Shortest Paths, Visibility, and Optimization Problems in Planar Curvilinear Objects *

Elefterios A. Melissaratos and Diane L. Souvaine
Department of Computer Science, Rutgers University, New Brunswick, NJ 08903

May 15, 1990

1 Introduction

Souvaine and Dobkin [10],[2] defined the *splinegon* as a curved polygon in which the region bounded by each curved edge and the line segment joining its endpoint is always convex. They developed techniques and algorithms for solving many problems on splinegons, using the same structure as the original straight-line solutions and achieving the same asymptotic complexities. One polygon technique, however, is decomposition into the triangles or arbitrary convex pieces. Curved objects can be inherently non-convex and some curved objects cannot be triangulated without adding additional vertices, both on the boundary and in the interior. But horizontal visibility and monotone decomposition (linear-time equivalent to triangulation on polygons [8]) can be computed equally efficiently on splinegons. Dobkin *et al.* conjecture that problems for which the best known polygonal algorithm depends on triangulation or convex decomposition may in fact be solvable equally efficiently using monotone decomposition in algorithms easily modified to accommodate splinegons [3],[2],[10].

In this paper, we study a collection of optimization and visibility problems for which the best known polygonal algorithms require a balanced decomposition of the polygon and/or the shortest path tree, both derived from a triangulation of the polygon. We begin by defining a linear time and space refinement of the horizontal visibility decomposition of a splinegon, the *bounded degree decomposition*, in which each component is either a triangle (straight line or curvilinear) or a quadrilateral with two opposite sides on the boundary of the splinegon. This decomposition facilitates the generalization of the polygonal shortest path tree algorithm [6] and algorithm for computing the factor graph of a polygon [1],[5] to splinegons with no asymptotic penalty. These tools can then be used, as in the straight-line polygon case, to generate efficient algorithms for visibility problems such as visibility from an edge, ray-shooting and two-point shortest paths (for polygonal algorithms, see [1], [5], and [6], respectively), and optimization problems such as computing minimum-length area-separator and the maximum inscribed triangle (for polygonal algorithms, see [9]). Although most of the algorithms for curvilinear objects obtain the same asymptotic complexity as their polygonal counterparts, the maximum inscribed triangle algorithms does not. We conjecture that some curvilinear problems are inherently more difficult that their polygonal counterparts.

*This research was supported in part by NSF grant CCR-88-03549 and DIMACS Center STC88-09648.

2 Preprocessing Splinegons

Theorem 2.1 *Any simple splinegon can be decomposed in such a way that each component has at most three neighbors, in the same asymptotic time complexity as triangulating a simple polygon.*

Proof: To begin, preprocess the edges of the splinegon such that each edge is monotone with respect to both the x and the y axes, *i.e.* insert the extrema of each splinegon edge with respect to either axes as a new vertex, without duplicating edge endpoints. The convexity of the splinegon edges means that each edge can have at most one minimum and at most one maximum relative to each axis, and, consequently, the additional number of edges or vertices is at most $4n$. If the constant time suffices to compute the extrema of any edge, then this preprocessing uses $O(n)$ time. Next, decompose the splinegon into horizontal trapezoids (with curved sides) [11], [3], producing a linear number of new vertices. Ordinarily, there will be at most one interior vertex per trapezoid edge, but in some applications several vertices may have the same y-coordinate, producing an arbitrary number of vertices within a base of a trapezoid. Thus some trapezoids could have an unlimited number of neighbors. Refine the curvilinear trapezoids case by case, adding new vertices only on splinegon boundaries so that every component has at most three neighbors and the dual of the decomposition is a binary tree as in polygonal triangulations (fig. 1). □

Theorem 2.2 *The shortest path tree inside a simple splinegon with a designated root can be computed in $O(n)$ time, given the bounded degree decomposition.*

