Algorithmica (1988) 3: 473-485

Algorithmica

Decomposition and Intersection of Simple Splinegons[1]

David P. Dobkin,[2] Diane L. Souvaine,[2,3] and Christopher J. Van Wyk[4]

Abstract. A splinegon is a polygon whose edges have been replaced by "well-behaved" curves. We show how to decompose a simple splinegon into a union of monotone pieces and into a union of differences of unions of convex pieces. We also show how to use a fast triangulation algorithm to test whether two given simple splinegons intersect. We conclude with examples of splinegons that make the extension of algorithms from polygons to splinegons difficult.

Key Words. Computational geometry, Curves, Simplicity testing, Intersection detection, Monotone decomposition, Convex decomposition.

1. Introduction. A major failing of many results in computational geometry is that they work only on well-behaved objects. This is a severe limitation in applying computational geometry algorithms to practical problems since most real shapes are not polygons at all, let alone convex polygons. Despite the proliferation of work in this area, a recent survey [15] and a recent book [17] contain few results that involve curved objects.

To combat this, Souvaine [21] has defined the splinegon as a generalization of the polygon. Splinegons provide an analytical framework within which to study algorithms on curved objects. Souvaine presents three methods for extending polygonal algorithms to splinegons and applies them to a broad class of examples. Here we focus specifically on algorithms for decomposing simple splinegons into better-behaved pieces, and detecting whether two simple splinegons intersect.

Let $\tau(n)$ be the time required to triangulate a simple polygon that has n vertices. Computing a triangulation is linear-time equivalent to computing for each vertex v of a polygon the zero, one, or two edges that are internally visible to v in the horizontal direction [3], [6]. Tarjan and Van Wyk have given an $O(n \log\log n)$-time algorithm for the problem of computing internal horizontal-vertex-visibility information [22], which, by the above-mentioned connection with triangulation, shows that $\tau(n) = O(n \log\log n)$; no lower bound better than the trivial $\tau(n) = \Omega(n)$ is known.

[1] Work on this paper by David A. Dobkin and Diane L. Souvaine was partially supported by National Science Foundation Grants MCS 83-03926 and DCR 85-05517. Diane L. Souvaine was also partially supported by an Exxon Foundation Fellowship.

[2] Department of Computer Science, Princeton University, Princeton, NJ 08544, USA.
[3] Current address: Department of Computer Science, Rutgers University, New Brunswick, NJ 08903, USA.
[4] AT&T Bell Laboratories, Murray Hill, NJ 07974, USA.

Received September 12, 1986; revised April 14, 1987. Communicated by D. T. Lee.

Tarjan and Van Wyk also show that given both the internal and external horizontal-vertex-visibility information for a polygon (both of which can be computed using any triangulation algorithm), one can test whether the polygon is simple in linear time. It is straightforward to extend both the algorithms for internal horizontal-visibility computation and for simplicity testing to splinegons. We use both extended algorithms to obtain our results.

The rest of this paper is organized as follows. We define splinegons formally in Section 2. In Section 3 we extend the notion of monotone decomposition from polygons to simple splinegons. This result is not difficult, but it illustrates several pitfalls that one often encounters in computing with curved objects.

In Section 4 we consider the problem of extending the convex decomposition of a simple polygon to splinegons. We show that several extensions fail because not all splinegons admit such decompositions. Our best result is a method for decomposing a splinegon into a union of differences of unions of convex pieces. This result uses the monotone decomposition of a simple splinegon with a simple carrier polygon.

In Section 5 we use $O(\tau(n))$-time simplicity testing on splinegons to derive an $O(\tau(n))$-time algorithm that detects whether the boundaries of two simple n-sided splinegons intersect. This result involves a novel application of Jordan sorting [10]. Detecting whether the areas of two splinegons intersect is a simple corollary of this result [2]. Heretofore, even the restricted problem of whether two simple n-gons intersect was not known to be solvable in $o(n \log n)$ time. We present this result in the framework of splinegons in order to state it in a more generally useful form.

In Section 6 we discuss two limitative results on splinegons that shed light on why they sometimes pose more difficult problems than polygons. Section 7 contains brief concluding remarks.

2. Definitions. A *splinegon* S can be formed from a polygon P on n vertices, $v_1, v_2, \ldots, v_n$, by replacing each line segment $\overline{v_i v_{i+1}}$ with a curved edge e_i which also joins v_i and v_{i+1} and which satisfies the following condition: the region S-seg$_i$ bounded by the curve e_i and the line segment $\overline{v_i v_{i+1}}$ must be convex.[5] Note that the new edge need not be smooth. The polygon P is called the *carrier polygon* of the *splinegon S*.

