A Unified Scheme for Detecting Fundamental Curves in Binary Edge Images

Tetsuo Asano,* Naoki Katoh,† Takeshi Tokuyama‡

Abstract

We present a unified scheme for detecting digital components of various planar curves in a binary edge image. A digital component of a curve is the set of input *edge points* from each of which the horizontal or vertical distance to the curve is at most 0.5.

Our algorithm outputs all curve components containing at least k points (k is a given threshold) in $O(n^d)$ time (if $d \geq 2$) and linear space, where n is the number of points, and d is a measure that reflects the complexity of a family of curves; for example, $d = 2, 3$, and 5 for lines, circles, and ellipses, respectively. For most of the popular families of curves, our only primitive operations are algebraic operations of bounded degree and comparisons. We also propose an approximate algorithm for computing an approximation solution with error ratio $\epsilon = 1 - \alpha$ (called an α-sensitive solution), whose time complexity is $O((t/\epsilon)^{d-1}n)$, which is very efficient if the threshold-ratio $t = n/k$ is small.

Keywords: Algorithm, Digital Curve, Arrangement, Image Recognition.

*School of Information Science, Japan Advanced Institute of Science and Technology, Tatsunokuchi, Ishikawa, 923-12, Japan

†Department of Architecture and Architectural Systems, Graduate School of Engineering, Kyoto University, Kyoto, 606-01, Japan

‡Graduate School of Information Sciences, Tohoku University, Sendai, 980-8577, Japan

1 Introduction

One of the most fundamental problems in pattern recognition is how to detect all the digital line components in a given binary edge image. A number of methods such as the Hough Transform [6, 7, 9, 25, 21] have been proposed for this purpose, and generalized to the problem of detecting digital curves [20, 28]. However, none of them seems able to guarantee that all line components will be detected. The algorithm based on the L_1-dual transform [3, 4] presented by the first two authors of the present paper was the first to guarantee perfect detection of all line components of a given set of n edge points in an $N \times N$ grid in $O(nN^2)$ time and $O(n)$ space.

In this paper, we present a unified scheme for detecting components of various planar curves, and show that it has several features desirable in algorithms for curve detection.

1.1 Definition of curve detection problems

Suppose that we are given a two-dimensional unit integral grid G of size $N \times N$, and a *binary edge image* consisting of n *edge points* (or black dots), which form a subset S of grid points of G.

For a plane curve defined by an equation $f(x, y) = 0$ (we often use f as an abbreviated symbol for the curve itself), its digital image $Im(f)$ in the grid G is the set of grid points p such that the curve f intersects one of the grid-edges adjacent to p at a point whose distance from p is less than or equal to 0.5. In other words, the digital image is a discretized image of the curve f on the grid, and is obtained by enumerating the intersections between the curve and grid-edges, and generating the grid point nearest to each intersection. Here, if the curve intersects at the midpoint of a grid-edge, the digital image contains both the endpoints of the edge. Figure 1 shows digital images of ellipses.

The portion $Im(f) \cap S$ of the digital image in the binary edge image is called the *digital component* of f in S. We are also given a family $\mathcal{F}$ of planar curves defined over the square region containing G. A subset P of S is called a digital component with respect to $\mathcal{F}$ if P is a digital component of a curve f of $\mathcal{F}$. A digital component whose size is greater than or equal to a given threshold k is called an *effective* component. With these settings, the

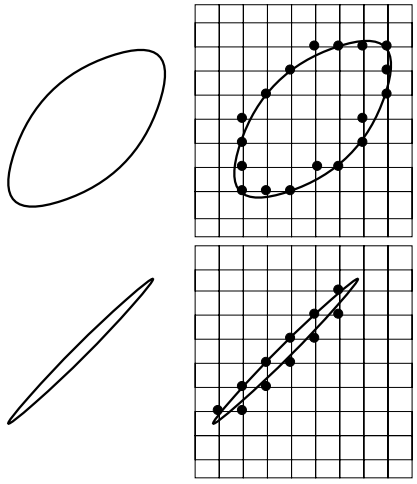

Figure 1: Digital image of ellipses

curve component detection problem consists of enumerating effective digital components with respect to $\mathcal{F}$.

Figure 2 shows an input binary edge image[1] for the line detection ($\mathcal{F}$ is the family of lines) problem. The output of a heuristic algorithm [1] is illustrated in Figure 3, in which segments representing dense parts of the lines are visualized.

1.2 Desirable features of a curve detection scheme

A good scheme for curve detection should include the following *features*, according to which curve detection algorithms should be evaluated:

1. **Detection ability:** It should be guaranteed that all effective digital curve components or their approximations can be detected.

2. **Simple operations:** It is desirable that an algorithm should require only algebraic operations and comparisons. If we cannot avoid using other operations, we should minimize their number. To be precise, we should avoid having to solve nonlinear equations as possible as we can.

[1]This is one of the input figures used for a program contest on line detection organized by the Special Interest Group on Computer Vision of Information Processing Society of Japan. By courtesy of Dr. T. Wada of Kyoto University (contest chair).

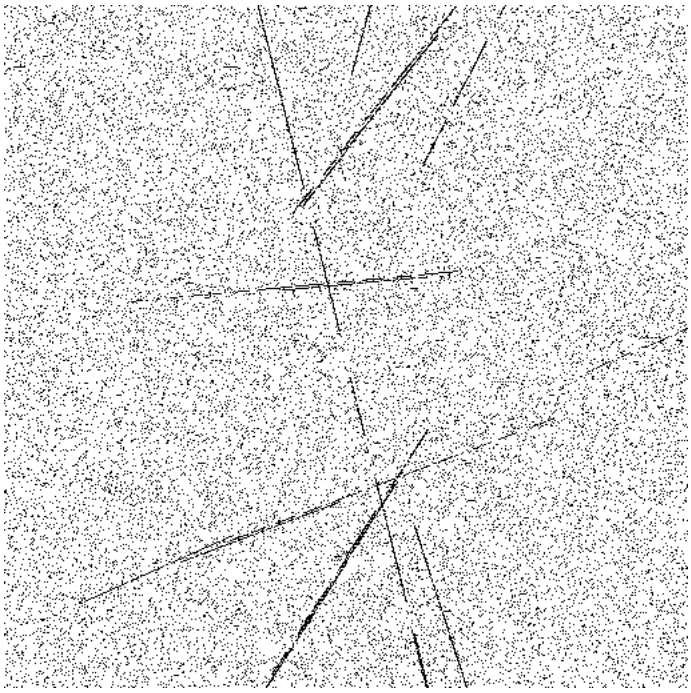

Figure 2: Input grid points

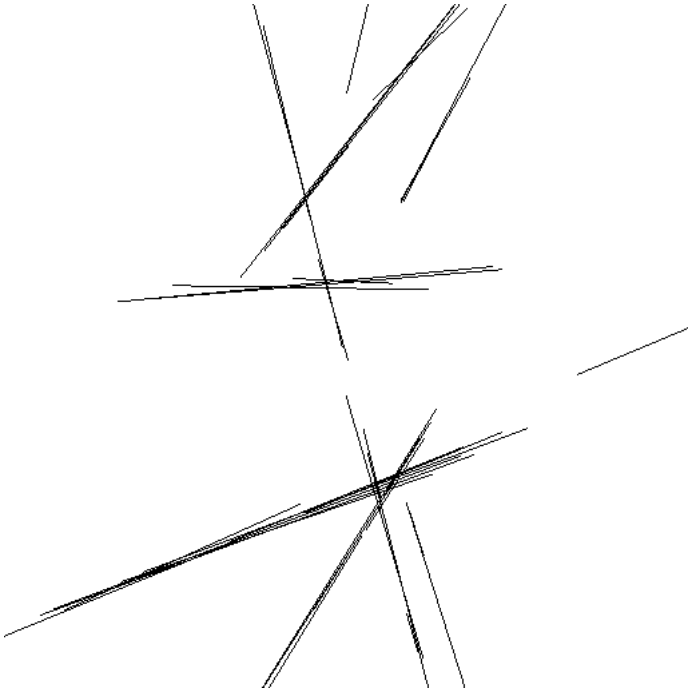

Figure 3: Detected digital line components

3. **Linear space complexity:** It is crucial to develop algorithms with small (preferably, linear in n) space complexity in practical applications.

