is removed from a set $S(c_1)$. That is, $S(c_2)$ is computed as follows:

$$S(c_2) = \begin{cases} S(c_1) \cup \{p_i\}, & \text{if } f_i^{(0)} \text{is crossed} \\ S(c_1) - \{p_i\}, & \text{if } f_i^{(1)} \text{is crossed} \end{cases}$$

In this way, we can enumerate all the cells in the arrangement. The lemma 5.2 allows us to determine whether a set associated with a cell is maximal or not. Whenever we encounter a maximal cell, we output its corresponding set of black points as a line component if its cardinality exceeds a threshold.

Thus, if those sinusoidal curves are pseudolines, i.e., if they can be treated as if they are lines, we can rely on Topological Walk algorithm. Unfortunately, this arrangement is not a pseudoline arrangement in the range $0 \leq \theta \leq \pi/4$. This is a major difficulty compared to the L_∞-width based line detection given in [3], [4] in which we only need to consider a line arrangement. More precisely, a pair of curves of the same type (upper or lower) intersect at most once, but a pair of curves of different types may intersect twice. Such an arrangement is called an arrangement of *pseudo parabolas* [14]. Our algorithm is based on *Topological Walk* method [2], which only works for a pseudoline arrangement. Therefore, we want to cut the arrangement by $O(n)$ vertical lines into pseudoline arrangements. However, the best known method needs $O(n^{5/3})$ vertical lines to cut an arrangement of n pseudo parabolas into pseudoline arrangements [14] in general (a speudo parabola is a curve having the same property as a parabola, that is, each pair of them intersect at most twice.)

Fortunately, our arrangement comes from dual curves of grid points, and also we can assume the width r is not very large in the line detection application. In the following, we show that we can cut the arrangement into pseudoline arrangements by at most $(2r+1)^2$ vertical lines. Since we assumed $r < \sqrt{n}$ (usually r is a small constant), addition of these lines does not increase the asymptotic complexity of the arrangement.

Consider when two curves intersect twice. As is said above, double intersections occur only for different

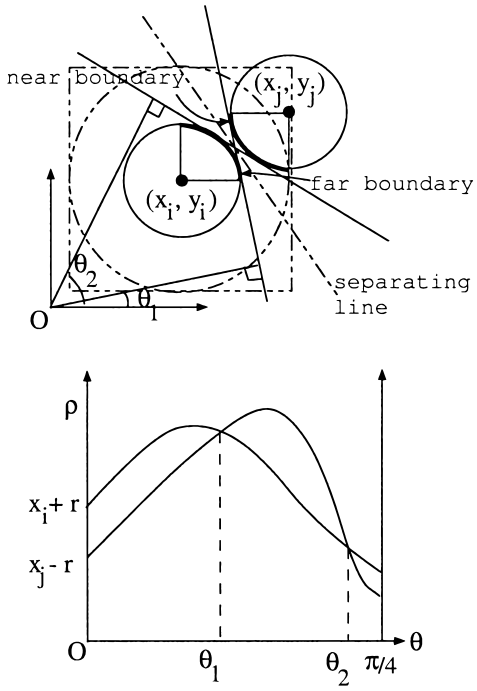


Fig. 3 Pictorial description for double intersections.

types of curves. Figure 3 shows the upper curve $f_i^{(0)}$ corresponds to the line tangent to the circular arc in the first quadrant with respect to a point (x_i, y_i). The lower curve $f_j^{(1)}$ is similarly described. So, the intersections between them correspond to common tangents of the two circular arcs defined in the figure. Their slopes are less than -1 if the point (x_j, y_j) lies in the interior of the square of side length $4r$ and in the exterior of the circle of radius $2r$ both centered at (x_i, y_i) and moreover below the line of slope 1 passing through the point (x_i, y_i). Moreover, they are separated by a slope which is perpendicular to the line passing through the two points (x_i, y_i) and (x_j, y_j). Thus, the slope of the separation line is defined by a pair of grid points (x_i, y_i) and (x_j, y_j) satisfying the condition described above. This leads to the fact that there are only $(2r+1)^2$ different such slopes.

This is just a rough sketch. More formal discussions follow. Suppose that $f_i^{(0)}$ and $f_j^{(1)}$ intersect twice each other. Then, there exist two tubes such that their far boundaries contain $p_i = (x_i, y_i)$ and near boundaries contain $p_j = (x_j, y_j)$.

