Optimizing a Strip Separating Two Polygons

Gill Barequet[*]   Barbara Wolfers[†]

Abstract

We consider the problem of finding a strip separating between two polygons, whose intersection with a third (convex) polygon is of maximum area. We present an optimal linear-time algorithm for computing the optimum strip. When the third polygon is not convex, the running time of the algorithm is quadratic in the size of the input. The application in mind is the piecewise-linear surface interpolation in simple branching cases, where the sought volume branches from one contour in one slice into two contours in the other slice.

Key Words: plane sweep; branching surfaces; polyhedra; surface reconstruction.

1 Introduction

In this paper we consider the following problem in the plane:

Problem 1 *Given two line-separable polygons P, Q and a convex polygon R, find the strip S^* separating between P and Q, whose intersection with R has maximum area.*

Note that not only do P and Q not intersect but their convex hulls are also disjoint. The third polygon R may intersect P and/or Q.

We present an optimal linear-time algorithm for solving Problem 1. The algorithm uses a "rotating-calipers"-like procedure [23]: it rotates the separating strip (supporting P and Q), maintains the area of the intersection of the strip and R, and identifies the slope in which the area of the intersection assumes its maximum.

[*]Center for Geometric Computing, Dept. of Computer Science, Johns Hopkins University, Baltimore, MD 21218, USA. E-mail: `barequet@cs.jhu.edu`

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

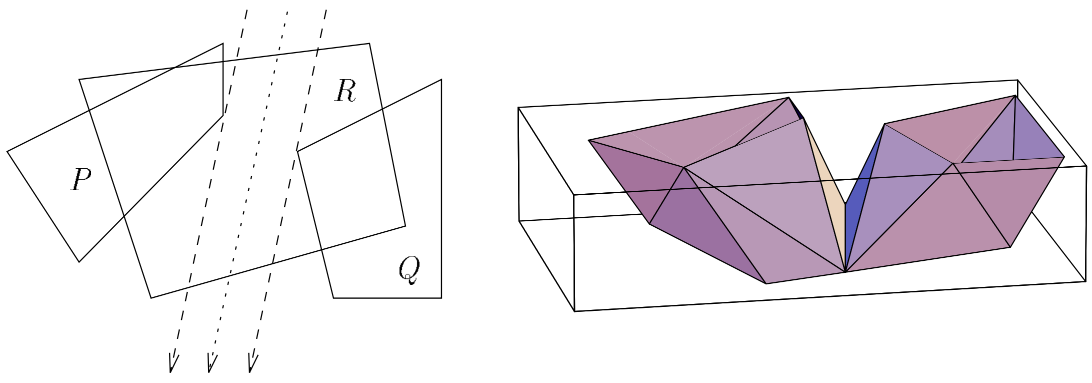
The rotating-calipers method can be viewed as a simple rotational version of the plane-sweep technique. This method was introduced by Toussaint for solving problems in computational geometry, e.g., computing the smallest-area rectangle enclosing a given set of points. The idea of the method is performing a rotational sweep of some shape, while keeping some invariants and maintaining some information. In this paper we present a nontrivial application of this method to Problem 1.

Here is a brief overview of the algorithm. We first identify the range of slopes which allow nonempty strips between P and Q; our aim is to rotate a maximum-width separating strip in this range. We collect all the critical events of the rotation—all the vertices of P, Q, and R through which such strip passes. Then we rotate the maximum-width strip in a plane-sweep manner. At each event we update the status of the rotation and investigate the range of strips between the last two events. In particular, we compute the strip (within the last slope interval) with maximum-area intersection with R. At the end of the rotation we obtain the optimum separating strip. There are several degenerate situations which we identify either in a preprocessing step or during the rotation.

The main motivation for solving this problem arises in a special 1-to-2 case of interpolating between parallel polygonal slices, in which the goal is to construct a solid object bounded by one contour (in one slice) and two contours (in the other slice). The general interpolation problem has an important application to medical imaging, where cross-sections of human organs are obtained by various technologies (CT, MRI, etc.), and the interpolated object can be displayed, analyzed, or even have a 3-dimensional model fabricated.

The input to the reconstruction problem usually consists of a pair of parallel planar slices, each consisting of a collection of simple noncrossing polygons. The goal is a polyhedral solid model whose cross-sections along the given planes coincide with the input slices. Only a few methods [1, 2, 4] handle the general case without limiting the number of contours, their geometries, or their containment hierarchies. Many of the earlier works studied the simplest 1-to-1 case, where each slice contains only one contour. These works either sought a global optimization of some objective function [10, 15, 20, 22, 24, 25] or used a local tiling-advancing rule [3, 7, 9, 11, 13, 14].

