Finding the Maximum Subset with Bounded Convex Curvature

Mikkel Abrahamsen[*1] **and Mikkel Thorup**[†2]

1 **Department of Computer Science, University of Copenhagen, Copenhagen, Denmark**
miab@di.ku.dk

2 **Department of Computer Science, University of Copenhagen, Copenhagen, Denmark**
mikkel2thorup@gmail.com

Abstract

We describe an algorithm for solving an important geometric problem arising in computer-aided manufacturing. When machining a pocket in a solid piece of material such as steel using a rough tool in a milling machine, sharp convex corners of the pocket cannot be done properly, but have to be left for finer tools that are more expensive to use. We want to determine a tool path that maximizes the use of the rough tool. Mathematically, this boils down to the following problem. Given a simply-connected set of points P in the plane such that the boundary ∂P is a curvilinear polygon consisting of n line segments and circular arcs of arbitrary radii, compute the maximum subset $Q \subseteq P$ consisting of simply-connected sets where the boundary of each set is a curve with bounded convex curvature. A closed curve has bounded convex curvature if, when traversed in counterclockwise direction, it turns to the left with curvature at most 1. There is no bound on the curvature where it turns to the right. The difference in the requirement to left- and right-curvature is a natural consequence of different conditions when machining convex and concave areas of the pocket. We devise an algorithm to compute the unique maximum such set Q. The algorithm runs in $O(n \log n)$ time and uses $O(n)$ space.

For the correctness of our algorithm, we prove a new generalization of the Pestov-Ionin Theorem. This is needed to show that the output Q of our algorithm is indeed maximum in the sense that if Q' is any subset of P with a boundary of bounded convex curvature, then $Q' \subseteq Q$.

1998 ACM Subject Classification I.3.5 Computational Geometry and Object Modeling

Keywords and phrases plane geometry, bounded curvature, pocket machining

Digital Object Identifier 10.4230/LIPIcs.SoCG.2016.4

1 Introduction

The motivation for our work comes from the generation of toolpaths for pocket machining. Pocket machining is the process of cutting some specified pocket in a piece of material – in our case most likely a piece of metal – using a milling machine. We first describe the clean mathematical problem that we solve and afterwards explain how it relates to pocket machining.

* Research partly supported by Mikkel Thorup's Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme.
† Research partly supported by an Advanced Grant from the Danish Council for Independent Research under the Sapere Aude research career programme.

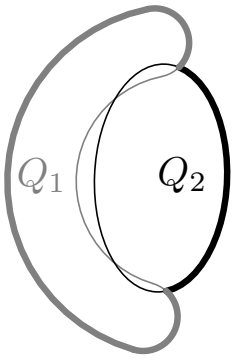

Figure 1

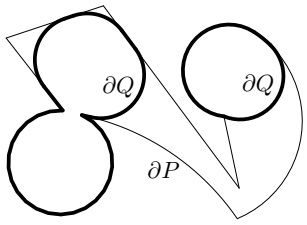

Figure 2

Consider a simply-connected closed subset of the plane and the weakly simple, closed curve around its boundary which we traverse in counter-clockwise direction. We say that the curve is convex where it turns left and concave where it turns right. We say it has *bounded convex curvature* if it turns left with curvature at most 1. There is no bound on the right-curvature, and we may even have sharp concave corners. We say that a set in the plane has bounded convex curvature if all the connected components are simply-connected and the weakly simple, closed curve around the boundary of each connected component has bounded convex curvature. Similarly, we say that a closed curve has *bounded curvature in general* if it turns to the left and to the right with curvature at most 1.

A nice composition property of sets of bounded convex curvature is that if we take two such sets Q_1 and Q_2, unite them, and fill out any holes, then the resulting set, denoted $Q_1 \uplus Q_2$, has bounded convex curvature. See Figure 1, where the boundary of $Q_1 \uplus Q_2$ is the thick curve. This composition property does not hold if we demand that the curvature is bounded in general.

The input to our problem is a simply-connected closed set of points P in the plane such that the boundary ∂P is a curvilinear polygon consisting of n line segments and circular arcs of arbitrary radii. We present an algorithm that in $O(n \log n)$ time finds the unique maximum subset $Q \subseteq P$ of bounded convex curvature, that is, Q contains any other $Q' \subseteq P$ of bounded convex curvature. See Figure 2 for an example.

We note that the uniqueness of a maximal subset Q of bounded convex curvature follows from the composition property; for if there was another $Q' \subseteq P$ of bounded convex curvature that was not contained in Q, then $Q \uplus Q'$ would also be contained in P and have bounded convex curvature, contradicting the maximality of Q.

The boundary of Q will be a curvilinear polygon consisting of $O(n)$ line segments and circular arcs. A very useful property of the boundary of Q is that all concave arcs and vertices are also on the boundary of P. Indeed, it is easy to verify that if there is a concave arc or vertex on the boundary of Q which is not on the boundary of P, then Q is not maximal. A similar reasoning implies that if the boundary of P is a simple curve, then so is the boundary of Q.

We now describe different contexts in which this problem appears naturally. The general problem is that we are given an area S of the plane whose boundary is represented as a curvilinear polygon. There is a thin layer of material in S close to the boundary ∂S of S. The goal is to remove that layer without removing anything from outside S. We are given a rough tool, and we want to remove as much as possible of the thin layer, leaving as little as possible for finer tools that are more expensive to use. The output is a toolpath for the

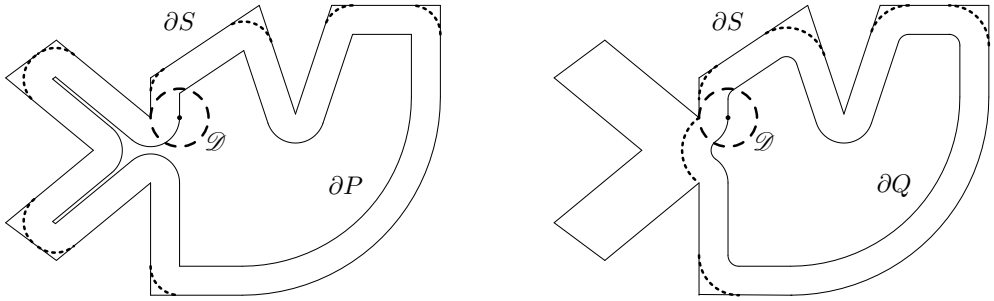

Figure 3 A pocket bounded by ∂S. To the left is shown the boundary ∂P of the inwards offset of S by r. To the right is shown the boundary ∂Q of the maximum subset with bounded convex curvature of P. The dotted arcs in the corners show the boundary of the material in S that cannot be removed by $\mathcal{D}$ using the two toolpaths.

rough tool consisting of one or more curvilinear polygons. The tool is a disk $\mathcal{D}$ of radius r, where r is bigger than the width of the layer we wish to remove. The toolpath is the path which the center of $\mathcal{D}$ is following and the material cut away is the area swept by $\mathcal{D}$ as its center follows the toolpath. The reason that we only have to handle a thin layer close to the boundary of S is because the area farther from the boundary is removed beforehand by tools that are less precise since they do not get close to the boundary. Thus we may assume that the material at all points at some distance $\delta \leq r$ from the boundary have been removed. Some of the points closer to the boundary may also have been removed, but this only makes it easier for our tool to move. With this in mind, when the tool follows a weakly simple closed curve, we think of it as removing the interior of the curve plus every point at distance at most r to the curve.