Lemma 5.3: If $|x_i - x_j| > 2r$ or $|y_i - y_j| > 2r$, then $f_i^{(0)}$ and $f_j^{(1)}$ intersect at most once in the range $0 \leq \theta \leq \pi/4$.

Proof If we set $t = \tan(\theta/2)$, the equations $\rho = x_i \cos\theta + y_i \sin\theta + r$ and $\rho = x_j \cos\theta + y_j \sin\theta - r$ become $(1+t^2)(\rho - r) = x_i(1-t^2) + 2y_i t$ and $(1+t^2)(\rho + r) = x_j(1-t^2) + 2y_j t$, respectively, each of which has at most two roots. If $|x_i - x_j| > 2r$, there exists a vertical tube going through the interval between p_i and p_j. Therefore, one of the above roots corresponds to a positive θ and the other to a negative θ. Hence, we have at most one intersection in the range $0 \leq \theta \leq \pi/4$. The case $|y_i - y_j| > 2r$ can be proved similarly. □

Lemma 5.4: Suppose that the angle between the vertical line and line $p_i p_j$ is α_{ij} (we chose the one in $[0, \pi/2]$). If $\alpha_{ij} > \pi/4$, $f_i^{(0)}$ and $f_j^{(1)}$ intersect at most once in the range $0 \leq \theta \leq \pi/4$. Otherwise, $f_i^{(0)}$ and $f_j^{(1)}$ intersect at most once in each of ranges $0 \leq \theta < \alpha_{ij}$ and $\alpha_{ij} < \theta \leq \pi/4$.

Proof Routine one, and hence omitted. □

Let $\{a_1, \ldots, a_M\}$ be the Farey series of rank $2r$, which is the increasing sequence of numbers represented by a/b such that $1 \leq a \leq b \leq 2r$. We divide the angle interval $[0, \pi/4]$ into buckets $B_k = (\tan^{-1}(a_{k-1}), \tan^{-1}(a_k)]$ $(k = 1, 2, .., M)$, where $a_0 = 0$.

Lemma 5.5: Within each bucket B_k, $f_i^{(0)}$ and $f_j^{(1)}$ intersect at most once.

Proof If $|x_i - x_j| \leq 2r$ and $|y_i - y_j| \leq 2r$, then, $\tan\alpha_{ij} \in F_{2r}$ if $\alpha_{ij} \leq \pi/4$. Hence, the lemma follows from Lemma 5.4. □

Hence, we have the following theorem:

Theorem 5.6: If we partition the (θ, ρ) plane into slabs by vertical lines $\theta = \tan^{-1}(a_k)$, $k = 1, 2, .., M$, the set of curves $f_i^{(e)}$ $(i = 1, 2, .., n,\ e = 0, 1)$ forms a pseudoline arrangement within each slab.

Hence, we can partition the plane by $M \leq (2r+1)^2/2 = O(n)$ vertical lines, so that the arrangement is pseudoline arrangement within each slab. This corresponds to the fact that there are at most $(2r+1)^2$ tubes of width $2r$ circumscribing a given set of grid points. Recall that we assumed $r = O(\sqrt{n})$.

6. What to be Detected

6.1 Optimization Conditions

We consider the following conditions for the digital line component Q reported by algorithms:

- Cardinality: The cardinality of Q is greater than a given threshold.
- Maximality: There exists no digital line component Q' satisfying $Q' \supset Q$.
- Density: There exists a "dense segment" in Q.

In usual methods, only cardinality condition is discussed. Maximality is important to reduce the redundancy of the output. If maximality is expensive to attain, we need some weaker condition to avoid redundant outputs; Indeed, we consider *restricted maximality* defined later. The density condition is requested since we

usually only need lines containing dense segments in computer vision applications.

Since the slope of a line is less than -1 if $\theta \in [0, \pi/4]$, the density of a segment is fairly represented by the density of its projection to the ordinate. For a vertical interval I, let $Q(I)$ be the subset of Q whose projection falls in I. Let $|I|$ be the width of I. There are several candidate optimization criteria for dense intervals:

1. Fixed-width dense interval: Interval I of width w containing at least t elements of Q, where w and t are given thresholds.
2. No-gap dense interval: Interval of width at least w without a gap of width g, where a gap is a subinterval in which no point of Q is located.
3. Parametric dense interval: The interval I satisfying $|Q(I)| - \tau|I| > \mu$ for given parameters τ and μ.