Several works handled the more involved 1-to-2 branching case. This case is usually solved by merging the two contours in one slice into one contour by adding a "bridge" between them and by applying a 1-to-1 algorithm [7, 9, 17, 20]. The major caveat in this approach is that the bridge in one slice may be inconsistent with the geometry in the other slice. Other works propose to *split* the single contour into two parts and to interpolate separately between each part and the corresponding contour in the other slice. Boissonnat and Geiger [5] used the *external Voronoi diagram* in one slice for splitting contours in the other slice in branching cases. Choi and Park [6] formed a "partitioning chain" for splitting a contour in one slice, based on a skeleton of the region between the two contours of the other slice. Such a partition of the single



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

Figure 1: Splitting a contour in a branching case

contour in one slice considers to some extent the geometry of the contours in the other slice. The interested reader is referred to the comprehensive reviews [18, 19, 21].

We propose a solution for Problem 1 as a tool for handling the 1-to-2 branching case, in which one convex contour $\overline{R}$ in one slice is interpolated with two contours P, Q in the other slice. We also denote by R the projection of $\overline{R}$ onto the plane that contains P and Q. The idea is to split $\overline{R}$ into two parts and to separately interpolate the two parts with P and Q, such that the combination of the two interpolations will be a feasible solution in the sense that 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.

Thus, we seek a pleasing split of $\overline{R}$. We observe that every split of $\overline{R}$ parallel to a line separating between P and Q (in their containing plane) yields a feasible solution. Each set of all parallel line-separators (with equal slope) defines a strip between P and Q. In order to find such a strip which also considers the geometry of $\overline{R}$, we compute the strip (between P and Q) whose intersection area with R is maximal. This choice is a heuristic compromise between considering only P and Q (e.g., by splitting $\overline{R}$ along the direction of the widest strip between P and Q) and considering only R (e.g., by splitting $\overline{R}$ into two polygons of equal area). Note that we do not necessarily compute the widest strip between P and Q. We propose to split $\overline{R}$ along a line parallel to the computed strip, e.g., the median of the strip, and to interpolate two surfaces between the two parts of $\overline{R}$ to P and Q (by using any 1-to-1 algorithm). An example of such interpolation is illustrated in Fig. 1. We have not fully implemented the algorithm, but only coded the intersection-area function (between two successive events) for experimenting with the number of local extrema it has.

The paper is organized as follows. In Section 2 we give a more precise definition of the problem and some notation. Section 3 presents an overview of the algorithm (more detailed than the one given above). The following two sections describe in

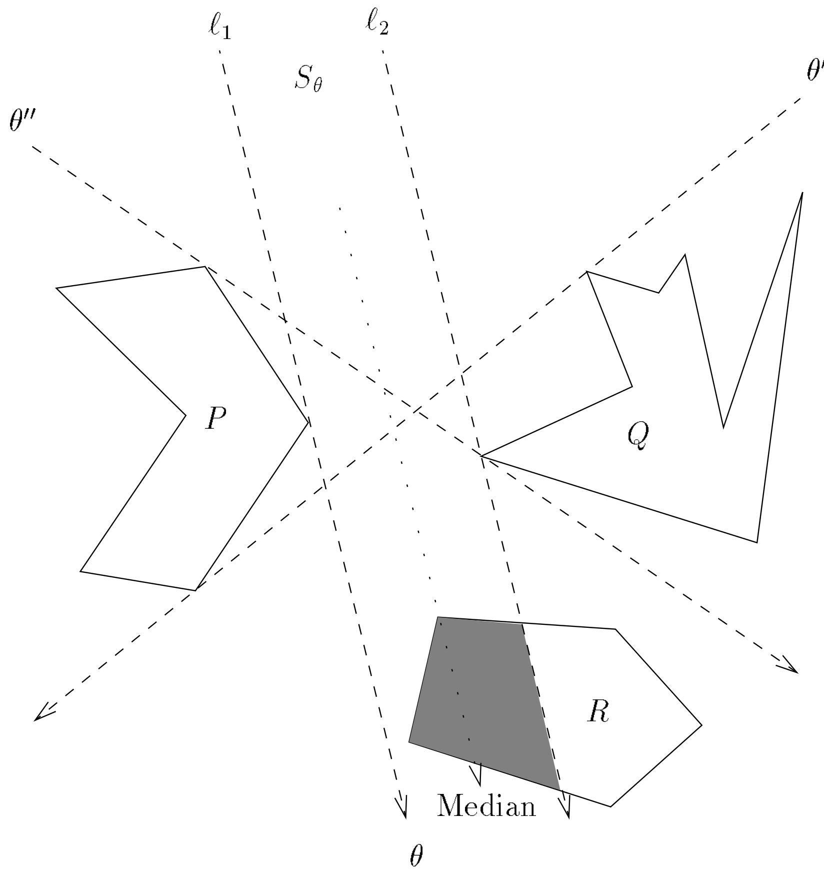


Figure 2: A strip separating between two polygons

detail the main phases of the algorithm: Section 4 describes the initialization of the rotational sweep, and Section 5 gives the details of the sweep itself. Section 6 analyzes the complexity of the algorithm. In Section 7 we describe the application of the algorithm to the surface interpolation problem in simple branching cases. In Section 8 we give a few extensions to our algorithm, and we end in Section 9 with some concluding remarks.