4. **Small output size:** We need a method to compress outputs, since the total number of possible digital components usually becomes very large. One possible way is to output only *maximal components*, where a digital component is *maximal* if there is no other digital component that properly includes it. Another possible way is to output neither each digital component nor the corresponding curve equation, but to aggregate information on digital components in a compact form by outputting regions of the parameter space defining $\mathcal{F}$ such that the digital component associated with each point of each region is effective.

5. **Efficient approximation:** If it is too expensive to output all effective components, an approximation algorithm must be considered. For each effective digital curve component P, a curve f such that $Im(f)$ contains a large portion of P should be output. It is desirable that the performance of the approximation algorithm can be theoretically guaranteed. Moreover, the computing cost should be cheap and the output should be small.

1.3 Defects of standard methods

Detecting lines in a digital image is the first step in recognizing some geometric shape. Although the word "line components" seems to be clear enough, it is vague from a mathematical point of view. This may be the reason for the scarcity of theoretical studies of the performance guarantees of line detection algorithms.

Among the various existing techniques for line detection, the standard algorithm is based on the Hough Transform [6, 7, 9, 25]. It transforms an edge point (x_i, y_i) into a sinusoidal curve $\rho = x_i \cos\theta + y_i \sin\theta$ in the parameter plane associated with ρ and θ. Since ρ is the orthogonal distance from the origin to the line passing through (x_i, y_i) with a normal vector of angle θ, the intersection of two such curves corresponds to the line passing through the two points in the original plane. Thus, one might easily imagine that the problem is one of finding all the intersections at which many such curves meet. But actually, this is not the

case, since we are interested not in exact fitting of edge points to a line but in approximate fitting. This is one of the difficulties.

A more standard transformation in computational geometry is the Duality Transform for mapping a point (x_i, y_i) in the (x, y) plane to a line $v = x_i u + y_i$ in the (u, v) plane. In fact, the original algorithm proposed by Hough [13] was based on the duality transform. One advantage of the sinusoidal transformation is that the range in the (ρ, θ) plane can be limited to a small region (note that $|\rho| \leq \sqrt{2}N$ and $0 \leq \theta < \pi$). This fact suggests a bucketing technique. We partition the parameter plane into $M \times M'$ equal-sized buckets (usually both M and M' are $O(N)$). Then, for each edge point (x_i, y_i) we compute $\rho_j = x_i \cos\theta_j + y_i \sin\theta_j$ for each angle $\theta_j \in (\theta_1, \theta_2, ..., \theta_{M'})$ in order to put a vote in the bucket specified by (ρ_j, θ_j). After the voting is over, we choose those buckets that contain more votes than a given threshold, and output lines corresponding to those buckets.

Here a question arises. What are we trying to detect? Compared with the above-mentioned reasonable definition of a line component, we will find that the Hough Transform method outputs digital lines based on a less concrete definition as long as a bucketing technique in the transformed plane is used.

A major disadvantage of the Hough Transform method is that it cannot be guaranteed to detect all effective line components; that is to say, it does not have perfect detection ability in our model. To overcome this disadvantage, Asano and Katoh [3, 4] proposed an algorithm that can guarantee perfect detection of all line components in $O(nN^2)$ time and $O(n)$ space. They also introduced a new notion of $\alpha-$sensitivity concerning detection ability, which relaxes the 100% detection ability by a factor of $(1-\alpha)$, and proposed a $1/2-$sensitive approximate algorithm that requires $O(nN \log n)$ time.

The family of circles is one of the most fundamental families of curves. Generalizing the Hough Transform method, one can design an algorithm for detecting circles. The idea is as follows: Consider a cone $z^2 = (x - x_i)^2 + (y - y_i)^2,\ 0 \leq z \leq N$, for each edge point (x_i, y_i) $i = 1, 2, .., n$ in the binary edge image. If three such cones intersect at one point (x, y, z), the circle of radius z with its center at (x, y) passes through the corresponding three edge points. Thus, if we put votes in a three-dimensional bucket according to such cones, we can detect circle components. The time complexity is $O(nN^2)$, and a large amount of space $O(N^3)$ is

required. Because of the large space complexity, this approach seems impractical.

Moreover, like line detection, the above algorithm does not have perfect detection ability. Indeed, in order that each bucket represent a digital component, the sum of the height, width, and breadth of a bucket should be at most 1; otherwise, there may be no circle component containing all the points whose votes go to that bucket. Thus, for example, the set of four points $(a, 0), (-a, 0), (0, a + 1), (0, -a - 1)$ forming a digital circle centered at origin with radius $a + 0.5$ gives at most three votes to each bucket.

There have been several papers [6, 9, 16, 25, 27] on the problem of detecting digital components of planar curves such as circles and ellipses. However, to the authors' knowledge, every existing algorithm requires more than linear space; moreover, no existing algorithm has perfect detection ability.

1.4 Outline of our scheme and results

Generalizing the idea of [3], we treat in a uniform manner the problem of detecting digital curve components, and propose algorithms that have most of the desirable features listed above.

Detection ability

Let us consider the region (Figure 4) obtained as the Minkowski sum of the curve and the union of four half grid-edges (around the origin). Based on a standard technique in motion planning and computational geometry [26], it can be proved that the digital image is the set of grid points in the closure of the region. Moreover, roughly speaking, if the family of the curves has d continuous parameters, it suffices to consider a curve f such that $Im(f)$ has d edge points on the boundary of the region.

Hence, for every set of d edge points, one can determine the curve that has them as boundary points of the region by solving the corresponding simultaneous equations on the parameters, and output the curve if $Im(f)$ contains k or more edge points. Although this scheme includes feature 1, we should improve it to include the other four features.

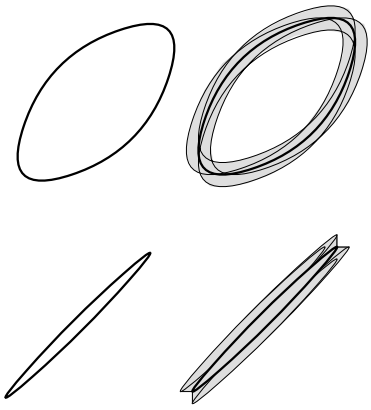

Figure 4: Regions defining digital images

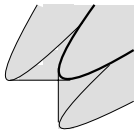

Figure 5: Tangent lines on the boundary

Avoiding non-linear equations

To realize feature 2 (simple operations), we want to avoid solving non-linear equations. We call an edge point *critical* if the curve f intersects one of its adjacent grid-edges at the midpoint. It seems that an edge point on the boundary of the region is always a critical point. The condition that a point $p = (x_0, y_0)$ is critical gives an equation $f(x_0 - 0.5, y_0) = 0$ if f cuts the grid-edge to the left of p. Therefore, if the above equation is linear with respect to the parameters defining f, we can compute f from d boundary points without solving a nonlinear equation.

Unfortunately, the above argument has a flaw. For example, in the detection of ellipses, if f is very skinny, horizontal and vertical tangent segments to the ellipse appear on the boundary of the region. Figure 5 is a magnified figure of the lower-left part of the region for the skinny ellipse in Figure 4.

Hence, it may happen that the region boundary has non-critical edge points lying on the vertical or horizontal boundary segments, and that these points together with some other

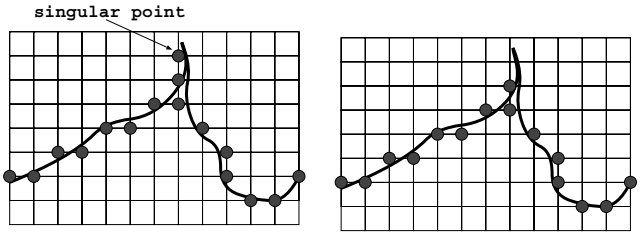

Figure 6: An $Im(f)$ containing a singular point (left), and its corresponding $RIm(f)$ (right)

boundary points define the ellipse. The condition that a point (x_0, y_0) should lie on one of the vertical tangent line of a ellipse $x^2 + a_1xy + a_2y^2 + a_3x + a_4y + a_5 = 0$ gives a quadratic equation $(x_0a_1 + a_4)^2 = 4(x_0a_3 + a_5 + x_0^2)a_2$. Hence, in order to compute the ellipse, we need to solve simultaneous quadratic equations on the parameters. For more complicated families of curves, the computation becomes very expensive.

