Optimizing a Corridor between Two Polygons with an Application to Polyhedral Interpolation*

Gill Barequet† Barbara Wolfers‡

Abstract

We consider the problem of finding a corridor (a separating strip) between two polygons, whose intersection with a third (convex) polygon is of maximum area. The application in mind is the interpolation in simple branching cases, where the sought volume branches from one contour in one slice into two polygons in another parallel slice. We present a linear-time plane-sweep algorithm which computes such a corridor. When the third polygon is not convex the running time of the algorithm is quadratic in the size of the input.

Keywords: surface reconstruction, plane-sweep, branching surfaces, slice interpolation, polyhedra.

1 Introduction

The problem of reconstructing the boundary of a solid object from a series of parallel planar cross-sections has attracted much attention in the literature during the past two decades. The main motivations for this problem come from medical imaging applications and from geographic information systems.

The input usually consists of a series of parallel planar slices, each consisting of a collection of simple non-crossing polygons (output of an edge-detection process applied to the raw data). The goal is a polyhedral solid model whose cross-sections along the given planes coincide with the input slices. A natural simplification of the problem is to consider only a single pair of successive parallel slices, and to construct a solid model within the layer delimited by the planes of the slices, which interpolates between the given slices. The concatenation of these models gives a solution model for the full problem.

There has been an extensive literature on this problem. Many of the earlier works studied the simple variant, where each slice contains only one contour. These studies either sought a global optimization of some objective function [FKU, Sh, SP, Ke, WA, WW], or used a local advancing rule, after the tiling starting points at the two contours were somehow determined [CS, GD, KSD, EPO].

Several works handled the more involved case, where one of the slices (or both) contains more than one contour. These works include [Sh, ZJH, EPO, MSS, CS, Sh, EPO, MSS, BG, CP, Bo, BG, BS]. In the full version we review these works in detail. The fairly simple 1-to-2 branching cases are usually solved by merging the two contours in one slice into one contour by adding "bridges" between them [CS, Sh, EPO, MSS]. As noted in several works (e.g., in [Sh, BS]), the bridges in one slice may be inconsistent with the geometry of the other slice, if they are not added with care.

In this work we deal with simple branching cases, in which one (convex) contour in one slice (denoted by R') is associated with two contours in the other slice (denoted by P and Q). We also denote by R the projection of R' onto the plane that contains P and Q. We preprocess R' by splitting it into two parts, which will be associated with P and Q, such that the combination of the two interpolations will be a feasible solution. That is, the two interpolated surfaces will be valid 2-manifolds, they will not intersect, and their combination will form a valid polyhedron bounded by the three input contours and the interpolated surfaces.

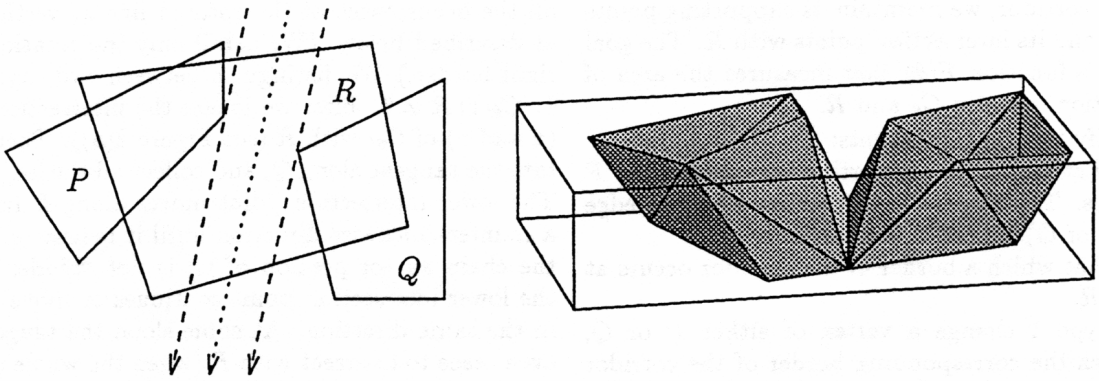
We observe that every split of R' along a line parallel to a separator of P and Q yields a feasible solution. Each set of all parallel separators form a "corridor" between P and Q. In order to find such corridor which also considers the geometry of R', we compute the corridor that maximizes the area of intersection with R. This corridor provides a direction of splitting (e.g., by using its median). The motivation for this choice is that this corridor is a compromise between considering only P and Q (e.g., by splitting R' along the direction of the widest corridor between P and Q), and considering only R (e.g., by splitting R' into two polygons of equal size).