Proof: In [6] the authors describe a linear time and space algorithm for finding the shortest paths from a vertex v of a triangulated simple polygon to all the other vertices. The union of these paths form a tree called as *the shortest path tree* which subdivides P so that each region corresponds to a *funnel* based on some polygon edge i, denoted $F_v(i)$ (fig. 2a). Extending the edges of each funnel up to their intersection with the funnel's base produces a refined subdivision of P called the *shortest path map* (fig. 2b) The subdivision of a specific edge i is denoted by $S_v(i)$ and its size denoted by s_i^v. For any point x in P, $anchor^s(x)$ represents the last vertex on the shortest path from s to x (see [6], [7]).
The main step of the polygon algorithm is as follows: given a funnel and a triangle, with one of its sides coincident with the funnel base, split the funnel into at most two funnels which have bases the other two edges of the triangle. In splinegons, however, a convex chain of a funnel is a concatenation of straight line segments and convex curved segments. If the shortest path map is formed by extending the straight line segments of the funnels and the tangents at the endpoints of the curved segments up to the intersection of the corresponding splinegon edge, for a point x moving within a single component of the shortest path map, $anchor^s(x)$ is not a constant function but instead takes its values along a particular convex segment of the splinegon boundary. Thus, the funnel splitting operation may create a new vertex either in the funnel or in the boundary of a new component or on both. Luckily, a region with 2 children must be a straight-edged triangle, so only regions with at most 1 child need new processing. Splitting these funnels may involve computing tangents from a point to a curve or between a pair of curved edges and, since each node is a quadrilateral, there are are two splitting points rather than just one. But we may assume that each curved operation requires constant time [2]. Careful analysis of splitting in finger search trees shows that the total work over all zero or one child cases remains $O(n)$ and thus the recursive formula used in [6] to prove the linearity

of the entire algorithm still applies.[1]

Theorem 2.3 *The factor graph and the augmented factor graph of a simple splinegon S can be computed in $O(n)$ time and space, given the bounded degree decomposition.*

Proof: The polygonal algorithm extends easily to splinegons [1]. The balanced decomposition tree of S can be created in linear time and space [6]. Let S_l be a splinegon at level l in the decomposition tree. Let d_l be the bisecting diagonal of S_l. It is known that the boundary of S_l consists of some edges of the initial splinegon as well as of some bisecting diagonals. The factor graph is an augmented decomposition tree with edges between d_l and the bounding diagonals of S_l. In some applications, the hourglass corresponding to each pair of diagonals is also needed. These edges and hourglasses are added bottom-up, as in the original polygon algorithm [1]. If for every splinegon component in levels $1...k$, the hourglasses between any pair of bounding diagonals has been computed, level $k+1$ is obtained by "deleting" all the diagonals at that level and computing the hourglasses between any bounding diagonal of the left component and any bounding diagonal of the right component by concatenating two hourglasses at level k. Since one hourglass can use $O(n)$ space, the augmented factor graph could use $O(n^2)$ space overall. By keeping each edge of an hourglass only at the highest level in which it appears in the tree, the augmented factor graph can be constructed in $O(n)$ time. Its size is $O(n)$ and each node has degree $O(\log n)$. □

3 Applications

Theorem 3.1 *Computing the part of the boundary of a simple splinegon P of n vertices which is visible from an edge requires $O(n)$ time given the horizontal visibility decomposition of P.*

Proof: Given an edge j of P, we wish to compute all points x on the boundary of P for which there exists at least one point y on j such that $xy \subseteq P$. The polygon algorithm uses the fact that if edges i, j are visible from each other then the shortest paths from p_{j+1} to p_i ($SP_{p_{j+1}}(p_i)$) and from p_{i+1} to p_j ($SP_{p_{i+1}}(p_j)$) are inward disjoint convex chains [6]. For splinegons, this fact does not hold (fig. 3). We present a new method for both polygons and splinegons based on local computations. Find the shortest path maps from p_j and p_{j+1} respectively. Merge $S_{p_j}(i)$ and $S_{p_{j+1}}(i)$ into a linear-sized subdivision M. If x moves along an elementary segment I of M, the anchors of x with respect to the endponts j remain unchanged. Thus we can unambiguously refer to $anchor^{p_j}(I)$ and $anchor^{p_{j+1}}(I)$. For each I, perform the following simple test: If $anchor^{p_{j+1}}(I) <> anchor^{p_j}(I)$ then for every point x on I, x is visible from edge j. □

Theorem 3.2 *Given the factor graph of a simple splinegon S, a query point q and a ray passing through q, the first intersection of the ray with the splinegon can be reported in $O(\log n)$ time.*

