Fitting a two-joint orthogonal chain to a point set

J. M. Díaz-Báñez[*] M. A. López[†] M. Mora[‡] C. Seara[§] I. Ventura[¶]

15th December 2009

Abstract

We study the problem of fitting a two-joint orthogonal polygonal chain to a set S of n points in the plane, where the objective function is to minimize the maximum orthogonal distance from S to the chain. We show that this problem can be solved in $\Theta(n)$ time if the orientation of the chain is fixed, and in $\Theta(n \log n)$ time when the orientation is not a priori known. We also consider some variations of the problem in three-dimensions where a polygonal chain is interpreted as a configuration of orthogonal planes. In this case we obtain $O(n)$ and $O(n \log n)$ time algorithms depending on which plane orientations are fixed.

1 Introduction and definitions

Fitting a curve of a certain type to a given point set in the plane is a fundamental problem with applications in fields as diverse as statistics, computer graphics, and artificial intelligence. A special case of this problem is the so called *polygonal approximation problem* or *polygonal fitting problem*, where a polygonal chain with k corners or joints is fitted to a data set so as to minimize the approximation error according to some agreed upon metric. This problem is closely related to that of approximating a piecewise-linear curve with n edges by one with fewer edges, except that the input is now also a chain. Applications of this problem arise in cartography, pattern recognition, and graphic design [6, 9, 22], and has received much attention in computational geometry [2, 7, 15, 18, 24].

In the *Min-Max problem* a polygonal chain with k joints is fitted to a data set with the goal of minimizing the maximum vertical distance from the input points to the chain. This problem was first posed by Hakimi and Schmeichel [16] and solved in $O(n^2 \log n)$ time. The complexity has since been improved, first by Wang et al. [27] to $O(n^2)$ time and then by Goodrich [13] to $O(n \log n)$ time.

We consider the case in which the approximating curve is an *orthogonal polygonal chain*, i.e., a chain of consecutive orthogonal line segments where the extreme segments are half-lines with the same

[*]Departamento de Matemática Aplicada II, Universidad de Sevilla, Spain, dbanez@us.es. Partially supported by grants BFM2003-04062 and MTM2006-03909.

[†]Department of Mathematics, University of Denver, 2360 South Gaylord Street, Denver, CO 80208, USA, mlopez@du.edu.

[‡]Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain, merce.mora@upc.edu. Supported by projects MEC MTM2009-07242 and Gen. Cat. DGR 2009GR1040.

[§]Departament de Matemàtica Aplicada II, Universitat Politècnica de Catalunya, Spain, carlos.seara@upc.edu. Supported by projects MEC MTM2009-07242 and Gen. Cat. DGR 2009GR1040.

[¶]Departamento de Matemática Aplicada II, Universidad de Sevilla, Spain, iventura@us.es. Partially supported by grants BFM2003-04062 and MTM2006-03909.

slope, *the slope of the orthogonal polygonal chain.* The case in which this slope is given was first solved by Díaz-Báñez and Mesa [11] in $O(n^2 \log n)$ time, and subsequently improved by Wang [26] to $O(n^2)$ time, and by Lopez and Mayster [21] to $\min\{n^2, nk \log n\}$ time. Very recently, Fournier and Vigneron [12] give an $O(n)$ time algorithm if the points are sorted by their x-coordinates, and an $O(n \log n)$ time algorithm for the unsorted case. These authors give an $\Omega(n \log k)$ lower bound for the decision problem and thus prove the optimality of their algorithm for the unsorted case when $k = \Theta(n)$

Let $S = \{p_1, \dots, p_n\}$ be a set of n points in the plane in general position, i.e., no three points on a line. When $k \geq 1$ and $0 \leq \theta < 180$, a *k-orthogonal polygonal chain with orientation θ, $\mathcal{O}_{k,\theta}$,* is a chain of $2k-1$ consecutive orthogonal segments such that the extreme segments are in fact half-lines with slope $\tan(\theta)$. Thus, $\mathcal{O}_{k,\theta}$ consists of k segments with slope $\tan(\theta)$ and $k-1$ segments with slope $\tan(\theta+90)$ (Figure 1). Clearly, $\mathcal{O}_{k,\theta}$ is always monotone with respect to its orientation.

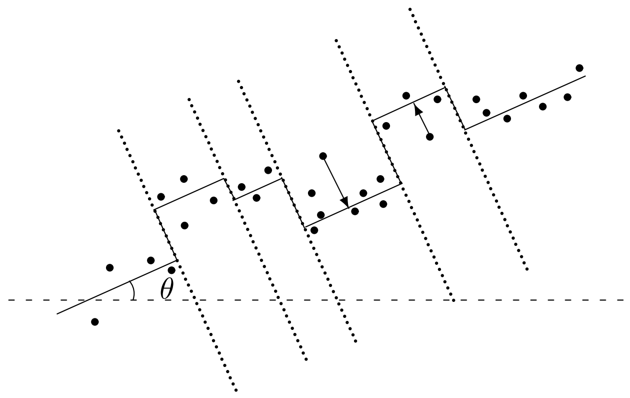

Figure 1: A 6-orthogonal polygonal chain dividing the plane into 6 strips.

We deal with the problem of fitting a k-orthogonal polygonal chain $\mathcal{O}_{k,\theta}$ to the set S. Fitting $\mathcal{O}_{k,\theta}$ to S means to locate θ-oriented segments $s_i(\theta)$, $i = 1, \dots, k$, according to a given optimization criterion. We consider the Min-Max criterion, illustrated in Figure 1 and defined as follows. Let $l_i(\theta)$ be the line passing through $p_i \in S$ with orientation $\theta + 90$. The fitting distance between p_i and $\mathcal{O}_{k,\theta}$, denoted by $d_f(p_i, \mathcal{O}_{k,\theta})$, is given by

$$d_f(p_i, \mathcal{O}_{k,\theta}) = \min_{p \in l_i(\theta) \cap \mathcal{O}_{k,\theta}} d(p_i, p).$$

Notice that d_f is not the Euclidean distance. However, we can assume that this distance is the Euclidean distance between p_i and a point on a segment with orientation θ in $\mathcal{O}_{k,\theta}$. The *error tolerance* of $\mathcal{O}_{k,\theta}$ with respect to S, denoted by $\mu(\mathcal{O}_{k,\theta}, S)$, is the maximum fitting distance between the points of S and $\mathcal{O}_{k,\theta}$, i.e.,

$$\mu(\mathcal{O}_{k,\theta}, S) = \max_{p_i \in S} d_f(p_i, \mathcal{O}_{k,\theta}).$$

Definition 1 *The k-fitting problem for S with the Min-Max criterion consists of finding an orthogonal polygonal chain $\mathcal{O}_{k,\theta}$ such that its error tolerance $\mu(\mathcal{O}_{k,\theta}, S)$ is minimized.*

Notice that if the orientation of $\mathcal{O}_{k,\theta}$ is fixed, for example $\theta = 0$, then the k-fitting problem consists of finding an x-monotone rectilinear path formed by $2k-1$ segments with minimum error tolerance where the fitting distance is just the vertical distance [11].