Splinegons form a rich class of geometric objects. Souvaine discusses many examples, and presents natural extensions of the notions of simplicity and convexity from polygons to splinegons [21]. We note here only that the convexity of a carrier polygon does not imply the convexity of its splinegon, and that the simplicity of a splinegon and its carrier polygon are completely unrelated (see Figure 1).

Given an n-sided simple splinegon S, we classify its edges as *concave-in* or *concave-out* as follows. A line segment edge is concave-in. If e_i is not a line segment, and for any point $p \in$ S-seg$_i$ a line segment that joins p to e_i intersects

[5] Subscripts are always interpreted modulo n.

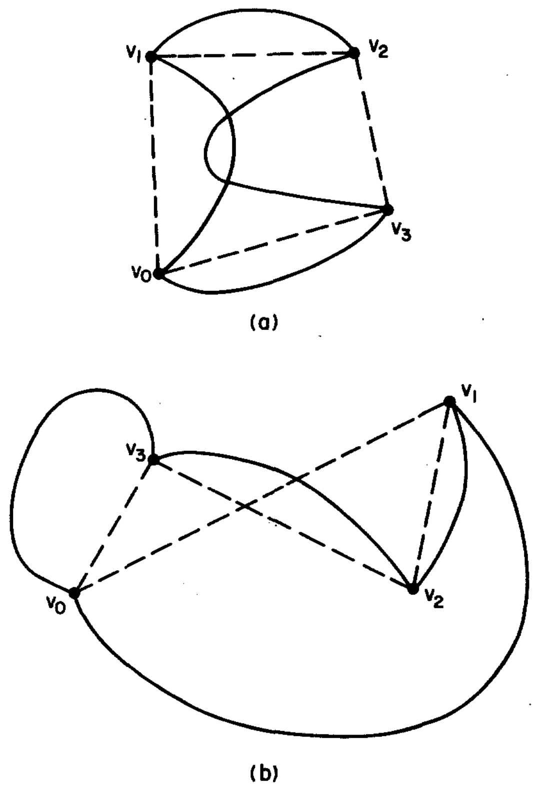

Fig. 1. Two four-sided splinegons, with dashed lines showing carrier polygons. In (a) the carrier polygon is convex, but the splinegon is neither simple nor convex; in (b) the splinegon is simple, but the carrier polygon is not.

the interior of S, then e_i is a concave-in edge. If e_i is not a line segment and for any point $p \in$ S-seg$_i$ a line segment that joins p to e_i intersects the exterior of S, then e_i is a concave-out edge.

Geometric algorithms on linear objects use certain primitive procedures (e.g., calculating segment–segment intersections) that can be done in constant time. For splinegons, the analogous primitives are more complex and may involve unsolvable problems (e.g., finding exact roots of fifth-degree polynomials). We avoid this difficulty by postulating the existence of the following oracles, which we use as primitive operations in our algorithms:

(1) Compute the intersection of two curved edges, or the maximum and minimum separation between them.
(2) Compute the intersection of a line with a curved edge.
(3) Given a curved edge and either a direction or a point, report both the point and the direction of a line that supports the edge of that point.
(4) Determine the line that supports a pair of curved edges.

All of the algorithms in this paper require $\Theta(n)$ of these operations on an n-sided splinegon, so we will not account separately for these operations in the time bounds for the algorithms.

3. Monotone Decomposition. A splinegon S is *y-monotone* if there exist two points $p_{\text{low}}, p_{\text{high}} \in S$ that partition S into two chains that are monotone in the y-direction. In this section we discuss the decomposition of an arbitrary splinegon S into y-monotone splinegons. In the case of polygons, the decomposition is an easy consequence of the horizontal-vertex-visibility decomposition [3], [6], [14], to which we now turn.

Given a simple polygon, the horizontal line segments that join a vertex to its visible edge or edges define a partition of the polygon into trapezoids. This partition can be computed in $O(\tau(n))$ time on an n-sided simple polygon [22]. Assuming that no polygon vertices have the same y-coordinate, each trapezoid contains exactly two polygon vertices: one on its top edge and one on its bottom edge. Given the horizontal-vertex-visibility partition, one computes a y-monotone decomposition by adding an edge between the polygon vertices of any trapezoid that has a polygon vertex lying in the middle of a horizontal edge. In the rest of the paper we refer to such vertices as *y-notches.*

The algorithm for horizontal-vertex-visibility partition of a polygon relies on two key properties of the edges:

(1) Each edge crosses any horizontal line at most once.
(2) If a set of edges crosses two horizontal lines, the order in which they cross the horizontal lines is the same on both lines.