Here is a brief overview of our algorithm. We first identify the range of slopes which allow feasible corridors between P and Q; we rotate a maximum-width corridor in this range. We collect all the critical events of the rotation—all the vertices of P, Q, and R, through which such corridor passes. Then we rotate the maximum-width corridor in a plane-sweep manner. At each event we update the status of the rotation, and compute the corridor within the last slope interval with maximum-area intersection with R. At the end of the rotation we obtain the optimum corridor. Although the

*Work on this paper by the first author has been supported by the Israeli Ministry of Science, Eshkol Grant 0562-1-94.

†School of Mathematical Sciences, Tel-Aviv University, Tel-Aviv 69978, Israel. E-mail: barequet@math.tau.ac.il

‡Institut für Informatik, Freie Universität Berlin, Takustraße 9, 14195 Berlin, Germany. E-mail: wolfers@inf.fu-berlin.de



(a) Input data (top view) (b) Interpolated surface (isometric view)

Figure 1: Splitting a contour in a branching case

algorithm is quite simple, its details and analysis are delicate.

Finally, we use the optimum corridor for determining the line along which we split the polygon R'. Two piecewise-linear surfaces are interpolated between the two parts of R' to P and Q. For this purpose we use any of the algorithms cited above. An example of such interpolation is illustrated in Figure 1. We remark that the solution to any interpolation problem is not uniquely defined, and the measure of 'goodness' of a proposed solution is rather subjective and intuitive.

The paper is organized as follows. In Section 2 we give a more precise definition of the problem. In Section 3 we discuss the concept of a corridor between two polygons with a maximum intersection-area with a third polygon. Section 4 presents an overview of the algorithm. Then, Section 5 describes the initialization of the rotational-sweep, and Section 6 gives the details of the sweep itself. In Section 7 we describe the application of the algorithm to the surface-interpolation problem in simple branching cases. Section 8 analyzes the complexity of the algorithm. In Section 9 we give an extension to our algorithm, and we end in Section 10 with some concluding remarks.

2 Statement of the Problem

We are given a pair of parallel planar slices: the first slice (Π_1) consists of one convex polygon (R'), while the second slice (Π_2) consists of two line-separable polygons (P and Q). Each polygon is given as a circular list of vertices, each specified by its (x, y) coordinates. The polygons are oriented in the counter-clockwise direction.

The problem we address is splitting R' into two parts R'_P and R'_Q, to be interpolated with the polygons P and Q. The approach suggested in this paper is splitting R' by a line ℓ parallel to a separator between P and Q. The direction of ℓ is chosen such that the intersection of a maximum-width strip between P and Q (in this direction) with R is of maximum area.

3 A Corridor with Max-Area Overlap

We assume w.l.o.g. that there is a vertical separator between P and Q, and that P is the left polygon. A *corridor* C between P and Q is an infinite strip bounded by two parallel lines ℓ_1 and ℓ_2, where ℓ_1 is tangent to P and ℓ_2 is tangent to Q, and both ℓ_1 and ℓ_2 separate between P and Q. In the sequel we also refer to ℓ_1 and ℓ_2 as the *borders* of C. The *slope* θ of C is the slope of ℓ_1 and ℓ_2; we thus denote the corridor by C_θ. The *median* of a corridor is the line parallel to and equidistant from its borders.

Assume that R' is split along a line ℓ parallel to a corridor and within it, and interpolate two surfaces between P and Q to the corresponding parts of R'. It is easily seen that if the two interpolated surfaces are valid, then their union is a feasible solution to the whole interpolation problem. Indeed, the two surfaces cannot intersect, since they are separated by any plane that contains ℓ (within Π_1) and a line parallel to ℓ which separates between P and Q (within Π_2). The two reconstructed solids share the segment which is the intersection of ℓ with R'. As explained in Section 1, we choose the corridor between P and Q with maximum overlap with R. In many practical interpolation cases (especially when R' is convex) this choice leads to an 'intuitive' branching.