We focus here on the case where k is small, in fact $k = 2$, and the points in S are not sorted. We study the 2-fitting problem for S with fixed orientation (the *oriented 2-fitting problem*) and the problem of finding the best orientation for fitting a two-joint orthogonal polygonal chain to S (the *un-oriented 2-fitting problem*). We also consider the extension of the problem to three-dimensions where an orthogonal polygonal chain is a configuration of orthogonal planes. See Chen and Wang [8] for recent results on some variants of this problem including NP-hardness results in three dimensions.

Outline of the paper. In Section 2 we study the oriented fitting problem in the plane. In Section 3 we study the un-oriented 2-fitting problem in the plane. Finally, in Section 4, we study the oriented 2-fitting problem in three-dimensions.

2 The oriented fitting problem

In this section we consider the oriented k-fitting problem for S, i.e., the case where the orientation θ of the k-orthogonal chain that fits S is fixed. Without loss of generality we assume that $\theta = 0$. Thus, we are looking for an x-monotone rectilinear path, $\mathcal{O}_{k,0}$ or $\mathcal{O}_k$, consisting of an alternating sequence of k horizontal and $k-1$ vertical segments with minimum error tolerance.

Often, the algorithms proposed in the literature for these kind of fitting problem assume that the input points are given in sorted order. Recently, Fournier and Vigneron [12] give an $O(n \log n)$ time algorithm for the oriented k-fitting problem when the points are unsorted and prove its optimality when $k = \Theta(n)$. The running time of the algorithm from Lopez and Mayster [21] is $\min\{n^2, nk \log n\}$ which is $O(n \log n)$ when k is a constant. For the sorted case, Fournier and Vigneron [12] present an optimal $O(n)$ time algorithm, and an $\Omega(n \log k)$ time lower bound for the decision problem for the unsorted case. Here we consider the oriented k-fitting problem for the case $k = 2$ for the unsorted case.

Let $S = \{p_1, \ldots, p_n\}$, where $p_i = (x_i, y_i)$. We can compute $y_{\max} = \max\{y_1, \ldots, y_n\}$ and $y_{\min} = \min\{y_1, \ldots, y_n\}$ in linear time. The oriented 1-fitting problem then is solved in $O(n)$ time by finding the horizontal line $y = (y_{\max} + y_{\min})/2$.

Let $\mathcal{O}_2$ denote an optimal solution to the oriented 2-fitting problem for a set S. Then $\mathcal{O}_2$ consists of two horizontal half-lines joined by a vertical segment contained in a vertical line ℓ^* which partitions S into subsets S_1 and S_2, namely the points of S to the left and to the right of ℓ^* respectively. Since ℓ^* must minimize the maximum error tolerance of S_1 and S_2, the following is apparent.

Lemma 1 *Line ℓ^* separates the two points in S with y-coordinates $y_{\min}$ and $y_{\max}$.*

Linear time algorithm for oriented 2-fitting.

Any vertical line ℓ between two points of S induces a candidate solution to the oriented 2-fitting problem whose cost is given by $\max\{\epsilon_1, \epsilon_2\}$, where ϵ_1 (resp. ϵ_2) denotes the tolerance of the subset of S to the left (resp. right) of ℓ. Our algorithm performs a binary search based on the following observation:

Lemma 2 *If ℓ is not optimal and $\epsilon_1 < \epsilon_2$ (resp. $\epsilon_1 > \epsilon_2$) then ℓ^* lies to the right (resp. left) of ℓ.*

We now outline the algorithm. Let ℓ denote the vertical line through the median x-coordinate of S. Partition S into subsets S_1 and S_2 to the left and right of ℓ, respectively. Compute also the tolerances ϵ_1 of S_1 and ϵ_2 of S_2, and store the two witness pairs of points responsible for the tolerances. All this can be computed in $O(n)$ time [10]. If $\epsilon_1 = \epsilon_2$, stop the algorithm, as $\ell^* = \ell$. If $\epsilon_1 < \epsilon_2$, in $O(n/2)$ time compute the median of S_2, reset ℓ as the vertical line at this median value, compute the subsets S_2' and S_2'' of the bipartition of S_2 produced by the new median, and compute the tolerances ϵ_2' and ϵ_2'' of S_2' and S_2''. The next move of ℓ (left or right) is determined based on the maximum of ϵ_2'' and ϵ where ϵ is the error tolerance of all points to the left of ℓ, which can be obtained in constant time by the two witness points for ϵ_1 and the two witness points for ϵ_2'. Store the tolerance values and the corresponding witness pairs as temporary values. Next compute and update the tolerances once we know the next move of ℓ. (If $\epsilon_1 > \epsilon_2$ we proceed in a symmetric way). Compare the new tolerance values and continue recursively translating the line ℓ left or right by computing the new median of a subset with half of the points and updating the new tolerance values (left and right) from the old ones. At all times we have two unions at the extremes and an unknown zone in between containing at most two strips. The points in the zone are known but, in general, are not sorted.

Clearly, the time complexity of the algorithm is $T(n) = T(n/2) + O(n) = O(n)$, using a linear time median finding algorithm [10]. By Lemma 1 the optimal line ℓ^* will be located between the two points in S with y-coordinates $y_{\min}$ and $y_{\max}$. The algorithm stops when either the tolerance values ϵ_1 and ϵ_2 are equal, or when translating ℓ left and right the bigger of the two tolerances switches sides. In this last case the solution will be the best of the two. Since the algorithm performs a binary search on a unimodal function, the method is correct. Notice that the solution (position of line ℓ or bipartition of S) is not unique because in an optimal solution some points can belong to S_1 or S_2 without changing the solution. Notice also that our algorithm does not sort the input points. We have the following result.

Theorem 1 *The oriented 2-fitting problem can be solved in $\Theta(n)$ time and space.*

By the $\Omega(n \log k)$ lower bound for the decisional oriented k-fitting problem [12] in the unsorted case, it is clear that if $k = \omega(1)$ there is no linear time algorithm. Thus, we raise the following open question more from a theoretical than from a practical point of view.

Open problem 1 *For which values of $k \geq 3$ does there exist a linear time algorithm for the oriented k-fitting problem?*

2.1 An $O(n \log n)$-time algorithm

We now describe an $O(n \log n)$-time algorithm for the oriented 2-fitting problem whose interest derives not from its time complexity but from the fact that it will be used as a preprocessing step in the $O(n \log n)$-time algorithm for the un-oriented 2-fitting problem discussed in Section 3. We start by introducing a basic tool.