Let P be the inwards offset of S by r, that is, P is the subset of S of points with distance at least r to the pocket boundary ∂S. See Figure 3. P is the set of all allowed positions of the center of $\mathcal{D}$. If we had complete control over the tool, then we would be able to remove the material in all of P and at all points with distance at most r from P by letting the tool center traverse the boundary ∂P. However, there are restrictions on what tool paths we can follow, e.g., we cannot count on following a toolpath with sharp corners in a precise way.

We are now ready to describe the first application where we want to compute the maximum subset of bounded convex curvature. Assume that the tool can only follow a path which has bounded curvature in general. Assume furthermore that the tool can turn at least as sharply as its own boundary, that is, $r \geq 1$.

Using our algorithm for bounded convex curvature, we are able to identify the maximum area that can be removed by the tool such that the toolpath has bounded curvature in general. First we compute the above set P which is the inwards offset of S by r. The boundary ∂P can be computed from the Voronoi diagram of ∂S [7]. Clearly the toolpath has to stay inside P. We now note that every concave part of ∂P has curvature at most $1/r \leq 1$. Next we use our algorithm to find the maximum subset $Q \subseteq P$ of bounded convex curvature, see Figure 3. The area cut away as $\mathcal{D}$ follows ∂Q is the unique maximum subset that can be cut out of S using a tool with radius $r \geq 1$ and such that the toolpath has bounded convex curvature. However, all concave arcs and vertices on ∂Q stem from P which has concave curvature at most $1/r \leq 1$. It follows that using the toolpath ∂Q, we cut out the maximum subset of S under the condition that the toolpath has bounded curvature in general. This

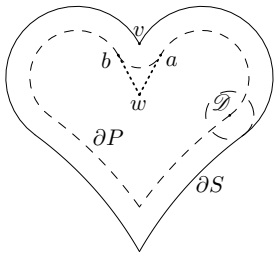

Figure 4

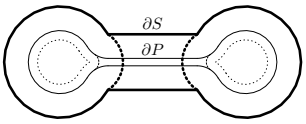

Figure 5

also implies that we remove the maximum subset of the thin layer of material close to the pocket boundary ∂S.

In the above example, the set P had bounded concave curvature. In particular, all concave arcs on ∂P have radius at least r, and for each concave arc A on ∂P of radius r and center v, there is an associated concave vertex v of ∂S. Let a and b be the first and last point on A. When the tool follows A, the corner v will be on the tool boundary $\partial \mathcal{D}$, and the slightest imprecision will blunt the corner v. A recommended alternative [10] is that we substitute A with two line segments aw and wb tangential to A at a and b, respectively, thus creating a sharp concave corner w on the toolpath, see Figure 4. Using this technique, the corner v will be cut much sharper and more precise. One can think of various variations of this technique, since we can "casually" stop and turn the tool at any point on its way to w because the remaining toolpath already ensures that all material will be cut away. This shows that we cannot in general assume that there is any bound on the concave curvature of the input toolpath. We also note the asymmetry with convex corners and arcs, where overshooting a convex corner implies an illegal cut through the boundary of S.

We shall now provide a completely different explanation for the need for bounded convex curvature. The point is that it is often preferable for the surface quality of the resulting part that the tool moves with a constant speed. Recall that the tool is only removing a thin layer close to the pocket boundary. The width of this layer is typically a deliberately chosen small fraction of the tool radius r. When moving at constant speed, a convex turn implies a higher engagement of the tool in the sense of the amount of material removed per time unit. In concave turns the engagement is only decreased. A too high engagement could break the tool, and therefore we must bound the convex curvature of the tool path.

These and other issues related to the machining of corners have been extensively studied in the technical literature on pocket machining. See for instance the papers [5, 6, 9, 11, 14]. There are several previous papers suggesting methods to get bounded convex curvature, but none of them guarantees an optimal solution like ours. One idea for how to handle convex corners is to replace each of them by a convex circular arc as deep in the corner as possible. This is suggested and studied in the papers [5, 9, 11]. However, in all the papers it is assumed that every corner is formed by two line segments which are sufficiently long (relative to the angle between them) that a tangential corner-rounding arc of sufficient size can be placed inside the wedge they form. As can be seen in Figure 3, this is not always the case, and rounding a toolpath can require more complicated modifications.

One heuristic used to obtain a non-trivial subset of bounded convex curvature is the *double offset method*, where we offset P inwards by 1 and then offset the result outwards by 1 and use that as Q. This can be computed in $O(n \log n)$ time using Voronoi diagrams [13]. However, the method does not in general result in the maximum subset of bounded convex

curvature. See for instance Figure 5, where P has bounded convex curvature and all material in S will be machined using ∂P as the toolpath. ∂S is the thick solid curve and ∂P is the thin solid curve. The innermost dotted curves are the boundary of the inwards offset of P by 1. The area of S between the thick dashed arcs will not be machined if the double offset method is used.

In Section 2 we describe an algorithm that computes the maximum subset of bounded convex curvature of P. In Section 3 we describe how to implement the algorithm so that it uses $O(n)$ space and runs in $O(n \log n)$ time. Many proofs and some definitions are omitted due to limited space. We refer to the full version [1].

1.1 General notation and conventions

If M is a set of points in the plane, ∂M denotes the boundary of M and $\overline{M}$ denotes the closure of M. Let γ be a simple curve and x and y two points on γ. Then $\gamma[x, y]$ is the portion of γ between x and y including x and y. Similarly, $\gamma(x, y)$, $\gamma[x, y)$, and $\gamma(x, y]$ are used in the obvious way when none or one of x, y is included. If γ is a closed curve, $\gamma[x, y]$ denotes the portion of γ from x to y in the counterclockwise direction. As before, round parenthesis can be used to exclude one or both endpoints. If γ is not closed, the order of x and y does not matter. We say that a point $z \in \gamma$ is an *inner point* of γ if z is not an endpoint of γ.

1.2 Mathematical foundation: The theorem of Pestov and Ionin

An oriented, simple curve γ is in the class $\mathcal{L}^+$ if γ turns to the left with curvature at most 1, but γ is allowed to turn to the right arbitrarily sharply. There can be corners on γ where γ is not differentiable, but in such corners γ has to turn to the right. These notions are defined rigorously in [1]. Note that a portion of a closed curve with bounded convex curvature traversed in counterclockwise direction is a curve in $\mathcal{L}^+$.

The correctness of the algorithm presented in this paper depends on the following generalization of the Peston-Ionin Theorem. We refer to [1] for a proof.

▶ **Theorem 1.** *Consider an open disk D of arbitrary radius and a curve γ in $\mathcal{L}^+$ from a to b such that $\gamma \cap \overline{D} = \{a, b\}$. Let R be the region bounded by $\partial D[b, a]$ and γ. If R contains D, then R contains an open unit disk U such that there exists a point $c \in \partial U \cap \gamma(a, b)$.*

▶ **Corollary 2.** *Every closed curve γ with bounded convex curvature contains an open unit-disk in its interior.*

The original theorem by Pestov and Ionin [12] was similar to Corollary 2, but γ was assumed to have bounded curvature in general. Howard and Treibergs [8] proved Corollary 2 for closed curves γ with bounded convex curvature for a more restricted class of curves. Ahn et al. [2] proved a theorem similar to Theorem 1, but assuming that γ has bounded curvature in general. Hence, the version of the theorem presented here generalizes all previous versions known to the authors of the present paper.

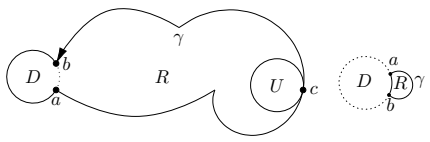

Figure 6 Cases where Theorem 1 does and does not apply.