Here note that a direct way of evaluating density of $Q(I)$ by $|Q(I)|/|I|$ is not linear. Since the density of the segment also depends on the slope, we also need the versions where parameters and thresholds are sensitive to the slope of the line. Parametric dense interval is useful since we can handle any non-linear concave objective function measuring density by using a parametric optimization approach [10] if we can detect parametric dense interval efficiently.

6.2 Line Segment Detection

We often want to report line segments rather than lines themselves. The problem is called *line segment detection*. If we consider the fixed-width dense interval condition, we want to report the maximal intervals (and the associated segments) each of which is represented as a union of fixed-width dense intervals. For the no-gap dense condition, we want to report maximal no-gap dense intervals. For the parametric dense interval, we report the segment corresponding to the interval I maximizing $|Q(I)| - \tau|I|$.

6.3 L_∞-Width Based Line Component

Instead of considering tubes of thickness $2r$, we may consider tubes of L_∞ (i.e. vertical or horizontal) width $2r$ to represent a line. If the line l has the equation $y = ax + b$ and $0 \le a \le 1$, the thickness of a tube of L_∞ width $2r$ around l is $2r/\sqrt{1+a^2}$, and hence it depends on the slope a. However, this gives a practical approximation of lines to be detected. This formulation was given in [3], [4], and solved by using geometric duality which maps a point (c, d) to the dual line $y = cx - d$. In [3], [4], the cardinality condition and maximality condition were discussed, and $O(n^2)$-time algorithm was given to report the set of all maximal line components satisfying the cardinality condition if we restrict the lines of slopes between 0 and 1. Lines of other slopes are handled by applying reflection transformation to the original data. Although we mainly consider thickness-based line components in this paper, we can similarly consider density conditions for this L_∞ case, and can design algorithms with the same asymptotic time complexities.

6.4 Cardinality Condition

Now, we see how our optimality conditions for the line component can be examined during the topological walk. We execute topological walk within each of slabs $S_1, \ldots, S_M$ associated with angle buckets $B_1, \ldots, B_k$.

For each cell C in the arrangement, let $Q(C)$ be the line component corresponding to C; in other words, the set of tube regions containing C is $\{R_i : p_i \in Q(C)\}$. When we proceed the walk from an edge e to the next edge e', the cells $C(e)$ and $C(e')$ above e and e', respectively, may be different. If so, there is a curve $f_i^{(\sigma)}$ through the endpoint shared by e and e' which separates these cells. If $C(e)$ is above $f_i^{(\sigma)}$, then $Q(C(e')) = Q(C(e)) \cup \{p_i\}$ for $\sigma = 0$ and $Q(C(e)) = Q(C(e')) \cup \{p_i\}$ otherwise. The case in which $C(e)$ lies below the curve is symmetric. Therefore, it is very easy to keep the count of $|Q(C(e))|$ for the edge e visited, and hence the cardinality condition can be checked in $O(1)$ time at each cell visited.

6.5 Maximality Condition with Angle Restriction

Let L_k be the set of lines whose slope is in the angle bucket B_k $(k = 1, 2, .., M)$. We say a line component Q is *maximal in* L_k if it is represented by some line $l \in L_k$, and there exists no line component $Q' \supset Q$ represented by a line in L_k. If a line component is maximal in L_k for some k in $\{1, 2, .., M\}$, we say it is *restricted maximal*. We design our algorithm to report only restricted-maximal line components.

We consider the arrangement restricted to the slab dual to the angle bucket B_k. If we cut the boundary of a cell at its rightmost and leftmost vertices, we have two chains of edges, one above the cell called the *upper chain* and the other called the *lower chain*. A cell $C(e)$ corresponds to a maximal line component in L_k if and only if all edges in its upper chain are segments of upper curves and all edges in its lower chain are segments of lower curves. This is because $C(e)$ is exactly the intersection of the upper half planes of the pseudolines in the lower chain and the lower half planes of the pseudolines in the upper chain (this does not hold for pseudo parabolas).

In the arrangement, each edge e is adjacent to exactly two cells $C(e)$ and $D(e)$, above and below e, respectively. Also, each vertex is the leftmost vertex of a unique cell. As we have shown, we know the whole information concerning $C(e)$ when we walk on e to the

right (in its second visit), but we do not know complete information concerning the cell $D(e)$. However, if e is a leaf edge of the upper horizon tree, we know the information concerning the part of the upper chain of $D(e)$ to the left of e.