Proof: Locate the query point q in the decomposition [4]. As in [1], find the the diagonal crossed by the shooting ray closest to the root of the decomposition tree. Descend the augmented factor graph as follows: at each node visited check either its $L(v)$ or $R(v)$ list; for each w in $L(v)$, test if the ray from q avoids the hourglass corresponding to the edge (v,w);

[1] We can also compute the shortest path tree in linear time directly from the horizontal visibility decomposition without the overhead of computing the bounded degree decomposition by using multiway splitting of the funnels. Some applications require the bounded degree decomposition, however.

at a leaf, no such hourglass exists but the edge of the splinegon intersected by the ray can be computed in $O(1)$ time. To achieve the $O(\log n)$ complexity, transform the factor graph so that it has bounded degree. Using fractional cascading, the $O(\log n)$ intersection tests between convex chains and line can all be accomplished in $O(\log n)$ time. This algorithm differs from the original polygon algorithm [1] in only one respect. In the polygon algorithm, the test of whether a line intersects an hourglass is transformed to the dual problem of point inclusion in a convex polygon, solvable using a variant of binary search. Since no duality tranforms are known to apply to curved objects, we solve the line-hourglass intersection problem directly using binary search on the two convex chains bounding the hourglass. □

Theorem 3.3 *Given the factor graph of a simple splinegon S and two query point p and q, the shortest path from p to q and its length can be reported in $O(\log n + k)$, where k is the number of segments in the path.*

Proof: The polygon algorithm [5] extends directly. Locate the points p, q in the decomposition [4]. Find the lowest common ancestor d of the regions containing p, q. Starting from p, q follow the path to d, collecting all diagonals which separate p and q. Between any pair of consecutive diagonals corresponds an edge of the factor graph. Concatenate the hourglasses which correspond to these pairs of diagonals, producing a "big" hourglass. Find the tangents from p, q to this hourglass. The $O(\log n)$ query and linear space and preprocessing results from augmenting only the subgraph of the initial factor graph consisting of the first $O(n/\log n)$ nodes. □

Theorem 3.4 *The minimum length area-separator of a simple splinegon can be computed in $O(n^2)$ time and $O(n)$ space.*

Proof: If two points x and y lie on the boundary of simple splinegon P and define a directed line segment $xy \subseteq P$ that separates P into two sets P_L and P_R with given areas K_L and K_R, respectively, then xy is called an *area-separator*. An area-separator for a splinegon may not exist; for example, there are splinegons that cannot bisected (i.e. $K_L = K_R$). A direct extension of the polygon algorithm [9] finds the minimum length separator, if one exists, and reports the degeneracy otherwise.

Assume that point x is on edge i and y on edge j. The idea is to fix a particular edge j, find the shortest path maps from its endpoints, and merge $S_{p_j}(i)$, $S_{p_{j+1}}(i)$. The area of P to the left (resp. right) of $SP_{p_{j+1}}(p_i)$ is denoted $AL_{p_{j+1}}(p_i)$ (resp. $AR_{p_{j+1}}(p_i)$) . For each elementary segment I of i such that $AL_{p_{j+1}}(p_i) \leq K_L$, and $AR_{p_j}(p_{i+1}) \leq K_R$, we wish to find points $x = (x_1, y_1)$ and $y = (x_2, y_2)$ on I and j respectively such that a) the $anchor^{p_j}(I)$ and $anchor^{p_{j+1}}(I)$ do not lie in the same open halfspace defined by $\overrightarrow{xy}$, b) the area of the curvilinear quadrilateral $p_i x y p_{j+1}$ equals $K_L - AL_{p_{j+1}}(p_i)$, and c) length of xy is minimum. This continuous optimization problem can be solved in $O(1)$. The linearity of the shortest path trees implies a total of $O(n)$ elementary segments, yielding an $O(n)$ algorithm to find the separator xy when y lies on a fixed edge i. Summing all over edges i we get an $O(n^2)$ time algorithm. □

Theorem 3.5 *The maximum area or perimeter triangle inscribed in a simple splinegon can be found in $O(n^4)$ time and $O(n)$ space.*