2 Algorithm

2.1 Preliminaries

We assume that a simply-connected region P_1 in the plane is given such that the boundary ∂P_1 consists of a finite number of line segments and circular arcs of arbitrary radii. For ease of presentation, we assume that ∂P_1 is a simple curvilinear polygon, but our algorithm works as long as ∂P_1 is weakly simple. We can therefore without loss of generality assume that P_1 is an open set. We set $P \leftarrow P_1$ and our algorithm keeps removing parts of P while maintaining the invariant that $\overline{P}$ contains every subset of $\overline{P_1}$ of bounded convex curvature. In the end, $\overline{P}$ itself has bounded convex curvature and it follows that $\overline{P}$ is the unique maximum subset of $\overline{P_1}$ of bounded convex curvature.

The region P is always a collection of disjoint, open, simply-connected sets each of which is bounded by a simple curvilinear polygon. P is represented by its boundary ∂P, which is a collection of disjoint, closed, simple curves where no curve is contained in the interior of another. The input P_1 is defined by one such curve. The open region enclosed by each curve is one connected component of P. ∂P is represented as a set of points known as the *vertices* of ∂P and a set of line segments and circular arcs known as the *arcs* of ∂P. We think of line segments as circular arcs with infinite radius and therefore in most cases use the word *arcs* for both circular arcs and line segments. Depending on the context, we may consider a vertex as a point or a set containing a single point. An *object* of ∂P is a vertex or an arc. We use the convention that an arc includes its endpoints. Every two arcs of ∂P are disjoint except possibly at the endpoints, and for each vertex there are two arcs having an endpoint at that vertex. This way, the arcs form the closed curves bounding P. We always use n to denote the number of vertices of the input ∂P_1.

The boundary of each connected component of P is oriented in counterclockwise direction and we say that a point moving on ∂P is moving *forward* (resp. *backward*) if it is following (resp. not following) the orientation of the boundary. Similarly, we orient every arc following the orientation of the boundary of the component containing it. We denote the endpoints of an arc A as $s(A)$ and $t(A)$, so that a point moving forward along A moves from $s(A)$ to $t(A)$. An arc which turns to the left when traversed in forward direction is called a *convex arc*. A *concave arc* is defined analogously. Line segments are regarded as both concave and convex arcs at the same time. A vertex v is *convex* if the interior angle of ∂P at v is strictly less than π. If the angle is strictly more than π, v is *concave*.

Let A be an arc of ∂P and $a \in A$ a point on A. Then $\mathbf{n}_A(a)$ is the unit-normal of A at a which points to the left relative to the orientation of A. We say that two arcs A and B are *tangential* if $t(A) = s(B)$ and $\mathbf{n}_A(t(A)) = \mathbf{n}_B(s(B))$. Note that A and B are tangential if and only if the vertex $t(A)$ is neither convex nor concave.

2.2 High-level description of the algorithm

A high-level description of the algorithm is given as Algorithm 1. The basic format is to maintain a stack Σ of pointers to objects of ∂P causing the convex curvature condition on ∂P to be violated. An object $\sigma \in \Sigma$ can be a convex vertex, a convex arc of radius less than 1, or a special *cut arc* of radius 1, where one or both of the endpoints might be convex vertices. In each iteration of the loop at line 3, we test if the object σ on the top of Σ is still on ∂P (it might have been removed in another iteration) and if so, we eliminate it by removing from P a subset $\mathcal{V} \subseteq P$. The object σ appears on the boundary $\partial\mathcal{V}$ and not on the boundary of $P \setminus \mathcal{V}$. By *performing a cut* or simply a *cut*, we mean the process of removing $\mathcal{V}$

Algorithm 1: `SubsetOfBoundedConvexCurvature`(P)

```
1 Add all convex arcs of ∂P with a radius less than 1 to Σ.
2 Add all convex vertices of ∂P to Σ.
3 while Σ ≠ ∅
4     Let σ be the topmost element on Σ and remove σ from Σ.
5     if σ ⊂ ∂P and σ is not a perfect cut arc
6         Cut away V from P and let C_j, j = 1, ..., t, be the new cut arcs after the cut.
7         Add each new cut arc C_j to Σ.
8 return P
```

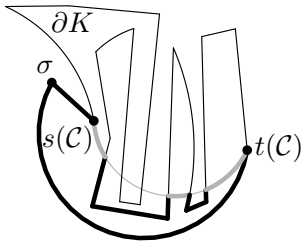

Figure 7

from P. It is important to choose $\mathcal{V}$ such that $\mathcal{V} \cap Q = \emptyset$ for every set $Q \subset \overline{P_1}$ of bounded convex curvature. Theorem 1 will be used to prove this.

It is possible that a cut splits P into more components. The algorithm will then keep working on each component separately. A cut can introduce new unit-radius cut arcs on ∂P, the endpoints of which can be convex vertices. Therefore, these new cut arcs are added to Σ so that the new convex vertices are eventually removed. The algorithm terminates when Σ is empty, which means that P has bounded convex curvature.

Let P_i be the set P in the beginning of iteration $i = 1, 2, \ldots$ of the loop at line 3 in Algorithm 1. We shall prove that the algorithm terminates after $k = O(n)$ iterations. Hence we have $P_1 \supseteq P_2 \supseteq \cdots \supseteq P_k$. The challenge is to define $\mathcal{V}$ in each iteration so that $k = O(n)$, $P \leftarrow P \setminus \mathcal{V}$ can be computed efficiently, and $\mathcal{V} \cap Q = \emptyset$ for every set $Q \subset \overline{P_1}$ of bounded convex curvature.

2.3 Specifying a cut using an arc $\mathcal{C}$

Let σ be an object from Σ that is to be eliminated in a certain iteration of Algorithm 1, and let K be the connected component of P such that $\sigma \subset \partial K$. If we conclude that K does not contain any non-empty set of bounded convex curvature, we set $\mathcal{V} = K$, so that all of the component K is removed from P. Otherwise, we specify $\mathcal{V}$ using a circular arc $\mathcal{C}$ of unit radius. See Figure 7, where $\mathcal{C}$ is grey and $\mathcal{V}$ is the area enclosed by the thick closed curve. Assume for now that we have defined $\mathcal{C}$. Let $s(\mathcal{C})$ and $t(\mathcal{C})$ be the points where $\mathcal{C}$ starts and ends in counterclockwise direction, respectively. The endpoints are included in $\mathcal{C}$ and are points on ∂K. They will be defined so that $\sigma \subseteq \partial K[s(\mathcal{C}), t(\mathcal{C})]$. It will follow from the definition of $\mathcal{C}$ that there will be at most two vertices on $\partial K(s(\mathcal{C}), t(\mathcal{C}))$. For efficiency, the algorithm may choose an arc $\mathcal{C}$ that intersects $\partial K(s(\mathcal{C}), t(\mathcal{C}))$.

The arc $\mathcal{C}$ divides K into open regions $R_1, \ldots, R_r$, which are the connected components of $K \setminus \mathcal{C}$. Assume without loss of generality that among all these regions, exactly the regions

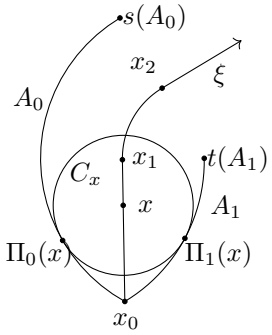

Figure 8

$R_1, \ldots, R_s$, $s \leq r$, contain a part of $\partial K(s(\mathcal{C}), t(\mathcal{C}))$ on their boundary. We note that $s = 1$ if $\mathcal{C}$ does not intersect $\partial K(s(\mathcal{C}), t(\mathcal{C}))$, as is the case in Figure 7. Define $\mathcal{V}$ as the union $\bigcup_{i=1}^{s} K \cap \overline{R_i}$. Then $K \setminus \mathcal{V}$ is open and hence so is our new $P \leftarrow P \setminus \mathcal{V}$. The arc $\mathcal{C}$ will be carefully chosen so that $\mathcal{V} \cap Q = \emptyset$ for every set $Q \subset \overline{P_1}$ of bounded convex curvature.