Our method is to keep two check-bits, namely U-bit and L-bit, at each leaf edge in the upper horizon tree. For each leaf edge e, U-bit (resp. L-bit) of e is "on" if all edges to the left of e in the lower (resp. upper) chain of $C(e)$ (resp. $D(e)$) are segments of lower (resp. upper) curves; otherwise, it is "off." If we come to the branch of leaf edges and execute an elementary step, we report the cell to the left of the branching vertex as a maximal cell if and only if both the L-bit of the upper leaf edge of the branch and the U-bit of the lower leaf edge are "on." At each elementary step, the L-bit and U-bit of the newly created pair of leaf edges can be obtained in $O(1)$ time from the erased pair of leaf edges. Therefore, the maximality condition can be checked in $O(1)$ amortized time.

Theorem 6.1: All line components satisfying both cardinality and restricted maximality conditions can be computed in $O(n^2)$ time and $O(n)$ space

The restricted maximal condition is definitely weaker than maximality; indeed the size of output might be much larger than the number of all maximal line components satisfying the cardinality condition. This is different from the L_∞-width based line detection [3], [4], for which we can only report maximal line components among lines of slope angles between 0 and $\pi/4$. If a non-maximal component Q is reported in the algorithm represented by a line in L_k, Q must be contained in the intersection $tube_r(l) \cap tube_r(l')$ for $l \in L_j$ and $l' \in L_k$ for some $j \neq k$. If $|j-k| \neq 1$, $tube_r(l) \cap tube_r(l')$ is a diamond which can contain at most $4r^4$ grid points. With the threshold $t > 4r^4$, Q is dominated by only maximal line components in L_{k+1} or L_{k-1}.

6.6 Density Conditions

6.6.1 Fixed-Width Dense Interval

For a line component Q, a vertical interval I is called *effective* if every subinterval of width w of I has at least t elements, where w and t are given thresholds. We would like to find every possible line component Q with at least one effective interval, and for such a component, we want to report all maximal effective intervals (this second requirement is discussed in Sect. 6.7). If Q is given, we can easily compute all of its maximal effective intervals in linear time. However, since Topological Walk algorithm walks in the arrangement while updating the information of a line component $Q(C(e))$ for the current edge e, we want an efficient dynamic data structure.

We have the point set $Q(C(e))$. For each point $p_i = (x_i, y_i)$, we associate a vertical interval $I_i = [y_i, y_i + w)$. The following is an easy observation:

Lemma 6.2: A vertical interval $(s-w, s]$ of width w ($w \leq s \leq N$) is effective for a line component Q if and only if there are at least t intervals I_i associated with $p_i \in Q$ containing s.

From the above lemma, we can use the well-known interval tree data structure. We construct a binary static interval tree T on $[1, N]$. An interval $J(v)$ is associated for each node v of T, and $J(v) = J(l(v)) \cup J(r(v))$ (disjoint union) holds, where $l(v)$ and $r(v)$ are left and right sons of v, respectively, and $J(root) = [1, N]$. We consider the segment tree storing the set of segments $\{I_i : p_i \in Q\}$. Each segment is decomposed into at most $\log N$ *canonical intervals* and stored in the tree. For each v, we count the number $m(v, Q)$ of I_i ($p_i \in Q$) which has the canonical interval $J(v)$ as its piece. Also, we define $M(v, Q) = \max_{u \succeq v} m(u, Q)$, where $u \succeq v$ means that u is a descendant of v in T. The value of M can be computed by using the formula $M(v, Q) = \max\{M(l(v), Q), M(r(v), Q)\} + m(v, Q)$. Our data structure is the tree T such that each node v stores both $m(v, Q)$ and $M(v, Q)$.

Lemma 6.3: The line component Q has a fixed-width dense interval if and only if $M(root, Q) \geq t$ for the given threshold.

The tree data structure takes $O(N)$ space, and it is easy to see that we can maintain it in $O(\log N)$ time if a point is inserted or deleted from Q. It may happen that $N > n$. However, in such a case, we can remove empty columns from the grid to compress the tree size to $O(n)$.