Both properties are satisfied by simple splinegons whose edges are all monotone in the y-direction. Thus we generalize the notion of horizontal-vertex-visibility partition to splinegons by adding (at most two) vertices to each side so that all sides are monotone in the y-direction. The horizontal line segments that join vertices to visible edges partition the splinegon into *visibility cells*; visibility cells are bounded by one or two horizontal edges and by y-monotone portions of two splinegon sides; all splinegon vertices occur on the horizontal edge of some visibility cell.

Given the horizontal-vertex-visibility partition of a splinegon, there is one more twist to computing a y-monotone decomposition. We cannot extend the polygon algorithm directly because there might be no obvious way to connect two vertices of a splinegon (see Figure 2). Thus we amend the polygon algorithm as follows: for any visibility cell with a y-notch v, if the line segment l that connects the two vertices of the cell crosses either of the two sides of the cell, let e be the edge that l touches closer to v; let l' be the line through v that is tangent to e at a relative interior point of e; add the line segment from v to $l' \cap e$ as an edge of the y-monotone decomposition. This operation adds a vertex to the splinegon; the total number of vertices added in this way is at most the number of y-notches.

The result of this amended algorithm is a correct, y-monotone decomposition of the splinegon. However, the carrier polygon of some of the y-monotone pieces might not be simple (see Figure 3). Should one desire to simplify the carrier

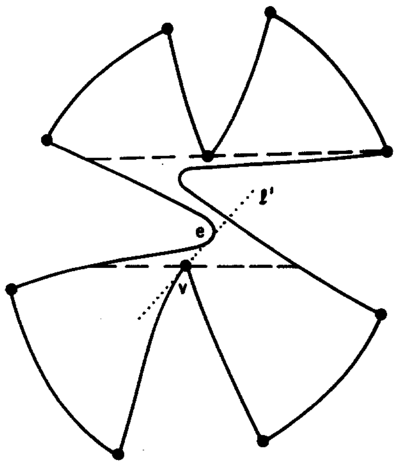

Fig. 2. A ten-sided splinegon divided into five visibility cells. To compute a y-monotone decomposition, we must further subdivide the visibility cell bounded by dashed line segments by adding vertices along both of its curved edges. One of these vertices is at $l' \cap e$.

polygon, it is easy to do so in linear time by scanning from p_{low} to p_{high}, adding new vertices horizontally opposite existing vertices whenever necessary to keep the carrier polygon simple. This step adds no more than one vertex per existing vertex.

We summarize the above results in the following theorem:

THEOREM. *An n-sided splinegon can be decomposed into y-monotone pieces with*

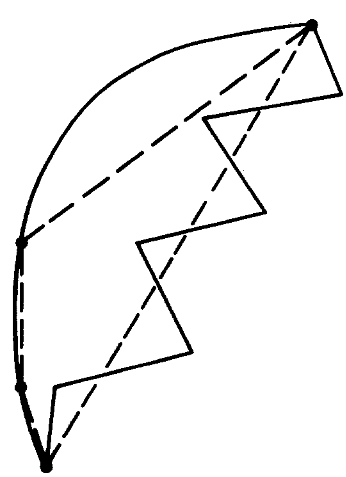

Fig. 3. A y-monotone splinegon with eight sides and a nonsimple carrier polygon (using the long dashed edge and the seven solid line segments) can also be represented as a ten-sided splinegon with a simple carrier polygon (using the three shorter dashed edges and the seven solid line segments).

simple carrier polygons in $O(\tau(n))$-time. The total number of vertices in the decomposition is $O(n)$.

4. Convex Decomposition. The problem of decomposing a simple polygon into convex pieces has received attention in various forms for several years [1], [5], [7], [11], [12], [14], [18]. The motivation for this decomposition is to solve problems on more complicated general polygons by combining the solutions to the problem on convex subpolygons, so decomposition into an infinite number of convex pieces is not interesting. Thus we use the term "convex decomposition" to mean a decomposition into a finite number of convex pieces.