4 Overview of the Algorithm

We consider the following 2D problem: Given two polygons P and Q separable by a line, and a convex polygon R, find a corridor C^* between P and Q such that the area of the intersection of C^* and R is maximal. Note that not only P and Q do not intersect but their convex hulls are also distinct.

Let the total number of vertices of P, Q and R be n. We use a plane-sweep algorithm in order to solve the problem in $O(n)$ time. While traditionally a line is swept in some direction, we rotate the widest possible corridor between the two extreme slopes θ' and θ''. Each slope $\theta' \leq \theta \leq \theta''$ corresponds to a corridor C_θ. While

rotating the corridor, we maintain its supporting points on P and Q and its intersection points with R. The goal is to obtain a function $\mathcal{F}(\theta)$ that measures the area of the intersection between C_θ and R.

We identify two types of events:
1. A slope θ at which the supporting vertex of either P or Q changes. This type of event occurs when an edge (of either P or Q) supports the corridor.
2. A slope θ at which a border of the corridor occurs at a vertex of R.

Events of type 1 change a vertex of either P or Q, around which the corresponding border of the corridor is rotated. Events of type 2 cause a change in the set of edges of R that are intersected by the corridor borders. We store events of type 1 in two lists (one for vertices of P and one for vertices of Q). We construct four more lists of vertices of R, at which events of type 2 occur (two for each border, from which one is for the upper envelope of R and one is for the lower envelope). All the lists are sorted in increasing order of the corresponding corridor slope. Finally, we merge the six lists into one event queue.

Since R is convex, the borders of every corridor between P and Q intersect with at most four edges of R. The algorithm is the following:

1. **Initialization.**
(a) Compute the event queue $\theta_0 = \theta' \leq \theta_1 \leq \theta_2 \leq \ldots \leq \theta_{k-1} \leq \theta_k = \theta''$;
(b) Put $\mathcal{F}(\theta_0) := 0$, $\text{area}_{\max} := 0$, and $\theta_{\max} :=$ indefinite;
2. **Sweeping.**
For $i = 1$ to k **do**
(a) Compute the function $\mathcal{F}(\theta)$ for $\theta_{i-1} \leq \theta \leq \theta_i$;
(b) Compute the slope θ_i^* for which $\mathcal{F}(\theta_i^*) = \max_{\theta_{i-1} \leq \theta \leq \theta_i} \mathcal{F}(\theta)$;
(c) **if** $\mathcal{F}(\theta_i^*) > \text{area}_{\max}$ **then** put $\text{area}_{\max} := \mathcal{F}(\theta_i^*)$ and $\theta_{\max} := \theta_i^*$;
od

5 Initializing the Sweep

In this step we compute the events of the rotational-sweep algorithm. First, we compute $CH(P)$ and $CH(Q)$ (since only vertices of P and Q which lie on their convex hulls may contribute events of type 1). Then we compute the extreme slopes θ' and θ'' and the corresponding supporting vertices of P and Q. We set the direction of rotation to be counter-clockwise. Thus we obtain the two lists of events of type 1; their first (resp. last) vertex supports the corridor $C_{\theta'}$ (resp. $C_{\theta''}$).