▶ **Lemma 3.** *$\mathcal{V}$ is connected and for each $i = 1, 2, \ldots$, the boundary ∂K of each connected component K of P_i is a simple curvilinear polygon.*

Consider the case where we remove a proper subset $\mathcal{V}$ from a connected component K of P_i to obtain P_{i+1}. One or more arcs $\mathcal{C}_1, \ldots, \mathcal{C}_t$ of ∂P_{i+1} are subsets of $\mathcal{C}$ and not arcs of ∂P_i. We denote the arcs $\mathcal{C}_1, \ldots, \mathcal{C}_t$ as *new cut arcs* of ∂P_{i+1}. An arc of ∂P_{i+1} which is a subset of a new cut arc of ∂P_j, $j \leq i$, is not a new cut arc of ∂P_{i+1}. An arc of ∂P_{i+1} is a *cut arc* if it is the subset of a new cut arc of ∂P_j for some $j \leq i + 1$. We say that a cut arc $\mathcal{C}$ of ∂P is *perfect* if none of the endpoints of $\mathcal{C}$ are convex vertices.

▶ **Lemma 4.** *Every cut arc of ∂P is convex.*

2.4 The bisector curve ξ

A specific type of bisector curve, illustrated in Figure 8, turns out to be useful when describing how to define the arc $\mathcal{C}$ specifying a cut and when proving the correctness of our algorithm. The curve is somewhat related to the medial axis of ∂P. Before introducing the curve, we need the concept of an *osculating circle*.

Let A be an arc of ∂P. A circle C *osculates* A at $a \in A$ if the interior of C is disjoint from A, C and A have the point a in common, and the center x of C is a point $a + \mathbf{n}_A(a) \cdot c$ where $c \geq 0$. Recall that $\mathbf{n}_A(a)$ is the unit-normal to A in a pointing to the left. If $c = 0$, C is a single point. Note that a is the point on A closest to x. A special case happens when A is an arc on C, in which case A is a convex arc and C osculates A at every point on A.

▶ **Observation 5.** *Let A be an arc of ∂P and x an arbitrary point. There exists at most one circle C with center x that osculates A. If such a circle C exists, the radius of C is the distance to the closest point a on A from x and C osculates A in a. If A is a convex arc, then the radius of C is at most the radius of A.*

We denote the bisector curve as $\xi = \xi(A_0, A_1, x_0)$. A_0 and A_1 are arcs of ∂P and x_0 is the point where ξ begins. ξ consists of at most three intervals to be defined below, each of which is part of a conic section. Depending on x_0, ξ might start in the second or third interval. From the context where ξ is used, it will be clear in which interval ξ starts.

Often, but not always, A_0 and A_1 are neighboring arcs and x_0 is their common endpoint, i.e., $x_0 = t(A_0) = s(A_1)$. In this case, x_0 will be a convex vertex of ∂P (as in Figure 8) or A_0 and A_1 are tangential. If A_0 and A_1 are tangential, one of A_0 and A_1 will be a cut arc.

For each point $x \in \xi$, we are going to define a circle C_x and points $\Pi_0(x) \in A_0 \cap C_x$ and $\Pi_1(x) \in A_1 \cap C_x$. The circle C_x has its center at x. The functions Π_0 and Π_1 are the *projections* associated to ξ and C_x is the *clearance circle* associated to ξ at x.

Interval **(1)** consists of points x such that C_x osculates both A_0 and A_1 at points $\Pi_0(x)$ and $\Pi_1(x)$, respectively. Interval **(2)** consists of points x where $s(A_0) = \Pi_0(x)$ and C_x osculates A_1 at a point $\Pi_1(x)$ or $t(A_1) = \Pi_1(x)$ and C_x osculates A_0 at a point $\Pi_0(x)$. Interval **(3)** consists of points x such that $s(A_0) = \Pi_0(x)$ and $t(A_1) = \Pi_1(x)$.

Assume that x_0 is contained in interval **(1)**. In the following, we think of x as a point that traverses ξ from x_0.

Interval (1): ξ consists of the points x such that there is a circle C_x with center x that osculates A_0 and A_1 at points $\Pi_0(x)$ and $\Pi_1(x)$, respectively. Hence, this interval consists of points x on ξ such that the distance to the closest points on A_0 and A_1 is the same. In the general case, as x traverses ξ, $\Pi_0(x)$ moves continuously backward and $\Pi_1(x)$ moves continuously forward. We eventually reach a point x_1 where C_{x_1} contains $s(A_0)$ or $t(A_1)$, and ξ continues with the interval defined in **(2)**. A special case occurs if $t(A_0) = s(A_1)$ and A_0 and A_1 are tangential. Then x moves in the direction $\mathbf{n}_{A_0}(t(A_0))$ until x is the center x_1 of A_0 or A_1. This happens since one of the arcs is a cut arc. Until x reaches x_1, the projections are constant, $\Pi_0(x) = \Pi_1(x) = t(A_0)$. If x_1 is the center of A_0, we define $\Pi_0(x_1) = s(A_0)$. Similarly, if x_1 is the center of A_1, we define $\Pi_1(x_1) = t(A_1)$.

Interval (2): ξ has reached a point x_1 such that C_{x_1} contains $s(A_0)$ or $t(A_1)$. If $s(A_0) = t(A_1)$, ξ stops here. Otherwise, if C_{x_1} contains both $s(A_0)$ and $t(A_1)$, interval **(2)** is degenerate, $x_2 = x_1$, and ξ continues with the interval described in **(3)**. Otherwise, assume without loss of generality that C_x contains $s(A_0)$ and osculates A_1. From here, ξ consists of points x such that there is a circle C_x with center x that contains $s(A_0) = \Pi_0(x)$ and osculates A_1 at a point $\Pi_1(x)$. When x moves along ξ, $\Pi_1(x)$ moves continuously forward. At some point, we reach a point x_2 where C_{x_2} contains $s(A_0)$ and $t(A_1)$, and ξ continues with the interval defined in **(3)**.

Interval (3): At the last point x_2 of interval **(2)**, C_{x_2} contains both of the points $s(A_0)$ and $t(A_1)$. Now, ξ consists of the points x such that there is a circle C_x with center x containing both $\Pi_0(x) = s(A_0)$ and $\Pi_1(x) = t(A_1)$, i.e., ξ follows a half-line. We define the direction of the half-line to be the counterclockwise rotation by an angle of $\pi/2$ of the vector $t(A_1) - s(A_0)$.

When defining the arc $\mathcal{C}$ that specifies the cut we want to perform, we are often searching for the first point x_3 on ξ where C_{x_3} has radius 1. We shall then define $\mathcal{C} = C_{x_3}[\Pi_0(x_3), \Pi_1(x_3)]$. Yap [13] showed that each of the intervals **(1)**–**(3)** is a part of a conic section. Using elementary geometry we can decide for each interval in $O(1)$ time if it contains a point x_3 such that C_{x_3} has radius 1.

Note that when $x_0 = t(A_0) = s(A_1)$ and x_0 is a convex vertex of ∂P, then a portion of ξ beginning at x_0 is a subset of the medial axis of ∂P. However, x_3 (as defined above) need not be on the medial axis and can even be outside P, also when $\mathcal{C} = C_{x_3}[\Pi_0(x_3), \Pi_1(x_3)]$ becomes a perfect cut arc. See Figure 9 for an example.