In [19, 23] the maxima problem for a point set S in the plane is considered. Concretely, given two points $p_i, p_j \in S$, the following dominance relation is established: *p_i dominates p_j ($p_j \prec p_i$), if $x_j \leq x_i$ and $y_j \leq y_i$.* The relation $\prec$ is a partial order in S. A point $p_i \in S$ is called *maximal* if there does not exists $p_j \in S$ such that $i \neq j$ and $p_i \prec p_j$. The maxima problem consists of finding all the maximal points of S under dominance. One can formulate maxima problems for each quadrant in the plane. We are interested in the set of maximal points for S with respect to the four quadrants which form the *rectilinear convex hull* of S, also known as *orthogonal convex hull* (Figure 2). Each set of maximal points has a *total ordering* that can be stored in a height balanced search tree [23].

Theorem 2 [19] *The maxima problem for S with respect to any of the four quadrants can be solved optimally in $\Theta(n \log n)$ time and $O(n)$ space.*

$O(n \log n)$-time algorithm for oriented 2-fitting.

1. Let $x_{\max}$, $x_{\min}$, $y_{\max}$, and $y_{\min}$ denote the respective maximum and minimum of the x and y-coordinates of the points in S. Without loss of generality assume $x_{\min} = y_{\min} = 0$, $x_{\max} = c$, and $p_1 = (0, y_1)$, $p_n = (c, y_n)$, $p_i = (x_i, y_{\max})$, and $p_j = (x_j, 0)$ (Figure 2), i.e., the rectangle with corners at $(0, 0)$ and $(c, y_{\max})$ is the axis-parallel bounding box for S. Assume further that p_i is strictly to the left of p_j and thus, by Lemma 1, the vertical line ℓ lies between p_i and p_j. (If both have the same x-coordinate the solution is trivial.)

   By Theorem 2, in $O(n \log n)$ time we can compute the *rectilinear convex hull* of S formed by the staircases structure as in Figure 2. Notice that staircases of opposite quadrants can intersect. Since p_i is to the left of p_j, then the third quadrant staircase gives the lower point on the left of ℓ and the first quadrant staircase gives the upper point on the right of ℓ.

2. By Lemma 1 the vertical line $\ell := (x = a)$ is between x_i and x_j. In order to find its correct location we do a *binary search* over the points in the staircase structure (first and third quadrant) in $O(\log n)$ time getting the best balance between the error tolerance on the left and on the right side of ℓ, i.e.,

$$\min_{x_i \le a < x_j} a \quad \text{subject to} \quad \max_{x_k \le a}\{y_{\max} - y_k\} \ge \max_{x_m > a} y_m \tag{1}$$

   or

$$\max_{x_i < a \le x_j} a \quad \text{subject to} \quad \max_{x_k \le a}\{y_{\max} - y_k\} \le \max_{x_m > a} y_m \tag{2}$$

   For at least one of the equations (1) or (2) there exists a solution. In case (1), the error tolerance of S is given by the points to the left of ℓ and, in case (2), by the points to the right of ℓ. In constant time compute this error tolerance given by the difference between the bigger and smaller y-coordinates of the points to the left or right of line ℓ.

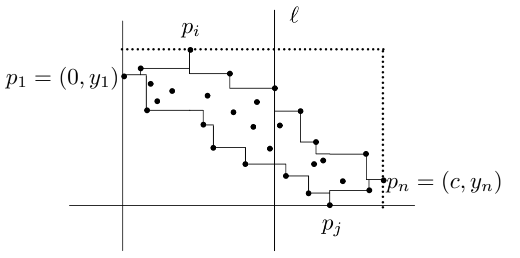

Figure 2: A rectilinear convex hull of S formed by the maximal points of S.

If p_i is to the right of p_j, then the algorithm is similar with the obvious changes: the second quadrant staircase gives the upper point to the left of ℓ and the fourth quadrant staircase gives the lower point to the right of ℓ.

Notice that once the rectilinear convex hull of S is obtained, the oriented 2-fitting problem can be solved in $O(\log n)$ time; this is a key component for the algorithm of Section 3.

The above staircase structure can be used to design $O(n \log n)$ time algorithms for the oriented 3-fitting and 4-fitting problems as well. We consider 3-fitting first. Let ℓ_1 and ℓ_2 denote, respectively, the two vertical lines containing the two vertical segments of the solution. By Lemma 1, at least one of ℓ_1 and ℓ_2 must lie between $y_{\max}$ and $y_{\min}$. Assume that ℓ_1 is to the left of ℓ_2. There are at most a linear number of locations for ℓ_1 between two consecutive points of the staircase structure. For each of these locations a binary search over the staircase structure to the right of ℓ_1 yields the optimal location for ℓ_2 in $O(\log n)$ time. Details are omitted but the binary search depends on whether the location of ℓ_2 is either between $y_{\max}$ and $y_{\min}$ or to the right of $y_{\min}$.

It is clear that for the oriented 4-fitting problem with three vertical lines ℓ_1, ℓ_2, and ℓ_3, we can proceed in a similar way, first fixing the location of the median line, say ℓ_2, in each of the linear number of possible locations, and then finding the locations of ℓ_1 (to the left of ℓ_2) and ℓ_3 (to the right of ℓ_2) by binary search on the staircase structure.

As a consequence of the discussion above, both the oriented 3- and 4-fitting problems can be solved in $O(n \log n)$ time and $O(n)$ space. The same result but using different techniques can be achieved by the proposals in [12, 14, 21].

3 The un-oriented 2-fitting problem

In this section we consider the problem of fitting S using an un-oriented 2-orthogonal polygonal chain $\mathcal{O}_{2,\theta}$ with free orientation θ. Notice that the un-oriented 1-fitting problem for S is equivalent to the problem of computing the width of S. If we know the convex hull of S, this problem can be solved in $O(n)$ time using rotating calipers [17]. Otherwise, computing the width of S has an $\Omega(n \log n)$-time lower bound [20]. Therefore the un-oriented 1-fitting problem for S can be solved optimally in $\Theta(n \log n)$ time.

Before studying the un-oriented 2-fitting problem we introduce some notation and tools which will be useful later. We start by reviewing some definitions and results from Avis et al. [3] concerning the computation of un-oriented Θ-maximal points of a planar point set S.

