# Spiral tool paths for high-speed machining of 2D pockets with or without islands

Abrahamsen, Mikkel

*Published in:*
Journal of Computational Design and Engineering

*DOI:*
10.1016/j.jcde.2018.01.003

*Publication date:*
2019

*Document version*
Publisher's PDF, also known as Version of record

Journal of Computational Design and Engineering 6 (2019) 105–117

Contents lists available at ScienceDirect

Journal of Computational Design and Engineering

journal homepage: www.elsevier.com/locate/jcde

# Spiral tool paths for high-speed machining of 2D pockets with or without islands ☆

Mikkel Abrahamsen [1]

*Department of Computer Science, University of Copenhagen, Universitetsparken 1, DK-2100 København Ø, Denmark*

ARTICLE INFO

*Article history:*
Received 28 November 2017
Received in revised form 3 January 2018
Accepted 16 January 2018
Available online 13 July 2018

*Keywords:*
Spiral-like path
Medial axis
Smoothing
High-speed machining

ABSTRACT

We describe new methods for the construction of spiral tool paths for high-speed machining. In the simplest case, our method takes a polygon as input and a number $\delta > 0$ and returns a spiral starting at a central point in the polygon, going around towards the boundary while morphing to the shape of the polygon. The spiral consists of linear segments and circular arcs, it is $G^1$ continuous, it has no self-intersections, and the distance from each point on the spiral to each of the neighboring revolutions is at most $\delta$. Our method has the advantage over previously described methods that it is easily adjustable to the case where there is an island in the polygon to be avoided by the spiral. In that case, the spiral starts at the island and morphs the island to the outer boundary of the polygon. It is shown how to apply that method to make significantly shorter spirals in some polygons with no islands than what is obtained by conventional spiral tool paths. Finally, we show how to make a spiral in a polygon with multiple islands by connecting the islands into one island.



## 1. Introduction

A fundamental problem often arising in the CAM industry is to find a suitable tool path for milling a pocket that is defined by a shape in the plane. A CNC milling machine is programmed to follow the tool path and thus cutting a cavity with the shape of the given pocket in a solid piece of material. The cutter of the machine can be regarded as a circular disc with radius $r$, and the task is to find a tool path in the plane such that the swept volume of the disc, when the disc center is moved along the path, covers the entire pocket. We assume for simplicity that the given pocket is consisting of all the points where the center of the cutter is allowed to be so that the tool path is allowed to be anywhere in the pocket and nowhere outside.

Some work has been made on spiral tool paths that morph a point within the pocket to the boundary of the pocket (Banerjee, Feng, & Bordatchev, 2012; Bieterman & Sandstrom, 2003; Chuang & Yang, 2007; Held & Spielberger, 2009, 2014; Held & de Lorenzo, 2018; Huang, Lynn, & Kurfess, 2017; Huertas-Talón, García-Hernández, Berges-Muro, & Gella-Marín, 2014; Patel & Lalwani, 2017; Romero-Carrillo, Torres-Jimenez, Dorado, & Díaz-Garrido, 2015; Xu, Sun, & Zhang, 2013; Zhou, Zhao, & Li, 2015). The method described by Held and Spielberger (2009) yields a tool path that (i) starts at a user-specified point within the pocket, (ii) ends when the boundary is reached, (iii) makes the cutter remove all material in the pocket, (iv) has no self-intersections, (v) is $G^1$ continuous,[2] (vi) makes the width of the material cut away at most $\delta$ at any time, where $\delta$ is a user-defined constant called the *stepover*. Held and de Lorenzo (2018) simplified the method by Held and Spielberger. Note that we must have $\delta < r$, since otherwise some material might not be cut away. See Fig. 2(c) for an example of such a spiral tool path. It is the result of an algorithm described in the present paper, but has a similar appearance as the spirals described by in Held and Spielberger (2009) and Held and de Lorenzo (2018). We refer to Held and Spielberger (2009) for a detailed discussion of the benefits of spiral tool paths in high-speed machining compared to various other tool path patterns and more information on CNC milling in general.

We describe an alternative construction of spirals that also satisfy the previously mentioned properties of the construction of

Peer review under responsibility of Society for Computational Design and Engineering.

☆ A preliminary version of this paper appeared at ASME 2015 International Design Engineering Technical Conferences and Computers and Information in Engineering Conference (IDETC/CIE 2015).

[1] A large part of this work was made when the author worked for Autodesk, Inc. While preparing the paper, the author was partly supported by Mikkel Thorup's Advanced Grant DFF-0602-02499B from the Danish Council for Independent Research under the Sapere Aude research career program.

*E-mail address:* miab@di.ku.dk

[2] A plane curve is $G^1$ *continuous* or *tangent continuous* if a parameterization of the curve by arclength is differentiable.

https://doi.org/10.1016/j.jcde.2018.01.003
2288-4300/

Held and Spielberger (2009). In practice, it is very common that there is one or more islands in the pocket that should be avoided by the cutter, for instance, if there are areas of material that should not be machined to the same depth. It is only described by Held and Spielberger (2009) how to handle simply-connected pockets, i.e., there must be no islands. In their following paper (Held & Spielberger, 2014), it is described how one can handle a pocket with an island by connecting it to the boundary with a "bridge", effectively changing the topology of the pocket in order to get rid of the island. Thus, the resulting spiral morphs a point to a shape consisting of the island, the bridge, and the pocket boundary. Similarly, Patel and Lalwani (2017) describes how to partition a pocket with an island into simply-connected regions. A big advantage of our method is that it has a natural extension to pockets with one island in the sense that the spiral morphs the shape of the island to the pocket boundary, see Fig. 6(b) for an example. This is our biggest new contribution. We exploit the fact that the Voronoi diagram of a pocket with an island consists of exactly one cycle and trees rooted at that cycle to define a wave that starts at the boundary of the island at time 0 and propagates outwards, reaching the outer boundary at time 1. This wave is used to define our spiral tool path. We shall demonstrate natural applications of this method to make significantly shorter spirals for pockets with no islands than one can obtain with spirals that morphs a point to the pocket boundary, see Fig. 10 (see Fig. 11). We also show how to handle pockets with multiple islands, see Fig. 12(b).

Banerjee et al. (2012) and Romero-Carrillo et al. (2015) also described methods to compute a spiral that morphs an island to the exterior boundary of a pocket. However, their methods only work if the entire pocket boundary is visible from the island or, even stricter, if a line segment with one endpoint at the island boundary and the other one at the pocket boundary can be swept over the entire pocket area – while never crossing over any part of the exterior of the pocket or the island – by letting the endpoints traverse the boundaries monotonically forward in clockwise direction. These methods would for instance not work for a pocket with an island such as in Fig. 7. Our method works with no assumption on the geometry of the pocket and the island.

The paper is based on the author's experiences while developing the morphed spiral strategy for the Autodesk™ CAM products (HSMWorks, Inventor HSM®, and Fusion 360®). The morphed spiral seems to be quite popular among the users. Due to the abundant number of real-world parts that have been available during the development, we guarantee that it is possible to make an efficient industrial-strength implementation of the algorithms described here.

We use Held's `VRONI` library for the computation of Voronoi diagrams (Held, 2001). All figures in the paper are automatically generated using our implementation of the algorithms.

The rest of the paper is structured as follows: In Section 2, we describe our basic method for making a spiral that morphs a point to the boundary in a simply-connected pocket. Section 3 describes how the method is adapted to a pocket with one island. Using that method, we describe in Section 4 an alternative spiral in simply-connected pockets which will be superior to the one from Section 2 in many cases. In Section 5, we show how to construct at spiral around arbitrarily many islands by first connecting the islands into one island. Finally, we conclude the paper in Section 6 by suggesting some future paths of development of spiral tool paths.