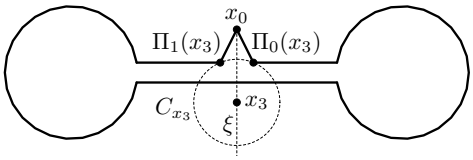

Figure 9

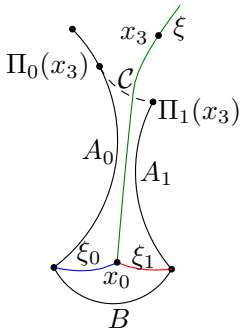

Figure 10

▶ **Observation 6.** *If A_k is a convex arc, $k = 0, 1$, and $x_0 = t(A_0) = s(A_1)$, then for each $a \in A_k$, there exists $x \in \xi$ such that C_x osculates A_k at a, and by Observation 5, the radius of C_x is at most the radius of A_k.*

2.5 Defining the arc $\mathcal{C}$ specifying a cut

The objects in Σ are either **(a)** convex arcs of ∂P_1 with a radius less than 1, **(b)** convex vertices of ∂P_1, or **(c)** cut arcs. Below, we describe how to define $\mathcal{C}$ when the object σ picked by Algorithm 1 is of each of these types.

Case (a): σ is a convex arc with radius less than 1. Consider the curve $\xi = \xi(\sigma, \sigma, x_0)$, where x_0 is the center of σ, which begins in interval **(3)**. Choose the first point $x_3 \in \xi$ such that the associated clearance circle C_{x_3} has radius 1 and define $\mathcal{C} = C_{x_3}[\Pi_0(x_3), \Pi_1(x_3)] = C_{x_3}[s(\sigma), t(\sigma)]$. Said in another way, we define $\mathcal{C}$ to be the unit-radius arc counterclockwise from $s(\sigma)$ to $t(\sigma)$ which spans an angle of less than π.

Case (b): σ is a convex vertex of ∂P. Let A_0 be the arc of ∂P before σ and A_1 the arc after σ. Consider the curve $\xi = \xi(A_0, A_1, x_0)$, where $x_0 = \sigma$. Find the first point $x_3 \in \xi$ where C_{x_3} has a radius of 1. If such a point does not exist, we are in the case where $s(A_0) = t(A_1)$, and we let $\mathcal{V}$ be the complete component K of P where σ appears on the boundary (in this case, the boundary of K only has two arcs, A_0 and A_1). Otherwise, we define $\mathcal{C} = C_{x_3}[\Pi_0(x_3), \Pi_1(x_3)]$.

Case (c): σ is a cut arc B. We assume that one of the endpoints of B is a convex vertex, since otherwise B is perfect and hence ignored by the algorithm. If one of the endpoints, say $s(B)$, is a concave vertex, then $t(B)$ is a convex vertex, and we do as if $\sigma = t(B)$ as described in case **(b)**. We now assume that none of the endpoints of B are concave vertices. See Figure 10. Let A_0 and A_1 be the arcs preceding and succeeding B on ∂P, respectively. Let $\xi_0 = \xi(A_0, B, s(B))$ and $\xi_1 = \xi(B, A_1, t(B))$. For each $b \in B$, follow the ray starting at b with direction $\mathbf{n}_B(b)$ and define $d_k(b)$ to be the distance from b to the first intersection point with ξ_k, $k = 0, 1$. Since $d_0(s(B)) = 0 < d_1(s(B))$ and $d_1(t(B)) = 0 < d_0(t(B))$, the continuity of ξ_k implies there must be an intersection point x_0 between ξ_0 and ξ_1. By Observation 6, we know that the distance from x_0 to B is at most 1. We now consider the curve $\xi = \xi(A_0, A_1, x_0)$ with associated projections Π_0, Π_1, and clearance circle C_x. The clearance circle C_{x_0} of ξ at x_0 is the same as for ξ_0 and ξ_1. If C_{x_0} osculates A_0 and A_1, ξ begins in interval **(1)**. If C_{x_0} osculates one of A_0, A_1 and contains an endpoint of the other,

ξ begins in interval **(2)**. Otherwise, C_{x_0} contains $s(A_0)$ and $t(A_1)$, and ξ begins in interval **(3)**. We find the first point $x_3 \in \xi$ where C_{x_3} has radius 1. If such a point does not exist, we are in the case where $s(A_0) = t(A_1)$, and we let $\mathcal{V}$ be the complete component K of P where σ appears on the boundary. Otherwise, we define $\mathcal{C} = C_{x_3}[\Pi_0(x_3), \Pi_1(x_3)]$.

One may object to the algorithm that since a connected component of P can split into more components by a cut, the arcs on Σ are not necessarily well-defined, since subsets of the arc originally added to Σ can be on the boundary of many different connected components of P. However, since the arcs are convex arcs of radius at most 1, the following lemma shows that it is not a problem.

▶ **Lemma 7.** *Let A be an arc of ∂P_1 which is a convex arc of any radius or a concave arc with radius at least 1. For any $i \geq 1$, there exists at most one connected component K of P_i such that (a subset of) A is an arc of ∂K.*

2.6 Proof of correctness of Algorithm 1

In this section, we prove that the set $\mathcal{V}$ to be cut away from P is disjoint from every subset of $\overline{P_1}$ of bounded convex curvature. The proof is by contradiction. The idea is that if there is a subset of $\overline{P_1}$ of bounded convex curvature that overlaps $\mathcal{V}$, then Theorem 1 gives the existence of a unit disk in $\mathcal{V}$ which cannot be there. In general, $\mathcal{V}$ can have a complicated shape and is not easy directly to reason about. Therefore, we describe a superset of $\mathcal{V}$ consisting of *simplices*, which are sufficiently simple that the argument goes through.

A *simplex* S is a closed region and has a boundary ∂S which is a closed, simple curve consisting of two, three, or four circular arcs. One of the arcs D on the boundary is the *door*. The door D has a radius of at most 1 and is clockwise when ∂S is traversed in counterclockwise order. We denote the following as the *simplex condition*, which every simplex is required to satisfy: Let U be the closed disk such that ∂U contains the door D. Then $U \cup S$ does not contain any unit-disk unless U has radius 1, in which case U is the only unit-disk contained in $U \cup S$.

▶ **Lemma 8.** *There exists a set $\mathfrak{S}$ of at most five simplices such that if $\mathcal{S} = \bigcup_{S \in \mathfrak{S}} S$, then $\mathcal{V} \subset \mathcal{S}$ and $\partial\mathcal{S} \subseteq \partial P[s(\mathcal{C}), t(\mathcal{C})] \cup \mathcal{C}$. There is a tree $\mathfrak{T}$ with the nodes being $\mathfrak{S}$ with the following properties. The root of $\mathfrak{T}$ is a simplex with the door $\mathcal{C}$. If a curve $\phi \subset \overline{P}$ exits a simplex $S_0 \in \mathfrak{S}$, it is either through the door of S_0 or through an arc which is the door of a child of S_0 in $\mathfrak{T}$, and in the latter case, ϕ enters the child. If S_0 has a parent in $\mathfrak{T}$ and ϕ exits S_0 through the door of S_0, then ϕ enters the parent.*

An example of the tree $\mathfrak{T}$ is given in Figure 11 for case **(c)** from Section 2.5. The following lemma is the heart of our proof of correctness. Recall that the curves in $\mathcal{L}^+$ are open, oriented curves that turn to the left with curvature at most 1.