6.6.2 No-Gap Dense Interval

To maintain the gap condition, we maintain the ordered list L of the x-coordinate values of Q. Moreover, if difference of two adjacent entries of L is less than g, we give a bidirectional pointer between them. These pointers divide L into groups, each of which corresponds to an interval without a gap. We also maintain this grouping, and also the list of length of groups. If there exists a group of length at least w, the line component Q has a no-gap dense interval. Maintenance of this data structure is a Union-Find-Split problem, and hence can be done in $O(\log \log N)$ time using $O(N)$ space. If $n \leq N$, we can do it in $O(\log n)$ time using $O(n)$ space.

6.6.3 Parametric Dense Interval

We fix the parameter τ. For a line component Q, let us compute the interval I maximizing $\Phi(I) = |Q(I)| - \tau|I|$ for the parameter τ. We again use the interval tree T. For the canonical interval $J(v)$ at the node v, we store the following three subintervals, together with

the associated values of the function Φ; (1) $J(v, Q, all)$ maximizing $|Q(I)| - \tau|I|$, (2) $J(v, Q, right)$ maximizing $|Q(I)| - \tau|I|$ under the condition that I contains the right end of $J(v)$, and (3) $J(v, Q, left)$ maximizing $|Q(I)|-\tau|I|$ under the condition that I contains the left end of $J(v)$. If we know these three intervals for two sons $l(v)$ and $r(v)$ of v, we can compute these for v in constant time. The line component Q has a parametric dense interval if and only if $\Phi(J(root, Q, all)) > \mu$. We can compress the data size to n if $N > n$, and it is easy to maintain the data structure in $O(\log n)$ time per update of Q.

Hence, we have the following theorem:

Theorem 6.4: All line components satisfying all of cardinality, maximality, and density conditions can be computed in $O(n^2 \log n)$ time and $O(n)$ space for each of density criteria. For the no-gap density criterion, time complexity can be decreased to $O(n^2 \log\log N)$ using $O(n + N)$ space.

Our density criteria depends on the width of the vertical projection of the segment. The real density also depends on the slope of the line; Of course, steeper line is denser with the same density interval condition, since we project it to the ordinate. We could assign a different threshold and parameter value for each bucket of angles to reduce the effect of slope. Moreover, within a bucket, Topological Walk algorithm reports the line component when it visits the rightmost vertex of its associated cell. We can run right-to-left topological walk instead of usual left-to-right one to report the line component at the leftmost vertex of the cell. In the primal plane, the leftmost vertex corresponds to the line of smallest θ, i.e. the line steepest among those representing the line component (recall that the slope is $\tan(\theta - \pi/2)$). Hence, we can compute the slope of the line corresponding to the vertex, and tune the parameters suitably. For the fixed-width density interval criterion, the parameter w must remain unchanged during topological sweep; however, the threshold t can be tuned to, say, $t/\cos\theta$. For the no-gap and parametric conditions, we can tune w and μ during Topological Walk, respectively.

6.7 Line Segment Detection

Let us consider the problem of detecting line segments with endpoints instead of infinite lines. We can apply the above algorithm and compute all maximal dense intervals in each line. However, it is not assured that we report $O(n^2)$ segments; Accordingly, the time complexity might increase. Indeed, the same segment component may be reported several times. We want to avoid such redundancy as far as possible.

For each maximal line segment component S, we consider the steepest tube containing S. This must correspond to the leftmost vertex of a cell in the arrangement (or the left vertical boundary of the slab), and we know the information of the cell when we visit the vertex in the right-to-left topological walk. Therefore, for each cell C, we only consider the segment component containing the points, say p_i and p_j, corresponding to curves intersecting at its leftmost vertex. If we consider the fixed-width dense interval condition, we report the maximal interval J such that any point in J is contained in at least one fixed-width dense interval under the condition $y_i, y_j \in J$. Such an interval is unique, and is computed in $O(\log n)$ time by using the interval tree data structure. For the no-gap dense condition, we want to report a maximal no-gap dense interval containing both y_i and y_j, and can be done in $O(\log n)$ time. For the parametric dense interval, we just start with the leaf nodes containing y_i and y_j, respectively, and climb up the tree to compute the optimal interval containing them in $O(\log n)$ time easily. Therefore, we have the following:

Theorem 6.5: We can compute a set of dense line segment components containing all maximal dense line segment components in $O(n^2 \log n)$ time and $O(n)$ space for each density criterion.

Unfortunately, there may be a non-maximal line segment component (in the restricted sense) in the output set, since a maximal line segment in a line component may not be maximal as a point set.