Definition 2 [3] *A ray from a point $p \in S$ is called a maximal ray if it passes through another point $q \in S$. A cone is defined by a point p and two rays C and D emanating from p. A point $p \in S$ is an un-oriented Θ-maximal with respect to S if and only if there exist two maximal rays, C and D, emanating from p with an angle at least Θ between them so that the points of S lie outside the (Θ-angle) cone defined by p, C and D (Figure 3(a)).*

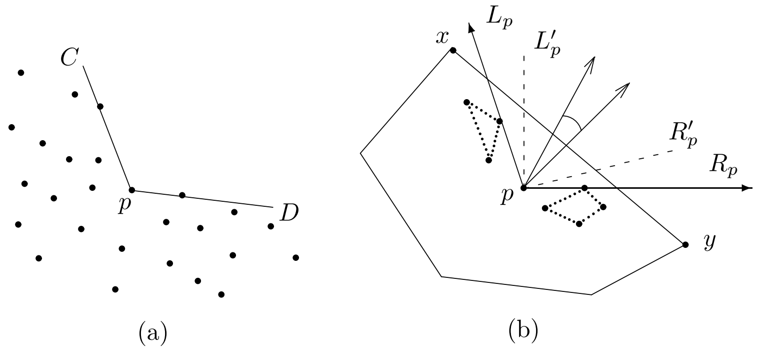

Figure 3: Un-oriented Θ-maximum with respect to S.

Theorem 3 [3] *All un-oriented Θ-maximal points of S for $\Theta \geq \pi/2$ can be computed in $O(n \log n)$ time and $O(n)$ space, and the algorithm is optimal for fixed values of Θ.*

For $\Theta = 90$, the output of the algorithm of Theorem 3 is the list of all the un-oriented 90-maximal points that are apices of the wedges that have bounding rays (crossing an edge of $CH(S)$) with aperture angle at least 90. For every such maximal point p the output also contains the two rays L_p and R_p bounding the widest empty wedge on the left and on the right, respectively (Figure 3(b)). Since the aperture angle is at least 90, then each maximal point p can have at most three disjoint wedges. In constant time we can compute the set of orientations of the bisectors of the (90-angle) cones with apex in p contained in the wedge defined by p, L_p and R_p: compute the ray R'_p (resp. L'_p) from p which is perpendicular to L_p (resp. R_p), the bisectors of the (90-angle) cones formed by R_p, p, L'_p and by R'_p, p, L_p are the extremes of the set of orientations of bisectors (Figure 3(b)). This set of orientations can be translated into an orientation interval in $\mathbb{S}^1$. Thus, each maximal point $p \in S$ can have at most three disjoint orientation intervals in $\mathbb{S}^1$ such that for each orientation inside these intervals the point p is 90-maximal. Notice that all the points in the boundary of the convex hull of S are 90-maximal, and that the total number of orientation intervals is linear.

Now we consider the un-oriented 2-fitting problem. An optimal solution for this problem is given by an orthogonal polygonal chain $\mathcal{O}_{2,\theta}$ with orientation θ such that the error tolerance of S with respect to $\mathcal{O}_{2,\theta}$ is minimum. Clearly, this is equivalent to the problem of determining a line ℓ_θ with slope $\tan(90+\theta)$ that splits S into subsets S_{l_θ} and S_{r_θ}, where $e^u_{l_\theta}$ and $e^b_{l_\theta}$ ($e^u_{r_\theta}$ and $e^b_{r_\theta}$) are the points responsible for the error tolerance of S_{l_θ} (S_{r_θ}) and such that $\mathcal{O}_{2,\theta}$ minimizes the error tolerance of S over all values of θ. Accordingly, the error tolerance is given by the following formula, where $d_\theta(p,q)$ denotes the distance between parallel lines through p and q with orientation θ.

$$\mu(\mathcal{O}_{2,\theta}, S) = \max_{p_i \in S} d(p_i, \mathcal{O}_{2,\theta}) = \max\{d_\theta(e^u_{l_\theta}, e^b_{l_\theta}), d_\theta(e^u_{r_\theta}, e^b_{r_\theta})\}.$$

Let $y_{\min,\theta}$ and $y_{\max,\theta}$ be the minimum and maximum y-coordinates of the points in S when the coordinate system is rotated by angle θ. The following lemma is a generalization of Lemma 1.

Lemma 3 *Given an orientation θ, an optimal solution for the un-oriented 2-fitting problem with orientation θ is defined by a line ℓ_θ passing through a point of S which separates the points of S with y-coordinates $y_{\min,\theta}$ and $y_{\max,\theta}$.*

Description of the un-oriented 2-fitting algorithm. The goal of our approach is to adapt the $O(n \log n)$-time algorithm for the oriented 2-fitting problem described earlier to account for continuous changes in the orientation θ, looking for the optimal $\mathcal{O}_{2,\theta}$ chain in the process. To do this we update the staircase structure as θ varies and use Lemma 3 to look for an optimal solution.

• *Initialization*: The starting situation is the staircase structure formed by the four sets of maximal points with respect to the four quadrants of the coordinate system when $\theta = 0$. Analogously to [19, 23], we use a height-balanced search tree to store and compute each of the four sets of maximal points with insertions or deletions in optimal $O(n \log n)$ time. Thus, we compute the staircase structure and its corresponding optimal solution in $O(n \log n)$ time as we did for the oriented case.

• *Update as θ sweeps over* $[0, 90]$: As we rotate the coordinate system according to the orientation θ in discrete steps from $\theta = 0$ to $\theta = 90$ to compute the un-oriented optimal solution, we identify the four quadrants by its oriented bisectors, i.e., by the oriented lines with slopes $\tan(\theta + 45)$, $\tan(\theta + 135)$, $\tan(\theta + 225)$, and $\tan(\theta + 315)$. The staircases are formed by the four sets of maximal points in S with respect to the bisectors of the current four quadrants. Notice that for any θ, a point is in the staircases if and only if it is a 90-maximal point for some of the four orientations above, i.e., when at least one of these four orientations lies in the orientation intervals defined by the point.

The main idea of the algorithm is to rotate the coordinate system by θ, and update the staircase structure by inserting or deleting points to each of the staircases as the orientation θ changes. To do this we maintain four ordered lists of the current un-oriented 90-maximal points of S with respect to the bisectors with orientations $\theta + 45$, $\theta + 135$, $\theta + 225$, and $\theta + 315$. The lists correspond to the sequences of points in the four staircases. More precisely, the staircase structure will be maintained with insertions and deletions of points induced by the changes in θ. Notice that for any orientation θ the staircase structure has linear size, and updating a point on it can be done in $O(\log n)$ time as in the $\theta = 0$ case [19, 23].

As θ changes, the four staircases can be modified because either a new point of S becomes 90-maximal or some current 90-maximal point of S has to be deleted. To determine the sequence of events, as θ changes, we use Theorem 3 to pre-compute in $O(n \log n)$ time the set of all un-oriented 90-maximal points of S together with their respective orientation intervals in $\mathbb{S}^1$. These orientation intervals are the intervals where each point is an un-oriented 90-maximal point for S. Notice that a point can be 90-maximal for at most 3 (disjoint) orientation intervals and, consequently, the total number of changes in the staircase structure is linear.