▶ **Lemma 9.** *Let γ be a curve in $\mathcal{L}^+$ contained in $\overline{P}$. Let $\mathcal{C}$ be an arc specifying an area $\mathcal{V}$ to be removed from P and suppose that γ starts at a point $a \in \mathcal{C}$. If γ leaves $\overline{\mathcal{V}}$ again, it is through a point on $\mathcal{C}[s(\mathcal{C}), a]$.*

Proof. Assume for contradiction that γ leaves $\overline{\mathcal{V}}$ through a point not as stated in the lemma. Since the boundary of $\mathcal{V}$ is contained in $\partial P \cup \mathcal{C}$, γ leaves $\overline{\mathcal{V}}$ from a point on $\mathcal{C}(a, t(\mathcal{C})]$. Consider the tree $\mathfrak{T}$ as described in Lemma 8. Let S be a simplex with maximum distance in $\mathfrak{T}$ to the root such that γ enters S. Let D be the door of S and $s(D)$ and $t(D)$ be the beginning and end of D in counterclockwise order, respectively. Let $\mathcal{D}$ be the open disk such that D is an arc on the circle $\partial\mathcal{D}$. It follows from Lemma 8 that the first time γ enters S,

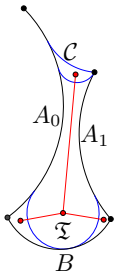

Figure 11

it is through a point on D. Let $e \in D$ be the point closest to $s(D)$ such that γ eventually enters S through e. After e, γ must leave S through a point f on $D(e, t(D)]$, since otherwise γ enters the interior of the closed, simple curve defined by $\gamma[a, e]$, $D[s(D), e]$, $\partial P[s(\mathcal{C}), s(D)]$, and $\mathcal{C}[s(\mathcal{C}), a]$, which it cannot exit again by assumption. Hence, Theorem 1 implies that the region bounded by $\partial\mathcal{D}[f, e]$ and $\gamma[e, f]$ contain a unit disk different from $\mathcal{D}$. That contradicts the simplex condition of S. ◀

By the following lemma, we conclude that when the algorithm terminates, $\overline{P}$ is the maximum subset of $\overline{P_1}$ of bounded convex curvature. The lemma follows easily from Lemma 9.

▶ **Lemma 10.** *For every $i = 1, 2, \ldots$, $\overline{P_i}$ contains Q for every set $Q \subset \overline{P_1}$ of bounded convex curvature.*

2.7 Proof that Algorithm 1 performs a linear number of iterations

Since the set $\mathcal{V}$ that we remove from P is contained in one connected component of P, the algorithm only makes changes to one connected component of P at a time. A vertex v of ∂P is *improper* if v is a convex vertex and v is not a vertex of ∂P_1. An improper vertex can be introduced in two different ways: 1) If the arc $\mathcal{C}$ specifying a cut intersects a cut arc and the intersection point v is on $\partial\mathcal{V}$, then v becomes a convex and hence improper vertex. 2) If the arc $\mathcal{C}$ begins or ends at the endpoint $v \notin \partial P_1$ of a cut arc which is not a convex vertex, v might be a convex and hence improper vertex after the cut has been performed.

Consider the first time an improper vertex appears. Let v be the vertex and let K be the connected component such that $v \in \partial K$. Then v will be removed by the subsequent cut performed in K since a cut arc with an endpoint at v is closest to the top of Σ among all objects in $\Sigma \cap \partial K$. It follows inductively that there can be one or two improper vertices in each component, and after the next iteration performed in the component, none of them will be left.

A vertex v of ∂P which is not a point on ∂P_1 (note the difference between being a *vertex* of ∂P_1 and being a *point* on ∂P_1) can only be created if the arc $\mathcal{C}$ specifying a cut to be performed in P intersects a cut arc at v. It follows that v is a convex and hence improper vertex. Equivalently, every proper vertex is on ∂P_1.

Consider an endpoint c of the arc $\mathcal{C}$ specifying a cut to be performed in P_i, where $P_{i+1} = P_i \setminus \mathcal{V}$. If c is an inner point of an arc A of ∂P_i, Lemma 4 together with Observation 5 gives that A is not a cut arc since the circle containing $\mathcal{C}$ osculates A at c. Hence, A is (a subset of) an arc of ∂P_1 and $c \in \partial P_1$. Furthermore, due to the way $\mathcal{C}$ is defined, an improper vertex of ∂P_i will never be chosen as c. Also in this case, we get that c is on ∂P_1. We summerize these observations in the following lemma.

▶ **Lemma 11.**

1. *There are at most two improper vertices of ∂K for each connected component K of P. If there are two, they are endpoints of the same cut arc.*
2. *All improper vertices of ∂K, where K is a connected component of P, are removed by the following cut performed in K.*
3. *The endpoints $s(\mathcal{C})$ and $t(\mathcal{C})$ of an arc $\mathcal{C}$ specifying a cut are both points on ∂P_1.*

▶ **Lemma 12.** *The algorithm performs $O(n)$ iterations.*

Sketch of proof. We divide the iterations into the types 1–5 defined below and prove that there is a linear number of each type. We bound the iterations of type 1 by observing that each such iteration corresponds to introducing an edge in a plane multigraph with $2n$ vertices, where each pair of vertices can be connected by at most $O(1)$ edges. Hence, Euler's formula implies that $O(n)$ edges are created in total. The iterations of types 2–5 can be bounded by the number of iterations of type 1 or by observing that each iteration completely removes one of the objects of ∂P_1 from ∂P. Hence there are $O(n)$ of each type.

Consider what happens during iterations i when the algorithm picks an object σ from Σ. Let $\mathcal{V}$ be the set removed from P_i during the iteration, i.e., $P_{i+1} = P_i \setminus \mathcal{V}$, where possibly $\mathcal{V} = \emptyset$. Lemma 11 implies that the iteration is of one of the following types.

1. $\mathcal{V}$ is specified by an arc $\mathcal{C}$ and each endpoint of $\mathcal{C}$ is either a vertex of ∂P_1 or an inner point of an arc of ∂P_1 which is not a vertex of ∂P_i.
2. $\mathcal{V}$ is specified by an arc $\mathcal{C}$ and one or both endpoints of $\mathcal{C}$ is a vertex v of ∂P_i which is not a vertex of ∂P_1. Hence, such a vertex v is the endpoint of a cut arc of ∂P_i.
3. A complete connected component K is removed from P, i.e., $\mathcal{V} = K$.
4. $\mathcal{V} = \emptyset$ and nothing happens since σ is not anymore on ∂P.
5. $\mathcal{V} = \emptyset$ and nothing happens since σ is a perfect cut arc.

Consider cuts of type 1. We see that each such cut corresponds to adding an edge to a plane graph G with $2n$ nodes. There is one node in G for each arc and vertex of ∂P_1. We choose a curve ϕ from $s(\mathcal{C})$ to $t(\mathcal{C})$ contained in $\mathcal{V}$ as the edge corresponding to the cut. This is possible since $\mathcal{V}$ is connected by Lemma 3. A vertex v of ∂P_1 is incident to ϕ if v is an endpoint of ϕ. An arc A of ∂P_1 is incident to ϕ if an endpoint of ϕ is an inner point of A. Let A_0 and A_1 be the objects of ∂P_1 incident to ϕ such that $s(\mathcal{C}) \in A_0$ and $t(\mathcal{C}) \in A_1$. Let C be the circle containing $\mathcal{C}$. Observe that if A_0 (resp. A_1) is an arc, then C osculates A_0 at $s(\mathcal{C})$ (resp. A_1 at $t(\mathcal{C})$).