## 2. Computing a spiral in a pocket without islands

In this section we describe a method to compute a spiral in a given simply-connected 2D pocket $\mathcal{P}$, see Fig. 2(c). In practice, the boundary of a pocket is often described by line segments and more advanced pieces of curves, such as circular arcs, elliptic arcs, and splines. However, it is always possible to use a sufficiently accurate linearization of the input, so we assume for simplicity that $\mathcal{P}$ is a polygon.

Our algorithm first constructs a polyline spiral, see Fig. 2(b). The polyline spiral must respect the stepover $\delta$, i.e., the distance from every point to the neighboring revolutions and the distance from the outermost revolution to the boundary of $\mathcal{P}$ is at most $\delta$. In Section 2.8 we devise a method for rounding the polyline spiral to get a $G^1$ continuous spiral consisting of line segments and circular arcs.

The corners of the polyline spiral are points on the edges of the Voronoi diagram of $\mathcal{P}$, and there is a corner at each intersection point between the spiral and the Voronoi diagram. We only consider the part of the Voronoi diagram inside $\mathcal{P}$. We have found that we get better results in practice by modifying the Voronoi diagram slightly. We describe these modifications in Section 2.7 to avoid too many technical details here. See Fig. 1 for a concrete example of the modifications we make on the Voronoi diagram. Let $\mathcal{VD} = \mathcal{VD}(\mathcal{P})$ be the modified Voronoi diagram of the pocket $\mathcal{P}$. Like the Voronoi diagram of $\mathcal{P}$, the modified diagram $\mathcal{VD}$ has the following properties which are necessary and in principle also sufficient for the computation of the spiral:

1. $\mathcal{VD}$ is a plane tree contained in $\mathcal{P}$,
2. each leaf of $\mathcal{VD}$ is on the boundary of $\mathcal{P}$,
3. there is at least one leaf of $\mathcal{VD}$ on each corner of $\mathcal{P}$,
4. all the faces into which $\mathcal{VD}$ divides $\mathcal{P}$ are convex.

### 2.1. The wave model

We imagine that a wave starts at time $t = 0$ at the point $p_0$ inside $\mathcal{P}$. The wave moves out in every direction such that at time $t = 1$, it has exactly the same shape as $\mathcal{P}$. The shape of the wave at a specific time is called a *wavefront*. The wave is growing in the sense that if $0 \leqslant t_1 \leqslant t_2 \leqslant 1$, the wavefront at time $t_1$ is contained in the wavefront at time $t_2$. We choose $p_0$ as a point in the diagram $\mathcal{VD}$ and consider $\mathcal{VD}$ as a tree rooted at $p_0$. We define the time at which the wave hits each node and the speed with which it travels on each edge in $\mathcal{VD}$. The speed of the wave is always constant or decreasing. Thus, we create a continuous map $\theta : \mathcal{VD} \mapsto [0, 1]$ that assigns a time value between 0 and 1 to each point on $\mathcal{VD}$. If $p$ is a point moving along a path on $\mathcal{VD}$ from $p_0$ to any leaf, the value $\theta(p)$ increases monotonically from 0 to 1. For each time $t \in [0, 1]$, the wavefront is a polygon inside $\mathcal{P}$ and the vertices of the wavefront are all the points $p$ on $\mathcal{VD}$ such that $\theta(p) = t$. Note that there is exactly one such point on each path from $p_0$ to a leaf of $\mathcal{VD}$ for a given $t \in [0, 1]$.

We define a time step $\Delta = 1/r$ for some integer $r$ and compute the wavefront at the times $t \in \{0, \Delta, 2\Delta, \ldots, r\Delta\}$, where $r\Delta = 1$, see Fig. 2(a). By *wavefront i*, we mean the wavefront at time $i\Delta$. We choose $r$ such that the distance from each point on wavefront $i$ to each of the wavefronts $i - 1$ and $i + 1$ is at most $\delta$ when $i > 0$ and $i < r$, respectively. For each $i = 1, \ldots, r$, we compute a revolution of the polyline spiral by interpolating between the wavefronts $i - 1$ and $i$. We describe in Sections 2.5 and 2.6 how to make the wavefronts and the interpolation such that the stepover is respected between neighboring revolutions.

### 2.2. Choosing the starting point $p_0$ and the number of revolutions of the spiral

In order to get a spiral with small length, we try to minimize the number of revolutions. Consider the longest path from $p_0$ to a leaf in $\mathcal{VD}$. The length of a path is the sum of edge lengths on the path. If $h$ is the length of the longest path, then $\lceil h/\delta \rceil + 1$ wavefronts are

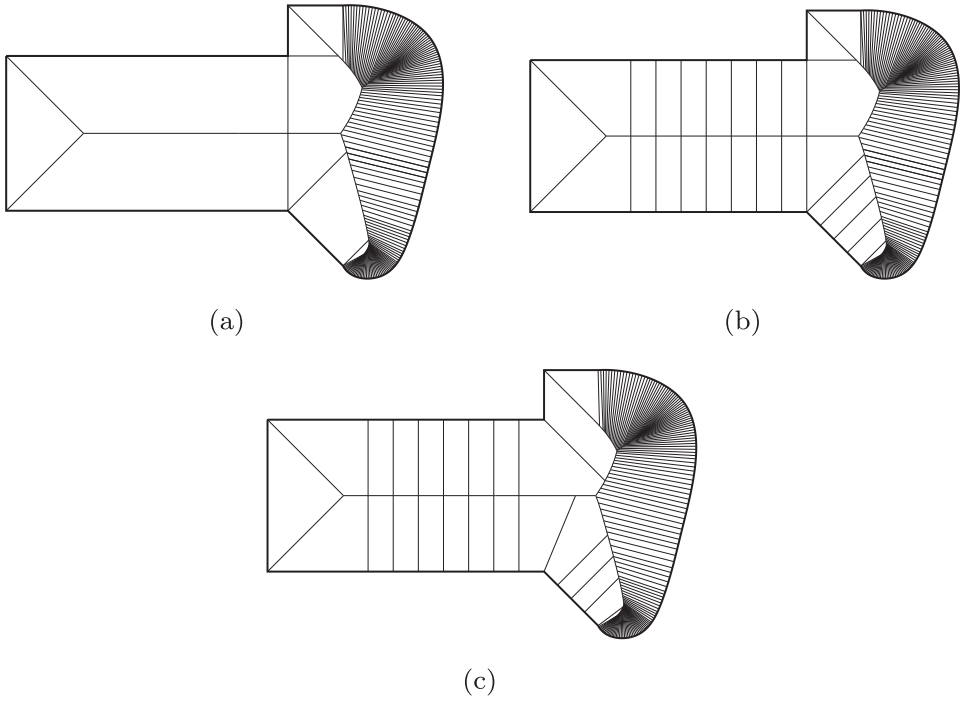



**Fig. 1.** (a) The Voronoi diagram. (b) The Voronoi diagram enriched with equidistantly placed segments perpendicular to long edges. (c) The final diagram $\mathcal{VD}$ where double edges going to concave corners of $\mathcal{P}$ are replaced by their angle bisector.