2 Statement of the Problem and Notation

We are given two line-separable polygons P and Q and a third convex polygon R in the plane. (There is no restriction on the intersections of R with P and/or Q.) Each polygon is given as a circular list of vertices, each specified by its (x, y) coordinates. The polygons are oriented in the counterclockwise direction. We assume without loss of generality that P and Q can be separated by a vertical line and that P is the left polygon.

A strip S is said to be separating between the polygons P and Q when its two borders ℓ_1 and ℓ_2 support the polygons; say, 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 (see Fig. 2). We seek the strip separating between P and Q whose intersection with R is of maximum area. We assume in the following that there exists a strip separating between P and Q with a nonempty intersection with R and that there does not exist any separating strip that fully covers R. We consider the degenerate cases (that do not fulfill our assumptions) separately (see Section 7).

The *slope* θ of S is the slope of ℓ_1 and ℓ_2; we thus denote the strip by S_θ. For a pair of line-separable polygons P and Q, the slope θ of any strip separating between the polygons is found within some range $[\theta', \theta'']$, where θ' and θ'' are the slopes of the two extreme separators tangent to both P and Q. The *median* of a strip is the line parallel to and equidistant from its two borders.

3 Overview of the Algorithm

We use a rotational sweep algorithm in order to solve Problem 1 in optimal time, linear in the size of the input. While traditionally a line is swept in some direction (e.g., a vertical line is swept along the x-axis), we rotate the widest possible strip between the two extreme slopes θ' and θ''. Each slope $\theta' \leq \theta \leq \theta''$ corresponds to a strip S_θ. While rotating the strip, we maintain its supporting points on the polygons P and Q and its intersection points with the polygon R. The goal is to obtain a function $\mathcal{F}(\theta)$ that measures the area of the intersection between S_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 strip.
2. A slope θ at which a border of the strip 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 strip is rotated. Events of type 2 cause a change in the set of edges of R that are intersected by the strip borders. Specifically, one such edge of R is replaced by another one, or two edges of R are added to or removed from this set.

We store events of type 1 in two lists. The first list contains the vertices of P, sorted by the corresponding strip slope in increasing order, at which events of type 1 occur. The second list is constructed in the same manner for Q.

In addition, we construct four more lists of vertices of R, at which events of type 2 occur. While rotating the strip, we collect events of this type separately for each of its two borders. For each border, we collect its occurrences at vertices of the upper and the lower envelope of R separately. In total we have four lists of events of type 2 which are sorted in increasing order of the corresponding strip slopes.

After constructing the six lists of events we merge them into one list.

Since R is convex, the two borders of every strip separating between P and Q intersect at most four edges of R (each border either intersects with two edges of R or does not intersect with R at all).

The algorithm is the following:

```
1. Initialization:
   (a) Compute the six sorted lists of events and merge them into one 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 intersection-area function $\mathcal{F}(\theta)$ for $\theta_{i-1} \leq \theta \leq \theta_i$;
      (b) Compute the slope $\theta_i^*$ 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
```

In the next two sections we describe how to perform the initialization and the sweeping stages of the algorithm in linear time.

4 Initializing the Sweep

In this step we compute the events of the rotational sweep algorithm.

First, we compute CH(P) and CH(Q), the convex hulls of the polygons P and Q, respectively. Note that 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 counterclockwise. Thus we obtain the two lists of events of type 1. These are ordered lists of vertices of P and Q (taken in the counterclockwise direction); their first (resp., last) vertex supports the strip $S_{\theta'}$ (resp., $S_{\theta''}$).

Now we construct the four event lists of type 2, which are contributed by vertices of R. In order to do that, we rotate the common tangent to P and Q with slope θ' counterclockwise 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. (That is, we once