Proof: For three points x, y, z on the boundary of a simple splinegon P to define an inscribed triangle, they must be pairwise visible. Fix edges i and j and find the maximum triangle xyz

such that x lies on any edge k but y and z lie on i and j respectively. Repeat the process for all the pairs i and j. To do so, construct the shortest path maps from the endpoints of edges i and j. Merge these four shortest path maps to create a set of elementary segments on the boundary of P. Let seg_i, seg_j and seg_k represent arbitrary triple of elementary segments on edges i, j and k, respectively. The goal is to find points x, y, z on seg_k, seg_i, seg_j respectively such that $anchor^{p_i}(x)$ and $anchor^{p_{i+1}}(x)$ lie in the opposite half-planes defined by the line through x and y, also $anchor^{p_j}(x)$ and $anchor^{p_{j+1}}(x)$ lie in the opposite half-planes defined by the line through x and z and finally $anchor^{p_j}(y)$ and $anchor^{p_{j+1}}(y)$ lie on the opposite sides of yz and the area of xyz is maximum. This is a continuous optimization problem (possibly infeasible) with a constant number of constraints in a constant dimensional space, thus solvable in constant time.

This algorithm calls the shortest path algorithm $O(n^2)$ times, using $O(n^3)$ time. The number of triples of elementary segments for a triple of edges is $O((s_k^{p_{i+1}} + s_k^{p_j} + s_k^{p_i} + s_k^{p_{j+1}})s_i^{p_{j+1}} s_j^{p_i})$. Thus the total number of calls of to the continuous optimization procedure

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{k=1}^{n}((s_k^{p_{i+1}} + s_k^{p_j} + s_k^{p_i} + s_k^{p_{j+1}})s_i^{p_{j+1}} s_j^{p_i})$$

Since $s_k^{p_i} < n$, the above sum is $O(n^4)$. In the polygon case, at least two of $anchor^{p_i}(x)$, $anchor^{p_j}(y)$ and $anchor^{p_k}(x)$ must lie on the boundary of the triangle. Thus, it was possible to reduce the number of triples of elementary segments considered overall to a quadratic number, yielding a cubic algorithm [9]. This reduction is not possible in the splinegon case, leaving an asymptotic gap between the polygon and splinegon solutions for this problem. □

References

[1] B. Chazelle, L. Guibas, "Visibility and intersection problems in the plane," *Proc. of ACM Symp. Comp. Geom.*, 1985.

[2] D. Dobkin, D. Souvaine, "Computational geometry in a curved world." *Algorithmica 5 (1990).*

[3] D. Dobkin, D. Souvaine, C. Van Wyk, "Decomposition and intersection of splinegons." *Algorithmica,* 3 (1988), 473-485.

[4] H. Edelsbrunner, L. Guibas, G. Stolfi, "Optimal point location in monotone subdivisions." *SIAM J. Comput.*, 1986.

[5] L. Guibas, J. Hershberger, "Optimal shortest path queries in a simple polygon." Proc. ACM Symp. on Comp. Geometry, 1987.

[6] L. Guibas, J. Hershberger, D. Leven, M. Sharir, R. Tarjan, "Linear time algorithms for Visibility and shortest path problems inside triangulated simple polygons." *Algorithmica 2 (1987)*, 209-233.

[7] J. Hershberger, "An Optimal Visibility Graph Algorithm for triangulated simple polygons." *Algorithmica 4* (1989), 141-155.

[8] A. Fournier and D. Y. Montuno, "Triangulating simple polygons and equivalent problems," *ACM Transactions on Graphics 3* (1984), 153-74.

[9] E. A. Melissaratos, D. L. Souvaine, "On Solving Geometric Optimization Problems Using Shortest Paths," *Proc. of the 6th ACM Symp. on Comp. Geometry*, June, 1990.

[10] D. L. Souvaine, "Computational Geometry in a Curved World." Ph.D. Thesis, Princeton University, October, 1986.

[11] R.E.Tarjan, C. Van Wyk, "Triangulation of a simple polygon," *SIAM Journal of Computing* (1987).