Finally, let us consider the problem of reporting the union of all the dense line segment components as a point set. Although we report at most $O(n^2)$ line segments in the previous algorithm, each line segment component may have many points and hence a naive method may need $O(n^3)$ time to compute the union.

Theorem 6.6: The union of all dense line components can be computed in $O(n^2 \log n)$ time and $O(n)$ space.

Proof We use a simple trick that works for each of our density criteria. We color the points in P in *red* and *white* in our algorithm. Initially, all points are white, and when Topological Walk finds a dense line segment component, we change all white points in the component into red, and update the coloring. If there are k white points in the line component, we can update the coloring using additional $O(k \log n)$ time if we store the current points in a segment tree. Since the total number of points is n, the total computation time remains $O(n^2 \log n)$. □

7. Some Difficulties in Implementation

We have implemented a preliminary version of the algorithms presented in this paper: although we have described an algorithm for detecting line components based on traversal on subdivided arrangements of sinusoidal curves, our program uses lines instead of si-

nusoidal curves. So, each black point is transformed into a tube of width $2r$ (vertical width since we are looking for lines of slopes between -1 and 1), which is a region bounded by two parallel lines corresponding to the upper and lower curves defined in this paper. The implementation is not easy due to possible numerical errors and degeneracies. Especially, degeneracy is a serious problem for this application: recall that each black point is transformed into a tube bounded by two lines based on duality transform. Since coordinates of such a point are integers, we may have very high degeneracies, that is, $O(N)$ lines may meet at one point. This problem can be resolved by using exact computation based on rational numbers. Another difficulty occurs when an upper line of a tube happens to coincide with a lower line of another tube for a different point. It happens when $r = \frac{1}{2}$. To avoid this type of degeneracy, we use an irreducible fraction for r with its denominator greater than 2, say $r = 4/7$.

We are now trying to extend our program to deal with some of the conditions described in the paper and also to traverse on the subdivided arrangements of sinusoidal curves in a robust way.

Generally speaking it is not so easy to implement Topological Walk for a set of curves defined by sine and cosine functions. Fortunately, in our case it is rather easy. For exact implementation we must exactly identify and resolve degeneracy that more than two curves meet at one point and exactly check whether an intersection lies above, on, or below a curve. We shall show all these can be done.

Let $\rho = x_i \cos\theta + y_i \sin\theta + c_i$ and $\rho = x_j \cos\theta + y_j \sin\theta + c_j$ be two arbitrary curves, where c_i and c_j are r or $-r$. Setting $t = \tan(\theta/2)$, their intersection must satisfy the following equation:

$$\frac{1-t^2}{1+t^2}x_i + \frac{2t}{1+t^2}y_i + c_i = \frac{1-t^2}{1+t^2}x_j + \frac{2t}{1+t^2}y_j + c_j.$$

Since $1 + t^2 \neq 0$, we have a quadratic equation in t.

$$(1-t^2)x_i + 2ty_i + c_i(1-t^2) = (1-t^2)x_j + 2ty_j + c_j(1-t^2).$$

Thus, a solution to the equation must be of the form:

$$t_{ij} = a_{ij} + b_{ij}\sqrt{c_{ij}},$$

where a_{ij} and b_{ij} are rational numbers and c_{ij} an integer. This type of numbers can be handled exactly, say by using LEDA. It is also obvious that we can judge vertical relationship between such an intersection of two curves and another curve.

8. Conclusions

Hough Transform is a well-established scheme for line detection and in fact, it is commonly used in practice. However, several questions about tradeoff between its computational cost and performance of line detection has been left unanswered. In this paper we began with a mathematical formulation of the problem and presented a totally different scheme based on decomposition of arrangement of parabolas into that of pseudolines combined with Topological Walk over it. We have shown that our new scheme could resolve several important open problems left by Hough Transform. The original Hough Transform detects lines by counting the number of points near parameterized lines. Thus, it seemed impossible to take the density and no-gap conditions into considerations without any increase of computational costs. Our new scheme can do it with $O(\log n)$-time overhead. Thus, its running time is either $O(n^2)$ or $O(n^2 \log n)$ depending on which constraints are required. Note that it depends only on the number n of edge points, not on the image size ($N \times N$). We implemented an algorithm for finding digital lines based on Topological Walk not on an arrangement of sinusoidal curves but on that of lines.