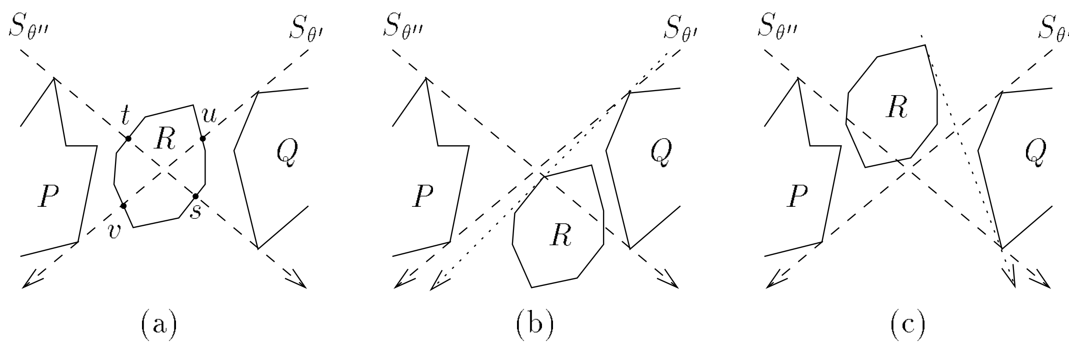


Figure 3: Locating events of type 2

keep the line tangent to Q and once to P.) These two rotations represent the two borders of all the feasible strips. We collect all the occurrences of the rotated line at vertices of R as described below. (We detail only the first rotation, during which we collect the events of type 2 that correspond to the right border; the second rotation is analogous.) We distinguish between two cases:

1. $S_{\theta'} \cap R \neq \emptyset$. First we locate the intersection points (u and v) of $S_{\theta'}$ with R (see Fig. 3a). Then we rotate the tangent along Q and collect all the vertices that the tangent hits. The lower intersection point moves along R from v in a counterclockwise 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" by Q) 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 counterclockwise 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 counterclockwise order along R), and vertices between s to u may be collected at most four times. The collected vertices are already sorted (in two lists) according to their slopes.

2. $S_{\theta'} \cap R = \emptyset$ (see Figs. 3b and 3c). In this case we first locate the first right border of a feasible strip that touches R. For this purpose we compute the tangent to R (from above) and to Q (from above if R is below $S_{\theta'}$ or below if R is above $S_{\theta'}$). (The existence of such a tangent is guaranteed by the assumption that a feasible strip exists—see Section 2). 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 strip (separating P and Q) does not intersect with R. In this case the right border does not contribute any events of type 2.

Now we merge all six event lists into one queue sorted according to event slope. Thus, fetching the next event during the rotation is trivial.

5 Sweeping

The key idea is to maintain a small amount of information which does not change between successive events, to perform simple updates at events (each in constant time), and to investigate each interval of slopes defined by two successive events (also in constant time).

Consider an interval of strip slopes between two successive events θ_{i-1} and θ_i. While the strip is rotated in this interval, there is no change either in the edges of R intersected by the strip borders or in the vertices of P and Q that support the strip. These six variables define precisely the status of the sweep.

The rotational sweep starts at the degenerate strip $S_{\theta'}$. The first two supporting vertices $p \in P$ and $q \in Q$ form the first pair of points around which the strip is rotated. In case $S_{\theta'}$ occurs at an edge of P or Q, there are two events with the same slope, causing a degenerate interval. We may easily omit this degeneracy by considering only the vertex with higher (counterclockwise) position as an event.

Now we compute the function $\mathcal{F}(\theta)$ that measures the area of the intersection between R and S_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 $[\theta_{i-1}, \theta_i]$. Actually, $\mathcal{F}_i(\theta)$ is based on the *change* of area between the strip $S_{\theta_{i-1}}$ and S_θ, $\theta \in [\theta_{i-1}, \theta_i]$.

The rotation starts with a strip of slope $\theta_0 = \theta'$. The width of S_{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 whose endpoints are the vertices $p \in P$ and $q \in Q$ that support the strip in the current interval. We identify four cases of the status of the rotated strip, depending on the relation between the edges of R intersected by the strip borders and the segment s:

1. All the (at most four) intersections of edges of R with the borders of the strip (ℓ_1 and ℓ_2) lie on the same side of s (see Fig. 4).

2. The line containing s separates between the intersections of edges of R with ℓ_1 and the intersections of edges of R with ℓ_2 (see Fig. 5a).