To know in advance the sequences of events, i.e., the values of θ where insertions or deletions of points occur, we proceed as follows. Suppose that we have computed the orientation intervals for each point $p_i \in S$. Figure 4 represents the set of these orientation intervals. A point $p \in S$ can have at most 3 disjoint orientation intervals. We sweep the set of these intervals from 0 to 360°, keeping track, for each orientation θ, of the set of 90-maximal points for that orientation which is the set of intervals intersected by the sweep line.

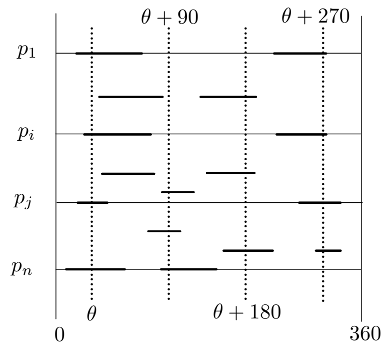

Figure 4: Sweeping the orientation intervals of the 90-maximal points of S.

Thus, the algorithm performs a sweep of these intervals from $\theta = 0$ to $\theta = 90$ by vertical lines corresponding to orientations θ, $\theta + 90$, $\theta + 180$, and $\theta + 270$, stopping at each event (interval endpoint) and updating the staircase structure. Since the points in S are in general position, only a constant number of updates can occur at each event. We compute the optimal solution for the staircase structure between two consecutive events by computing a line ℓ_θ with slope $\tan(\theta + 90)$, as explained below. In order to cover all the orientations of the plane, we also run the algorithm as θ changes from 90° to 180°, which can be handled analogously.

Consider consecutive events θ_1 and θ_2. Lemma 3 implies that for a fixed value $\theta \in [\theta_1, \theta_2]$, the line ℓ_θ that gives the optimal solution has to separate the points with the current y-coordinates $y_{\min,\theta}$ and

$y_{\max,\theta}$. Thus, the optimal solution is determined by two pairs of points, either (i) $(y_{\max,\theta}, e^b_{l_\theta})$ and $(e^u_{r_\theta}, y_{\min,\theta})$ if $y_{\max,\theta}$ is to the left of $y_{\min,\theta}$, or (ii) $(e^u_{l_\theta}, y_{\min,\theta})$ and $(y_{\max,\theta}, e^u_{r_\theta})$ if $y_{\max,\theta}$ is to the right of $y_{\min,\theta}$, giving the error tolerance in S_{l_θ}, *the left error tolerance*, and the error tolerance in S_{r_θ}, *the right error tolerance*, respectively. To compute the optimal solution between two consecutive events we use the following lemma.

Lemma 4 *Let $[\theta_1, \theta_2]$ be an orientation interval corresponding to consecutive events. The optimal solution for the un-oriented 2-fitting problem in this interval occurs found either at an endpoint, i.e., at θ_1 or θ_2, or at an orientation $\theta_0 \in [\theta_1, \theta_2]$ where the left and right error tolerances are equal.*

Proof. Let $\theta_0 \in [\theta_1, \theta_2]$ be the orientation of the optimal solution in $[\theta_1, \theta_2]$. Assume that the left and right error tolerances of the optimal solution for θ_0 are given by the point pairs (p_i, p_k) and (p_j, p_m), respectively. Let p_i and p_j be the points with maximum and minimum y-coordinate, respectively, for any orientation in $[\theta_1, \theta_2]$. The identify of these points does not change in the interval, as otherwise we would get a new orientation interval. Assume that p_i is to the left of p_j. Other cases can be handled analogously.

The left error tolerance can be written as a function $w_1(\theta) = d(p_i, p_k)\cos(\theta_{ik} - \theta)$, where θ_{ik} is the orientation of the line passing through p_i and p_k, and θ is the current orientation in the rotation process. This function increases or decreases, and the unique increasing/decreasing change can occur if, during the rotation, p_k passes from one side to the other side of line $\ell_{\theta,i}$ with orientation θ going through p_i, a predictable event (Figure 5(a), and (b)). An entirely analogous situation occurs with the right error tolerance with the function $w_2(\theta) = d(p_j, p_m)\cos(\theta_{jm} - \theta)$ (Figure 5(c) and (d)).

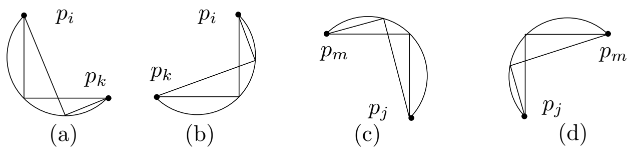

Figure 5: (a) and (b) Variation of the left error tolerance depending on whether p_k is on the right or left side of $\ell_{\theta,i}$, (c) and (d) variation of the right error tolerance depending on whether p_m is on the left or right side of $\ell_{\theta,j}$.

If both functions $w_1(\theta)$ and $w_2(\theta)$ increase, the optimal solution is found at the endpoint θ_1, as otherwise we can rotate clockwise, decreasing both error tolerances in the process (Figure 6(a)). If both functions $w_1(\theta)$ and $w_2(\theta)$ decrease, the optimal solution is found at the endpoint θ_2 as a counterclockwise rotation decreases both error tolerances (Figure 6(c)). Analogous is the case when $w_1(\theta)$ increases and $w_2(\theta)$ decreases, or viceversa, but both functions do not intersect. Otherwise, the intersection of both functions gives the optimal solution in an orientation θ_0 when the left and right error tolerances are equal (Figure 6(b)). This can be detected because there is a change of the maximum error tolerance from the right error tolerance in θ_1 to the left error tolerance in θ_2 or viceversa. □

As a consequence of Lemma 4, the optimal solution for a (non-starting) interval orientation $[\theta_1, \theta_2]$ can be computed in $O(\log n)$ time. Summarizing: (1) the number of events for the staircase structure as θ changes from 0 to 90° is linear, (2) any update can be done in $O(\log n)$ time, (3) for a fixed value of θ an $O(\log n)$ binary search produces the optimal location of the line ℓ_θ, its

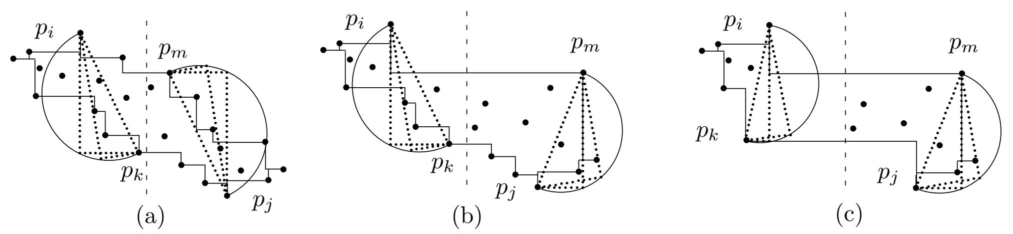

Figure 6: Variations of the left and right error tolerances.

corresponding error tolerance, and allows us to maintain the minimum one. We conclude that the un-oriented 2-fitting problem can be solved in $O(n \log n)$ time and $O(n)$ space.