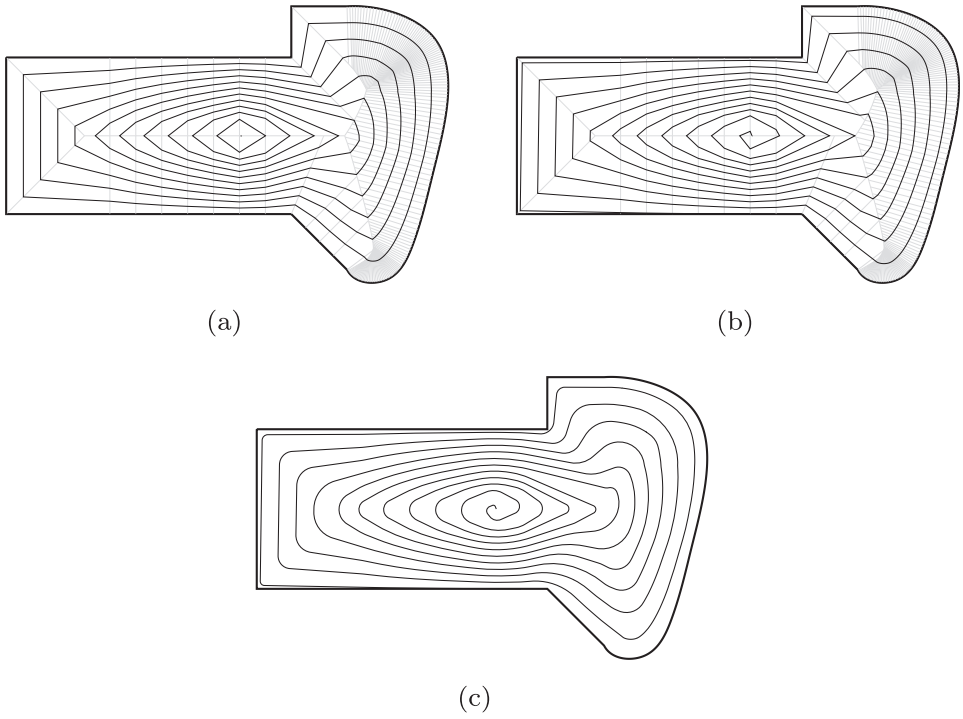



**Fig. 2.** Wavefronts (a), polyline spiral (b), and the final $G^1$ continuous spiral (c) in the same polygon $\mathcal{P}$. The diagram $\mathcal{VD}$ is in gray.

necessary and sufficient for the stepover to be respected among all neighboring wavefronts. Therefore, we choose $p_0$ as the point in $\mathcal{VD}$ that minimizes the longest distance to a leaf in $\mathcal{VD}$. That is a unique point traditionally known as the *center* of $\mathcal{VD}$. Handler (1973) gives a simple algorithm to compute $p_0$ in time proportional to the size of $\mathcal{VD}$. The center will most likely not be a node in $\mathcal{VD}$, but an interior point on some edge. In that case, we introduce a node at $p_0$ by splitting the edge into two edges.

### 2.3. Our representation of $\mathcal{VD}$

We consider $\mathcal{VD}$ as a directed, rooted tree with the node `Root` at $p_0$ being the root. For a node $n$ in $\mathcal{VD}$, we let $\texttt{Pt}[n]$ be the position of the node $n$. Let $\mathcal{VD}[n]$ be the subtree rooted at node $n$. For a node $n \neq \texttt{Root}$, we store a pointer $\texttt{ParentE}[n]$ to the edge having end node $n$. We say that edge $\texttt{ParentE}[n]$ is the *parent edge* of node $n$ and of any edge having start node $n$. We also store an array

ChildEs[$n$] of the edges going out of $n$ sorted in counterclockwise order with the edge following ParentE[$n$] being the first. For Root, the choice of the first child edge does not matter. For each edge $e$, we store pointers Start[$e$] and End[$e$] to the start and end nodes of $e$. We also store an index $i = \texttt{IndexInStart}[e]$ such that $\texttt{ChildEs}[\texttt{Start}[e]][i] = e$. If $e$ is an edge, we say that Start[$e$] and End[$e$] are *incident* to $e$ and that $e$ is *incident* to Start[$e$] and End[$e$]. For an edge $e_1$ and node $n$ incident to $e_1$, we let $\texttt{NextCCW}(e_1, n) = e_2$, where $e_2$ is the edge after $e_1$ among the edges incident to $n$ in counterclockwise order. The function NextCCW can be implemented so that it runs in constant time using the values defined here.

Using NextCCW, we can traverse all of $\mathcal{VD}$ in the counterclockwise direction in linear time. We start setting $(n, e) = (\texttt{Root}, \texttt{ChildEs}[\texttt{Root}][0])$. In each iteration, we let $n$ be the other node incident to $e$ and then set $e = \texttt{NextCCW}(e, n)$. We stop when we have traversed every edge, i.e., when $(e, n) = (\texttt{Root}, \texttt{ChildEs}[\texttt{Root}][0])$ at the end of an iteration. Note that each edge $e$ is visited twice, once going down the tree $\mathcal{VD}[\texttt{Start}[e]]$ and once going up.

### 2.4. Defining the movement of the wave

In the following, we outline the model after which we define the movement of the wave. Let Hgt[$n$] for each node $n$ be the length of the longest path from $n$ to a leaf in $\mathcal{VD}[n]$. All the Hgt values can be computed in linear time by traversing $\mathcal{VD}$ once. For each node $n$, we define the time TmNd[$n$] where the wave reaches $n$. We set $\texttt{TmNd}[\texttt{Root}] = 0$. We also define the speed VeNd[$n$] that the wave has when it reaches $n$. We set $\texttt{VeNd}[\texttt{Root}] = \texttt{Hgt}[\texttt{Root}]$. The wave starts at the root at time $t = 0$ and travels with constant speed VeNd[Root] on the paths to the farthest leafs in $\mathcal{VD}$. (Due to our choice of the starting point $p_0$, there will always be at least two paths from $p_0$ to a leaf with the maximum length.) Hence, it reaches those leafs at time $t = 1$. On all the shorter paths, we make the wave slow down so that it reaches every leaf at time $t = 1$. Generally, when the speed needs to be decreased along some path, we let the speed be a piecewise linear function of time, so that the first 25% of the path is used to slow down and the speed is thereafter constant.

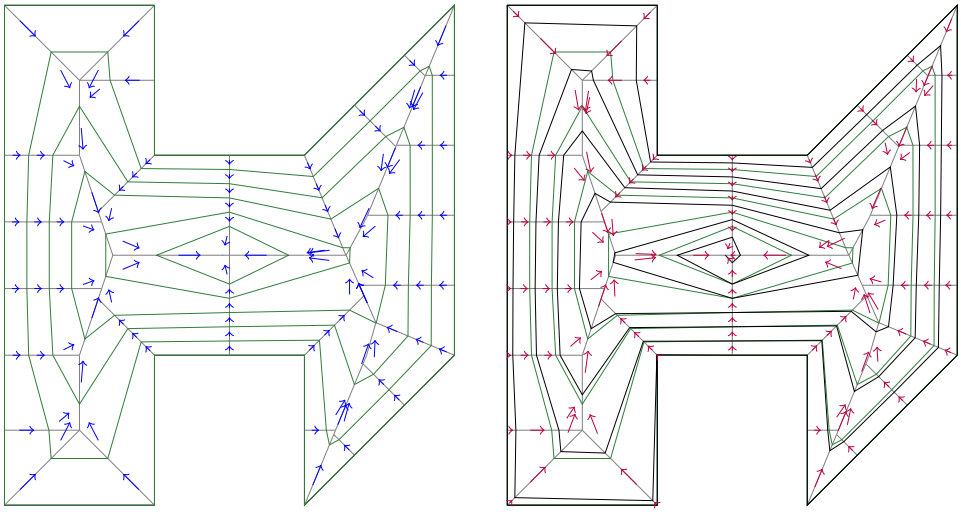
When the movement of the wave is defined, we may define $\texttt{GetPt}(e, t)$ as the point on edge $e$ with $\theta(\texttt{GetPt}(e, t)) = t$, where $\texttt{TmNd}[\texttt{Start}[e]] \leqslant t \leqslant \texttt{TmNd}[\texttt{End}[e]]$.