Any polygon can be decomposed into a union of convex pieces, $\bigcup_i A_i$ [1], [5], [7], [11], [12], [14], [18]. Direct extension of any of these results to the case of splinegons is impossible: the splinegon in Figure 4(a) cannot be decomposed into a finite number of convex pieces.

However, Figure 4(a) also suggests a promising amendment to the problem. For any representation of a splinegon S, each of the regions S-seg$_i$'s is convex,

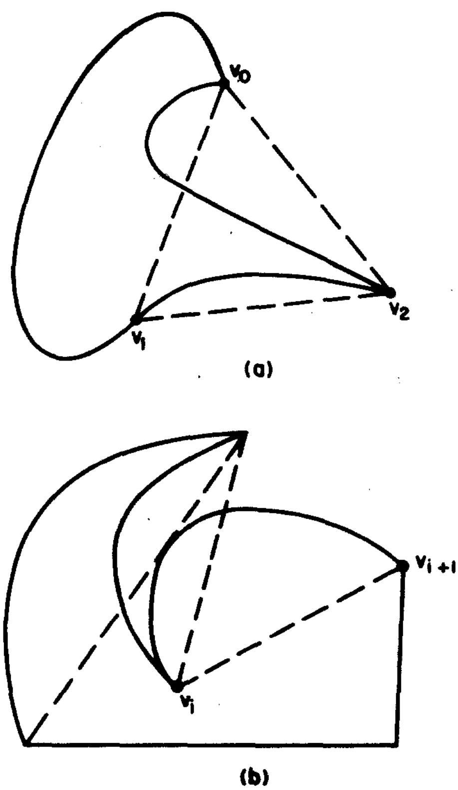

Fig. 4. Simple splinegons that cause various decomposition schemes to fail. The splinegon in (a) cannot be represented as $\bigcup_i A_i$ for any finite choice of convex A_i; in (b) the edges incident to v_i are tangent at v_i, so a decomposition of the form $\bigcup_i A_i - \bigcup_i B_i$ for any finite choice of convex A_i and B_i must be incorrect in a neighborhood of v_i.

so we could try to decompose the splinegon into a difference of two unions of convex sets: $\bigcup_i A_i - \bigcup_i B_i$. Sets A_i would include some convex decomposition of the carrier polygon, together with the S-seg's of concave-in edges of S. Sets B_i would be the S-seg's of concave-out edges of S.

There are several problems with this scheme. Existing algorithms work only on simple polygons, but a simple splinegon can have a nonsimple carrier polygon (Figure 1(b)). Even if the carrier polygon is simple, however, the scheme of decomposing S into a difference of unions of convex sets, $\bigcup_i A_i - \bigcup_i B_i$, is flawed. Figure 4(b) shows a simple splinegon with a simple carrier polygon; no matter how much the carrier is refined, any convex decomposition of the form $\bigcup_i A_i - \bigcup_i B_i$ must be incorrect near vertex v_i.

If the carrier polygon were simple, then we could form a kind of "chained" convex decomposition of S: begin with a convex decomposition of the carrier polygon, then unite in the concave-in edges and subtract out the concave-out edges *in order.* For example, we might write the splinegon in Figure 4(a) as $((\Delta v_0 v_1 v_2 \cup \text{S-Seg}_0) - \text{S-seg}_1) - \text{S-seg}_2$. Although it is correct, this decomposition has several disadvantages: it is guaranteed to consist of $\Omega(n)$ convex pieces, and the ordering of the union and subtraction operations makes it awkward to parallelize algorithms that use it. Another disadvantage to requiring that the carrier polygon be simple is discussed in Section 6: there exist n-sided splinegons whose smallest simple carrier polygons have $\Omega(n^2)$ vertices.

All of these examples lead us to suggest an alternate scheme for decomposing a splinegon into convex pieces. First, apply the algorithm of the last section to produce a decomposition into y-monotone pieces with simple carrier polygons. Each of these pieces can be decomposed into convex pieces in several ways. The result of any of these efforts is a decomposition of the splinegon interior in the form $\bigcup_j (\bigcup_i A_{ij} - \bigcup_i B_{ij})$, with convex A_{ij} and B_{ij}.

A naive way to perform the convex decomposition of a y-monotone splinegon with a simple carrier polygon is to form the convex decomposition of the carrier polygon, then unite in the concave-in S-seg's and subtract out the concave-out S-seg's. This results in a decomposition whose size is $\Omega(n)$.