In order to overcome this difficulty, we slightly weaken the detection ability (feature 1), so that the definition of the effective components is replaced by a novel one, which we call *eligible components*. This change enables us not only to obtain a method for realizing feature 2, but also to apply efficient computational-geometric methods such as the Topological Walk algorithm ([2, 5]) in a line arrangement and r-cutting ([8]) of a hyperplane arrangement.

A grid point $p = (i, j)$ in $Im(f)$ is called a *singular* grid point of f if (1) f intersects each of p's adjacent half-edges $((i, j), (i, j + 0.5))$, $((i, j - 0.5), (i, j))$, $((i, j), (i + 0, 5, j))$, and $((i - 0.5, j), (i, j))$ to p at an even number of points, and (2) f goes through none of the endpoints of these half-edges. Otherwise, it is called *regular*. Figure 6 gives an example of a singular grid point of a curve f. Around a singular grid point, f is a very skinny even-fold curve that is hard to distinguish from a one-fold curve in a digital image. We do not consider a curve that has many singular grid points in its digital image.

$RIm(f)$ is the set of all the regular grid points with respect to f in G. It will be shown in Section 3 (Lemma 3.1) that $RIm(f)$ is the set of grid points intersecting a region bounded by translated copies of f. Thus, we can avoid solving the nonlinear equations caused by the presence of non-critical edge points on the boundary of the region. The set $S \cap RIm(f)$ is the *regular component* of f in S. A regular digital component whose size is greater than or equal to a given threshold k is called an *eligible component*.

Note that lines have no singular grid points in their digital images. A circle can have

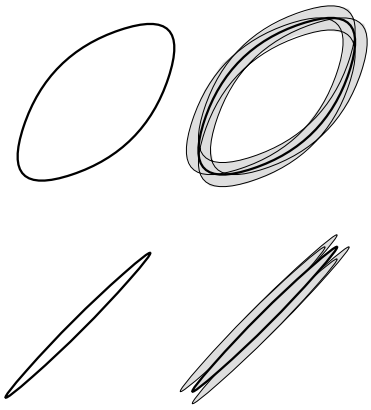

Figure 7: Regions of regular grid points of ellipses

a singular grid point only if its diameter is less than 1, and we need not concern ourselves with a digital component of such a small circle in practice. For a ellipse, the regions of the regular grid points are shown in Figure 7, and we can notice a slight difference between the region for the skinny ellipse in this figure and that in Figure 4.

Curve detection scheme and algorithm

For the detection of planar curve components, we will propose a unified scheme for detecting all eligible digital curve components for a family $\mathcal{F}$ of curves. Let us consider families of curves defined by an expression $f[a_1, .., a_d](x, y) = 0$ that are polynomials for x, y, and d parameters $a_1, .., a_d$, together with parameter constraints called *secondary constraints* which are inequalities and equalities independent of x and y. In the expression explicitly given in equation (2) in Section 3.1, we assume that $f[a_1, .., a_d](x, y)$ is linear with respect to the parameters.

The above assumption is necessary for analyzing the algorithm, and if we want to handle a family of curves for which f is represented by a non-linear expression with respect to parameters, we linearize it by increasing the number of parameters and secondary constraints if necessary. For example, the family of conchoids defined by $(x - a)^2(x^2 + y^2) = b^2x^2$ is linearized to have an expression $(x^4 + x^2y^2) + a_1(x^3 + xy^2) + a_2(x^2 + y^2) + a_3x^2 = 0$, together with two secondary constraints $a_1^2 = 4a_2$ and $a_3 \leq 0$; hence, we need three parameters, although the number of degrees of freedom of the parameters is two. For many families

of curves in practical use, such as circles, hyperbolas, ellipses, and cubic curves, we have expressions such that the constraints do not decrease the number of degrees of freedom of the parameters, as will be shown in Section 3.

The algorithm to be presented runs in $O(n^d + n \log n)$ time and $O(n)$ space for such a family of curves, under some restrictions on the secondary constraints. The algorithm realizes features 1, 2, and 3. Because of the definition of the regular digital image and linearity of the parameters, the problem of detecting curves whose regular digital images contain digital components is naturally transformed into one of examining a $d-$dimensional arrangement to find appropriate cells. Focusing on critical cases of such curves, we can reduce the problem to one of searching for an arrangement of lines (not "curves"!) in a two-parameter space, which enables us to apply the Topological Walk algorithm.

When specialized to line detection, our algorithm runs in $O(n^2)$ time and $O(n)$ space, and outputs only maximal digital line components. This improves the time complexity of Asano and Katoh [3, 4] if $n = o(N^2)$. When specialized to the detection of ellipses (resp. circles), our algorithm computes all digital elliptic (resp. circular) components in $O(n^5)$ (resp. $O(n^3)$) time and $O(n)$ space. A number of algorithms have been proposed for detecting digital circle or ellipse components [16, 17, 24, 27], but to the authors' knowledge, no existing algorithm guarantees detection of every such component. In this sense, therefore, ours is the first algorithm with guaranteed performance and linear space complexity. Moreover, it is conjectured that these computational complexities are optimal within a constant factor.

α-sensitive approximation

In order to realize features 4 and 5, we also consider α-sensitive approximate solutions. For a digital curve component P containing m points and for $0 < \epsilon < 1$, its *ϵ-approximation* is a curve f of $\mathcal{F}$ such that $Im(f)$ contains at least $(1-\epsilon)m$ points of P. A subfamily of $\mathcal{F}$ is called an *α-sensitive family* for the point set S if for any eligible component P of S, there exists a $(1-\alpha)$-approximation of P in the subfamily.

We will propose an algorithm based on r-cutting theory [8] that runs in $O((t/(1-\alpha))^{d-1}n)$ time, outputting an α-sensitive family with $O((t/(1-\alpha))^d)$ curves. Here, $t = n/k$ is the threshold-ratio. In many practical cases, t is small; thus, the above algorithm becomes

efficient. We can also output a set of cells in the r-cutting as the compressed information requested in the feature 4. We will also propose a $1/2-$*sensitive* approximation algorithm for line detection, which is faster than the approximation algorithm of Asano and Katoh [3, 4].

Detecting polygonal chains

Our scheme can be applied to the problem of detecting all digital components of transformed copies of a given polygonal chain P by a combination of translation, rotation, and scaling. This problem is strongly related to geometric matching problems [12, 14, 15].

We can apply our idea to the case in which $\mathcal{F}$ is the family of polygonal chains similar to the given polygonal chain P, and solve the problem in $O((mn)^4)$ time and $O(mn)$ space, where m is the number of edges of the polygonal chain.

2 Detection of Lines and Circles

In this section, we discuss procedures for detecting lines and circles as special cases of our main result for the general curves given in Section 3; As well as being good examples for giving an intuitive grasp of our method, they are very popular problems in computer vision that allows us compare our method with the existing popular methods mentioned in Section 1.3.

2.1 Line detection

An edge point (x_i, y_i) is contained in the digital image of a line $y = ux + v$ with $|u| \leq 1$ if and only if $ux_i + v - 0.5 \leq y_i \leq ux_i + v + 0.5$. This is equivalent to the inequality $-x_iu + y_i - 0.5 \leq v \leq -x_iu + y_i + 0.5$, which corresponds to a strip defined by two parallel lines in the (u, v) plane. Line components corresponding to lines with slope u, $|u| > 1$, can be detected in a similar manner by exchanging x and y coordinates. Note that there is no singular grid point with respect to a line.

Since we have n edge points, we have n such strips, which together define an arrangement of lines. If m such strips intersect at a cell in the arrangement, the cell corresponds to a line component of size m. If m is more than the given threshold k, the cell is called an *eligible*

cell. Thus, the problem is reduced to sweeping the arrangement to detect all the eligible cells. We have $2n$ lines and $O(n^2)$ cells. We can visit all the cells by using the Topological Walk algorithm of Asano et al. [2, 5]. (An explanation of the Topological Walk algorithm is given in Section 4.) Moreover, without increasing the time complexity, we can modify the algorithm so that we can selectively output all the maximal eligible line components (we do not give a proof of this here, since it is not the main topic of the paper).