### 2.5. Constructing the wavefronts

We make a spiral with $r = \lceil \frac{\texttt{Hgt}[\texttt{Root}]}{\delta'} \rceil$ revolutions, where $\delta' = 0.95 \cdot \delta$. The number 0.95 is a hyper-parameter that we have found to work well in practice. We use the slightly smaller stepover $\delta'$ so that the maximum distance between two neighboring revolutions is smaller than $\delta$. That gives more flexibility to smooth the spiral later on as described in Section 2.8. Using a number closer to 0 instead of 0.95 results in more revolutions and thus a longer spiral, while a number closer to 1 increases the curvature of the resulting spiral, since there will not be as much freedom to smooth the spiral with circular arcs and at the same time respecting the stepover. We set $\Delta = 1/r$ and compute a wavefront for each of the times $\{0, \Delta, 2\Delta, \ldots, r\Delta\}$. The two-dimensional array Wf stores the wavefronts so that the wavefront at time $i\Delta$ is the array Wf[$i$]. Wavefront $i$ is constructed by traversing $\mathcal{VD}$ once and finding every point on $\mathcal{VD}$ with time $i\Delta$ in counterclockwise order. Let $e$ be an edge we have not visited before, and let $n = \texttt{Start}[e]$ and $m = \texttt{End}[e]$. There is a corner of wavefront $i$ on $e$ if $\texttt{TmNd}[n] < i\Delta \leqslant \texttt{TmNd}[m]$. If that is the case, we add $\texttt{GetPt}(e, i\Delta)$ to the end of Wf[$i$]. We also make one corner in Wf[0] for each of the child edges of the root, ChildEs[Root], and all these corners are copies of the point Pt[Root]. Using this construction, there is exactly one corner of each wavefront on each path from Root to a leaf of $\mathcal{VD}$.

For each corner Wf[$i$][$w$], we store the length of the part of the wavefront up to the corner, i.e. $\texttt{WfLng}[i][0] = 0$ and $\texttt{WfLng}[i][w] = \sum_{j=1}^{w} \|\texttt{Wf}[i][j] - \texttt{Wf}[i][j-1]\|$, for $w \geqslant 1$. Here $\|\cdot\|$ is the Euclidean norm. We also store the total length of Wf[$i$] as TtlWfLng[$i$].

We introduce a rooted tree with the wavefront corners as the nodes. See Fig. 3(a). The parent of a corner $\texttt{Wf}[i][w], i > 0$, is the unique corner $\texttt{Wf}[i-1][pw]$ on wavefront $i-1$ on the path from Wf[$i$][$w$] to Root. We store $pw$ as PaWf[$i$][$w$], i.e., the parent of Wf[$i$][$w$] is $\texttt{Wf}[i-1][\texttt{PaWf}[i][w]]$.



(a) (b)

**Fig. 3.** The construction of a polyline spiral in a polygon $\mathcal{P}$: (a) The wavefronts in green and blue arrows from each wavefront corner $\texttt{Wf}[i][w]$ to its parent $\texttt{Wf}[i-1][\texttt{PaWf}[i][w]]$. The diagram $\mathcal{VD}$ is in gray. (b) The polyline spiral in black obtained by interpolating between the wavefronts. The purple arrows are from each corner Sp[$i$] of the spiral to its parent Sp[Pa[$i$]].

Since the distance from each wavefront corner to each of its children is at most $\delta'$, we get that the distance from a point on one wavefront to the neighboring wavefronts is at most $\delta'$. Furthermore, since the wave is moving with positive speed towards the leafs of $\mathcal{VD}$ and the faces into which $\mathcal{VD}$ divides $\mathcal{P}$ are convex, neighboring wavefronts do not intersect each other. From the order in which the corners of a wavefront are constructed, it is also clear that a wavefront does not intersect itself.

### 2.6. Interpolating between the wavefronts

We construct a polyline spiral stored as an array $\texttt{Sp}$. For each $i = 1, \ldots, r$, we construct one revolution of the spiral by interpolating between wavefront $i-1$ and wavefront $i$. Every corner of the spiral is a point on $\mathcal{VD}$. There is exactly one spiral corner on the path in $\mathcal{VD}$ from each wavefront corner $\texttt{Wf}[i][w]$ to its parent $\texttt{Wf}[i-1][\texttt{PaWf}[i][w]]$. Assume for now that we know how to choose the actual corners of the polyline spiral. We shall get back to this shortly.

The first corner $\texttt{Sp}[0]$ is on the root node of $\mathcal{VD}$, and for every other corner $\texttt{Sp}[s], s > 0$, we store a parent index $\texttt{Pa}[s]$, such that the parent $\texttt{Sp}[\texttt{Pa}[s]]$ is the first corner we meet on the path in $\mathcal{VD}$ from $\texttt{Sp}[s]$ to the root. Fig. 3(b) shows the resulting polyline spiral and the parents of each corner. We define the spiral such that the distance between a spiral corner and its parent is at most $\delta'$. It follows that the distance from a point on the polyline spiral to the neighboring revolutions is at most $\delta'$.

Here we describe how to define the $\texttt{Pa}$-pointers. When we have constructed a spiral corner $\texttt{Sp}[s]$ which is on the path from $\texttt{Wf}[i][w]$ to its parent wavefront corner $\texttt{Wf}[i-1][pw]$, we know that the first spiral corner on the path from $\texttt{Wf}[i][w]$ to the root is $\texttt{Sp}[s]$, and we store this information as $\texttt{PaSp}[i][w] = s$. Therefore, when we have made a new spiral corner $\texttt{Sp}[r]$ on the path from $\texttt{Wf}[i+1][w']$ to its parent $\texttt{Wf}[i][pw']$, the parent of $\texttt{Sp}[r]$ is defined to be $\texttt{Pa}[r] = \texttt{PaSp}[i][pw']$. By doing so, the corner $\texttt{Sp}[\texttt{Pa}[r]]$ will be the first spiral corner on the path from $\texttt{Sp}[r]$ to the root.

In the following we describe how to choose the corners of the polyline spiral. We assume that we have finished the revolution of the spiral between wavefronts $i-2$ and $i-1$ and we show how to make the revolution between wavefronts $i-1$ and $i$. While constructing the revolution between wavefronts $i-2$ and $i-1$, we have stored for each wavefront corner $\texttt{Wf}[i-1][v]$ the index