We say that A_0 and A_1 are *parallel* if:

- A_0 and A_1 are concentric arcs, one is concave, the other is convex, and the radius of the convex is 2 larger than the radius of the concave.
- A_0 and A_1 are two parallel line segments (in the ordinary sense) with opposite directions and distance 2.
- A_1 (resp. A_0) is a vertex while A_0 (resp. A_1) is convex arc with radius 2 and center A_1 (resp. A_0).

If A_0 and A_1 are parallel, there are infinitely many unit-circles that osculates A_0 or A_1 like C does. Otherwise, there are only $O(1)$. Hence, by Euler's formula, we are adding $O(n)$ edges to G that connect non-parallel objects. One can verify that even if A_0 and A_1 are parallel, we make $O(1)$ edges between them in G. ◀

Since every cut arc is eventually picked from Σ by Algorithm 1, the linear bound on the number of iterations leads to the following lemma.

Algorithm 2: `PerformCut`$(\mathcal{C})$

```
1  U ← {s(C)}.
2  while U ≠ ∅
3      Remove a point e from U. If e is not a vertex of ∂P, create a vertex at e.
4      a ← e.
5      repeat
6          Let b ← TraverseP(C, a, e). If b is not a vertex of ∂P, create a vertex at b.
7          Mark all objects on ∂P[a, b] as removed, except for vertices a and b.
8          If b ∈ ∂P(s(C), t(C)), add b to U.
9          if C enters or leaves P at b
10             a ← TraverseC(C, b). If a is not a vertex of ∂P, create a vertex at a.
11             Change ∂P by setting the arc succeeding a and preceding b to C[a, b].
12         else
13             a ← e.
14     until a = e
```

▶ **Lemma 13.** *The total number of cut arcs created while running Algorithm 1 is $O(n)$. Likewise, the total number of different vertices appearing on ∂P while running Algorithm 1 is $O(n)$.*

We have now proved the following theorem.

▶ **Theorem 14.** *Given a simply-connected open region P_1 bounded by a curvilinear polygon consisting of n line segments and circular arcs, there is a unique maximum set $Q \subseteq \overline{P_1}$ of bounded convex curvature which contains every set $Q' \subseteq \overline{P_1}$ of bounded convex curvature. ∂Q consists of $O(n)$ line segments and circular arcs, and Algorithm 1 computes Q in $O(n)$ iterations.*

3 Implementation

In this section, we show how the algorithm can be implemented so that it uses $O(n \log n)$ time in total and $O(n)$ space.

Let $\mathcal{C}$ be the arc specifying a cut to be performed in P. Let $R_1, \ldots, R_r$ be the connected components of $P \setminus \mathcal{C}$, and assume without loss of generality that the sets $R_1, \ldots, R_s$, $s \leq r$, have to be removed since their boundaries contain a part of $\partial P(s(\mathcal{C}), t(\mathcal{C}))$. For ease of presentation we assume that $\mathcal{C}$ contains no vertices of ∂P except for possibly the endpoints $s(\mathcal{C})$ and $t(\mathcal{C})$. Algorithm 2 traverses the boundary of each of the regions $R_1, \ldots, R_s$ and introduces the new cut arcs in order to remove the regions from P.

The algorithm traverses $\partial P(s(\mathcal{C}), t(\mathcal{C}))$ from $s(\mathcal{C})$ and removes the regions $R_1, \ldots, R_s$ one by one. Each point in the set U is on the boundary of one of the regions $R_1, \ldots, R_s$ which has not so far been removed. The loop at line 5 traverses one such region.

The subroutine `TraverseP`$(\mathcal{C}, a, e)$ traverses ∂P from a until we meet e or a point where $\mathcal{C}$ enters or exits P. The point b where we stop traversing is returned.

The subroutine `TraverseC`$(\mathcal{C}, b)$ follows $\mathcal{C}$ from b through P until it exits P at some point a, which is returned. Since we assumed that no vertex of ∂P is an inner point of $\mathcal{C}$, it is uniquely defined which direction of $\mathcal{C}$ to follow.

Since P is simply-connected, the regions $R_1, \ldots, R_r$ induce a tree T such that there is an edge in T between two regions R_j and R_k if there is an arc $\mathcal{C}(x, y) \subset \partial R_j \cap \partial R_k$ for $x \neq y$. By Lemma 3, $\mathcal{V}$ is connected, so the regions $R_1, \ldots, R_s$ induce a subtree in T. Therefore, each of the regions in T is traversed at least once by Algorithm 2. On the other hand, a region is removed from P while its boundary is traversed, so each region is traversed at most once. Hence, we have the following lemma.

▶ **Lemma 15.** *Algorithm 2 traverses the boundary of each region $R_1, \ldots, R_s$ exactly once and thus correctly computes $P \leftarrow P \setminus \mathcal{V}$ as specified by the arc $\mathcal{C}$.*

The point $\texttt{TraverseC}(\mathcal{C}, b)$ is the first intersection point between $\mathcal{C}$ and ∂P when following $\mathcal{C}$ from b in the relevant direction. Cheng et al. [4] describes an efficient solution to the following problem. For a simple polygon V, preprocess V such that queries of the following kind can be answered efficiently: Given a circular unit-radius arc beginning at some point in the interior of V, find the first intersection point between the arc and V if it exists. The algorithm requires $O(n)$ space, uses $O(n \log n)$ time on preprocessing, and answers queries in $O(\log n)$ time, where n is the number of vertices of V.

It is straightforward to generalize the method of Cheng et al. to curvilinear polygons. We apply the preprocessing to the original input ∂P_1. Thus, by querying an arc following $\mathcal{C}$ from b in the direction through P, we know the point where $\mathcal{C}$ exits P_1. However, the arc $\mathcal{C}$ can exit P before it exits P_1, namely if and only if it crosses a cut arc of ∂P. In the following, we show how to detect if that is the case or not.

Since $\mathcal{C}$ enters or exits P at b, there exists a point $d \in \mathcal{C} \cap P$ such that $\mathcal{C}(d, b) \subset P$. In the following, d denotes such a point. It is not necessary to compute d, it is merely a tool in our analysis. We know from Lemma 11 that $s(\mathcal{C}), t(\mathcal{C}) \in \partial P_1$. Using the circular ray shooting data structure, we can find a closed subset $\mathcal{C}' \subseteq \mathcal{C}$ such that $s(\mathcal{C}'), t(\mathcal{C}') \in \partial P_1$, $\mathcal{C}'(s(\mathcal{C}'), t(\mathcal{C}')) \subset P_1$, and $b \in \mathcal{C}'$. Hence also $d \in \mathcal{C}'$. The following lemma says that if the arc $\mathcal{C}'$, when traversed in any direction from d, enters a removed area, i.e., a connected component of $P_1 \setminus P$, then it stays in that removed area.

▶ **Lemma 16.** *Let x be one of the endpoints of $\mathcal{C}'$. There is at most one cut arc A of ∂P which intersects $\mathcal{C}'(d, x)$. There is such an arc A if and only if $x \notin \partial P$. If A exists, $\mathcal{C}'(d, x)$ and A intersect at a unique point y, and $\mathcal{C}'[y, x) \subset P_1 \setminus P$.*

It is possible that $\mathcal{C}'(s(\mathcal{C}'), d)$ and $\mathcal{C}'(d, t(\mathcal{C}'))$ intersect the same cut arc B. That is easy to detect since we are given a point $b \in \partial P \cap \mathcal{C}'$ where $\mathcal{C}'$ enters of leaves P. Thus, we check if b is on a cut arc B and if $\mathcal{C}(b, x)$ intersects the same arc. Otherwise, every cut arc A satisfies that $A \cap \mathcal{C}'(s(\mathcal{C}), d) = \emptyset$ or $A \cap \mathcal{C}'(d, t(\mathcal{C})) = \emptyset$, and we have the following lemma.