Now we construct the four event lists of type 2 contributed by vertices of R. In order to do that, we rotate the common tangent to P and Q with slope θ' counter-clockwise until it becomes the other common tangent to P and Q with slope θ''. We perform two rotations: one along Q and one along P. These two rotations represent the two borders of all the feasible corridors. We collect all the occurrences of the rotated line at vertices of R as described below. (We detail only the rotation of the right border.) We distinguish between two cases:
1. $C_{\theta'} \cap R \neq \emptyset$. First we locate the intersection points (u and v) of $C_{\theta'}$ with R (see Figure 2(a)). Then we rotate the tangent along Q, and collect all the hit vertices. The lower intersection point moves along R from v in a counter-clockwise direction until it reaches s. In case the chain su (or portion of it) is not "hidden" by Q, the lower intersection point continues to move along R in the same direction. At some slope the tangent may even cease to intersect with R (when the whole chain su is not "hidden") and resumes the intersection when we rotate the tangent further. Now the intersection point moves along R in a clockwise direction until it reaches s again. Similarly, the upper intersection point (which starts at u) may first move along R in a clockwise direction (within the chain us), then change its direction to counter-clockwise until it reaches u again. Then it continues to move in the same direction until it reaches t. Thus, all the vertices of R from v to s and from u to t are collected once (in their counter-clockwise order along R), and vertices between s to u may be collected as many as four times. The collected vertices are already sorted (in two lists) according to their slopes.
2. $C_{\theta'} \cap R = \emptyset$ (see Figures 2(b,c)). In this case we first locate the first right border of a feasible corridor that touches R. For this purpose we compute the tangent to R from above and to Q from above (if R is below $C_{\theta'}$) or below (otherwise). In case its slope belongs to the feasible range $[\theta', \theta'']$, we initialize the collection at this slope and resort to case 1 above. Otherwise the right border of any corridor does not intersect with R, thus does not contribute any event of type 2.

Now we merge all the six lists into one queue.

6 Sweeping

The key idea is to maintain a small (constant) amount of information which does not change between successive events, and allows simple updates at events and fast investigation of each interval of slopes defined by two successive events. Consider an interval of corridor slopes between two events θ_{i-1} and θ_i. While the corridor is rotated in this interval, there is no change either in the edges of R intersected by the corridor borders or in the vertices of P and Q that support the corridor. These six pieces of information define precisely the status of the sweep.

The rotational sweep starts at the degenerate corridor $C_{\theta'}$. The two supporting vertices $p \in P$ and $q \in Q$ form the first pair of points around which the corridor is rotated. In case $C_{\theta'}$ occurs at an edge of P or Q, there are two events with the same slope, causing a degenerate interval. We may omit this degeneracy by considering only the vertex with higher (counter-clockwise) slope as an event. We compute the function $\mathcal{F}(\theta)$ that mea-

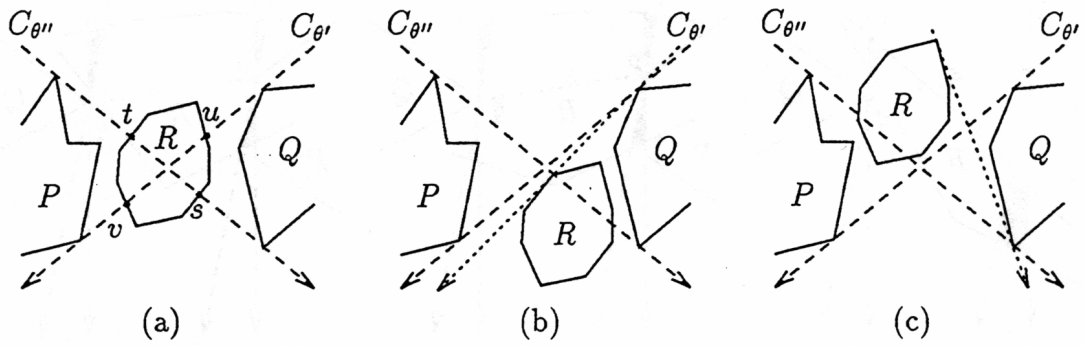


Figure 2: Locating events of type 2

sures the area of the intersection between R and C_θ. This function is composed of the functions $\mathcal{F}_i(\theta)$ (for $i = 1, \ldots, k$), where $\mathcal{F}_i$ is defined only at the ith slope interval $[\theta_{i-1}, \theta_i]$.

The rotation starts with a corridor of slope $\theta_0 = \theta'$. The width of C_{θ_0} is 0, hence $\mathcal{F}_1(\theta_0) = 0$. Assume that $\mathcal{F}(\theta_{i-1})$ has already been computed. We now compute the function $\mathcal{F}_i(\theta)$. Instead of computing it directly, we compute the difference between $\mathcal{F}_i(\theta)$ and $\mathcal{F}(\theta_{i-1})$. Finding the maximum difference in the ith interval will also imply the maximum of $\mathcal{F}$ in this interval.