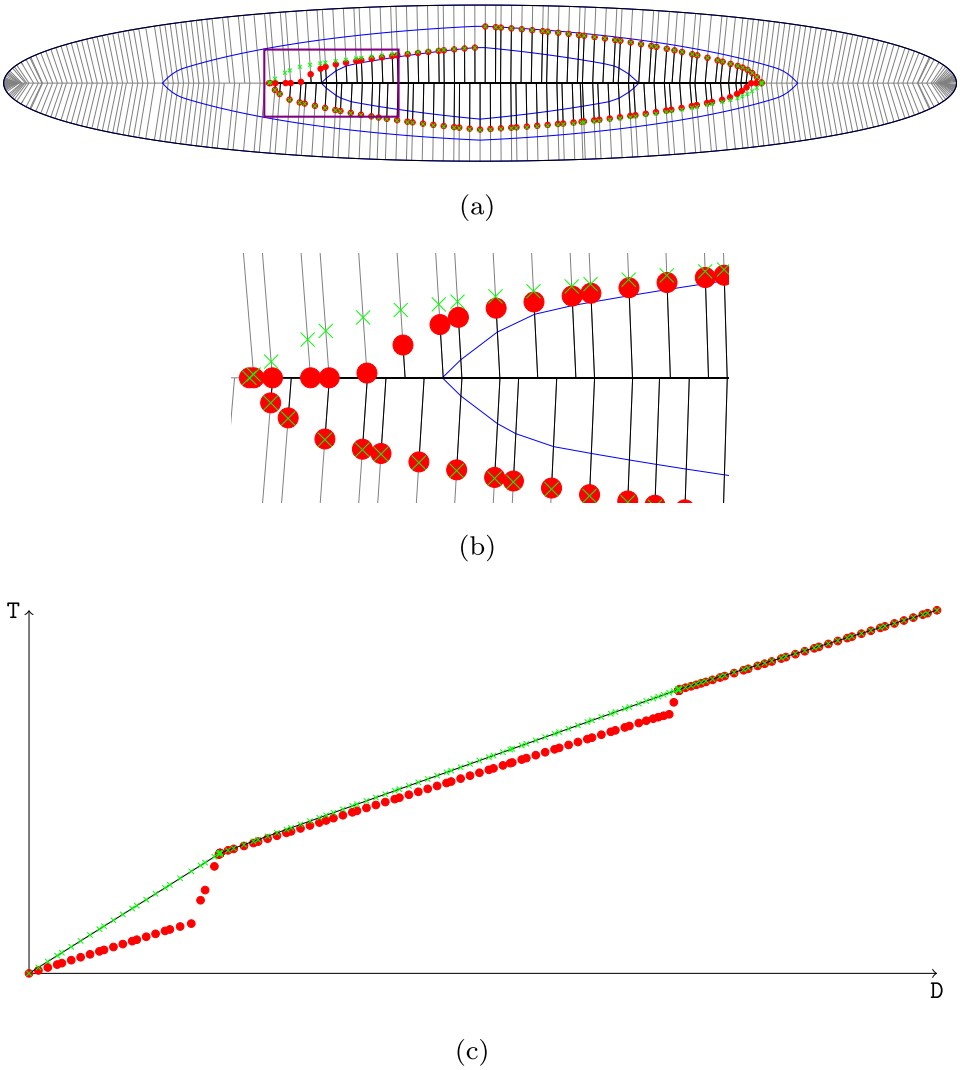



**Fig. 4.** (a) The interpolation between the two blue wavefronts. The marked part of the diagram $\mathcal{VD}$ is in black, the rest is in gray. The red circles are the points $\texttt{Q}[w]$. The green crosses are the resulting points of the polyline spiral stored in $\texttt{Sp}$ after the convexification process. The violet box contains the detail shown in (b). (c) Related values for the same interpolation: The points $(\texttt{D}[w], \texttt{T}[w])$ are red circles. The upper convex hull $\mathcal{F}$ of the points is a black curve, and the green crosses are the points $(\texttt{D}[w], \mathcal{F}(\texttt{D}[w]))$ on that hull.

$ps = \mathtt{PaSp}[i-1][v]$ such that $\mathtt{Sp}[ps]$ is the unique spiral corner on the path from $\mathtt{Wf}[i-1][v]$ to its parent $\mathtt{Wf}[i-2][\mathtt{PaWf}[i-1][v]]$. Now, for each wavefront corner $\mathtt{Wf}[i][w]$, we find the point $\mathtt{Q}[w]$ on the path to $\mathtt{Wf}[i-1][pw]$, where $pw = \mathtt{PaWf}[i][w]$, with time $t_w = (i-1)\Delta + \frac{\mathtt{WfLng}[i][w]}{\mathtt{TtlWfLng}[i]}\Delta$. If $\mathtt{Q}[w]$ is more than $\delta'$ away from $\mathtt{Sp}[\mathtt{PaSp}[i-1][pw]]$, we redefine $\mathtt{Q}[w]$ to be the point on the same path with distance exactly $\delta'$. We mark the path from $\mathtt{Q}[w]$ to the root of $\mathcal{VD}$. See Fig. 4.

When we have done the marking for each $w$, we traverse wavefront $i$ once more. For each wavefront corner $\mathtt{Wf}[i][w]$, we find the first marked point on the path to the root. We let $\mathtt{P}[w]$ be that point and $\mathtt{T}[w] = \theta(\mathtt{P}[w])$ be its time. We have that $\mathtt{T}[w] \geqslant t_w$, because a later wavefront corner $\mathtt{Wf}[i][w'], w' > w$, can mark more of the path from $\mathtt{Wf}[i][w]$ to the root. Therefore, we can have $\mathtt{P}[w] = \mathtt{P}[w+1]$ for some $w$. Using this construction, there is exactly one distinct $\mathtt{P}$-point on each path from a wavefront corner to the root. Furthermore, we know that the distance from $\mathtt{P}[w]$ to the spiral corner $\mathtt{Sp}[\mathtt{PaSp}[i-1][\mathtt{PaWf}[i][w]]]$ is at most $\delta'$.

The polyline defined by the points $\mathtt{P}[0], \mathtt{P}[1], \ldots$ is basically our interpolated spiral, but the points have a tendency to have unnecessarily sharp corners if $\mathcal{VD}$ is relatively dense, which is often the case for polygons $\mathcal{P}$ occurring in real-world problems. To avoid such corners, we apply a method which we denote as the *convexification*, see Fig. 4. Let $\mathtt{D}[w] = \sum_{v=0}^{w-1} \|\mathtt{P}[v] - \mathtt{P}[v+1]\|$ be the length of the polyline until $\mathtt{P}[w]$ and consider the points $(\mathtt{D}[w], \mathtt{T}[w])$. We compute the upper convex hull of these points, e.g. using the method of Graham and Yao (1983). Let $\mathcal{F}$ be the function whose graph is the upper hull. By definition, we have that $\mathtt{T}[w] \leqslant \mathcal{F}(\mathtt{D}[w])$ for each $w = 0, \ldots$. We now choose the corners of $\mathtt{Sp}$ in the following way: For each wavefront corner $\mathtt{Wf}[i][w]$ in order, we find the point $S$ on the path to the root with time $\mathcal{F}(\mathtt{D}[w])$. If $S$ is more than $\delta'$ away from the parent spiral corner $\mathtt{Sp}[\mathtt{PaSp}[i-1][\mathtt{PaWf}[i][w]]]$ (which will be the parent of the spiral corner we are constructing), we choose instead $S$ to be the point on the same path which is exactly $\delta'$ away. To avoid repetitions of the same point in $\mathtt{Sp}$, we add $S$ to the end of $\mathtt{Sp}$ if $S$ is different from the last point in $\mathtt{Sp}$. Since we get the spiral corners by moving the $\mathtt{P}$-points closer to wavefront $i$, we get exactly one distinct spiral corner on each path from a wavefront corner to the root. When $\mathcal{VD}$ is sparse like in Fig. 3, the convexification makes no visible difference between the $\mathtt{P}$-points and the final points in $\mathtt{Sp}$, but when $\mathcal{VD}$ is dense as in Fig. 4, the effect is significant.

We also add one revolution around $\mathcal{P}$ to the end of $\mathtt{Sp}$, which is used to test that the last interpolated revolution respects the stepover when the spiral is rounded later on.

The polyline spiral constructed as described clearly satisfies that the distance from a point on one revolution to the neighboring revolutions is at most $\delta'$. Furthermore, each revolution is between two neighboring wavefronts since all the corners of the interpolation between wavefronts $i$ and $i+1$ have times in the interval $[i\Delta, (i+1)\Delta[$. The wavefronts do not intersect as mentioned earlier, so different revolutions of the polyline spiral do not either. It is also clear from the construction that one revolution does not intersect itself. Therefore, the polyline spiral has all the properties that we require of our spiral except being $G^1$ continuous. How to obtain that is described in Section 2.8.

### 2.7. Modifying the Voronoi diagram

In this section, we outline some modifications we make on the Voronoi diagram of $\mathcal{P}$ before doing anything else. The result is the diagram $\mathcal{VD} = \mathcal{VD}(\mathcal{P})$. An example of the process can be seen in Fig. 1.

Long edges on $\mathcal{P}$ lead to long faces in the Voronoi diagram so that the wave is not moving towards the boundary of $\mathcal{P}$ in a natural way. It may result in degenerate one-dimensional wavefronts. Therefore, we add extra edges from long edges of the Voronoi diagram to the boundary in the following sense. Let $l_1$ and $l_2$ be two leafs of $\mathcal{VD}$ which are neighbors on $\partial\mathcal{P}$. Such a pair of nodes are on the same or on two neighboring corners of $\mathcal{P}$. Assume the latter, so that there is a segment $S$ on $\partial\mathcal{P}$ from $l_1$ to $l_2$ and a face $f$ of the Voronoi diagram to the left of $S$. Let $s = \mathtt{Pt}[l_2] - \mathtt{Pt}[l_1]$ be the vector from $l_1$ to $l_2$, $d = \|s\|$ be the length of $S$ and $m = \lceil d/\delta \rceil$. We want to subdivide $f$ into $m$ faces. Let $p_i = \mathtt{Pt}[l_1] + s \cdot \frac{i}{m}, i = 1, \ldots, m-1$, be interpolated points on $S$. Let $h_i = \overrightarrow{p_i, p_i + \hat{s}}$ be the half-line starting at $p_i$ with direction $\hat{s}$, where $\hat{s}$ is the counterclockwise rotation of $s$. For each $i = 1, \ldots, m-1$, we find the first intersection point between $h_i$ and the Voronoi diagram. Assume the intersection for some $i$ is a point $q$ on an edge $e$. If the smallest angle between $h_i$ and $e$ is larger than 50°, we split $e$ into two edges by introducing a node at $q$ and add a segment from that node to a new node at $p_i$. If the smallest angle is less than 50°, the Voronoi diagram is moving fast enough towards the boundary so that the wavefronts will be fine in that area without adding any additional edges.

