Polygonal Chain Simplification with Small Angle Constraints

Ovidiu Daescu* Anastasia Kurdia†

Abstract

We consider the problem of simplifying an n-vertex polygonal chain with *small* angle constraints in $\mathbb{R}^2$ and $\mathbb{R}^3$, thus closing the gap on the range of angles left in previous work on the problem. Specifically, we show that the *min-#* version of the polygonal chain simplification problem with small angle constraints can be solved in $O(n^2)$ time and space in $\mathbb{R}^2$, and in $O(n^2 \log^2 n)$ time, $O(n^2)$ space in $\mathbb{R}^3$.

1 Introduction

The problem of simplifying a polygonal chain with angle constraints was introduced in [6], and it was defined as follows: Given a polygonal chain $P = (p_1, p_2, \ldots, p_n)$ in $\mathbb{R}^2$ or $\mathbb{R}^3$, find another chain $P' = (p_{i_1}, p_{i_2}, \ldots, p_{i_m})$ such that (1) $1 = i_1 < i_2 < \ldots < i_m = n$; (2) the tolerance zone criterion is satisfied for a given tolerance $\varepsilon \geq 0$: for every $j = \overline{1, m-1}$ the vertices of the subpath $(p_{i_j}, p_{i_j+1}, \ldots, p_{i_{j+1}})$ of P are within distance ε from the line segment $\overline{p_{i_j} p_{i_{j+1}}}$ of P'; (3.min) the turn angle between any two consecutive line segments $\overline{p_i p_j}$ and $\overline{p_j p_k}$ of P' is at least a specified value $\delta(\overline{p_i p_j}) \in [0, \pi)$ (*min turn angle problem*); or (3.max) the turn angle between any two consecutive line segments $\overline{p_i p_j}$ and $\overline{p_j p_k}$ of P' is at most a specified value $\delta(\overline{p_i p_j}) \in [0, \pi)$ (*max turn angle problem*).

The problem was solved in [6] only for a subset of ranges of the turn angle δ, specifically for $\delta \in [0, \pi/2)$ for the (3.min) constraint and for $\delta \in [\pi/2, \pi)$ for the (3.max) constraint. Solving the problem when $\delta \in [\pi/2, \pi)$ for (3.min) and $\delta \in [0, \pi/2)$ for (3.max) remained open. In this paper we close the gap in the range of δ.

Let $ray(p_i p_j)$ be the ray originating at p_j, extending the line segment $\overline{p_i p_j}$ to infinity and not containing $\overline{p_i p_j}$. The turn angle between two line segments $\overline{p_i p_j}$ and $\overline{p_j p_k}$ is defined as the smallest angle needed to rotate the ray $ray(p_i p_j)$ at p_j such that it overlaps with the line segment $\overline{p_j p_k}$ [6] (see Fig. 1).

Using the tolerance-zone error criterion, the *min-#* problem (given an error-tolerance ε, minimize m) was solved in [6] in $O(n^2)$ time and space in $\mathbb{R}^2$, and in $O(n^2 \log n)$ time, $O(n^2)$ space, in $\mathbb{R}^3$. The *min-ε* problem (given m, minimize ε) was solved in $O(n^2 \log n)$ time and $O(n^2)$ space in $\mathbb{R}^2$, and in $O(n^2 \log^3 n)$ time and $O(n^2)$ space in $\mathbb{R}^3$. These bounds match the best known bounds for the unconstrained polygonal chain simplification problem with the tolerance-zone criterion.

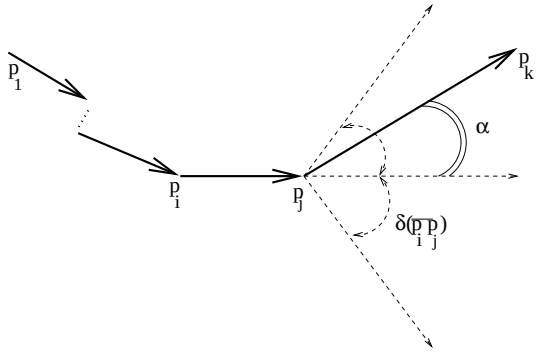

Figure 1: Angle constraint for the max turn angle problem in $\mathbb{R}^2$: α is a turn angle between $\overline{p_i p_j}$ and $\overline{p_j p_k}$; the edges $\overline{p_i p_j}$ and $\overline{p_j p_k}$ satisfy the angle constraint $\delta(\overline{p_i p_j})$.

A line segment $\overline{p_i p_j}$, $1 \leq i < j \leq n$, is a valid approximating segment (also called an ε-approximation segment) if the vertices $p_{i+1}, \ldots, p_{j-1}$ of P are within distance ε of $\overline{p_i p_j}$.