Let s be the line segment defined by the current vertices $p \in P, q \in Q$ that support the corridor in the current interval. We identify four cases of the status of the rotated corridor, relative to the edges of R intersected by the corridor borders and to the line segment s:
1. All the intersections of edges of R with the borders of the corridor (ℓ_1 and ℓ_2) lie on the same side of s.
2. The line s separates between the intersections of edges of R with ℓ_1 and the intersections of edges of R with ℓ_2.
3. Each border of the corridor intersects R in two points which are separated by the line s.
4. The intersection points of one border with R are separated by s, while the intersection points of the other border with R lie on the same side of s.
Figure 3 shows the first case. Figures 4(a,b,c) illustrate the other three cases (2, 3, and 4, respectively).

Consider, for example, case 1. The difference between $\mathcal{F}_i(\theta)$ and $\mathcal{F}(\theta_{i-1})$ is given by the difference between the two shown quadrangles. (In case one border does not intersect with R, the corresponding quadrangle is empty.) Let e_j (for $j = 1, \ldots, 4$) be the edges of R intersected by the borders of $C_{\theta_{i-1}}$, and let β_j be the angle between e_j to the corresponding border. Let d_j be the distance between the jth intersection point and the corresponding supporting point (p or q) along the corresponding border of $C_{\theta_{i-1}}$. Simple calculation shows that $\mathcal{F}_i(\theta) = \mathcal{F}(\theta_{i-1}) + \frac{\sin(\theta-\theta_{i-1})}{2} \sum_{j=1}^{4} \frac{c_j d_j^2 \sin\beta_j}{\sin(\theta-\theta_{i-1}+\beta_j)}$, where $c_1, c_4 = 1$ and $c_2, c_3 = -1$. When ℓ_1 (resp. ℓ_2) does not intersect with R, we simply set d_1 and d_2 (resp. d_3 and d_4) to 0.

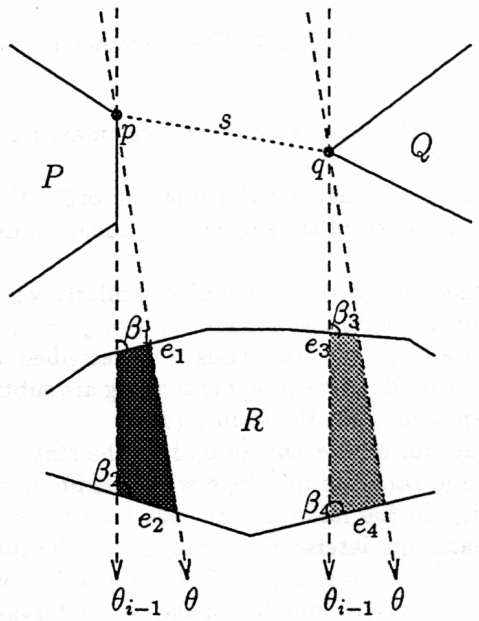


Figure 3: The difference between $C_{\theta_{i-1}} \cap R$ and $C_\theta \cap R$ (case 1)

Theorem 1 *The function $\mathcal{F}_i(\theta)$ has at most three local maxima within the interval $[\theta_{i-1}, \theta_i]$.*

Proof. We ignore the insignificant constant term $\mathcal{F}(\theta_{i-1})$ in the definition of $\mathcal{F}_i$, set $\alpha = \theta - \theta_{i-1}$, and investigate $\mathcal{F}_i$ as a function of α: $\mathcal{F}_i(\alpha) = \frac{1}{2}\sum_{j=1}^{4} \frac{c_j d_j^2 \sin\beta_j \sin\alpha}{\sin(\alpha+\beta_j)}$.