The un-oriented-2-fitting-algorithm.

We assume that for the current orientation θ the point with y-coordinate $y_{\max,\theta}$ is to the left of the point with y-coordinate $y_{\min,\theta}$; otherwise, we only update the changes in the staircase structure without computing the optimal solution. We repeat the algorithm for the alternative case.

1. Use the algorithm from Avis et al. [3] to compute in $O(n \log n)$ time the list of the un-oriented 90-maximal points of S and their orientation intervals in $\mathbb{S}^1$ where each point is un-oriented 90-maximal. Sort the arrangement of the orientation intervals according to their endpoints in such a way that when we sweep the arrangement we know which 90-maximal points are *active* in a current sweeping orientation, and which is the next incoming endpoint (Figure 4).

2. In $O(n \log n)$ time compute the horizontal/vertical staircase structure for S and the optimal solution as we did in the oriented 2-fitting problem. The staircase structure is formed by the 90-maximal points for S with orientations $0 + 45$, $90 + 45$, $180 + 45$, and $270 + 45$.

3. Sweep the arrangement of the orientation intervals with the four vertical lines. Each time that we reach an endpoint, either (1) a new un-oriented 90-maximal point enters the staircase structure, or (2) an active un-oriented 90-maximal point is deleted from the staircases. We update the changes in the staircases in $O(\log n)$ time, including also the possible changes of the points with minimum and maximum y-coordinates for the current orientation. Since we consider the points in general position, at most two aligned points are updated at the same time producing a constant number of changes. We use binary search to compute the separating line of the new optimal solution in $O(\log n)$ time, and store and update the information of the optimal solution.

Notice that a point enters one of the staircases at most once, so a point is updated a constant number times and the overall running time for updating changes is $O(n \log n)$ time. The running time of the algorithm is $O(n \log n)$ and the space is $O(n)$ since the lists and the arrangement of orientation intervals have linear size[1].

[1]Notice that the algorithm can maintain the rectilinear convex hull of S during the rotation in $O(n \log n)$ time and $O(n)$ space, improving on a recent result by Bae et al [5] who present an $O(n^2)$ time and $O(n)$ space algorithm for this problem. In their paper, the authors derive a space/time trade-off: $O(n^{3/2} \log^{7/3}(n))$ time and $O(n^{3/2} \log n)$ space are also possible.

Next, we show a reduction for the un-oriented 2-fitting problem from a MAX-GAP problem for points on the first quadrant of the unit circle [1, 20], establishing in the process a $\Omega(n \log n)$-time lower bound.

We reduce the MAX-GAP problem for points on the first quadrant of the unit circle centered at the origin of the coordinates system to our problem. This MAX-GAP problem has an $\Omega(n \log n)$ time lower bound in the algebraic decision tree model [1, 20]. Let $P = \{(x_1, y_1), \ldots, (x_n, y_n)\}$ be the set of points of an instance of MAX-GAP. Consider a symmetric copy of points in the third quadrant and new copies of the overall circle in other positions as in Figure 7. It is easy to see that the optimal un-oriented 2-fitting orthogonal chain defines the maximum gap for P and viceversa. This construction can be generalized to work with the un-oriented k-fitting problem, $k \geq 3$, using $k-1$ copies of the initial circle with the centers located with adequate distances between them.

Theorem 4 *The un-oriented* 2*-fitting problem can be solved in optimal* $\Theta(n \log n)$ *time and* $O(n)$ *space.*

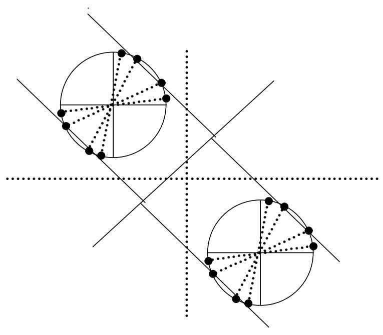

Figure 7: Construction in the proof of the lower bound for the un-oriented 2-fitting problem.

4 The oriented 2-fitting problem in 3D

In this section we consider the oriented 2-fitting problem in three-dimensions where a polygonal chain is interpreted as a configuration of two parallel half-planes, joined with an orthogonal strip. First, we give a short discussion of the 1-fitting problem.

To solve the oriented 1-fitting problem for S in $\mathbb{R}^3$ we proceed according to how much information about the solution plane we have, distinguishing between two cases: *Case i: the orientation of the solution plane is fixed.* Assume that the solution plane has normal $\overrightarrow{u} = (0, 0, 1) \in \mathbb{S}^2$. We solve this problem in $\Theta(n)$ time by computing the points with maximum and minimum z-coordinates. *Case ii: the orientation of the solution plane has one degree of freedom.* Assume that the orientation of the solution plane is orthogonal to $\overrightarrow{u} = (0, 1, 0) \in \mathbb{S}^2$. We solve this problem by projecting the points onto a plane with normal $\overrightarrow{u}$ and computing the width of the convex hull of the projected points with a rotating caliper. The total running time is $\Theta(n \log n)$. The $\Omega(n \log n)$ time lower bound comes from the computation of the width of a set of points in two-dimensions.

The oriented 2-fitting problem can be defined by three consecutive orthogonal planes. We refer to the plane Π producing the bipartition of S as the *separating plane* and to the two parallel planes that induce the error tolerances on either side of Π as the *supporting planes.* We distinguish among three cases depending on how the orientation for the solution is constrained: (1) the orientation of both the separating plane and the parallel supporting planes are fixed; (2) the orientation of the separating plane is fixed; and (3) the orientation of the parallel supporting planes is fixed. Next, we consider the three cases.

Case 1: the orientation of both the separating plane and the parallel supporting planes are fixed. Assume that the separating plane has normal $\overrightarrow{u_1} = (0, 1, 0)$ and that the parallel supporting planes have normal $\overrightarrow{u_2} = (0, 0, 1)$). We reduce the problem to two-dimensions as follows. Let $\overrightarrow{u_3} = \overrightarrow{u_1} \times \overrightarrow{u_2}$. We project the points in S onto a plane with normal $\overrightarrow{u_3} = (1, 0, 0)$ and solve it optimally in $O(n)$ time using the algorithm of Section 2.