▶ **Lemma 17.** *Let A be a cut arc of ∂P such that $s(A), t(A) \in \partial P_1$. Assume that $A \cap \mathcal{C}'(s(\mathcal{C}'), d) = \emptyset$ or $A \cap \mathcal{C}'(d, t(\mathcal{C}')) = \emptyset$. Then, the following two statements hold:*

- *A and $\mathcal{C}'(s(\mathcal{C}'), d)$ intersect if and only if the endpoints of $\mathcal{C}'$ and A appear in the order $s(\mathcal{C}'), t(A), t(\mathcal{C}'), s(A)$ on ∂P_1.*
- *A and $\mathcal{C}'(d, t(\mathcal{C}'))$ intersect if and only if the endpoints of $\mathcal{C}'$ and A appear in the order $s(\mathcal{C}'), s(A), t(\mathcal{C}'), t(A)$ on ∂P_1.*

See Figure 12, which illustrates Lemma 17. The black arc A is the only cut arc of ∂P. The thick part of the grey arc $\mathcal{C}$ is $\mathcal{C}'$.

Lemma 17 leads to our method for finding cut arcs intersecting an arc $\mathcal{C}'$ by searching after arcs with endpoints on specific portions of ∂P_1. We associate to each point p on ∂P_1 a unique number $\varphi(p) \in [0, n)$. Let the vertices of ∂P_1 be $v_0, v_1, \ldots, v_{n-1}, v_n$, where $v_0 = v_n$. We set $\varphi(v_i) = i$ for $i < n$. For the points p on the arc between two vertices v_i and v_{i+1}, we

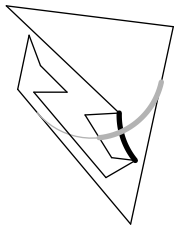

Figure 12

Algorithm 3: `TraverseC`$(\mathcal{C}, b)$

1 Use the circular ray shooting data structure to find the arc $\mathcal{C}' \subseteq \mathcal{C}$ such that $\mathcal{C}'(s(\mathcal{C}'), t(\mathcal{C}')) \subset P_1$, $s(\mathcal{C}'), t(\mathcal{C}') \in \partial P_1$, and $b \in \mathcal{C}'$.
2 Let $z \in \{s(\mathcal{C}'), t(\mathcal{C}')\}$ be the endpoint of $\mathcal{C}'$ when following $\mathcal{C}'$ from b through P.
3 If b is on a cut arc B of ∂P and $\mathcal{C}(b, z)$ intersects B at a point a, return a.
4 Ask the data structure Θ if any point $\theta(A)$ is in the rectangle(s) as specified by Lemma 18. If so, return the intersection point between A and $\mathcal{C}'(b, z)$.
5 Return z.

interpolate between i and $i+1$ to uniquely define $\varphi(p)$. For an arc A with $s(A), t(A) \in \partial P_1$, we define an associated point $\theta(A) \in [0, n) \times (0, 2n)$ in the following way:

$$\theta(A) = \begin{cases} (\varphi(s(A)), \varphi(t(A))) & \text{if } \varphi(s(A)) < \varphi(t(A)) \\ (\varphi(s(A)), \varphi(t(A)) + n) & \text{otherwise.} \end{cases}$$

Lemma 17 then leads to the following.

▶ **Lemma 18.** *Let A be a cut arc of ∂P such that $s(A), t(A) \in \partial P_1$. Assume that $A \cap \mathcal{C}'(s(\mathcal{C}'), d) = \emptyset$ or $A \cap \mathcal{C}'(d, t(\mathcal{C}')) = \emptyset$. Let $\theta(\mathcal{C}') = (x, y)$.*

- *A and $\mathcal{C}'(s(\mathcal{C}'), d)$ intersect if and only if*
  - *$y < n$ and $\theta(A)$ is in $[0, x) \times (x, y)$ or $(y, n) \times (x + n, y + n)$, or*
  - *$y \geq n$ and $\theta(A)$ is in $(y - n, x) \times (x, y)$.*
- *A and $\mathcal{C}'(d, t(\mathcal{C}'))$ intersect if and only if*
  - *$y < n$ and $\theta(A)$ is in $(x, y) \times (y, x + n)$, or*
  - *$y \geq n$ and $\theta(A)$ is in $[0, y - n) \times (y - n, x)$ or $(x, n) \times (y, x + n)$.*

For each cut arc A of ∂P where $s(A), t(A) \in \partial P_1$, we store the point $\theta(A)$ in a data structure Θ. It is necessary to add new points to and delete points from Θ as the algorithm proceeds, since new cut arcs are created and other no longer appear on ∂P. We need to be able to find a point $\theta(A)$ in a rectangle specified by $\mathcal{C}'$ as stated in Lemma 18. Therefore, we implement Θ as a fully dynamic orthogonal range searching structure as described by Blelloch [3]. Algorithm 3 sketches how to implement `TraverseC`.

It is now possible to bound the running time and memory requirement of Algorithm 1 when using our suggested implementation.

▶ **Theorem 19.** *Algorithm 1 can be implemented so that it runs in time $O(n \log n)$ and uses $O(n)$ space.*

References

1. M. Abrahamsen and M. Thorup. Finding the maximum subset with bounded convex curvature, 2016. URL: `http://arxiv.org/abs/1603.02080`.
2. H.-K. Ahn, O. Cheong, J. Matoušek, and A. Vigneron. Reachability by paths of bounded curvature in a convex polygon. *Computational Geometry*, 45(1):21–32, 2012.
3. G.E. Blelloch. Space-efficient dynamic orthogonal point location, segment intersection, and range reporting. In *Proceedings of SODA*, pages 894–903, 2008.
4. S.-W. Cheng, O. Cheong, H. Everett, and R. van Oostrum. Hierarchical decompositions and circular ray shooting in simple polygons. *Discrete and Computational Geometry*, 32:401–415, 2004.
5. H.S. Choy and K.W. Chan. A corner-looping based tool path for pocket milling. *Computer-Aided Design*, 35(2):155–166, 2003.
6. X. Han and L. Tang. Precise prediction of forces in milling circular corners. *International Journal of Machine Tools and Manufacture*, 88:184–193, 2015.
7. M. Held. Voronoi diagrams and offset curves of curvilinear polygons. *Computer-Aided Design*, 30(4):287–300, 1998.
8. R. Howard and A. Treibergs. A reverse isoperimetric inequality, stability and extremal theorems for plane-curves with bounded curvature. *Rocky Mountain Journal of Mathematics*, 25(2):635–684, 1995.
9. H. Iwabe, Y. Fujii, K. Saito, and T. Kishinami. Study on corner cut by end mill. *International Journal of the Japan Society for Precision Engineering*, 28(3):218–223, 1994.
10. S.C. Park and Y.C. Chung. Mitered offset for profile machining. *Computer-Aided Design*, 35(5):501–505, 2003.
11. V. Pateloup, E. Duc, and P. Ray. Corner optimization for pocket machining. *International Journal of Machine Tools and Manufacture*, 44(12):1343–1353, 2004.
12. G. Pestov and V. Ionin. The largest possible circle imbedded in a given closed curve. *Doklady Akademii Nauk SSSR*, 127(6):1170–1172, 1959.
13. C.K. Yap. An $O(n \log n)$ algorithm for the voronoi diagram of a set of simple curve segments. *Discrete and Computational Geometry*, 2(1):365–393, 1987.
14. Z.Y. Zhao, C.Y. Wang, H.M. Zhou, and Z. Qin. Pocketing toolpath optimization for sharp corners. *Journal of Materials Processing Technology*, 192:175–180, 2007.