The algorithmic approach in [6] is to construct a directed acyclic graph G_p on P containing all valid approximating segments for the corresponding subpaths of P and to compute the shortest p_1-to-p_n path in G_p satisfying the angle constraint using a dynamic programming algorithm. G_p is constructed in $O(n^2)$ time in $\mathbb{R}^2$ [3, 5] and in $O(n^2 \log n)$ time in $\mathbb{R}^3$ [2] using the algorithms for the unconstrained version. The problem of finding the shortest p_1-to-p_n path in G_p satisfying the angle constraint is solved by reduction to a so called off-line ball exclusion search problem (OLBES) [6].

Results. In this work we close the gap in the range of possible turn angles by giving solutions for $\delta \in [\pi/2, \pi)$ for min turn angle problem and for $\delta \in [0, \pi/2)$ for max turn angle problem. As is the case in [6], the min turn and max turn problems can be solved using the same technique, so we discuss the max turn angle version only. Following the notation of [6], we describe the reduction of the shortest path problem in G_p to a special instance of the *off-line ball inclusion search* problem (OLBIS),

*Department of Computer Science, University of Texas at Dallas, daescu@utdallas.edu. Daescu's research was partially supported by NSF grant CCF-0635013.

†Department of Computer Science, University of Texas at Dallas, akurdia@utdallas.edu

defined in Section 2. We then give solutions to the 1-dimensional (1D) and 2-dimensional (2D) OLBIS problems.

Surprisingly, in $\mathbb{R}^3$, the time complexity of the solution for small turn angle constraints (maximum turn angle $\delta \in [0, \pi/2)$) seems to be inherently higher by an $O(\log n)$ factor than that for large angle constraints (maximum turn angle $\delta \in [\pi/2, \pi)$) due to the difference in time complexity for querying the associated data structures. Specifically, we solve the *min-#* problem in $\mathbb{R}^3$ in $O(n^2 \log^2 n)$ time with $O(n^2)$ space. Our solution for small angle constraints in $\mathbb{R}^2$ matches the $O(n^2)$ time and space complexities of that in [6].

2 Preliminaries

We use the notations in [6]. Let $ACSP_j(k)$ denote the angle-constrained shortest path from p_1 to p_k in G_p such that the last segment of $ACSP_j(k)$ is $\overline{p_j p_k}$. At the end of iteration i, $ACSP_j(k)$ is available for $j = \overline{1, i}$ and $k = \overline{i+1, n}$. At iteration $i+1$, from the available $ACSP_j(i+1)$, $j \in 1, 2, \ldots, i$, $ACSP_{i+1}(k)$ is computed for every $k = \overline{i+2, n}$.

At iteration $i+1$ consider $ACSP_j(i+1)$, for some $j \in 1, 2, \ldots, i$. The line segment $\overline{p_j p_{i+1}}$ is an incoming edge at p_{i+1} in G_p. Let p_{i+1} be the center of a unit sphere S_{i+1}. Let $Cone(j, i+1)$ be the cone of directions at p_{i+1} satisfying the angle constraint for $\overline{p_j p_{i+1}}$. An outgoing edge $\overline{p_{i+1} p_k}$, with $i+1 < k$ and $k \in i+2, i+3, \ldots, n$, can succeed $\overline{p_j p_{i+1}}$ to extend an angle constrained (shortest) path in G_p only if $\overline{p_{i+1} p_k}$ is contained within $Cone(j, i+1)$.

Each nonempty $Cone(j, i+1)$ at p_{i+1}, $j < i+1$, is intersected with S_{i+1}; the resulting ball (spherical cap) is assigned a weight equal to the length of $ACSP_j(i+1)$ (the number of edges of $ACSP_j(i+1)$). For each outgoing edge $\overline{p_{i+1} p_k}$ at p_{i+1}, the ray supporting the line segment $\overline{p_{i+1} p_k}$ is also intersected with S_{i+1}. We have at most i weighted balls and at most $n-i-1$ points. For each point we need to find a minimum-weight ball on S_{i+1} such that the point is contained by the ball. Sorting the balls by their weight and adding the points at the end of the resulting sequence we obtain an instance of the OLBIS problem:

OLBIS (Off-Line Ball Inclusion Search): Given a sequence $\mathcal{E} = (e_1, e_2, \ldots, e_n)$ such that each e_i is either a ball B_i or a point p_i, for every point p_k find the smallest-index ball $B_j \in \{e_1, e_2, \ldots e_{k-1}\}$ such that $p_k \in B_j$, or report no such ball exists.

Note that the OLBIS problem is a more general problem, since in the chain simplification problem the points appear after all the balls in $\mathcal{E}$.