Consider the equation $\mathcal{F}_i'(\alpha) = \frac{1}{2}\sum_{j=1}^{4} \frac{c_j d_j^2 \sin^2\beta_j}{\sin^2(\alpha+\beta_j)} = 0$. We represent each denominator $\sin^2(\alpha+\beta_j)$ by the term $p_j\cos(2\alpha) + q_j\sin(2\alpha) + r_j$, where $p_j = -\frac{1}{2}\cos(2\beta_j)$, $q_j = \frac{1}{2}\sin(2\beta_j)$, and $r_j = \frac{1}{2}$. After multiplying out the denominators, this becomes a third-degree trigonometric equation in $\cos(2\alpha)$ and $\sin(2\alpha)$. Such an equation has at most 6 roots within a 2π-range of 2α. This can be verified by separating the odd powers of $\sin(2\alpha)$ from the even powers, substituting $\cos(2\alpha) = t$ and $\sin(2\alpha) = \sqrt{1-t^2}$, and squaring. Thus the equation has at most 6 roots within a π-range

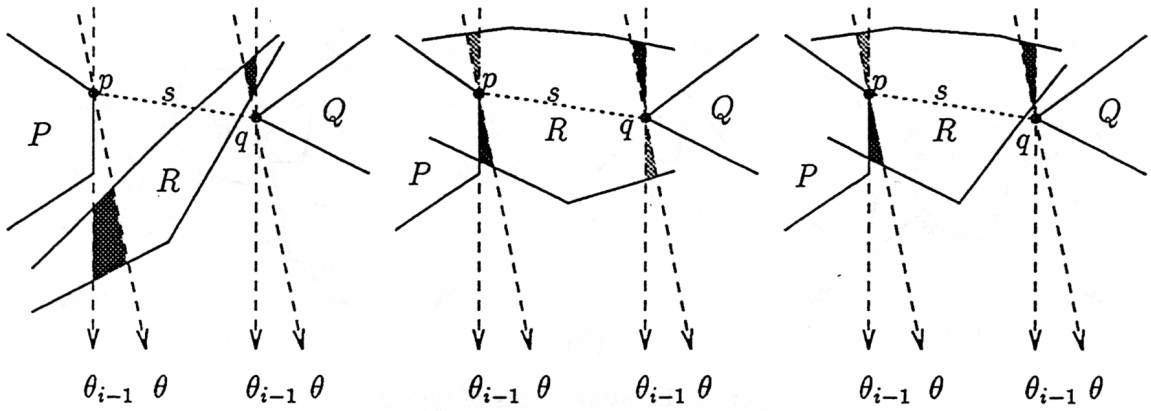


Figure 4: The difference between $C_{\theta_{i-1}} \cap R$ and $C_\theta \cap R$ (cases 2, 3, and 4)

of α. Hence there are at most 3 local maxima within this range. □

The global maximum (in this interval) of $\mathcal{F}_i(\theta)$ can be computed in constant time by using standard numerical methods.

The other three cases are handled similarly, where the only difference is in the signs of c_j (for $j = 1, \ldots, 4$). Figures 4(a,b,c) show the cases 2–4 described above. The areas with dark (resp. light) shading are subtracted from (resp. added to) the term $\mathcal{F}(\theta_{i-1})$.

To recap, for each event we update the status of the rotation and compute the slope with maximum area of intersection in the last interval. We keep track of the global maximum intersection area and of the intersection area of the current event. The latter is the constant term in the measure function of the next interval.

7 Application to Interpolation

The application is the interpolation between the slice that contains R' to the slice that contains P and Q. We use the algorithm described in Sections 5 and 6 for determining the line along which the polygon R' is split prior to the interpolation. First we find the corridor C_θ between P and Q with the maximum-area overlap with R. In case the two borders of C_θ intersect with R, we use the median of C_θ for splitting R'. In case a border of C_θ does not intersect with R, we narrow C_θ by translating the non-intersecting border (while keeping its slope θ unchanged) towards the nearest vertex $v \in R$, such that the border supports R at v, and use the median of the narrowed corridor for splitting R'.