The other modification we perform has to do with the *concave* corners of $\mathcal{P}$, which are the corners where the inner angle is more than 180°. Each concave corner $c$ on $\mathcal{P}$ leads to a face in the Voronoi diagram of all the points in $\mathcal{P}$ being closer to $c$ than to anything else on the boundary of $\mathcal{P}$. Therefore, there are two edges $e_1$ and $e_2$ of the Voronoi diagram with an endpoint on $c$. We have found that we get a better spiral if we remove these edges and instead add an edge following the angle bisector of the edges, i.e., we follow the bisector from $c$ and find the first intersection point $q$ with the Voronoi diagram and add an edge from $q$ to $c$. The reason that this process improves the resulting spiral is that the wavefronts will resemble $\mathcal{P}$ more because they will have just one corner corresponding to the corner $c$ (on the bisector edge on $\mathcal{P}$), instead of two (one on each of $e_1$ and $e_2$). We can only do this manipulation if the resulting faces are also convex. That is checked easily by computing the new angles of the manipulated faces and it seems to be the case almost always.

### 2.8. Rounding the polyline spiral

In this section, we describe a possible way for smoothing the polyline spiral to get a $G^1$ continuous spiral. See Fig. 5 for a comparison between the rounded and unrounded spiral from Fig. 2 using this method. Note that some of the arcs of the rounded spiral round multiple corners of the polyline spiral.

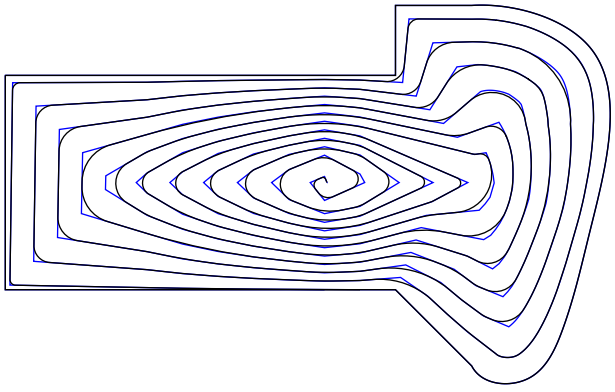


**Fig. 5.** The spirals from Fig. 2 together. The polyline spiral is blue and the rounded spiral is black.

For each corner on the polyline spiral, we substitute a part of the spiral containing the corner with a circular arc which is tangential to the polyline spiral in the endpoints. That gives a spiral which is $G^1$ continuous, i.e., having no sharp corners. For each index $i$, let $s_i = \mathtt{Sp}[i]\mathtt{Sp}[i+1]$ be the segment from $\mathtt{Sp}[i]$ to $\mathtt{Sp}[i+1]$ and $v_i = \mathtt{Sp}[i+1] - \mathtt{Sp}[i]$ be the vector from $\mathtt{Sp}[i]$ to $\mathtt{Sp}[i+1]$. Each arc has the start point $p$ on some segment $s_a$ and the endpoint $q$ on another segment $s_b, a < b$, so that the arc *substitutes* the part of the polyline spiral from $p$ to $q$. We say that the arc *rounds* the corners $a+1$ to $b$. We call the arc *tangential* if it is counterclockwise and its center is on the intersection of the half-lines $\overrightarrow{p, p+\widehat{v_s}}$ and $\overrightarrow{q, q+\widehat{v_r}}$ or it is clockwise and its center is on the intersection of the half-lines $\overrightarrow{p, p-\widehat{v_s}}$ and $\overrightarrow{q, q-\widehat{v_r}}$. It follows that if each corner is substituted by a tangential arc with positive radius, the resulting spiral is $G^1$ continuous.

We store a pointer $\mathtt{Arc}[i]$ to the arc that substitutes the corner $\mathtt{Sp}[i]$. The same arc can substitute multiple consecutive corners, so that $\mathtt{Arc}[i] = \mathtt{Arc}[i+1] = \ldots = \mathtt{Arc}[i+k-1]$. In that case, when $\mathtt{Arc}[i-1] \neq \mathtt{Arc}[i] \neq \mathtt{Arc}[i+k]$, the *neighbors* of $\mathtt{Arc}[i]$ are $\mathtt{Arc}[i-1]$ and $\mathtt{Arc}[i+k]$. Two different arcs must substitute disjoint parts of the polyline spiral for the rounded spiral to be well-defined. We subdivide each segment $s_a$ at a point $p_a \in s_a$ such that an arc ending at $s_a$ must have its endpoint at the segment $\mathtt{Sp}[a]p_a$ and an arc beginning at $s_a$ must have its startpoint on the segment $p_a\mathtt{Sp}[a+1]$. The point $p_a$ is chosen as a weighted average of $\mathtt{Sp}[a]$ and $\mathtt{Sp}[a+1]$ so that the arc rounding the sharpest of the corners $\mathtt{Sp}[a]$ and $\mathtt{Sp}[a+1]$ gets most space. Let $\phi_a \in (-\pi, \pi]$ be the angle at corner $\mathtt{Sp}[a]$ of the polyline spiral. We set $w_a = \frac{\pi - \phi_a}{2\pi - \phi_a - \phi_{a+1}}$ and choose $p_a$ as $p_a = (1 - w_a) \cdot \mathtt{Sp}[a] + w_a \cdot \mathtt{Sp}[a+1]$.

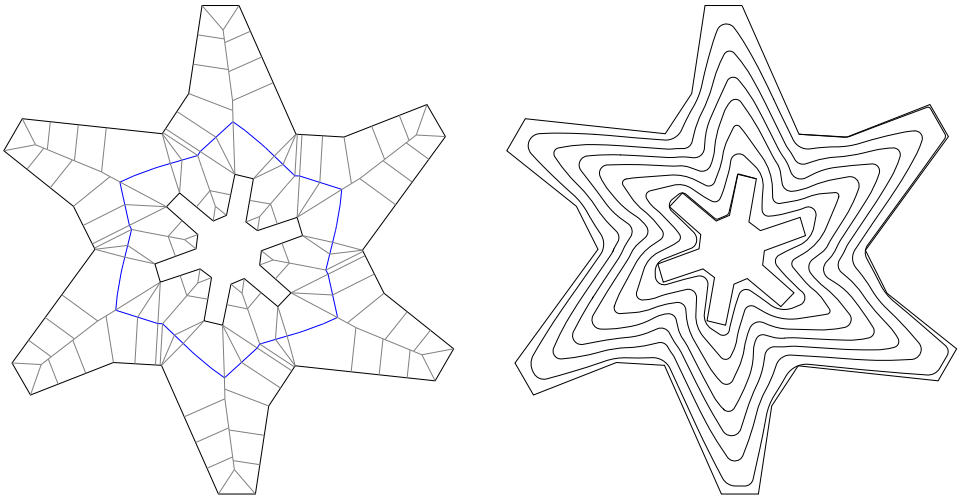
We keep a priority queue $Q$ (Cormen, Leiserson, Rivest, & Stein, 2009) of the arcs that can possibly be enlarged. After each enlargement of an arc, the resulting spiral respects the stepover $\delta$ and no self-intersections have been introduced. We say that an arc with these properties is *usable*. Initially, we let each corner be rounded by a degenerated zero-radius arc, and $Q$ contains all these arcs. Clearly, these initial zero-radius arcs are usable by definition. We consider the front arc $A$ in $Q$ and try to find another usable arc $A'$ that substitutes a longer chain of the polyline spiral. The new arc $A'$ normally has a larger radius than $A$. If possible, we choose $A'$ so that it also substitutes one or, preferably, two of the neighbors of $A$. Assume that we succeed in making a usable arc $A'$ substituting both $A$ and its neighbors $B$ and $C$. Now $A'$ rounds the union of the corners previously rounded by $A, B$, and $C$. For all these corners, we update the $\mathtt{Arc}$-pointers and we remove $A, B$, and $C$ from $Q$. We add $A'$ to $Q$. For every corner that $A'$ rounds, we also add the arcs rounding the children and parents of that corner to $Q$, since it is possible that these arcs can now be enlarged. If no larger usable arc $A'$ is found, we just remove $A$ from $Q$. The rounding process terminates when $Q$ is empty.