2.2 Circle detection

The region of a digital image of a circle $f(x, y) = x^2 + y^2 + ax + by + c = 0$ is defined by the four circles obtained by shifting the circle in the four orthogonal directions (North, South, East, and West). Since we can define the interior of the circle by the inequality $x^2 + y^2 + ax + by + c \leq 0$, a condition that an edge point (x_i, y_i) is contained in the regular image of the circle $f(x, y) = 0$ is described by
$[f(x_i + 0.5, y_i)f(x_i, y_i) \leq 0] \vee [f(x_i - 0.5, y_i)f(x_i, y_i) \leq 0] \vee [f(x_i, y_i + 0.5)f(x_i, y_i) \leq 0] \vee$
$[f(x_i, y_i - 0.5)f(x_i, y_i) \leq 0]$, which is equivalent to

$$[f(x_i + 0.5, y_i)f(x_i - 0.5, y_i) \leq 0] \vee [f(x_i, y_i + 0.5)f(x_i, y_i - 0.5) \leq 0] \tag{1}$$

if the radius of the circle is not less than 0.5.

The inequalities (1) define a polygonal region (union of a pair of wedges) in the $(abc)-$space. Thus, again it suffices to examine an arrangement of planes in the three-dimensional space. Unfortunately, no existing sweeping algorithm visits every cell in a three-dimensional arrangement in $O(n^3)$ time and $O(n)$ space.

In order to overcome this difficulty, for each edge point $p = (x_i, y_i)$ of S, we consider the condition that p is a critical point, which gives a linear equation, and we can delete one of the parameters using it. Hence, we can detect all the components containing p as a critical point by examining an arrangement in the two-dimensional plane. Since every circle component can be represented so that at least one edge point is critical, the algorithm detects all eligible circle components. Thus, we obtain an algorithm that runs in $O(n^3)$ time and $O(n)$ space.

3 General Curve Detection

3.1 Parameter space associated with a family of curves

We generalize the idea expressed in the previous section in order to develop algorithms for detecting all the digital components of more complex curves such as ellipses. To deal with various planar curves, we consider a family of planar curves defined by linear combination of $d+1$ linearly independent polynomial expressions for x and y with d parameters. More precisely, the family is specified by an equation of the form

$$f[a_1, ..., a_d](x, y) \equiv t_0(x, y) + \sum_{i=1}^{d} a_i t_i(x, y) = 0, \tag{2}$$

where each $t_i(x, y),\ 0 \leq i \leq d$ is a polynomial function for x and y, and each $a_i,\ 1 \leq i \leq d$ is a real parameter. We assume that d is a fixed constant. A family of planar curves defined by (2) is denoted by $\mathcal{F}$. We sometimes denote the parameter vector $a_1, .., a_d$ by $\mathbf{a}$. When understood from the context, a member of $\mathcal{F}$ is simply written as $f(x, y) = 0$.

In most applications, we need to impose some constraints on the parameters represented by the polynomial inequalities. Let $C(\mathcal{F})$, which we call *secondary conditions*, denote the set of constraints for $\mathcal{F}$. If $C(\mathcal{F})$ is the empty set, we say that F is a *free* family.

For example, the family $\mathcal{C}$ of circles is given by

$$(x^2 + y^2) + a_1 x + a_2 y + a_3 = 0 \tag{3}$$

with $C(\mathcal{C}) = \emptyset$. Hence, $\mathcal{C}$ is a free family.

The family $\mathcal{E}$ of ellipses is specified by the functional form

$$x^2 + a_1 xy + a_2 y^2 + a_3 x + a_4 y + a_5 = 0 \tag{4}$$

with

$$C(\mathcal{E}) = \{a_1^2 - 4a_2 < 0\}. \tag{5}$$

Hence $\mathcal{E}$ is not a free family.

Notice that with these definitions of $C(\mathcal{C})$ and $C(\mathcal{E})$, the families $\mathcal{C}$ and $\mathcal{E}$ contain imaginary circles and ellipses. Our algorithm does not output any such imaginary curves.

Remark 1. Besides lines, circles, ellipses, and hyperbolas, many mathematically interesting families of 2D curves can be expressed by using the above formulation. The family of lemniscates is defined by $(x^2+y^2)^2-2a(x^2-y^2)=0$ with the secondary condition $a>0$. This is a subfamily of Cassini's ovals, defined by $(x^2+y^2)^2-2a(x^2-y^2)=b$ with $a>0$. The family of Diocles's cissoids is defined by $(x-a)y^2=x^3$ $(a\neq 0)$, which can be written as $f[a_1](x,y)=a_1y^2+x^3-xy^2=0$, with a secondary condition $a_1\neq 0$. This family is extended to the family of generalized cissoids $f[\mathbf{a}](x,y)=-xy^2+a_1y^2+a_2x^3+a_3x^2+a_4x+a_5=0$, which contains many interesting curves, including Agnesi's witch and Descartes folium cartesii.

The above families of lemniscates and so on are neither translation-invariant nor rotation-invariant. If we want to deal with rotated copies of lemniscates, we have a nice expression $f[a_1,a_2](x,y)=(x^2+y^2)^2-a_1(x^2-y^2)-a_2xy=0$. The family of translated copies of generalized cissoids is expressed by $xy^2+a_1y^2+a_2xy+a_3y+a_4x^3+a_5x^2+a_6x+a_7=0$. However, for example, the family of translated copies of lemniscates needs many parameters and complicated secondary conditions, and hence is not suitable for a treatment using our framework.

Let D be a square region of size $N\times N$ containing G. Here we can observe the following fact: the curve $f(x,y)=0$ partitions D into a number of regions, so that the sign of the value of f is identical at every point in each such region. That is, the domain is partitioned into regions of positive signs and negative signs.

[**Definition 1**] Let f be an instance of a family $\mathcal{F}$ defined by equation (2). The planar curves obtained by shifting $f=0$ right, left, up, and down by 0.5 are denoted by $f_l=0, f_r=0, f_t=0$, and $f_b=0$, respectively. In other words, $f_r(x,y)\equiv f(x-0.5,y)$, for example. For notational convenience, $f_0(x,y)$ is defined to be $f(x,y)$. Five curves $f_\gamma(x,y)=0,\ \gamma\in\{0,l,r,t,b\}$, are referred to as the *characteristic curves* of f.

$region_r(f), region_l(f), region_t(f)$, and $region_b(f)$, defined below, are called the *right*, *left*, *top*, and *bottom* **regular image regions**, respectively.

$$region_\gamma(f)=\{(x,y)\in D \mid f_\gamma(x,y)\times f(x,y)\leq 0\}, \tag{6}$$

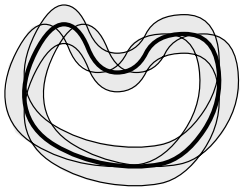

Figure 8: Regular image region of a curve defined by four characteristic curves.

where $\gamma \in \{r, l, t, b\}$. The union

$$region(f) = region_r(f) \cup region_l(f) \cup region_t(f) \cup region_b(f) \tag{7}$$

is simply called the *regular image region* of the curve $f = 0$ (see Figure 8).

Lemma 3. 1 *The set of regular grid points $RIm(f)$ with respect to f coincides with $G \cap region(f)$.*

Proof Let $p = (i, j)$ be a grid point in $region_r(f)$. Without loss of generality, assume that $f(p) > 0$. Then, $f(i + 0.5, j) \leq 0$ and the segment $((i, j), (i + 0.5, j))$ must intersect f an odd number of times. Thus, p is in $RIm(f)$. Conversely, if $((i, j), (i + 0.5, j))$ intersects f odd times, then it is easy to see that p is in $region_r(f)$. The other cases can be examined similarly. □

A curve family $f[a_1, ..., a_d](x, y)$ is continuous with respect to the parameters in the following sense.