Acknowledgments

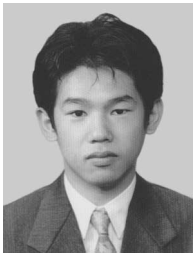
This work was partially supported by Grant in Aid for Scientific Research of the Ministry of Education, Science and Cultures of Japan.

References

[1] F. van der Stappen, Motion Planning amidst Fat Obstacles, Ph.D. Dissertation, Utrecht University, 1994.
[2] T. Asano, L. Guibas, and T. Tokuyama, "Walking in an arrangement topologically," Int. J. of Comput. Geom. and Appl. 4-2, pp.123–151, 1994.
[3] T. Asano and N. Katoh, "Variants for the Hough transformation for line detection," Comput. Geom. Theory and Appli. 6, pp.231–252, 1996.
[4] T. Asano, N. Katoh, and T. Tokuyama, "A unified scheme for detecting fundamental curves in binary edge images," Proc. 2nd ESA, Springer LNCS 855, pp.215–226, 1994.
[5] T. Asano and T. Tokuyama, "Topological walk revisited," IEICE Trans. Fundamentals, vol.E81-A, no.5, pp.751–756, 1998; Preliminary version in Proc. 6th CCCG, pp.1–6, 1994.
[6] C.M. Brown, "Inherent bias and noise in the Hough transform," IEEE Trans. Pattern Anal. & Mach. Intell., vol.5, no.5, pp.493–505, 1983.
[7] R.O. Duda and P.E. Hart, "Use of the Hough transformation to detect lines and curves in pictures," Comm. of the ACM, vol.15, pp.11–15, 1972.
[8] P.V.C. Hough, Method and Means for Recognizing Complex Patterns, U.S. Patent 3069654, 1962.
[9] J. Illingworth and J. Kittler, "The adaptive Hough transform," IEEE Trans. Pattern Anal. & Mach. Intell., vol.9, no.5, pp.690–698, 1987.
[10] N. Katoh and T. Ibaraki, "A parametric characterization and an ϵ-approximation scheme for the minimization of a quasiconcave program," Discrete Applied Math., vol.17 pp.39–66, 1987.
[11] H. Li, M.A. Lavin, and R.J. LeMaster, "Fast Hough transform," Comput. Vision Graphics Image Processing, vol.36, pp.139–161, 1986.
[12] J. Matoušek, "Range searching with efficient hierarchical cutting," Discrete & Computational Geometry, vol.10, pp.157–182, 1993.
[13] I.D. Svalbe, "Natural representation for straight lines and

the Hough transform on discrete arrays," IEEE Trans. Pattern Anal. & Mach. Intell., vol.11, no.9, pp.941–950, 1989.
[14] H. Tamaki and T. Tokuyama, "How to cut pseudo-parabolas into segments," Discrete & Computational Geometry, vol.19, pp.265–290, 1998.
[15] T. Wada, T. Fujii, and T. Matsuyama, "$\gamma - \omega$ Hough transform—Linearizing voting curves in an unbiased $\rho - \theta$ parameter space," IEICE Trans., vol.J75-D-II, no.1, pp.21–30, Jan. 1992.

Tetsuo Asano received the M.E. and Ph.D. degrees from Osaka University in 1974 and 1977, respectively. He joined Osaka Electro-Communication University in 1977 as a lecturer. He joined Japan Advanced Institute of Science and Technology in 1997 where he is now a full professor. He has been working in computational geometry and VLSI design from theory to applications. He is serving as an editor for several journals including Discrete and Computational Geometry, Computational Geometry: Theory and Applications, International Journal of Computational Geometry and Applications, Theory of Computing Systems, etc. He is a member of ACM, SIAM, IPS of Japan, and Operations Research Society of Japan.



Koji Obokata received the B.E., M.E., and Ph.D. degrees from Gunma University in 1992, 1995, and 1998, respectively. Since 1998 he has been an associate in School of Information Science, Japan Advanced Institute of Science and Technology. He is interested in distributed algorithm, graph theory, computational geometry and their applications. He is a member of IPS of Japan.

Takeshi Tokuyama received the B.S., M.S., and Ph.D. degrees in Mathematics from the University of Tokyo in 1979, 1981, and 1985, respectively. He joined IBM Research Laboratory in 1986, and worked on theoretical computer science and its applications. Since 1999, he has worked as a professor on Design and Analysis of Information Systems in Graduate School of Informations Sciences, Tohoku University.