We test if an arc is usable by measuring the distance to the arcs rounding the child and parent corners. That is easily done using elementary geometric computations.

Here we describe how to find the largest usable arc rounding the corners $a+1$ to $b, a < b$. We find the possible radii of tangential arcs beginning at a point on $p_a\mathtt{Sp}[a+1]$ and ending at a point on $\mathtt{Sp}[b]p_b$ using elementary geometry. There might be no such arcs, in which case we give up finding an arc rounding these corners. Otherwise, we have an interval $[r_{\min}, r_{\max}]$ of radii of tangential arcs going from $p_a\mathtt{Sp}[a+1]$ and ending on $\mathtt{Sp}[b]p_b$. We check if the arc with radius $r_{\min}$ is usable. If it is not, we give up. Otherwise, we check if the arc with radius $r_{\max}$ is usable. If it is, we use it. Otherwise, we make a binary search in the interval $[r_{\min}, r_{\max}]$ after the usable arc with the largest radius. We stop when the binary search interval has become sufficiently small, for instance $0.01 \cdot \delta$, and use the arc with the smallest number in the interval as its radius.

The order of the arcs in $Q$ is established in the following way: We have found that giving the arc $A$ the priority $P(A) = r(A)/r(C_{\max}) + 1/s(A)$ gives good results, where $r(A)$ is the radius of $A$, $s(A)$ is the size of the subtended angle from the center of $A$ in radians, and $r(C_{\max})$ is the radius of the maximum circle contained in $\mathcal{P}$. The front arc in $Q$ is the one with the smallest $P$-value. We divide by $r(C_{\max})$ to make the rounding invariant when $\mathcal{P}$ and $\delta$ are scaled by the same number. $r(C_{\max})$ can be obtained from the Voronoi diagram of $\mathcal{P}$, since the largest inscribed circle has its center on a node in the diagram. If $s(A) = 0$, we set $P(A) = \infty$, since there is no corner to round.

We see that arcs with small radii or large subtended angles are chosen first for enlargement. In the beginning when all the arcs in $Q$ have zero radius, the arcs in the sharpest corners are chosen first because their degenerated arcs have bigger subtended angles –



(a) (b)

**Fig. 6.** (a) A polygon $\mathcal{P}$ with an island $\mathcal{I}$, both in black. The diagram $\mathcal{VD}$ of $\mathcal{P} \setminus \mathcal{I}$ is drawn with the cycle $\mathcal{C}$ in blue and the other edges in gray. (b) The resulting spiral.

even though the radius is zero, we can still define the start and end angle of the arc according to the slope of the segments meeting in the corner and thus define the subtended angle of the arc.

If two or three arcs are substituted by one larger arc each time we succeed in making a larger arc, we are sure that the rounding process does terminate, since the complexity of the spiral decreases. However, it is often not possible to merge two or three arcs, but only to make a larger arc rounding the same corners as an old one. The rounded spiral gets better, but we cannot prove that the process terminates. In practice, we have seen fast termination in any tested example. A possible remedy could be only to allow each arc to increase in size without rounding more corners a fixed number of times.

## 3. Computing a spiral in a pocket with an island

In many practical applications, the area to be machined is not simply-connected but has one or more "islands" that should not be machined. It might be because there are physical holes in the part or areas of a thicker layer of material not to be machined in the same depth. Therefore, assume that we are given a polygon $\mathcal{P}$ and a single polygonal island $\mathcal{I}$ in the interior of $\mathcal{P}$. In Section 5 we suggest a method to deal with multiple islands. By $\mathcal{P} \setminus \mathcal{I}$, we denote the closed set of points which are in the interior or on the boundary of $\mathcal{P}$ but not in the interior of $\mathcal{I}$. We want to compute a spiral which is contained in $\mathcal{P} \setminus \mathcal{I}$ such that the user-defined step-over $\delta$ is respected between (i) two consecutive revolutions, (ii) the boundary of $\mathcal{I}$ and the first revolution, and (iii) the boundary of $\mathcal{P}$ and the last revolution. We also require that the spiral is $G^1$ continuous and has no self-intersections. See Fig. 6(b) for an example.

As in the case with a simply-connected pocket, we use a wave model to construct the spiral. We imagine a wave that has exactly the shape of $\mathcal{I}$ at time 0 and moves towards the boundary of $\mathcal{P}$, so that at the time 1, it has the shape of $\mathcal{P}$. We explain how to define a polyline spiral. The spiral should be rounded by a method similar to the one described in Section 2.8.

### 3.1. The Voronoi diagram of a pocket with an island

We use the Voronoi diagram of the set of line segments of $\mathcal{P}$ and $\mathcal{I}$. As in the case with no islands, we modify the diagram slightly. Let $\mathcal{VD} = \mathcal{VD}(\mathcal{P} \setminus \mathcal{I})$ be the modified polygon. Like the true Voronoi diagram, the modified diagram $\mathcal{VD}$ has the following properties:

1. $\mathcal{VD}$ is a connected, plane graph contained in $\mathcal{P} \setminus \mathcal{I}$,
2. each leaf of $\mathcal{VD}$ is on the boundary of $\mathcal{P}$ or $\mathcal{I}$,
3. there is at least one leaf of $\mathcal{VD}$ on each corner of $\mathcal{P}$ and $\mathcal{I}$,
4. all the faces into which $\mathcal{VD}$ divides $\mathcal{P} \setminus \mathcal{I}$ are convex,
5. $\mathcal{VD}$ contains exactly one cycle, the cycle is the locus of all points being equally close to the boundaries of $\mathcal{I}$ and $\mathcal{P}$, and $\mathcal{I}$ is contained in its interior.

The diagram $\mathcal{VD}$ is consisting of one cycle $\mathcal{C}$, which we also denote by the *central cycle*, and some trees growing out from $\mathcal{C}$, see Fig. 6(a). Each of the trees grows either outwards and has all its leaves on the boundary of $\mathcal{P}$ or inwards and has all its leaves on the boundary of $\mathcal{I}$. The general idea is to use the method described in Section 2 to define wavefronts in each of these trees separately. Afterward, we interpolate between the neighboring wavefronts in each tree and connect the interpolated pieces to get one contiguous spiral.

As in the case of a polygon without an island, we enrich the Voronoi diagram and remove double edges going to concave corners as described in Section 2.7.