Theorem 5 *The oriented 2-fitting problem in 3D can be solved in $\Theta(n)$ time and space if the orientations of both the separating plane and the parallel supported planes are fixed.*

Case 2: the orientation of the separating plane is fixed. Assume that the separating plane has normal $\overrightarrow{u} = (0, 1, 0)$. In $O(n \log n)$ time, sort the points in S along $\overrightarrow{u}$, i.e., by $\overrightarrow{u} \cdot p_i$ (e.g., by y-coordinate). According to this order, let $S_i = \{p_1, \dots, p_i\}$ and $S_{n-i} = \{p_{i+1}, \dots, p_n\}$ be the bipartition of S given by the separating plane passing through p_i. In order to compute the parallel supporting planes of S_i and S_{n-i} for determining which bipartition of S gives the optimal solution, we project the points of S_i and the points of S_{n-i} onto two planes parallel to the separating plane. We work with the convex hulls of the projected points. Let S'_i and S'_{n-i} be the projected points of S_i and S_{n-i}, and let $CH(S'_i)$ and $CH(S'_{n-i})$ be their respective convex hulls (Figure 8).

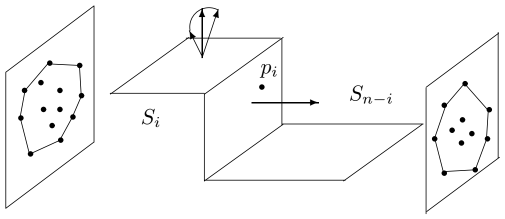

Figure 8: Sample configuration for case (2).

To find the optimal 2-fitting solution, we use two clockwise rotating calipers which rotate simultaneously over $CH(S'_i)$ and $CH(S'_{n-i})$ in discrete steps. Each step is defined by the minimum rotating angle of the two calipers on antipodal pairs. Suppose that at some step, the rotating caliper over $CH(S'_i)$ (resp. $CH(S'_{n-i})$) has antipodal points q_1 and q_2 (resp. q_3 and q_4). Let α (resp. α') be the angle of rotation with respect to the parallel supporting lines passing through q_1 and q_2 (resp. q_3 and q_4). Let w_1 (resp. w_2) be the width function of $CH(S'_i)$ (resp. $CH(S'_{n-i})$) in the rotation interval and $d_1 = d(q_1, q_2)$ (resp. $d_2 = d(q_3, q_4)$). The continuous and monotone width function w_1 (resp. w_2) depends on d_1 (resp. d_2) and $\cos(\alpha)$ (resp. $\cos(\alpha')$). The minimum of the maximum of the two width values is a minimum of the upper envelope of the two functions w_1 and w_2. Thus, we compute the minimum of the upper envelope in the rotation interval and the corresponding width (at most a linear number of intervals) and maintain the best solution. Next we describe the complete algorithm.

Algorithm for case 2. Take the median point p_i of S. Let S_i, S_{n-i}, S'_i, S'_{n-i}, $CH(S'_i)$ and $CH(S'_{n-i})$ be defined as above. In linear time find the minimum of the upper envelope of the two functions w_1 and w_2. Denote this minimum value by w. If w is determined by the intersection of w_1 and w_2, then the current separation plane is optimal and the algorithm can stop since the left and right error tolerances are equal. Otherwise, if w is determined by w_1, then the separation plane should move to the right (i.e., towards S_{n-i}). Now we let the new separating plane pass through the median point of S_i and repeat the above procedure; if w is determined by w_2, we move to the left and proceed similarly. The high-level framework is essentially a binary search. Thus, the optimal separating plane can be found in $O(\log n)$ steps. In each step, the running time is dominated by the computation of the convex hulls $CH(S'_i)$ and $CH(S'_{n-i})$, which is $O(n \log n)$ time. Hence, the running time of the whole algorithm is $O(n \log^2 n)$. A more sophisticated algorithm solves the problem in $O(n \log n)$ time. The idea is as follows. We can update $CH(S'_i)$ and $CH(S'_{n-i})$ in $O(\log n)$ time per point p_i changing from S_{n-i} to S_i or viceversa [4]. Because we are doing essentially a binary search, a point can change a certain number of times from $CH(S'_i)$ to $CH(S'_{n-i})$ or viceversa while we compute the median point to the left or to the right, and at some step it will be fixed for the remainder of the algorithm. Thus, we spend $O((n+n/2+n/4+n/8+\cdots)\log n) = O(n \log n)$ amortized time in the total updating process of the convex hulls. Notice that deleting points in the convex hulls can be considered as inserting points in the reverse process.

The $\Omega(n \log n)$ time lower bound comes from the following construction for a point set not in general position. Consider a set P of n points in the unit circle centered at the origin on the plane XZ of the coordinate system. Make two copies P_1 and P_2 of P putting their respective centers in the points $(0, 1, 0)$ and $(0, -1, 0)$. Let P_3 be a set of n equidistant points in the y-axis between the points $(0, 1, 0)$ and $(0, -1, 0)$. Let $S = P_1 \cup P_2 \cup P_3$ be the set of those $3n$ points. Assume that the separating plane has normal $\overrightarrow{u} = (0, 1, 0)$. It is easy to see that an optimal solution for our 2-fitting problem for S gives the the minimum width of the set P and viceversa.

Theorem 6 *The oriented 2-fitting problem in 3D can be solved in $\Theta(n \log n)$ time and $O(n)$ space if the orientation of the separating plane is fixed.*

Case 3: the orientation of the parallel supporting planes is fixed. Assume that the parallel supporting planes have normal $\overrightarrow{u} = (0, 0, 1)$. Sort the points in S by increasing z-coordinate and store the results in list A. Let $z_{\min}$ (resp. $z_{\max}$) be the first (resp. last) element of A. An optimal separating plane, produces a bipartition of S that separates the points with extreme projections along $\overrightarrow{u}$, i.e., $z_{\max}$ and $z_{\min}$. Let $S_{\max}$ (resp. $S_{\min}$) be the subset of points of S in the same half-space as $z_{\max}$ (resp. $z_{\min}$) in an optimal solution. Then, the projections of $S_{\max}$ and $S_{\min}$ on the XY plane are separable by the line resulting from the intersection between the separating plane and the plane XY (Figure 9).

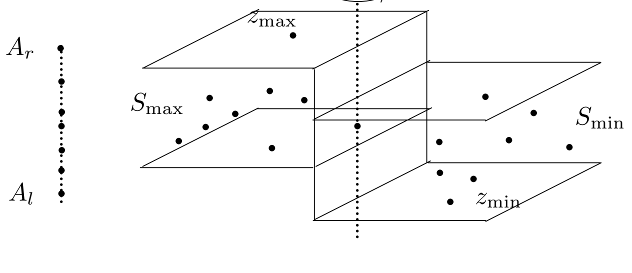

Figure 9: Sample configuration for case (3).

Algorithm for case 3. Let $S_{\max}$ and $S_{\min}$ be empty sets initially. We add $z_{\max}$ to $S_{\max}$ and remove it from A. Similarly, add $z_{\min}$ to $S_{\min}$ and remove it from A. The key observation is that we try to move as many elements from the front of A to $S_{\min}$ and from the rear of A to $S_{\max}$ as possible while keeping the two sets $S_{\min}$ and $S_{\max}$ separable by a plane parallel to the z-axis, and keeping the best current solution. To do so we proceed as follows.