An approach that offers greater promise for decomposing a y-monotone splinegon S is to form a splinegon S' by replacing each concave-out edge of S by a line segment that joins its two vertices. Note that the y-monotonicity of S guarantees that S' is simple. Splinegon S' is readily decomposed into the smallest possible number of convex pieces, opt(S').

THEOREM. *Splinegon S' can be decomposed optimally into the union of convex pieces (with or without Steiner points) using existing polygonal algorithms* [1], [7], [11], [12].

PROOF. Form polygon Q by replacing each concave-in edge of S' by a convex polygonal chain each of whose edges is tangent to the curved edge. The edges added to decompose Q are identical to the edges added to decompose S', so opt(S') = opt(Q). □

Given the convex decomposition of S', the concave-out S-seg's can be subtracted from the result to give a convex decomposition of S.

COROLLARY. *A monotone splinegon S can be decomposed into the union of the convex decomposition of S' (as defined above) and the difference of the concave-out S-segs. The number of pieces in the convex decomposition is* $\mathrm{opt}(S')+v(S)$, *where $v(S)$ is the number of concave-out S-seg's in S.*

5. Intersection Detection. Given two n-sided simple splinegons K_1 and K_2, we wish to detect in $O(\tau(n))$ time whether their boundaries have any points in common. The previous best-known result related to this problem is that detecting the intersection of simple polygons can be performed in $O(n \log n)$ time [20]. For convex polygons, $\Omega(n)$ is a lower bound on the time to detect boundary intersection, even if preprocessing is allowed; however, area intersection can be detected in $O(\log n)$ time [2].

Our approach is to create from K_1 and K_2 a merged splinegon M such that the boundaries of K_1 and K_2 are disjoint if and only if M is simple. Splinegon M consists of the edges of K_1 and K_2 together with a "bridge" between them that is composed of a constant number of edges. One way to find such a bridge was proposed by Hershberger [8]; it requires a linear-time algorithm for computing the convex hull of a simple splinegon [19], and uses two cases depending on whether the convex hull of one splinegon contains the other. Our method for finding a bridge uses Jordan sorting [10], an algorithm that plays a crucial role in $O(\tau(n))$-time simplicity testing.

ALGORITHM FOR INTERSECTION DETECTION

(1) Find the y-extent (minimum and maximum y-coordinates of any point) of K_1 and K_2. If the y-extents of K_1 and K_2 do not overlap, then $K_1 \cap K_2 = \varnothing$. Otherwise, choose a value y_{cut} that lies in both y-extents. Assume without loss of generality that each edge of K_1 and K_2 intersects the line $y = y_{\mathrm{cut}}$ in no more than two points.
(2) For $i = 1, 2$,
    (a) Find the sequence of points (p_j) at which K_i crosses $y = y_{\mathrm{cut}}$, in the order in which they appear around the boundary of K_i.
    (b) Let σ_i be the sequence (p_j) sorted in order of increasing x-coordinate.
    (c) Scan σ_i in order of increasing x, labeling each point. The label assigned to point p_j should be $touch_i$ if the boundary of K_i does not cross $y = y_{\mathrm{cut}}$ at p_j; in_i if the interior of K_i lies to the right of p_j; and out_i if the interior of K_i lies to the left of p_j.
(3) Merge σ_1 and σ_2 into a single sorted sequence σ.
(4) Scan σ for a consecutive pair of crossing points $q_i \in K_1$ and $q_2 \in K_2$ that are labeled with different subscripts.
(5) Pry each splinegon K_i slightly open at q_i, so that q_i is split into two points q_i+ and q_i-, with q_i+ above q_i-. Let M be the splinegon consisting of the "slightly opened" K_1 and K_2 together with two nonintersecting segments