3. Each border of the strip intersects R in two points which are separated by the line containing s (see Fig. 5b).

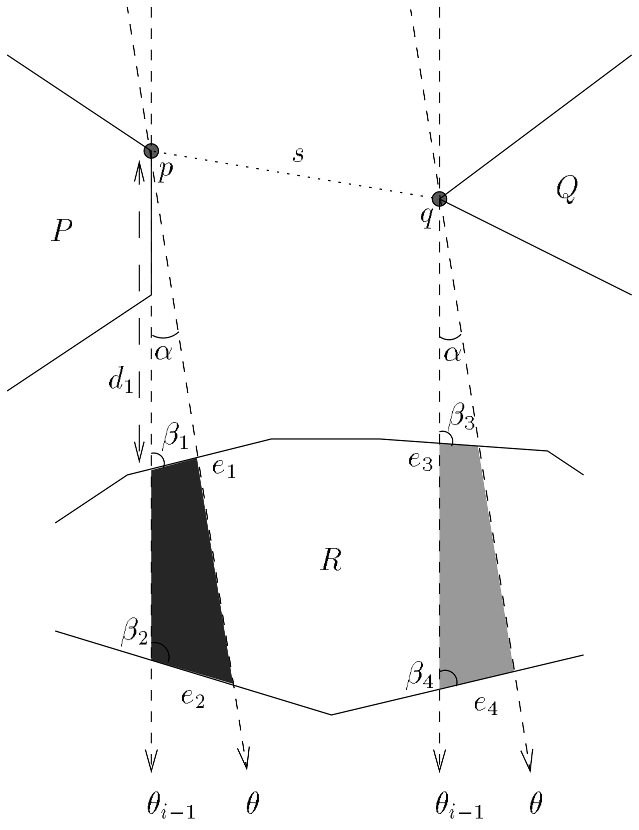


Figure 4: The difference between $S_{\theta_{i-1}} \cap R$ and $S_\theta \cap R$ (case 1)

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 (see Fig. 5c).

Consider, for example, case 1. All the intersections between the borders of S_θ and edges of R lie on the same side of s. In this case the difference between $\mathcal{F}_i(\theta)$ and $\mathcal{F}(\theta_{i-1})$ is given by the difference between two quadrangles. (In case one border does not intersect with R, the corresponding quadrangle is empty.) Since the rotation is performed in the counterclockwise direction, $\mathcal{F}_i(\theta)$ equals $\mathcal{F}(\theta_{i-1})$ plus the area of the right quadrangle minus the area of the left quadrangle.[1] Let e_j (for $j = 1, \ldots, 4$) be the edges of R intersected by the borders of $S_{\theta_{i-1}}$, and let β_j be the angle between e_j to the corresponding strip border. Let d_j be the distance from the jth intersection point (between $S_{\theta_{i-1}}$ and R) to the corresponding supporting point ($p \in P$ or $q \in Q$) along the corresponding border of $S_{\theta_{i-1}}$. Simple calculation shows that

$$\mathcal{F}_i(\theta) = \mathcal{F}(\theta_{i-1}) + \frac{\sin(\theta - \theta_{i-1})}{2} \sum_{j=1}^{4} \frac{c_j d_j^2 \sin \beta_j}{\sin(\theta - \theta_{i-1} + \beta_j)} \ ,$$

[1]Vice versa when R fully lies above s.

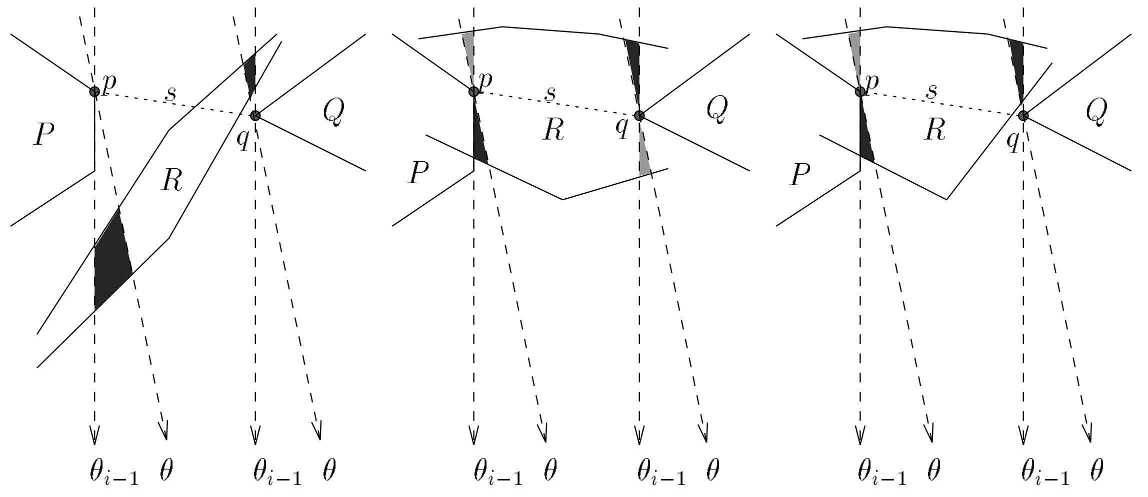


Figure 5: The difference between $S_{\theta_{i-1}} \cap R$ and $S_\theta \cap R$ (cases 2, 3, and 4)

where $c_1 = 1$, $c_2 = -1$, $c_3 = -1$, and $c_4 = 1$. When ℓ_1 (resp., ℓ_2) does not intersect with R (or when it is tangent to R at an edge), we simply omit the first and second (resp., third and fourth) terms.

The function $\mathcal{F}_i(\theta)$ is the sum of the constant $\mathcal{F}(\theta_{i-1})$ and of two monotone functions: one is decreasing (the negative area of the left quadrangle), and the other is increasing (the area of the right rectangle). This function has a constant number of local maxima in the interval $[\theta_{i-1}, \theta_i]$, as we show next.