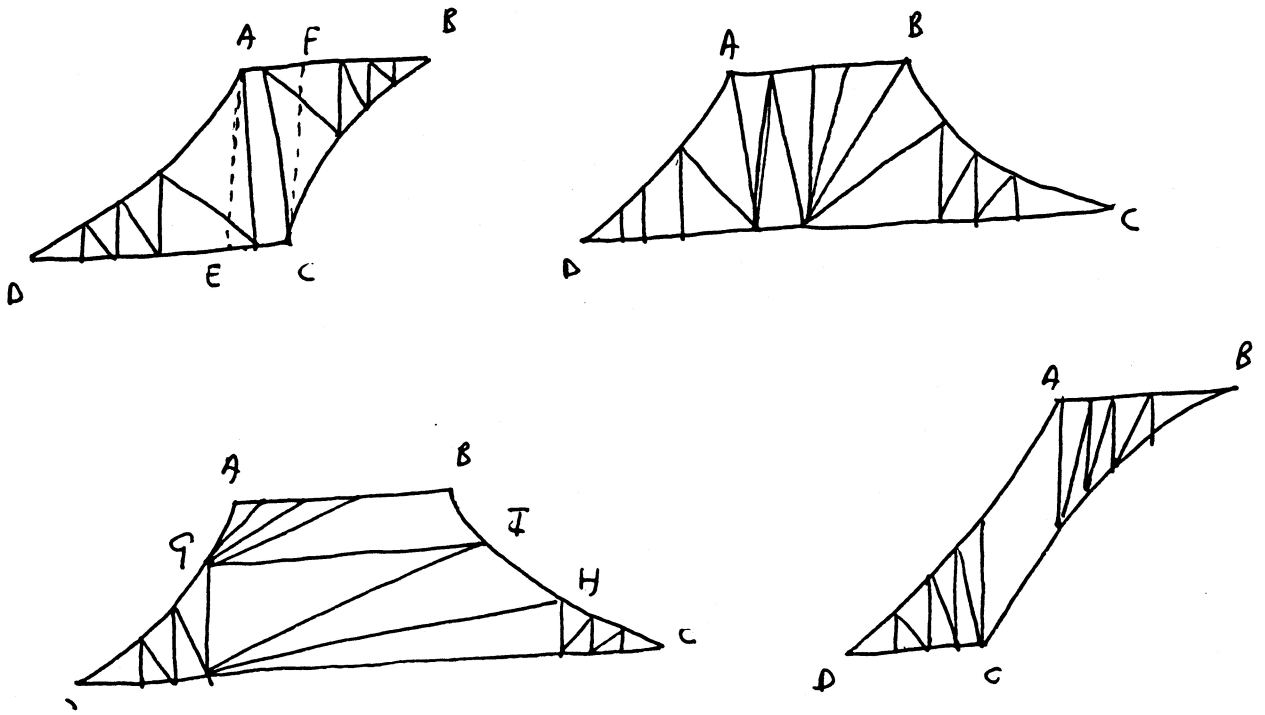
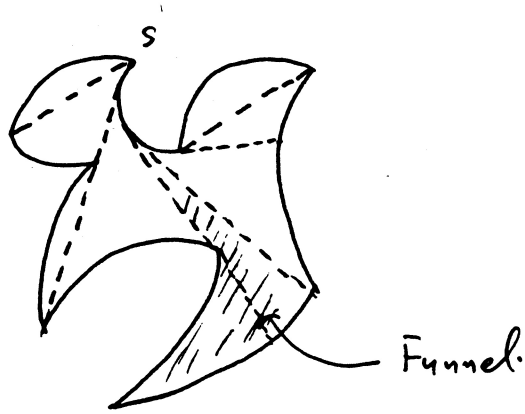


Figure 1: Refinement of degenerate trapezoids.



shortest path map.

Figure 2: a) Funnel; b) Shortest-Path Map

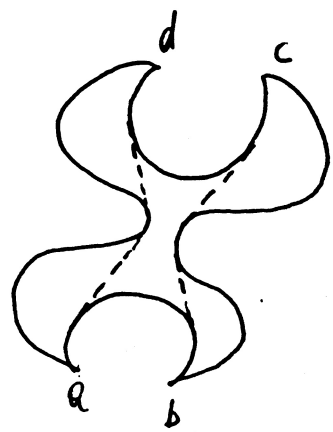


Figure 3:

Edges ab, cd are visible but the shortest paths from a to d and b to c are not inward convex.