Lemma 3. 2 *Consider any pair of parameter values $\mathbf{a} = [a_1, ..., a_d]$ and $\mathbf{b} = [b_1, ..., b_d]$ Suppose there is a point $u = (x_0, y_0)$ in the plane such that $f[a_1, ..., a_d](x_0, y_0) \geq 0$ and $f[b_1, ..., b_d](x_0, y_0) \leq 0$. Then, for any continuous path $\mathbf{p}(t)$ from $\mathbf{p}(0) = [a_1, ..., a_d]$ to $\mathbf{p}(1) = [b_1, ..., b_d]$ in the parameter space, there exists a value $0 \leq t_0 \leq 1$ such that $f[\mathbf{p}(t_0)](x_0, y_0) = 0$.*

Proof The function $f[s_1, ..., s_d](x_0, y_0)$ is a linear function with respect to s_i, and hence is continuous. Therefore, the lemma follows from the intermediate value theorem. □

3.2 Algorithm for a free family

In this subsection, we assume that $\mathcal{F}$ is a free family. Given digital component Q, we say that a curve f *fits* Q if $S \cap Int(region(f)) \subseteq Q \subseteq RIm(f)$, where $Int(region(f))$ means the interior of $region(f)$. For each eligible digital component, we will compute and output a curve fitting it.

Let $P = \{p_1, ..., p_m\}$ be a subset consisting of m points of S, and let $p_i = (x_i, y_i)$. Let π be a map from $\{1, 2, .., m\}$ to $\{0, r, l, t, b\}$. We consider the system of linear equations

$$f_{\pi(i)}[a_1, ..., a_d](x_i, y_i) = 0, \quad (i = 1, 2, ..., m)$$

with respect to parameters $a_1, .., a_d$. $V(P, \pi)$ denotes the subspace of the parameter space defined by this system of linear equations. The rank of (P, π) is the rank of the above system of linear equations. The pair (P, π) is called a *good pair* if the system is feasible and has rank m.

A curve $f[\mathbf{a}]$ is called a solution-curve of (P, π) if $\mathbf{a} \in V(P, \pi)$. We permit the case in which $P = \emptyset$, where $m = 0$, π is the empty map, and every curve becomes a solution-curve of (P, π).

For a regular digital component $Q \in S$, let m be the maximum integer such that a good pair (P, π) of rank m has a solution curve that fits Q. Since we permit $m = 0$, it is clear that the above m exists. This integer m is denoted by $m(Q)$, and is called the *contact-rank* of Q. (P, π) is called a saturated pair associated with Q if its rank is $m(Q)$.

For example, if $\mathcal{F}$ is the family of lines, given a line component Q containing at least two points, let $Q = RIm(l)$ for a line l. Then, we can rotate and slide l so that no point of Q lies outside the closure of the region of the line, and no point of $S - Q$ lies interior of the region, and obtain a line l' for which at least two points lie on the characteristic curves (indeed, lines). The contact-rank is two, and any pair of points on the characteristic curves form a saturated pair together with the associated π. As we can see in this example, P may contain points in $S - Q$. Similarly, we can see that the contact-rank with respect to the family of circles is three.

Note that $m(Q)$ is dependent on S, since we choose the above P from S. If S contains sufficiently large number of points, the contact-rank $m(Q)$ is almost always d for any Q

Unfortunately, it is not always the case, as shown in the following counterexample: Consider the family $f[a_1,..,a_d](x,y) = y + a_1 + a_2x + ... + a_dx^{d-1} = 0$ of 1-variable polynomial curves of degree at most $d-1$. For a usual point set, the contact-rank is d; however, if $S = \{(0,2i) : i = 1,2,...,n\}$, any characteristic curve goes through at most one point of S, and it cannot happen that more than one of f_t, f, and f_b intersect S for each f in the family. Hence, the contact-rank is at most three.

Lemma 3. 3 *Let $\mathcal{F}$ be a free family of planar curves, and let Q be an arbitrary regular digital component of $\mathcal{F}$. Let (P,π) be a saturated pair associated with Q, and let f and f' be any pair of solution-curves of it. Then, $RIm(f) = RIm(f')$, and they fit Q.*

Proof Let f_0 be a solution curve of (P,π) fitting Q, which must exist by definition. It suffices to show that $RIm(f) = RIm(f_0)$ and $RIm(f') = RIm(f_0)$. Thus, we assume that $f = f_0$. The vector of parameter values of f is denoted by $\mathbf{a}(f)$.

There exists a path $\mathbf{p}$ from $\mathbf{a}(f)$ to $\mathbf{a}(f')$ lying in $V = V(P,\pi)$. If $RIm(f') \neq RIm(f)$, there exists a point $v = (x_0,y_0)$ in the set difference of $RIm(f')$ and $RIm(f)$. It cannot happen that v is located on a characteristic curve of f, since any point lying on a characteristic curve of f must also lie on a characteristic curve of any curve corresponding to a parameter vector contained in V, because of the maximality of $m(Q)$.

Hence, $f_\gamma(x_0,y_0)f'_\gamma(x_0,y_0) \leq 0$ and $f_\gamma(x_0,y_0) \neq 0$ hold for a suitable γ in $\{0,r,l,t,b\}$. Thus, from Lemma 3. 2 , there exists a parameter value $\mathbf{a}(h(v))$ on $\mathbf{p}$ such that v is located on the associated characteristic curve $h(v)_\gamma$. If there is more than one such parameter value, we take the one nearest to $\mathbf{a}(f)$ on the path. We choose v so that $\mathbf{a}(h(v))$ is nearest to $\mathbf{a}(f)$ on the path. Then, $h(v)$ contains at least $m+1$ points on its characteristic curves, and fits Q. Furthermore, the associated equation is linearly independent of the system corresponding to (P,π). This contradicts the maximality of $m(Q)$. □

Corollary 3. 4 *Let $\mathcal{F}$ be a free family of planar curves, and let Q be an arbitrary regular digital component of $\mathcal{F}$. Let (P,π) be a saturated pair associated with Q, $p_i = (x_i,y_i)$ for $i = 1,2,..,m(Q)$, and let f be any solution-curve of it. Then, we can find a set of $d-m(Q)$ parameters satisfying the following property:*

For any arbitrary fixed values of the $d - m(Q)$ parameters, the system of equations

$$f[a_1, ..., a_d]_{\pi(i)}(x_i, y_i) = 0, \quad (i = 1, 2, ..., m(Q))$$

has a unique solution, and the corresponding curve g satisfies $RIm(g) = RIm(f)$.

Proof We have a feasible system consisting of $m(Q)$ linearly independent linear equations on parameters associated with (P, π). We can find $d - m$ unit vectors that we can add to the system to obtain a full-rank linear system of equations. Thus, if we set the associated parameters to have arbitrary fixed values, the system of linear equations defines a parameter vector $\mathbf{a}(g)$ associated with a curve g. Thus, the corollary follows from Lemma 3. 3 □

Theorem 3. 5 *We can detect all eligible digital components for a free family of curves $\mathcal{F}$ in $O(n^d + n \log n)$ time and $O(n)$ space without using non-algebraic operations.*

The rest of this section is devoted to proving Theorem 3. 5 . If $d = 1$, for each point $v \in S$, the region of the parameter a for $v \in RIm(f[a](x, y))$ becomes a union of at most four intervals in the parameter space. Thus, if we sort the endpoints of these intervals for all v, the problem can be solved by using a simple sweeping algorithm in $O(n \log n)$ time. From now on, we assume that $d \geq 2$, and design an $O(n^d)$ time and $O(n)$ space algorithm based on Corollary 3. 4 .

First, we detect all components whose contact-rank is not less than $d - 2$. For every subset P of size $d-2$ of S, we try all possibilities of the map π, and find all good pairs (P, π) of rank $d - 2$. Since the range of π is $\{0, l, r, t, b\}$, the above operation requires 5^{d-2} trials. For each good pair obtained, we perform the following procedure.

We consider the system of equations $f[a_1, ..., a_d]_{\pi(i)}(x_i, y_i) = 0$ $(i = 1, 2, ..., d - 2)$. We eliminate a set of $d - 2$ parameters by using the equations. Without loss of generality, we assume that the set is $\{a_1, a_2, \ldots, a_{d-2}\}$.

For a parameter vector $a \in V = V(P, \pi)$, we define $\tilde{a} = (a_{d-1}, a_d)$. For a point $p = (x_0, y_0) \in S - P$ and $\gamma \in \{0, l, r, t, b\}$, $f_\gamma[a](x_0, y_0) > 0$ (and < 0) defines a linear inequality on $\tilde{a}$ after elimination of the parameters $\{a_1, a_2, \ldots, a_{d-2}\}$. It may happen that the inequality is infeasible, or holds trivially. Otherwise, the inequality defines a halfplane in the (a_{d-1}, a_d)-plane.