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

Proof. We ignore the constant term $\mathcal{F}(\theta_{i-1})$ in the definition of $\mathcal{F}_i$. We also set $\alpha = \theta - \theta_{i-1}$ (the rotation of the strip relative to its initial slope at the current interval), and investigate $\mathcal{F}_i$ as a function of α instead of θ;

$$\mathcal{F}_i(\alpha) = \frac{1}{2}\sum_{j=1}^{4} \frac{c_j d_j^2 \sin\beta_j \sin\alpha}{\sin(\alpha+\beta_j)} \ ,$$

where $c_1 = 1$, $c_2 = -1$, $c_3 = -1$, and $c_4 = 1$.

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 first represent each denominator $\sin^2(\alpha+\beta_j)$ by the term $p_j\cos(2\alpha) + q_j\sin(2\alpha) + r_j$, with the appropriate coefficients $p_j = -\frac{1}{2}\cos(2\beta_j)$, $q_j = \frac{1}{2}\sin(2\beta_j)$, 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 six roots within a 2π-range of 2α. This can be verified by separating the odd powers of $\sin(2\alpha)$ from the even powers, by substituting $\cos(2\alpha) = t$ and $\sin(2\alpha) = \sqrt{1-t^2}$, and by squaring. Hence the equation has at most six roots within a π-range of α. This implies that there are at most three local maxima within this range, establishing the claim. $\square$

Hence, 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 in a similar manner, where the only difference is in the signs of c_j (for $j = 1, \ldots, 4$). Figs. 5a–5c 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 (the points $p \in P$ and $q \in Q$ that support the strip and edges of R that are intersected by the strip borders) 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 definition of the area function of the next slope interval.

6 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., [16]).

Then we initialize the algorithm by computing and merging six event lists. The two event lists contributed by P and Q (events of type 1) require first locating the first and last events (vertices) of P and Q (endpoints of subchains of the polygons) 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 event lists contributed by R (events of type 2). Finding the extreme (slope-wise) strips requires $O(\log n)$ time, while constructing the lists requires $O(n)$ time. Finally, we merge all the event lists into one queue (sorted according to slopes) in $O(n)$ time.

If there does not exist any feasible strip we halt the algorithm. In nondegenerate situations we invoke the main algorithm. Fetching an event is performed in constant time. As explained in Section 5, each event is handled also in $O(1)$ time. Since there are $O(n)$ events, the whole event queue is processed in $O(n)$ time. In case we detect during the course of the algorithm a strip that fully contains R, we compute (in $O(n)$ time) the area of R and halt the algorithm. Otherwise we run the entire algorithm until all events are processed.

To conclude, the whole algorithm runs in optimal $\Theta(n)$ 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 as many as four events. However, when $k = o(n)$ (e.g., when P and Q are very close, so that the feasible range of slopes is very small), the running time, except reading the input data and computing the two convex hulls in the preprocessing, 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 known to be convex or when their convex hulls are given as input to the algorithm.

7 Application to Interpolation

The application in mind is the interpolation of a solid object between two parallel slices, one of which contains one convex polygon $\overline{R}$ and the other which contains two polygons P and Q. We denote the planes containing the two slices by Π_1 and Π_2, which are assumed, with no loss of generality, to be parallel to the xy-plane. We suggest solving the interpolation problem by splitting $\overline{R}$ into two parts, $\overline{R}_P$ and $\overline{R}_Q$, to be associated with the polygons P and Q, and to interpolate a surface between each pair of associated polygons. The approach we suggest is to split $\overline{R}$ along some line ℓ (in Π_1) which is parallel to a separator (in Π_2) between P and Q. The direction of ℓ is chosen such that a maximum-width strip between P and Q in that direction "chops" as much as possible from R, the projection of $\overline{R}$ onto Π_2. It is easily seen that if the two interpolated surfaces are valid, then their union is a feasible solution to the entire interpolation problem. Indeed, the two surfaces cannot intersect, since they are separated by any plane that contains ℓ and a line (within Π_2) parallel to ℓ, which separates between P and Q. The two reconstructed solids share the segment which is the intersection of ℓ with $\overline{R}$. In many practical branching cases such a choice of a splitter of $\overline{R}$ leads to an "intuitively good" reconstruction.

Note that we specifically assume that the assignment of the contours is given, that is, the sought solid is assumed to branch from one contour ($\overline{R}$) into two contours (P and Q). Hence, we do not consider the option of connecting $\overline{R}$ to only one polygon (P or Q) even if their geometries suggest that.

We thus compute the strip S_θ separating between P and Q whose intersection with R is of maximum area. In case the two borders of S_θ intersect with R, we use the median of S_θ for splitting $\overline{R}$. In case one border of S_θ does not intersect with R, we shrink S_θ by translating that border (while keeping its slope θ unchanged) toward the nearest vertex $v \in R$, such that the new border supports R at v, and use the median of the shrunk strip for splitting $\overline{R}$ (see Fig. 6).