Two situations make the maximum-area corridor-overlap problem trivial:

1. No corridor between P and Q intersects with R. This case is identified during the initialization of the algorithm. The maximum-area overlap in this case is 0. This situation happens when R is "hidden" by P or Q. That is, when R fully lies in the same wedge defined by $C_{\theta'}$ and $C_{\theta''}$ as P (or Q), but does not "penetrate" towards the apex of the wedge.
2. There is a range of corridors (possibly only one) that fully cover R. This case is identified during the rotation of the corridor. The maximum-area overlap in this case is the area of R. This situation happens when R is "between" P and Q.

In both cases the relations between P, Q, and R are less restrictive (for the interpolation problem). Either because the position of R is too "bad" (in case 1) or too "good" (in case 2). So the choice of an "intuitively good" split of R' in case 1 becomes less obvious then in the regular situations, whereas many solutions in case 2 seem to be pleasing enough.

Since the choice of a corridor between P and Q of maximum intersection-area with R becomes irrelevant (in case 1) or of less significance (in case 2), we prefer in these cases the corridor between P and Q of maximum width. First, we use the algorithm described in [Ed] for finding the closest pair of vertices $p \in P, q \in Q$ which lie on their convex hulls. The corridor C^+ of maximum width between P and Q is supported by p and q and orthogonal to $\overline{pq}$. Then the slope of C^+ is used for splitting R'. We may use any line with this slope which splits R' into two non-empty polygons. Reasonable splitters are the line with the longest portion that lies in the interior of R', the line that splits R' into two polygons of equal areas, and the median of C^+.

8 Complexity Analysis

We measure the complexity of the algorithm as a function of n, the total number of vertices of the polygons P, Q, and R.

First we compute the convex hulls of the polygons P and Q. This step requires $O(n)$ time (see, e.g., [Me]). Then we initialize the algorithm by computing six event lists. The two lists of events of type 1 require first locating the first and last events (vertices) of P and Q and then constructing the event lists. The two tasks can be performed in $O(\log n)$ and $O(n)$ time, respectively.

We also compute four lists of events of type 2. Finding the extreme corridors requires $O(\log n)$ time, while constructing the lists requires $O(n)$ time. Finally, we merge all the event lists into one queue in $O(n)$ time.

In case we identify a degeneracy of type 1, we locate the widest corridor in $O(\log n)$ time and compute a splitter. Each of the first two proposed splitters can be found in $O(\log n)$ time (by using a binary search), while the third can be found in $O(1)$ time.

In non-degenerate situations we invoke the main algorithm. Each event is processed in constant time. Since there are $O(n)$ events, the whole event queue is processed in $O(n)$ time. In case we detect a degeneracy of type 2, we switch to the widest-corridor paradigm, which requires $O(\log n)$ time.

To conclude, the whole algorithm runs in $O(n)$ time. The algorithm is sensitive to the number of events k. In the worst case $k = \Theta(n)$, since every vertex of P and Q may contribute one event, and every vertex of R may contribute at most 4 events. However, when $k = o(n)$ (e.g., when P and Q are very close) the running time, except reading the input and computing the two convex hulls, is $O(\log n + k)$. In this case the computation of the convex hulls becomes the bottleneck of the algorithm, and may be omitted when P and Q are convex, or when their convex hulls are given as input.

9 Extensions

We now discuss a relaxation of the requirement that R be convex. Assume then that R is not convex. The intersection of a corridor C_θ with R may be more complex than before, since each of its borders may now cross more than two edges of R. Instead of maintaining (at most) two intersection points for each border, we have to maintain a larger set of intersections. Denote the size of this set within the slope range $[\theta_{i-1}, \theta_i]$ by m_i. Then, the function $\mathcal{F}_i(\theta)$ that measures the intersection area within this range is $\mathcal{F}_i(\theta) = \mathcal{F}(\theta_{i-1}) + \frac{\sin(\theta-\theta_{i-1})}{2} \sum_{j=1}^{m_i} \frac{c_j d_j^2 \sin\beta_j}{\sin(\theta-\theta_{i-1}+\beta_j))}$. By using the same argument as that given for Theorem 1, it is easily shown that $\mathcal{F}_i(\theta)$ has at most $m_i - 1$ local maxima in a π-range of θ. Therefore, processing an event may take as much as $O(n)$ time, making the total running time of the algorithm $O(n^2)$. However, the new solution may become irrelevant to the interpolation problem because R' may be split into more than two parts. We omit the discussion of this situation in this version of the paper.