The main differences between our solutions and those in [6] are in the construction and querying of the data structures associated with the OLBIS problem, versus those for the OLBES problem.

3 The $\mathbb{R}^2$ Problem

In $\mathbb{R}^2$, the cone projections correspond to intervals (1-dimensional balls). The 1-dimensional (1D) OLBIS problem can be solved by a simple greedy algorithm in $O(n \log n)$ time (see also [1] for the solution to a more general problem). Specifically, we first sort all points in the sequence. Then, starting with the first interval in $\mathcal{E}$, we locate the left endpoint of that interval in the sorted list of points and advance along the list until a point with value greater than the right endpoint of the interval is found. For each encountered point p, if the index of p in $\mathcal{E}$ is greater than the index of the interval we report p as included in that interval. Otherwise there is no interval that contains p and has index smaller than that of p. The points visited are removed from the sequence and the procedure is repeated with the next disk in $\mathcal{E}$.

Lemma 1 *Given a sequence $\mathcal{E}$ of n intervals (1-dimensional balls) and n points on the real line, the OLBIS answers for all points in $\mathcal{E}$ can be determined in $O(n \log n)$ time and $O(n)$ space.*

For the special case that arises in the min-# problem we can actually do better.

Lemma 2 *For a sequence $\mathcal{E}$ of n intervals (1-dimensional balls) and n points on the real line such that (i) the left and right endpoints of the intervals form sorted sequences and (ii) the points appear after all the intervals in $\mathcal{E}$ and are sorted by their values, the OLBIS answers for all points in $\mathcal{E}$ can be found in $O(n)$ time and space.*

Proof. First discard all the points that appear before the leftmost endpoint of the first interval, as they do not belong to any interval in $\mathcal{E}$. Then fix the left endpoint of the first interval I_1 and move along the real line until the right endpoint of I_1 is found; during this scan, enqueue the left endpoints of other intervals as they are encountered. I_1 is a minimum-index interval including all the points in $\mathcal{E}$ encountered so far. Next, dequeue the right endpoint r_2 of the next interval and report all query points encountered before reaching r_2 as belonging to the second interval. If the queue is empty and there are query points that have not yet been reported, advance to the first left endpoint available while discarding all encountered query points. This procedure is repeated until all query points are processed. It requires one pass along the input sequence and takes $O(n)$ time and space. □

We then have the following result for the polygonal chain simplification problem.

Theorem 3 *The polygonal chain simplification problem with small angle constraints in $\mathbb{R}^2$ can be solved in $O(n^2)$ time and space.*

4 The $\mathbb{R}^3$ Problem

We start by addressing the OLBIS problem. The key idea is to decide whether a query point is inside the union of a set of disks. The combinatorial complexity of the boundary of the union of n disks in $\mathbb{R}^2$ is known to be $O(n)$ [9], and it can be computed in $O(n \log n)$ time. However, as we will see below, there is no need to explicitly compute the union of disks to answer the queries. To decide whether a point is inside the union of disks, we use a standard transform that lifts a disk onto the unit paraboloid [7]. The image on the paraboloid of the circle bounding the disk defines a plane. A point is inside the disk if and only if its image on the paraboloid (the point lifted to the paraboloid) is below the plane. We call the halfspace defined by this plane and containing the image of the points inside the disk a proper halfspace. Consider the union of the corresponding proper halfspaces. Its intersection with the paraboloid and projected back to the plane gives the union of disks. The complement of this union is the intersection of a set of halfspaces, that is, a convex polytope. A point is inside the union of disks if it is outside this convex polytope.

Lemma 4 *Given a sequence $\mathcal{E}$ of n disks and n points, the OLBIS answers for all points in $\mathcal{E}$ can be determined in $O(n \log^2 n)$ time and $O(n \log n)$ space.*

Proof. We construct a complete binary tree T. Each leaf of T is associated with the complement of the proper halfspace for a disk D_j from $\mathcal{E}$ in order (i.e., the halfspace for D_1 is associated with the leftmost leaf, the second leaf from left is for the halfspace for D_2, and so on). At each internal node v we compute and store $I(v)$, the intersection of the halfspaces stored at the leaves of the subtree of T rooted at v. We also compute and store the index $i(v)$ of the rightmost leaf of the subtree rooted at the left child of v.

The intersection of half-spaces associated with the root of T is computed in divide-and-conquer fashion using the linear-time algorithm for intersecting three-dimensional convex polyhedra [4] and corresponds to a bottom-up computation in T. The intermediate results of the divide-and-conquer algorithm are intersections of halfspaces associated with internal nodes of T and are stored at those nodes. Thus, over all internal nodes $v \in T$, we can compute and store the intersections of halfspaces in $O(n \log n)$ time using $O(n \log n)$ space. Assigning $i(v)$ to each node $v \in T$ can be done in $O(n)$ time by a simple traversal of T. An n vertex polyhedron can be preprocessed for $O(\log n)$ time point inclusion queries in $O(n)$ time and space [8]. We preprocess the polyhedra associated with the nodes of T for point location queries.