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

1. No strip between P and Q intersects with R. This case is identified during the initialization step of the algorithm when we compute the events of type 2.

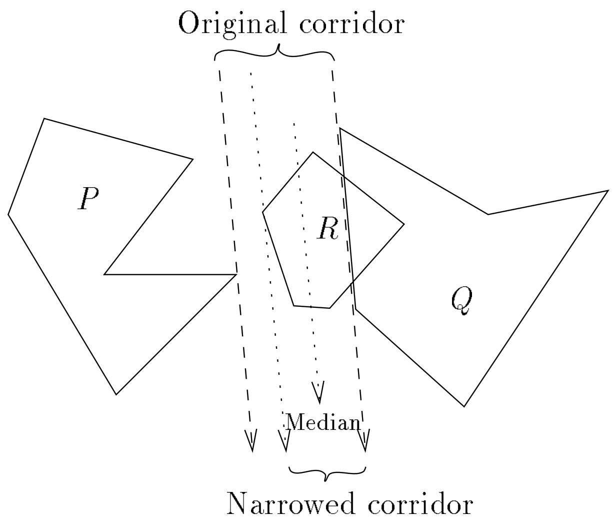


Figure 6: A shrunk strip

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 $S_{\theta'}$ and $S_{\theta''}$ as P (or Q), but does not "penetrate" toward the apex of the wedge.

2. There is a range of strips (possibly only one strip) that fully cover R. This case is identified during the rotation of the strip. 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" (case 1) or too "good" (case 2). The choice of an "intuitively good" split of $\overline{R}$ in case 1 therefore becomes less obvious than in the regular situations, whereas there seem to exist many pleasing solutions in case 2. In these cases we suggest to use the strip between P and Q of maximum width. First, one can use the algorithm described in [8] to find the closest pair of vertices $p \in P$ and $q \in Q$ which lie on their convex hulls. The strip S^+ of maximum width between P and Q is the strip supported by p and q and orthogonal to $\overline{pq}$. Then we use the slope of S^+ as the direction along which $\overline{R}$ is split. One may use any line with this slope which splits $\overline{R}$ into two nonempty polygons; the following are a few reasonable splitters:

- The line with the longest portion that lies in the interior of $\overline{R}$.
- The line that splits $\overline{R}$ into two polygons of equal areas.
- The median of S^+. (In case 2 above, we first shrink S^+ until it supports R from both sides).

Each of the first two splitters can be found in $O(\log n)$ time (by using a binary-search procedure), while the third splitter can be found in $O(1)$ time.

8 Extensions

So far we have assumed that the polygon $\overline{R}$ (and hence also R) is convex. We now discuss a modification of our method for relaxing this requirement.

Assume then that R is not convex. The intersection of a strip S_θ with R may now have a more complex shape than before, since each strip border 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 (e.g., in a balanced tree). 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

$$\mathcal{F}_i(\theta) = \mathcal{F}(\theta_{i-1}) + \frac{\sin(\theta - \theta_{i-1})}{2} \sum_{j=1}^{m_i} \frac{c_j d_j^2 \sin \beta_j}{\sin(\theta - \theta_{i-1} + \beta_j)} \ .$$

Maximizing this function is obviously harder than maximizing the previous one. By using the same argument as that given in the proof of 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 $\overline{R}$ may be split into more than two parts. Therefore we may invoke the algorithm on $\mathrm{CH}(R)$ and check whether $\overline{R}$ is split into exactly two parts. In such a case we split $\overline{R}$ and interpolate as before. In case $\overline{R}$ is split into more than two parts we attempt to translate the splitting line (within the strip) such that $\overline{R}$ is split into only two parts. Checking whether the solution for $\mathrm{CH}(R)$ actually splits $\overline{R}$ into two parts can be performed by using a Jordan-sorting of the edges of R, which requires $O(n)$ time. Investigating all the translations of the split can be performed by a simple plane-sweep procedure, which requires $O(n \log n)$ time. Any translation of the original splitting line, which is contained by the computed strip between P and Q and which splits $\overline{R}$ into two parts, yields a valid solution. A more ambitious goal may be to find a splitter of $\overline{R}$ into exactly two parts (if it exists), which is parallel to a feasible strip between P and Q and that optimizes some objective function. In some nonconvex situations (see, e.g., Fig. 7) the algorithm for the abstract geometric problem can produce counterintuitive (albeit valid) solutions for the interpolation problem. In such cases we may use $\mathrm{CH}(R)$ instead of R.

Finally, we note that in none of the cases discussed in this work (except when interpolating between convex polygons), the existence of a non-self-intersecting interpolation is guaranteed [12].