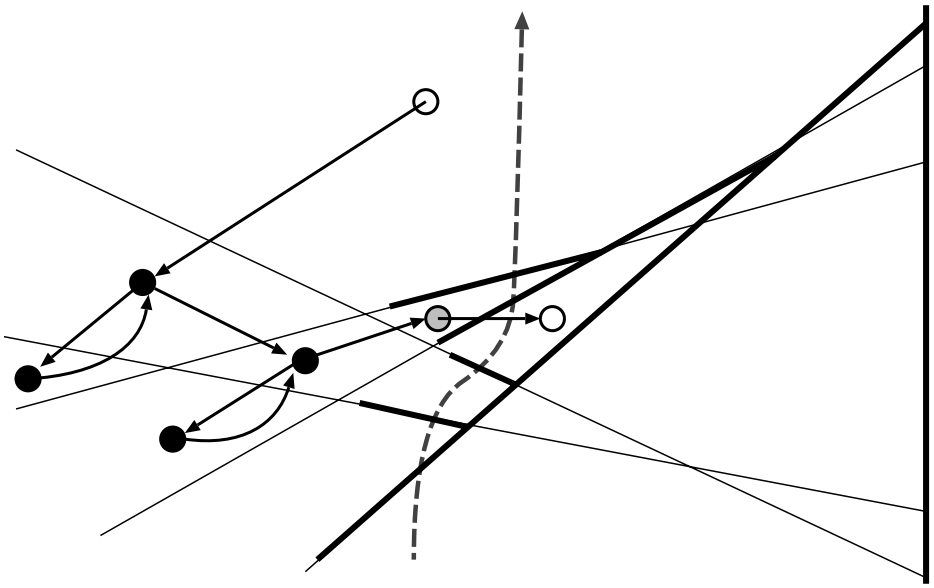

Figure 9: Topological Walk

The region of the parameter vectors a satisfying that that $RIm(f[a])$ contains the point p is bounded by five hyperplanes $f_\gamma[a](x_0, y_0) = 0$ for $\gamma \in \{0, l, r, t, b\}$ in the parameter space (parts of the hyperplanes may cross the interior of the region). Thus, the corresponding (not necessary connected) region $R(p)$ of $\tilde{a}$ is bounded by the boundary lines of the halfplanes obtained above. $R(p)$ is called the *characteristic region* of p (with respect to $\mathcal{F}$.) The characteristic region is just a generalization of a strip in the (u, v) plane defined in Section 2.1 for line detection, since the characteristic region of a point with respect to the family of lines is a strip.

The set of the bounding lines of these halfplanes for all p and γ forms an arrangement $\mathcal{A}$ of at most $5(n - d + 2)$ lines in the plane, and subdivides the plane into $O(n^2)$ cells. Thus, all we need to do is to enumerate all the cells of $\mathcal{A}$ that are contained in $k - d + 2$ or more characteristic regions. This task can be performed in $O(n^2)$ time and linear space by applying the Topological Walk algorithm of Asano et al. [2].

The Topological Walk algorithm dynamically generates a walk (the tour indicated by directed arcs in Figure 9) of the planar dual graph of the arrangement, keeping the information associated with the currently visited cell. In our application, we keep the list $L(c)$ of the characteristic regions containing the currently visited cell c by the walk. When the walk

passes through a line to visit a new cell, we examine the characteristic region bounded by this line, and update the list $L(c)$ as follows: If the region is currently in $L(c)$, and does not contain the new cell, we remove it from $L(c)$. If the region is not in $L(c)$, and does contain the new cell, we add it to $L(c)$. We store $L(c)$ as a table of size $O(n)$, so that the update can be done in $O(1)$ time. The algorithm finds all the cells such that the size of $L(c)$ becomes larger than or equal to $k - d + 2$ when the walk visits them.

The update of the other data structures necessary for the Topological Walk algorithm can be done in $O(1)$ amortized time per cell. We give a brief sketch of the algorithm below. See Asano-Tokuyama [5] for an outline, and Asano-Guibas-Tokuyama [2] for details of the algorithm. In [5], the algorithm is described as a dynamic depth-first search on the skeleton graph of the arrangement, instead of a dynamic walk in the dual graph.

While running the algorithm, we maintain a *sweep-pseudoline*, and we only know the part of the arrangement to the left of the sweep-pseudoline (Figure 9, dashed curve). We only keep the portion of the arrangement intersecting the current sweep-pseudoline, together with a dynamic data structure called the *upper horizon tree* (Figure 9, the tree consisting of bold edges) in order to attain an $O(n)$ working space requirement. A cell comes to the left of the current sweep-pseudoline when the rightmost vertex of the cell is deleted from the upper horizon tree. We report each eligible cell when it comes to the left of the sweep pseudoline. In Figure 9, the cells associated with black nodes have been already reported (if they are eligible), and the cell associated with a shaded node is currently being reported.

Since the number of good pairs (P, π) of rank $d - 2$ is $O(n^{d-2})$, the algorithm requires $O(n^d)$ time and $O(n)$ space to compute all regular image components whose contact-rank is not less than $d - 2$.

It is an easier task to compute regular image components with lower contact-rank. For each $m < d - 2$, we find all good pairs of rank m. For each good pair, find the set of $d - m$ parameters of Corollary 3. 4 , set the values of these parameters to zero, and compute the associated curve and its regular image. This takes a total of $O(n^{d-2})$ time.

Remark 2. Bentley-Ottmann's sweep line algorithm [22] is a popular method for sweeping planar objects. Unfortunately, since a sweep line intersects $\Omega(n)$ cells, each of which has a list of size $\Omega(n)$, a naive implementation requires $\Omega(n^2)$ space. The authors do not know how

to modify the sweep line algorithm so that it solves the above cell-finding task in $O(n^2 \log n)$ time with $O(n)$ space.

Topological Sweep [10] is another famous method for sweeping an arrangement of lines. The original implementation of the Topological Sweep algorithm keeps $O(n)$ cells as well as two trees, and thus the total table size of $L(C)$ for those cells becomes $\Omega(n^2)$ if we naively apply the implementation to the curve detection. The Topological Walk algorithm can be considered a variant of the Topological Sweep algorithm, in which we reduce the space complexity to $O(n)$.

3.3 Algorithm for a general case

Next, we consider the case in which $C(\mathcal{F})$ is not empty. Let $V(C(\mathcal{F}))$ be the region in the parameter space satisfying $C(\mathcal{F})$. This region contains the parameter vectors corresponding to the curves in $\mathcal{F}$. Recall that our algorithm for the free family generates an arrangement of lines in a two-dimensional subspace of the parameter space, and finds its cells corresponding to eligible components. In the general case, we need to detect whether each of the cells intersects $V(C(\mathcal{F}))$ or not, and report a point in the intersection if exists. We will show later that if we have a point in each connected component of the intersection of $V(C\mathcal{F})$ with each line of the arrangement, and also a point in each connected component of the intersection of $V(C\mathcal{F})$ with the plane, the above detection can be done in $O(1)$ amortized time per cell.

We also need to deal with the intersection of $V(C\mathcal{F})$ and higher dimensional subspaces. This motivates us the following assumption:

Assumption 1: For any subspace W of the parameter space, we can compute a set of $O(1)$ points of $V(C\mathcal{F}) \cap W$ in such a way that this set contains at least one point from each connected component of $V(C(\mathcal{F})) \cap W$; Moreover, the computation time T is small.

The computation time T is always independent of n, and is a small constant for popular families of curves; Indeed, we need not use non-algebraic operations for families such as ellipses, hyperbolas, parabolas, cissoids, and Cassini's ovals. However, if the secondary condition $C(\mathcal{F})$ is very complicated, T may be too large to ignore as a constant. Moreover,

we may be forced to use non-algebraic operations, so that our feature 2 is violated if the above computation is executed for a large number of subspaces W. We explicitly describe the contribution of T in time complexities when we state theorems, but ignore it elsewhere.