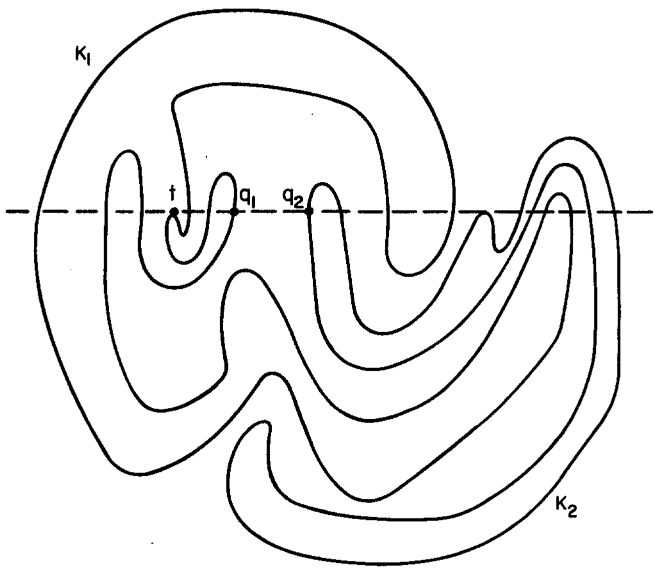

Fig. 5. Illustrating intersection detection for simple splinegons K_1 and K_2. Point t is labeled *touch*$_1$, q_1 is labeled *out*$_1$, and q_2 is labeled *in*$_2$. The bridge to build M will be constructed between q_1 and q_2.

that join q_1+ to q_2+ and q_1- to q_2-. The boundaries of K_1 and K_2 intersect if and only if M is not simple.

(See Figure 5.)

THEOREM. *This algorithm correctly detects whether the boundaries of two n-sided simple splinegons intersect in $O(\tau(n))$ time.*

PROOF. First we show that the algorithm is correct. Splinegon M is constructed by adding two edges to splinegons K_1 and K_2. These edges do not cross each other, and they are chosen so that they do not cross any edge of K_1 or K_2. Since K_1 and K_2 are both simple, M is nonsimple if and only if an edge of K_1 and an edge of K_2 have nonempty intersection.

Next we show that the total work done by the algorithm is $O(\tau(n))$. Step (1) can be performed in linear time. The definition of splinegons implies that the output of step (2a) has size at most $2n$, and that this output can be computed in $O(n)$ time. The algorithm of [10] can be used to perform step (2b) in $O(n)$ time. Steps (2c), (3), and (4) involve linear-time scans of lists of length $O(n)$. Step (5) can be performed in $O(\tau(n))$ time using the algorithm of [22]. □

It is easy to extend this result to detect area intersection [2].

COROLLARY. *It is possible to detect in $O(\tau(n))$ time whether the areas of two n-sided simple splinegons intersect.*

PROOF. Use the above algorithm to detect whether the boundaries of the two splinegons intersect. If they do not, the method described by Preparata and Shamos to determine whether a point lies inside a polygon [17] can easily be generalized to determine whether one splinegon lies entirely inside the other. □

Although it is not mentioned in the algorithm, the scan of step (4) can even obviate step (5): if σ' contains a sequence of the form $\langle in_1, in_2, out_1, out_2\rangle$, then the boundaries of the two splinegons intersect.

The assumption that the splinegons intersect the cutting line only in isolated points can be removed by having the splinegon oracle report only the leftmost and rightmost points of each connected component of the intersection. This technique is a generalization of that used by Van Wyk [23].

6. Limitative Results. Topologists often refer to "curvilinear triangulations" of surfaces [4], [9], [16]. In the context of splinegons, it is natural to consider constructing such a subdivision using additional edges that enjoy the same property as splinegon sides. The visibility cell of Figure 2 shows that this is impossible if we do not permit the introduction of additional vertices or "Steiner points." Thus, a curvilinear triangulation of an $n/2$-sided splinegon could have as many edges as a curvilinear triangulation of an n-sided splinegon. This has implications for the translation to splinegons of algorithms that proceed by hierarchical triangulation [13]; Souvaine discusses more details [21].

We now show that simplifying the carrier polygon can be quite expensive by constructing an n-sided splinegon G whose smallest simple carrier polygon has $\Omega(n^2)$ vertices. Begin by constructing an equilateral, equiangular polygonal path C of k segments, with vertices $v_0, v_1, \ldots, v_{k-1}$, $k>2$, such that C together with the line segment $\overline{v_0v_{k-1}}$ bounds a convex region. Let R be the (possibly infinite) open region bounded by $\overline{v_0v_{k-1}}$ and the two lines that contain $\overline{v_0v_1}$ and $\overline{v_{k-2}v_{k-1}}$, whose intersection with C is empty. Let h_1 be a point interior to the region bounded by C and $\overline{v_0v_{k-1}}$ and h_2 be a point in region R; let $H=\overline{h_1h_2}$. Let p and q be points in R such that $\overline{pv_0}$ and $\overline{qv_{k-1}}$ do not intersect H, but $\overline{pq}$ does intersect H. The following lemma implies that we can construct a splinegon edge from p to q that fits C "very tightly" in that any inscribed path must contain at least k segments:

LEMMA. *There exists a curve D that joins p and q such that*

(1) *D does not intersect H,*
(2) *$D\cup\overline{pq}$ bounds a convex region, and*
(3) *any polygonal path inscribed in D that does not intersect $H\cup C$ contains at least k segments.*

PROOF. Erect perpendicular bisectors to each of the segments of C. Define points w_i on these perpendicular bisectors as follows: $w_{-1}=p$; for $0\le i<k-1$, let w_i be the intersection of the line through w_{i-1} and v_i and the perpendicular

bisector of the edge $\overline{v_i v_{i+1}}$. Take D to be any convex curve between p and q that passes between each point w_i and the corresponding segment of C; such a D obviously exists.

The perpendicular bisectors containing the points w_i define sectors with respect to the center of the curve C. The key observation to the proof of the lemma is that any inscribed path in edge D that does not cross H must have a vertex in each of these sectors: any segment with endpoints on D that are not in adjacent sectors must intersect C by the way the points w_i were chosen. Since there are k sectors, any inscribed path in D that does not cross H has at least k vertices. □

(See Figure 6.)

Notice that any curve that lies between D and C has the same inscribed-paths property as D. The way to construct splinegon G is now clear: we pack many edges between C and D; each edge adds only one vertex to G, but adds at least k vertices to any simple carrier polygon.

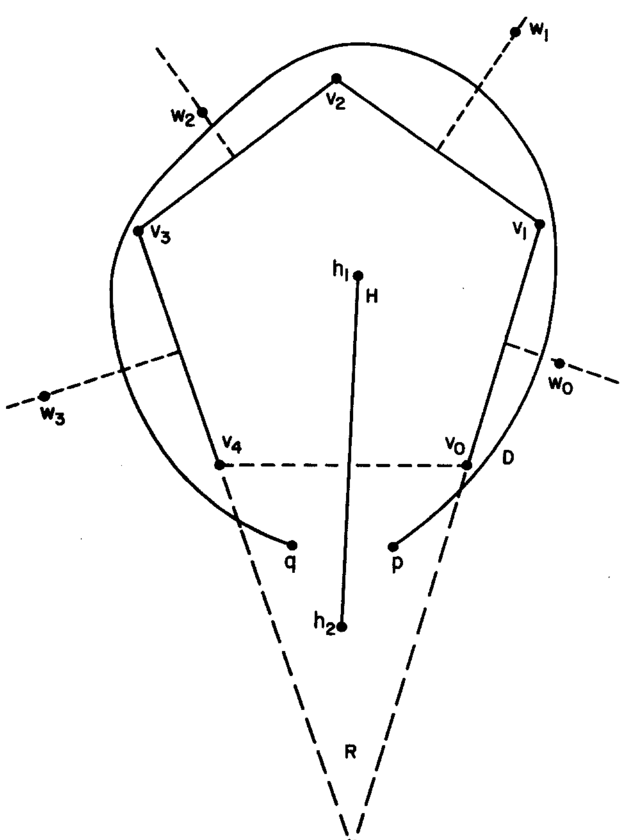

Fig. 6. Illustrating the proof of the lemma: any path inscribed in D that does not intersect either the polygonal path $v_0, \ldots, v_4$ or the line segment H must contain at least five segments.

To be more precise, put k vertices on the line segment $\overline{v_0 p}$ and k vertices on the line segment $\overline{v_{k-1}q}$. Join v_0 to h_1, and q to h_2, by line segments. Construct $2k+1$ curved edges that complete the boundary of G by joining v_{k-1} to p so that

(1) the boundary of G is simple;
(2) each vertex on $\overline{v_0 p}$, except for v_0, is adjacent to two vertices on $\overline{v_{k-1}q}$; and
(3) each vertex on $\overline{v_{k-1}q}$, except for q, is adjacent to two vertices on $\overline{v_0 p}$.

Splinegon G has $3k+4$ vertices. Let P_G be a simple polygon for G. By the above lemma, P_G has at least k vertices on any curved edge of G, of which $k-2$ are not original vertices of G. Since G has $2k+2$ curved edges, P_G has at least $(2k+2)(k-2)+3k+4=2k^2+k$ vertices.

This construction can obviously be modified to construct splinegons whose number of vertices when divided by three leaves a remainder of 0 or 2. Thus we have the following theorem.