While A is not empty do the following. Suppose the current two sets are separable (i.e., they can be separated by a plane parallel to the z-axis) and the first (resp. last) element of the current list A is A_l (resp. A_r). If $z_{\max} - A_l \leq A_r - z_{\min}$, then we put A_r into $S_{\max}$ and check whether the new set $S_{\max}$ can be separated from $S_{\min}$. If they are still separable, then we remove A_r from A and continue the above procedure; otherwise, any plane parallel to the z-axis which can separate the two sets $S_{\max} \backslash \{A_r\}$ and $S_{\min}$ is an optimal separating plane and the algorithm can stop. If $z_{\max} - A_l > A_r - z_{\min}$, we proceed similarly.

To determine whether the two sets above are separable, we can compute the convex hull of their projections on the XY plane. By using a dynamic convex hull data structure (with insertion only), each insertion can be updated in $O(\log n)$ time. Thus, the running time of the algorithm is $O(n \log n)$.

Theorem 7 *The oriented 2-fitting problem in 3D can be solved in $O(n \log n)$ time and $O(n)$ space if the orientation of the parallel supporting planes is fixed.*

Open problem 2 *Does the oriented 2-fitting problem for case 3 have an $\Omega(n \log n)$ time lower bound?*

Acknowledgements. These problems were proposed and partially solved during the *Third Spanish Workshop on Geometric Optimization*, July 4-8, 2006, El Rocío, Huelva, Spain. The authors would like to thank the other workshop participants for helpful comments. Three anonymous referees deserve our gratitude for their comments and useful suggestions which allowed us to improve the last two results in three-dimensions.

References

[1] E. M. Arkin, F. Hurtado, J. S. B. Mitchell, C. Seara, and S. S. Skiena. Some lower bounds on geometric separability problems. *International Journal of Computational Geometry and Applications*, Vol. 16, No. 1, (2006) pp. 1–26.

[2] B. Aronov, T. Asano, N. Katoh, K. Melhorn, and T. Tokuyama. Polyline fitting of planar points under min-sum criteria. *International Journal of Computational Geometry and Aplications*, Vol. 16, Nos. 2 and 3, 2006, pp. 97–116.

[3] D. Avis, B. Beresford-Smith, L. Devroye, H. Elgindy, E. Guvremont, F. Hurtado, and B. Zhu. Unoriented Θ-maxima in the plane: complexity and algorithms. *SIAM Journal of Computing*, Vol. 28, No. 1, 1999, pp. 278–296.

[4] D. Avis, H. Elgindy, and R. Seidel. Simple on-line algorithms for convex polygons. *Computational Geometry*, G. T. Toussaint, ed., North-Holland, Amsterdam, 1985, pp. 23–42.

[5] S. W. Bae, C. Lee, H-K. Ahn, S. Choi, and K-Y. Chwa. Computing minimum-area rectilinear convex hull and L-shape. *Computational Geometry: Theory and Applications*, Vol. 42, 2009, pp. 903–912.

[6] P. J. Burt. Fast filter transforms for image processing. *Computer Graphics and Image Processing*, Vol. 16, 1979, pp. 20–51.

[7] W. S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments or minimun error. *International Journal of Computational Geometry and Applications*, Vol. 6, 1996, pp. 59–77.

[8] D. Z. Chen and H. Wang. Approximating points by piecewise linear functions. *Manuscript.*

[9] F. Chin, A. Choi, and Y. Luo. Optimal generating kernel for image pyramids by piecewise fitting. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 14, 1992, pp. 1190–1198.

[10] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to algorithms.* MIT Press, 2001.

[11] J. M. Díaz-Báñez and J. A. Mesa. Fitting rectilinear polygonal curves to a set of points in the plane. *European Journal of Operations Research*, 130, 1, 2001, pp. 214–222.

[12] H. Fournier and A. Vigneron. Fitting a step function to a point set. *Lecture Notes in Computer Science*, 5193, 2008, pp. 442–453.

[13] M. T. Goodrich. Efficient piecewise-linear function approximation using the uniform metric. *Discrete and Computational Geometry*, Vol. 14, 1995, pp. 445–462.

[14] S. Guha and K. Shim. A note on linear time algorithms for maximum error histograms. *IEEE Transactions on Knowledge and Data Enginering*, Vol. 19, No. 7, 2007, pp. 993–997.

[15] L. J. Guibas, J. E. Hershberger, J. S. B. Mitchell, and J. S. Snoeyink. Approximating polygons and subdivisions with minimum-link paths. *International Journal of Computational Geometry and Applications*, Vol. 3, 1993, pp. 383–415.

[16] S. L. Hakimi and E. F. Schmeichel. Fitting polygonal functions to a set of points in the plane. *Graphical Models and Image Processing*, Vol. 53, 1991, pp. 132–136.

[17] M. E. Houle and G. T. Toussaint. Computing the width of a set. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 10, No. 5, 1988, pp. 761–765.

[18] H. Imai and M. Iri. Polygonal approximations of a curve-formulations and algorithms. *Computational Morphology*, G. T. Toussaint ed., North Holland, 1988.

[19] H. T. Kung, F. Luccio, and F. P. Preparata. On finding the maxima of a set of vectors. *Journal of ACM*, Vol. 22, 1975, pp. 469–476.

[20] D. T. Lee and Y. F. Wu. Geometric complexity of some location problems. *Algorithmica*, Vol. 1, 1986, pp. 193–211.

[21] M. A. Lopez and Y. Mayster. Approximating a set of points by a step function. *Journal of Visual Communication and Image Representation*, 2006.

[22] T. Pavlidis. Algorithms for shape analysis of contours and waveforms. *IEEE Transation on Pattern Analysis and Machine Intelligence*, 1980, pp. 301–312.

[23] F. P. Preparata and M. I. Shamos. *Computational Geometry, an Introduction.* Springer, 1988.

[24] G. T. Toussaint. On the complexity of approximating polygonal curves in the plane. *International Symposium on Robotics and Automation*, Lugano, Switzerland, 1985.

[25] K. R. Varadarajan. Approximating monotone polygonal curves using the uniform metric. *Symposium on Computational Geometry*, 1996.

[26] D. P. Wang. A new algorithm for fitting a rectilinear x-monotone curve to a set of points in the plane. *Pattern Recognition Letters*, Vol. 23, No. 1, 2002, pp. 329–334.

[27] D. P. Wang, N. F. Huang, H. S. Chao, and R. C. T. Lee. Plane sweep algorithms for polygonal approximation problems with applications. *Lecture Notes in Computer Science*, 762, 1993, pp. 515–522.