Lemma 3. 6 *Let $\mathcal{F}$ be a family of planar curves, and let Q be an arbitrary regular digital component of $\mathcal{F}$. Let (P, π) be a saturated pair associated with Q, and let f be its solution curve fitting Q. Let U be the connected component containing $a(f)$ (the parameter associated with f) in $V(C(\mathcal{F})) \cap V(P, \pi)$. Then, for any f' such that $a(f') \in U$, $RIm(f) = RIm(f')$.*

Proof Analogous to the proof of Lemma 3. 3 . □

Theorem 3. 7 *We can detect all the eligible digital components for a family of curves $\mathcal{F}$ in $O(n^d + n^{d-1}T + n \log n)$ time and $O(n)$ space, where T is the constant given in Assumption 1.*

Proof We consider only the cases in which $d \geq 2$ (it is easy if $d = 1$). The algorithm is almost the same as the one in Theorem 3. 5 . We find all the good pairs (P, π) of rank $d-2$.

Let us fix a good pair (P, π). We compute the 2-dimensional space $V = V(P, \pi)$, and as preprocess, we use Assumption 1 to compute a set of $O(1)$ points containing at least one point from each connected component of the feasible region $V(C(\mathcal{F})) \cap V$. We also preprocess each line L in the arrangement given in the proof of Theorem 3. 5 , so that we have a set of $O(1)$ points containing at least one point from each connected component of $V(C(\mathcal{F})) \cap L$. The preprocessing can be done in $O(n)$ space and $O(nT)$ time, because of Assumption 1.

A cell c of the arrangement intersects the feasible region either if it contains a connected component of $V(C(\mathcal{F})) \cap V$ or if an edge intersects the feasible region. In the former case, c must contain a precomputed point. An edge e on a line L of the arrangement intersects the feasible region either if it contains a connected component of $V(C(\mathcal{F})) \cap L$, or if one of the endpoints of the edge is in the feasible region. In the former case, e must contain a precomputed point.

We now invoke the Topological Walk algorithm, checking that precomputed points are contained in each visited cell and edge. Since the sum of the squares of the complexities of

cells is $O(n^2)$, this check can be done in $O(n^2)$ time for all the cells and edges. Then, we can output all the cells both intersecting W and contained in at least $k-d+2$ characteristic regions. Since we examine $O(n^{d-2})$ good pairs, the total time complexity is $O(n^d + n^{d-1}T)$.

For the digital components whose contact-rank is $m < d-2$, we compute all good pairs of rank m. For each good pair, we compute the parameter space V associated with it. Then, we compute the set of $O(1)$ points of $V(C(\mathcal{F})) \cap V$ given in Assumption 1, and compute the curve $f[a]$ for each such point a. This can be done $O(n^{d-2}T)$ time. □

4 α-sensitive curve detection

Although the algorithm in the previous section outputs all digital components in $O(n^d)$ time, the time complexity is often too expensive for practical use, especially if d is large. Also, the number of output eligible components may be large even if the threshold is high, unlike in the exact fitting problem [11], which computes the lines containing many data points. In this section, therefore, we consider cheaper algorithms that output an α-sensitive family.

Let us consider the space R^d of the parameters $(a_1, ..., a_d)$. The condition that an edge point $p = (x_0, y_0)$ lies on $f_\gamma(x, y)$ defines a hyperplane in R^d for each $f \in \mathcal{F}$ and $\gamma \in \{r, l, t, b, 0\}$. Thus, we have an arrangement of $5n$ hyperplanes corresponding to the point set S. An *r-cutting* of n hyperplanes is a space subdivision, such that at most n/r hyperplanes intersect each cell of the subdivision. The following theorem is known [8]:

Theorem 4. 1 *If d is a constant, there exists an r-cutting of size $O(r^d)$, which can be computed in $O(nr^{d-1})$ randomized time, together with the set of hyperplanes intersecting each cell.*

By using r-cutting, we can obtain the following algorithm, which is efficient if the threshold k is large, as is often the case in practice:

Theorem 4. 2 *An α-sensitive family containing $O((t/(1-\alpha))^d)$ curves can be computed in $O((t/(1-\alpha))^{d-1}n + (t/(1-\alpha))^d T)$ randomized time, where T is the constant given in Assumption 1, and $t = n/k$.*

Proof For simplicity, we first assume that the input family of the curves is a free family. We set $r = 5n/(1-\alpha)k = 5t/(1-\alpha)$, compute an r-cutting, select a representative point $v(c)$ (for example, a vertex) in each cell c, and output the set of all curves associated with the selected points, whose digital images contain at least αk points (counting points in the digital images can be done in $O(nr^{d-1})$ time in total).

For any digital curve component P, consider the associated point p in the space of parameters. The point p is located in a cell c of the cutting. Let $Q(c)$ be the digital curve component associated with the representative point $v(c)$. From the definition of the cutting, $P \setminus Q(c)$ contains at most $5n/r = (1-\alpha)k$ points. Therefore, the output set is an α-sensitive family of size $O((t/(1-\alpha))^d)$. This proves Theorem 4. 2 for a free family.

For the general case, we must choose $v(c)$ from the point set (if it is not empty) that satisfies the secondary condition $C(\mathcal{F})$. However, since a cell c of a cutting is a simplex, we can detect whether c has a point satisfying $C(\mathcal{F})$, and compute $v(c)$ (if one exists) in $O(T)$ time from our Assumption 1. Since there are $O((t/(1-\alpha))^d)$ cells, the time complexity becomes $O((t/(1-\alpha))^d T)$. □

Remark 3. We call a cell c in the r-cutting *active* if the representative point $v(c)$ is associated with a digital image containing at least αk points. If we further continue searching in the active cells of the cutting, we can obtain an exact algorithm that is sensitive to the number of eligible components with respect to the threshold αk. We may combine this algorithm with the Hough transformation method by subdividing the active cells of the cutting into buckets.

Remark 4. It is sometimes advantageous to output neither each digital component nor the corresponding curve equation, but to aggregate information on digital components in a compact form by outputting the region of the parameter space defining $\mathcal{F}$ for which a digital component exists. We can output the set of all active cells in the cutting together with the set of hyperplanes intersecting each of the cells. Then, the output size becomes $O((t/(1-\alpha))^{d-1}n)$, and if we want more digital images than the α-sensitive family of Theorem 4. 2 , we can search in each cell recursively.

Remark 5. Instead of the sophisticated Chazelle's cutting, we can apply random sampling to obtain a Monte-Carlo-type randomized algorithm that outputs an α-sensitive family of

size $O((t \log n/(1-\alpha))^d)$ in $O((t \log n/(1-\alpha))^{d-1}(n+T))$ time with high probability.

The above approximation method is useful only if the threshold k is large. However, if $n \gg N$, we can efficiently compute a 1/2-sensitive family of digital *line* components even if the threshold k is small.

Theorem 4. 3 *There exists a 1/2-sensitive family consisting of $O(N^2)$ digital line components. Such a family, together with the size of digital components of members of the family, can be computed in $O(N^{4/3}n^{2/3} \log n)$ time.*

Proof. Consider the family of $O(N^2)$ lines penetrating two grid points located on the boundary of the grid frame. It is not difficult to show that the associated family of digital components contains a 1/2-sensitive family. We can compute the size of a digital component $S \cap Im(f)$ by applying slab-range searching, because $Im(f)$ is a slab. If we apply a slab-range search data structure with $O(N^{4/3}n^{2/3})$ space [18], the query time is $O((n/N)^{2/3} \log n)$ and the preprocessing time is $O(N^{4/3}n^{2/3} \log n)$. □

5 Family of similar polygonal chains

In this section, we consider the problem of detecting all digital components of polygonal chains similar to a given polygonal chain. In other words, we detect copies of the given polygonal chain in the digital image transformed by translation, rotation, scaling, and reflection. We ignore reflection, since reflection does not affect asymptotic computational complexity.

Let P be a polygonal chain with m vertices. From the basic linear algebra, the family of polygonal chains obtained by translating, rotating, and scaling P is $\mathcal{F} = \{P(a_1, a_2, a_3, a_4) | a_i \in \mathcal{R} \text{ for } i = 1, 2, 3, 4\}$, where $P(a_1, a_2, a_3, a_4) = \{(x, y) : (a_1x + a_2y + a_3, a_2x - a_1y + a_4) \in P\}$.