THEOREM. *For any n, there exist splinegons whose simple carrier polygons have* $\Omega(n^2)$ *vertices.*

7. Conclusions and Open Problems. In this paper we have seen two kinds of splinegon problems. Decomposition into monotone or convex pieces presents challenging difficulties not encountered in the polygon case, whereas detection of intersections is a relatively straightforward extension of the result for polygons.

We have also shown a negative result on the potential complexity of finding a simple carrier polygon for a splinegon. This result demonstrates that a simple splinegon is a more powerful object than a simple polygon in its ability to represent shape information.

The techniques in this paper are likely to have wider application in turning polygon algorithms into splinegon algorithms.

References

[1] B. Chazelle and D. P. Dobkin, Optimal convex decompositions, in *Machine Intelligence and Pattern Recognition 2: Computational Geometry* (G. T. Toussaint, ed.), Elsevier Science, North Holland, Amsterdam, 1985, pp. 63-133.
[2] B. Chazelle and D. P. Dobkin, Intersection of convex objects in two and three dimensions, *J. Assoc. Comput. Mach.*, **34** (1987), 1-27.
[3] B. Chazelle and J. Incerpi, Triangulation and shape-complexity, *ACM Trans. Graphics*, **3** (1984), 135-152.
[4] J. Dugundji, *Topology*, Allyn and Bacon, Reading, MA, 1966.
[5] H.-Y. F. Feng and T. Pavlidis, Decomposition of polygons into simpler components: feature generation for syntactic pattern recognition, *IEEE Trans. Comput.*, **24** (1975), 636-650.
[6] A. Fournier and D. Y. Montuno, Triangulating simple polygons and equivalent problems, *ACM Trans. Graphics*, **3** (1984), 153-174.
[7] D. H. Greene, The decomposition of polygons into convex parts, in *Advances in Computing Research* (F. P. Preparata, ed.), Vol. 1, JAI Press, Greenwich, CT, 1983, pp. 235-259.
[8] J. Hershberger, private communication.

[9] J. G. Hocking and G. S. Young, *Topology*, Addison-Wesley, Reading, MA, 1961.
[10] K. Hoffman, K. Mehlhorn, P. Rosenstiehl, and R. E. Tarjan, Sorting Jordan sequences in linear time using level-linked search trees, *Inform. and Control*, **68** (1986), 170–184.
[11] J. M. Keil, Decomposing a polygon into simpler components, *SIAM J. Comput.*, **14** (1985), 799–817.
[12] J. M. Keil and J. R. Sack, Minimum decompositions of polygonal objects, in *Machine Intelligence and Pattern Recognition 2: Computational Geometry* (G. T. Toussaint, ed.), Elsevier Science, North Holland, Amsterdam, 1985, pp. 197–216.
[13] D. Kirkpatrick, Optimal search in planar subdivisions, *SIAM J. Comput.*, **12** (1983), 28–35.
[14] D. T. Lee and F. P. Preparata, Location of a point in a planar subdivision and its applications, *SIAM J. Comput.*, **6** (1977), 594–606.
[15] D. T. Lee and F. P. Preparata, Computational geometry—a survey, *IEEE Trans. Comput.*, **33** (1984), 1072–1101.
[16] S. Lefschetz, *Introduction to Topology*, Princeton University Press, Princeton, NJ, 1949.
[17] F. P. Preparata and M. I. Shamos, *Computational Geometry: An Introduction*, Springer-Verlag, New York, 1985.
[18] B. Schachter, Decompositions of polygons into convex sets, *IEEE Trans. Comput.*, **27** (1978), 1078–1082.
[19] A. A. Schäffer and C. J. Van Wyk, Convex hulls of piecewise-smooth Jordan curves, *J. Algorithms*, **8** (1987), 66–94.
[20] M. I. Shamos and D. Hoey, Geometric intersection problems, *Proceedings of the 17th Annual IEEE Symposium on Foundations of Computer Science*, Houston, TX, 1976, pp. 208–215.
[21] D. L. Souvaine, Computational geometry in a curved world, Ph.D. Dissertation, Princeton University, 1986.
[22] R. E. Tarjan and C. J. Van Wyk, An $O(n \log \log n)$-time algorithm for triangulating simple polygons, *SIAM J. Comput.* (to appear).
[23] C. J. Van Wyk, Clipping to the boundary of a circular-arc polygon, *Comput. Vision Graphics Image Process.*, **25** (1984), 383–392.