10 Conclusion

We have proposed an algorithm for finding a corridor between two polygons that maximizes intersection area with a third convex polygon. The algorithm runs in optimal time linear in the complexity of the input polygons. Relaxing the requirement that the third polygon be convex makes the algorithm run in quadratic time.

We applied this algorithm to the problem of surface interpolation in 1-to-2 branching cases. The single contour in one of the slices is split into two parts, which are interpolated separately with the two contours in the other slice. The main goal of our algorithm is to split the single contour with special care to the geometry of the two contours in the other slice, while most of the previous methods, that we are aware of, merge the contours of the other slice by adding "bridges", which might conflict the geometry of the single contour.

References

[Bo] J.D. Boissonnat, Shape reconstruction from planar cross sections, *CVGIP*, 44 (1988), 1–29.

[BG] J.D. Boissonnat and B. Geiger, Three dimensional reconstruction of complex shapes based on the Delaunay triangulation, in: *Proc. Biomed. Image Process. and Biomed. Vis.* (R.S. Acharya and D.B. Goldof, eds.), 1905, SPIE Press, Bellingham, WA, 1993, 964–975.

[BS] G. Barequet and M. Sharir, Piecewise-linear interpolation between polygonal slices, *Proc. 10th ACM Symp. on Computational Geometry*, 1994, 93–102; full version to appear in *Computer Vision and Image Understanding*, 63 (1996).

[CP] Y.-K. Choi and K.H. Park, A heuristic triangulation algorithm for multiple planar contours using an extended double branching procedure. *The Visual Comp.*, 10 (1994), 372–387.

[CS] H.N. Christiansen and T.W. Sederberg, Conversion of complex contour line definitions into polygonal element mosaics, *Computer Graphics*, 13 (1978), 187–192.

[Ed] H. Edelsbrunner, Computing the extreme distances between two convex polygons, *J. of Alg.*, 6 (1985), 213–224.

[EPO] A.B. Ekoule, F.C. Peyrin, and C.L. Odet, A triangulation algorithm from arbitrary shaped multiple planar contours, *ACM Trans. on Graphics*, 10 (1991), 182–199.

[FKU] H. Fuchs, Z.M. Kedem, and S.P. Uselton, Optimal surface reconstruction from planar contours, *Comm. of the ACM*, 20 (1977), 693–702.

[GD] S. Ganapathy and T.G. Dennehy, A new general triangulation method for planar contours, *ACM Trans. on Computer Graphics*, 16 (1982), 69–75.

[Ke] E. Keppel, Approximating complex surfaces by triangulation of contour lines, *IBM J. Res. and Dev.*, 19 (1975), 2–11.

[KSD] N. Kehtarnavaz, L.R. Simar, and R.J.P. De Figueiredo, A syntactic/semantic technique for surface reconstruction from cross-sectional contours, *CVGIP*, 42 (1988), 399–409.

[Me] A. Melkman, On-line construction of the convex hull of a simple polyline, *IPL*, 25 (1987), 11–12.

[MSS] D. Meyers, S. Skinner, and K. Sloan, Surfaces from contours, *ACM Trans. on Graphics*, 11 (1992), 228–258.

[Sh] M. Shantz, Surface definition for branching contour-defined objects, *Computer Graphics*, 15 (1981), 242–270.

[SP] K.R. Sloan and J. Painter, Pessimal guesses may be optimal: A counterintuitive search result, *IEEE TPAMI*, 10 (1988), 949–955.

[WA] Y.F. Wang and J.K. Aggarwal, Surface reconstruction and representation of 3D scenes, *Pat. Recog.*, 19 (1986), 197–207.

[WW] E. Welzl and B. Wolfers, Surface reconstruction between simple polygons via angle criteria, *J. of Symbolic Computation*, 17 (1994), 351–369.

[ZJH] M.J. Zyda, A.R. Jones, and P.G. Hogan, Surface construction from planar contours, *Computers and Graphics*, 11 (1987), 393–408.