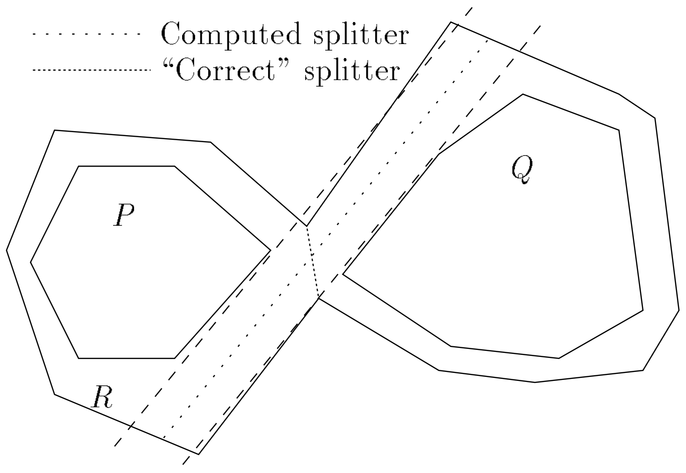


Figure 7: An example with a counter-intuitive solution

9 Conclusion

We have proposed in this paper an algorithm for finding a strip between two polygons that maximizes the area of intersection with a third convex polygon. The algorithm is optimal and runs in 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 have also presented the application of 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 application 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 a "bridge," which might conflict with the geometry of the single contour.

Acknowledgment

We thank Emo Welzl and Micha Sharir for supporting the visit of the first author to the Freie Universität Berlin, during which this work was carried out.

References

[1] C.L. Bajaj, E.J. Coyle, and K.-N. Lin, Arbitrary topology shape reconstruction from planar cross sections, *Graphical Models and Image Processing*,

58, 1996, 524–543.

[2] G. Barequet and M. Sharir, Piecewise-linear interpolation between polygonal slices, *Computer Vision and Image Understanding.* **63**, 1996, 251–272.

[3] S. Batnitzky, H.I. Price, P.N. Cook, L.T. Cook, and S.J. Dwyer III, Three-dimensional computer reconstruction from surface contours for head CT examinations, *J. of Computer Assisted Tomography.* **5**, 1981, 60–67.

[4] J.D. Boissonnat, Shape reconstruction from planar cross sections, *Computer Vision, Graphics and Image Processing.* **44**, 1988, 1–29.

[5] J.D. Boissonnat and B. Geiger, Three dimensional reconstruction of complex shapes based on the Delaunay triangulation, in *Proceedings, Biomedical Image Processing and Biomedical Visualization* (R.S. Acharya and D.B. Goldof, Eds.), pp. 964–975, 1905, SPIE Press, Bellingham, WA, 1993.

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

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

[8] H. Edelsbrunner, Computing the extreme distances between two convex polygons, *J. of Algorithms.* **6**, 1985, 213–224.

[9] 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.

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

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

[12] C. Gitlin, J. O'Rourke, and V. Subramanian, On reconstructing polyhedra from parallel slices, *Int. J. of Computational Geometry and Applications.* **6**, 1996, 103–122.

[13] N. Kehtarnavaz and R.J.P. De Figueiredo, A framework for surface reconstruction from 3D contours, *Computer Vision, Graphics and Image Processing.* **42**, 1988, 32–47.

[14] N. Kehtarnavaz, L.R. Simar, and R.J.P. De Figueiredo, A syntactic/semantic technique for surface reconstruction from cross-sectional contours, *Computer Vision, Graphics and Image Processing.* **42**, 1988, 399–409.

[15] E. Keppel, Approximating complex surfaces by triangulation of contour lines, *IBM J. of Research and Development.* **19**, 1975, 2–11.

[16] A. Melkman, On-line construction of the convex hull of a simple polyline, *Information Processing Letters.* **25**, 1987, 11–12.

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

[18] H. Müller and A. Klingert, Surface interpolation from cross sections, in *Focus on Scientific Visualization* (H. Hagen, H. Müller, and G.M. Nielson, Eds.), pp. 139–189, Springer-Verlag, Berlin, 1993.

[19] L.L. Schumaker, Reconstructing 3D objects from cross-sections, in *Computation of Curves and Surfaces* (W. Dahmen, M. Gasca, and C.A. Micchelli, Eds.), pp. 275–309, Kluwer Academic, Dodrecht/Norwell, MA, 1989.

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

[21] K.R. Sloan and J. Painter, From contours to surfaces: Testbed and initial results, in *Proceedings, CHI+GI, 1987*, pp. 115–120.

[22] K.R. Sloan and J. Painter, Pessimal guesses may be optimal: A counterintuitive search result, *IEEE Trans. on Pattern Analysis and Machine Intelligence.* **10**, 1988, 949–955.

[23] G.T. Toussaint, Solving geometric problems with the rotating calipers, in *Proceedings, IEEE MELECON, 1983*, pp. 1–4.

[24] Y.F. Wang and J.K. Aggarwal, Surface reconstruction and representation of 3D scenes, *Pattern Recognition.* **19**, 1986, 197–207.

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