The OLBIS queries for the points in $\mathcal{E}$ are answered as follows (all points are queried simultaneously). Let *root* be the root of T, let $left(root)$ be the left child of the root, and let $right(root)$ be the right child of the root. We select the points that fall outside $I(root)$ (i.e. the points that are inside of at least one disk) and partition the selected points based on inclusion in $I(left(root))$. The points that are outside $I(left(root))$ are given to the search on the subtree rooted at $left(root)$; the remaining points become input for the search on the subtree rooted at $right(root)$ with the only reservation that if a query point in the input set for $right(root)$ has a smaller index than the index of the rightmost leaf in the subtree rooted at $left(root)$ then this point does not belong to any disk preceding it in $\mathcal{E}$ and can be dropped.

Using point location queries it takes $O(\log n)$ time to answer an inclusion query in $I(v)$ at each level of T. There are $O(\log n)$ levels in T, giving $O(\log^2 n)$ query time for one point. Then, the general 2D OLBIS problem can be solved in $O(n \log^2 n)$ time with $O(n \log n)$ space. □

Notice that the extra $O(\log n)$ factor in time complexity, when compared to [6], is due to the query processing. We then obtain the following result for the polygonal chain simplification problem.

Theorem 5 *The* min-# *problem with angle constraints in $\mathbb{R}^3$ can be solved in $O(n^2 \log^2 n)$ time and $O(n^2)$ space.*

For the related *min-ε* problem we obtain:

Theorem 6 *The* min-ε *problem with angle constraints can be solved in $O(n^2 \log n)$ time and $O(n^2)$ space in $\mathbb{R}^2$ and in $O(n^2 \log^4 n)$ time, $O(n^2)$ space in $\mathbb{R}^3$.*

5 Conclusions

In this paper we presented solutions for the polygonal chain approximation problem with *small* angle constraints, thus closing the gap on the range of angles left in previous work [6]. Our solution for small angle constraints in $\mathbb{R}^2$ matches the $O(n^2)$ time and space complexities of that in [6].

Surprisingly, in $\mathbb{R}^3$, the time complexity of the solution for small turn angle constraints is higher by an $O(\log n)$ factor than that for large angle constraints [6] due to an extra $O(\log n)$ factor in processing queries. We leave the elimination of the extra $O(\log n)$ factor in this case as an open problem.

References

[1] Pankaj K. Agarwal, Lars Arge, and Ke Yi. An optimal dynamic interval stabbing-max data structure? In *SODA '05: Proceedings of the sixteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 803–812, Philadelphia, PA, USA, 2005. Society for Industrial and Applied Mathematics.

[2] G. Barequet, D.Z. Chen, O. Daescu, M.T. Goodrich, and J. Snoeyink. Efficiently approximating polygonal paths in three and higher dimensions. *Algorithmica*, 33(2):150–167, 2002.

[3] W. S. Chan and F. Chin. Approximation of polygonal curves with minimum number of line segments. In *ISAAC '92: Proceedings of the Third International Symposium on Algorithms and Computation*, pages 378–387, London, UK, 1992. Springer-Verlag.

[4] Bernard Chazelle. An optimal algorithm for intersecting three-dimensional convex polyhedra. *SIAM J. Comput.*, 21(4):671–696, 1992.

[5] Danny Z. Chen and Ovidiu Daescu. Space-efficient algorithms for approximating polygonal curves in two dimensional space. *Int. J. Comput Geometry Appl.*, 13(2):95–111, 2003.

[6] Danny Z. Chen, Ovidiu Daescu, John Hershberger, Peter M. Kogge, Ningfang Mi, and Jack Snoeyink. Polygonal path simplification with angle constraints. *Computational Geometry*, 32(3):173–187, 2005.

[7] Mark de Berg, Marc van Kreveld, Mark Overmars, and Otfried Schwarzkopf. *Computational Geometry: Algorithms and Applications*.

[8] David P. Dobkin and David G. Kirkpatrick. Determining the separation of preprocessed polyhedra: a unified approach. In *Proceedings of the seventeenth international colloquium on Automata, languages and programming*, pages 400–413, New York, NY, USA, 1990. Springer-Verlag New York, Inc.

[9] K. Kedem, R. Livine, J. Pach, and M. Sharir. On the union of jordan regions and collision-free translational motion amidst polygonal obstacles. *Discrete Comput. Geom.*, 1:59–71, 1986.