In order to compute all eligible digital components of members of $\mathcal{F}$ in a set S of n edge points, it suffices to compute the polygonal chains in each of which the regular image region contains at least four edge points of S on the boundary. A pair of an edge point p of S and an edge e of P is called a contact pair of $P(a_1, a_2, a_3, a_4)$ if p lies on the part associated with e of the boundary of the regular image region.

The edge e can be defined by $y = cx + d, s \leq x \leq t$. Accordingly, the corresponding edge in $P(a_1, a_2, a_3, a_4)$ is defined by $a_2x - a_1y + a_4 = c(a_1x + a_2y + a_3) + d, s \leq a_1x + a_2y + a_3 \leq t$. Since $p = (i, j)$ and e is a contact pair, the above edge must go through a point (say, $(i, j + 0.5)$) given by translating p vertically or horizontally with 0.5. The condition can be represented as the following system of linear inequalities with respect to a_1, a_2, a_3, and a_4:

$$ia_2 - (j + 0.5)a_1 + a_4 = c[ia_1 + (j + 0.5)a_2 + a_3] + d$$
$$s \leq ia_1 + (j + 0.5)a_2 + a_3 \leq t.$$

We fix two contact pairs among the $O(nm)$ possible contact pairs. Thus, we can eliminate two parameters, a_3 and a_4, from the above systems of inequalities. Then, gathering all bounding lines of halfplanes corresponding to the $4nm$ inequalities in these systems, we have an arrangement of $4nm$ lines in a plane. As in the previous sections, we can then find all eligible components (for each fixed pair of contact pairs) in $O(n^2m^2)$ time by applying the Topological Walk algorithm in this arrangement. Since there are $O(n^2m^2)$ choices of pairs of contact pairs, all eligible digital components in $\mathcal{F}$ can be computed in $O((mn)^4)$ time and $O(mn)$ space.

Furthermore, analogously to Theorem 4. 2 , we obtain the following:

Theorem 5. 1 *If the threshold is k, we can compute an α-sensitive family in $O((\frac{mn}{(1-\alpha)k})^3 mn)$ randomized time.*

6 Concluding remarks

Like the circle and ellipse, the parabola is an important curve and a fundamental element of the quadratic splines frequently used in CAD systems. A family of parabolas can be described by the same equations as (4) with a constraint of $4a_1^2 = a_2$. Our scheme gives an $O(n^5)$-time algorithm for detecting all eligible parabolic components. However, if we are allowed to solve a quadratic equation in constant time (or less than $O(\log n)$ time), we can eliminate a_4 and a_5 by using two points of S, and eliminate a_2 by using $4a_1^2 = a_2$. Each third point will then define an axis parallel parabola in a_1-a_3 space. Thus, searching in the arrangement of parabolas by walking on each parabola separately, we can obtain

an $O(n^4 \log n)$-time and linear-space algorithm. If we can use $O(n^2)$ space, we can further improve the time complexity to $O(n^4)$ time by applying a randomized algorithm [19] to construct an arrangement of algebraic curves.

We can also detect digital components of a polynomial surface in the three-dimensional grid space by using our framework. For example, since a plane is expressed by using three parameters, we can obtain an $O(n^3)$-time and $O(n)$-space algorithm to detect all digital planes.

As is shown in the literature [23], there may be various ways of defining a digital image of a planar curve. We have adopted a discretization using edge points, which is a natural and popular method in computer vision. Another natural idea is to define the digital image to be the set of grid points from each of which the Euclidean distance to the curve is not more than 0.5. Unfortunately, it seems that the problem becomes harder in general under this definition; in particular, for higher-order curves such as ellipses, the defining equation for the trajectory of points at a distance of 0.5 from a curve (say, ellipse) is very complicated.

References

[1] T. Asano, Unpublished experimental result (1994)

[2] T. Asano, L. Guibas, and T. Tokuyama, Walking in an Arrangement Topologically, *Int. J. of Comput. Geom. and Appl.* 4-2 (1994), pp. 123-151.

[3] T. Asano and N. Katoh, Number Theory Helps Line Detection in Digital Images, *Proc. ISAAC'93, Springer LNCS 762,* (1993), pp.313-322.

[4] T. Asano and N. Katoh, Variants for the Hough Transform for Line Detection, *Computational Geometry, Theory and Appl.,* 6 (1996), pp.231-252.

[5] T. Asano and T. Tokuyama, Topological Walk Revisited. *Proc. 6th Canadian Conference on Computational Geometry* (1994), pp.1-6.

[6] D.H. Ballard, Generalizing the Hough Transform to Detect Arbitrary Shapes, *Pattern Recognition*, 13-2 (1981), pp. 111-122.

[7] C.M. Brown, Inherent Bias and Noise in the Hough Transform, *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-5, No.5 (1983), pp. 493-505.

[8] B. Chazelle, An Optimal Convex Hull Algorithm and New Results on Cuttings, *Proc. 32nd IEEE Symp. on Foundation of Computer Science*, (1991), pp.29-38.

[9] 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.

[10] H. Edelsbrunner and L.J. Guibas, Topologically Sweeping an Arrangement, *J. Comp. and Sys. Sci.*, 38 (1989), pp. 165-194.

[11] L. Guibas, M. Overmars, and J-M. Robert, The Exact Fitting Problem for Points, *Proc. 3rd CCCG*, (1991), pp.171-174.

[12] P.J. Heffernan and S. Schirra. Approximate Decision Algorithms for Point Set Congruence, *Proc. 8th Symposium on Computational Geometry*, (1992), pp. 93-101.

[13] P.V.C. Hough, Method and Means for Recognizing Complex Patterns, *US Patent 3069654*, December 18, (1962).

[14] D.P. Huttenlocher and K. Kedem, Computing the Minimum Housdorff Distance of Point Sets under Translation, *Proc. 6th Symposium on Computational Geometry*, (1990), pp. 340-349.

[15] K. Imai, S. Sumino, and H. Imai, Minimax Geometric Fitting of Two Corresponding Sets of Points, *Proc. 5th Symposium on Computational Geometry*, (1989), pp. 276-282.

[16] C. Kimme, D.H. Ballard, and J. Sklansky, Finding Circles by an Array of Accumulators, *Comm. ACM*, 18 (1975), pp. 120-122.

[17] H. Maitre, Contribution to the Prediction of Performance of the Hough Transform, *IEEE Trans. Pattern Anal. Machine Intell.*, PAMI-8, No.5 (1986), pp. 669-674.

[18] J. Matoušek, Range Searching with Efficient Hierarchical Cuttings, *Proc. 8th ACM Computational Geometry*, (1992), pp.276-285.

[19] K. Mulmuley, A Fast Planar Partition Algorithm II, *J. ACM* 38, (1991), pp. 74-103.

[20] C. F. Olson, Decomposition of the Hough Transform: Curve Detection with Efficient Error Propagation, *Proc. ECCV96, Springer LNCS* **1064**, (1996), pp. 263-272.

[21] P. L. Palmer, J. Kittler, and M. Petrou, An Optimized Line Finder Using a Hough Transform Algorithm, *Computer Vision and Image Understanding* **67-1** (1997), pp. 1-23.

[22] F. P. Preparata and M. I. Shamos, *Computational Geometry, an Introduction,* Texts and Monographs in Computer Science, Springer-Verlag, 1985 (2nd ed 1988).

[23] A. Rosenfeld and R.A. Melter, Digital Geometry, Tech. Report, CAR–323, Center for Automatic Research, University of Maryland, 1987.

[24] P. Sauer, On the Recognition of Digital Circles in Linear Time, *Computational Geometry: Theory and Applications* , 2 (1993), pp. 287-302.

[25] J. Sklansky, On the Hough Technique for Curve Detection, *IEEE Trans. Comp.*, C-27, 10 (1978), pp. 923-926.

[26] J. T. Schwartz and M. Sharir, Algorithmic Motion Planning in Robotics, *in Handbook of Theoretical Computer Science* (ed. J. van Leeuwen), pp. 391-430 (1990).

[27] S. Tsuji and F. Matsumoto, Detection of Ellipses by a Modified Hough Transformation, *IEEE Trans. Comp.*, C-27, No. 8 (1978), pp. 777-781.

[28] L. Xu and E. Oja, Randomized Hough Transform (RHT): Basic Mechanism, Algorithms, and Computational Complexities, *CVGIP: Image Understanding*, **57-2** (1993), pp. 131-154.