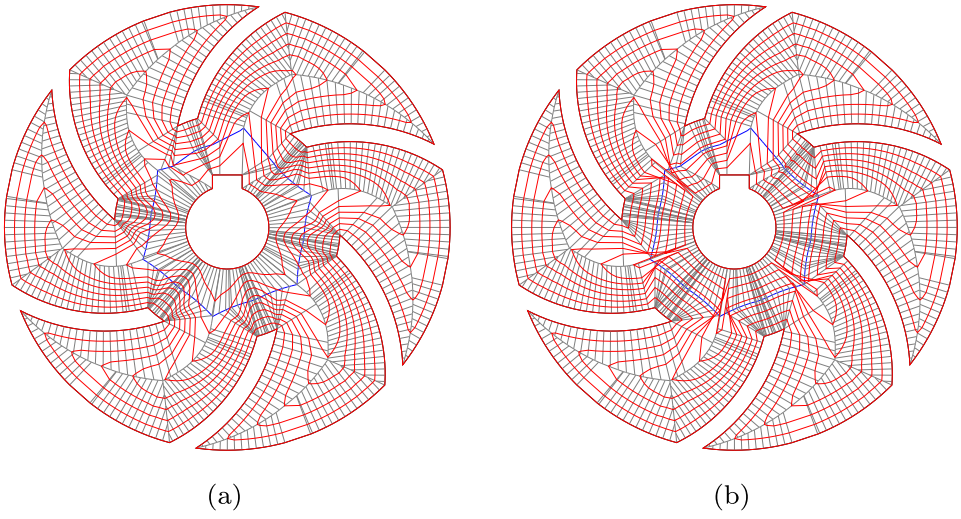
We want the trees to be *symmetric* in the sense that there is a tree $\mathcal{PT}_n$ with root $n \in \mathcal{C}$ and leafs on the boundary of $\mathcal{P}$ if and only if there is a tree $\mathcal{IT}_n$ with root $n$ and leaves on the boundary of $\mathcal{I}$. If for a node $n \in \mathcal{C}$ we only have one of the trees, say $\mathcal{IT}_n$, we add an edge from node $n$ to the closest point on the boundary of $\mathcal{P}$ and let $\mathcal{PT}_n$ be the tree consisting of that single edge. It follows from the properties of the Voronoi diagram that the added edge does not intersect any of the other edges.

We store $\mathcal{C}$ as a vector $[n_0, \ldots, n_{c-1}]$ of the nodes on $\mathcal{C}$ in counterclockwise order, such that there are trees $\mathcal{IT}_{n_i}$ and $\mathcal{PT}_{n_i}$ for each $i = 0, \ldots, c-1$. A *root node* is a node $n$ on $\mathcal{C}$. We let $\mathcal{T}_n = \mathcal{PT}_n \cup \mathcal{IT}_n$ be the union of the two trees rooted at node $n \in \mathcal{C}$ and consider $\mathcal{T}_n$ as a tree rooted at node $n$.

### 3.2. Defining the movement of the wave

On each tree $\mathcal{T}_n$ we now define a wave model similar to the one described in Section 2.1. The wave starts at time $t = 0$ on the leafs on the boundary of $\mathcal{I}$ and moves through $\mathcal{T}_n$ so that it hits the leafs on the boundary of $\mathcal{P}$ at time $t = 1$. We compute the wavefronts from $n$ and towards $\mathcal{I}$ and $\mathcal{P}$.

Let the *preferred time* of a root node $n$ be $t_n = \frac{\texttt{IslHgt}[n]}{\texttt{BndHgt}[n]+\texttt{IslHgt}[n]}$, where $\texttt{BndHgt}[n]$ and $\texttt{IslHgt}[n]$ is the length of the longest path



(a) (b)

**Fig. 7.** (a) Wavefronts in red when an appropriate smoothing has been applied to the times and speeds of the wave in the root nodes. $\mathcal{C}$ is in blue and the rest of $\mathcal{VD}$ is in gray. (b) Wavefronts when the preferred times and speeds have been used to define the movement of the wave.

to a leaf in $\mathcal{PT}_n$ and $\mathcal{IT}_n$, respectively. Similarly, if the time $\texttt{TmNd}[n]$ has already been defined, we let the *preferred speed* of $n$ be $v_n = \max\left\{\frac{\texttt{IslHgt}[n]}{\texttt{TmNd}[n]}, \frac{\texttt{BndHgt}[n]}{1-\texttt{TmNd}[n]}\right\}$.

A naive method to define the times and speeds of a root $n$ is to set $\texttt{TmNd}[n] = t_n$ and $\texttt{VeNd}[n] = v_n$. That will minimize the number of revolutions and give the most equidistant wavefronts on each tree $\mathcal{T}_n$. However, the abrupt changes in time and speed along the central cycle $\mathcal{C}$ results in a spiral which curves a lot. Instead, we might smooth the times and speeds in the root nodes around $\mathcal{C}$. This can be done in many different ways, and we shall leave the details for the reader. See Fig. 7.

### 3.3. Creating wavefronts

For a given root node $n \in \mathcal{C}$, we want at least $s_{\mathcal{IT}_n} = \frac{\texttt{IslHgt}[n]}{\delta'}$ revolutions of the spiral in the tree $\mathcal{IT}_n$ in order to respect the stepover $\delta' = 0.95 \cdot \delta$. Similarly, we want $s_{\mathcal{PT}_n} = \frac{\texttt{BndHgt}[n]}{\delta'}$ revolutions in $\mathcal{PT}_n$. Therefore, the time between two revolutions should be at most $\Delta_n = \min\left\{\frac{\texttt{TmNd}[n]}{s_{\mathcal{IT}_n}}, \frac{1-\texttt{TmNd}[n]}{s_{\mathcal{PT}_n}}\right\}$. Hence, we let $\Delta' = \min_{n\in\mathcal{C}}\{\Delta_n\}$ be the minimum over all such values. We let the number of revolutions be $r = \lceil 1/\Delta' \rceil$ and set $\Delta = 1/r$.

Each tree $\mathcal{T}_n$ contains an interval of the corners of each wavefront $i$. The corners of the subset are points on $\mathcal{IT}_n$ if $t \leqslant \texttt{TmNd}[n]$, otherwise they are on $\mathcal{PT}_n$. Let $r_n = \lfloor \frac{\texttt{TmNd}[n]}{\Delta} \rfloor$. The wavefronts $i = 0, \ldots, r_n$ are on $\mathcal{IT}_n$, while wavefronts $i = r_n + 1, \ldots, r$ are on $\mathcal{PT}_n$. As explained in Section 3.2, we do not use the same time $\texttt{TmNd}[n]$ for every node $n$ on $\mathcal{C}$. Therefore, we may get wavefronts crossing $\mathcal{C}$ and wavefronts crossing each other. We shall later explain how to avoid that the polyline spiral has self-intersections.

We suggest to store the wavefront corners in a two-dimensional array for each of the trees $\mathcal{IT}_n$ and $\mathcal{PT}_n$. The wavefront corners on $\mathcal{IT}_n$ of one wavefront are stored in an array $\texttt{IslWf}[n][j]$, where index $j$ corresponds to wavefront $i = r_n - j + 1$. In $\mathcal{PT}_n$, the array $\texttt{BndWf}[n][j]$ stores corners of wavefront $i = j + r_n$. Hence, the corners of each wavefront is stored locally in each tree $\mathcal{T}_n$.

In $\mathcal{IT}_n$, the parents of the corners of wavefront $i = 0, \ldots, r_n - 1$ are corners of wavefront $i + 1$. In $\mathcal{PT}_n$, the parents of the corners of wavefront $i = r_n + 2, \ldots, r$ are corners of wavefront $i - 1$. Therefore, all parents are on the wavefront one step closer to the root $n$. In both $\mathcal{IT}_n$ and $\mathcal{PT}_n$, we introduce fake wavefront corners at the root $n$ stored in the arrays $\texttt{IslWf}[n][0]$ and $\texttt{BndWf}[n][0]$, respectively, which are the parents of the corners in the arrays $\texttt{IslWf}[n][1]$ and $\texttt{BndWf}[n][1]$. Thus, these fake corners are not corners on wavefront $i$ for any $i = 0, \ldots, r$, but are merely made to complete the tree of parent pointers between corners of neighboring wavefronts.

We also need an array $\texttt{WfLng}$ containing *global* information about the length of each wavefront crossing all the trees $\{\mathcal{T}_n\}_n$ in

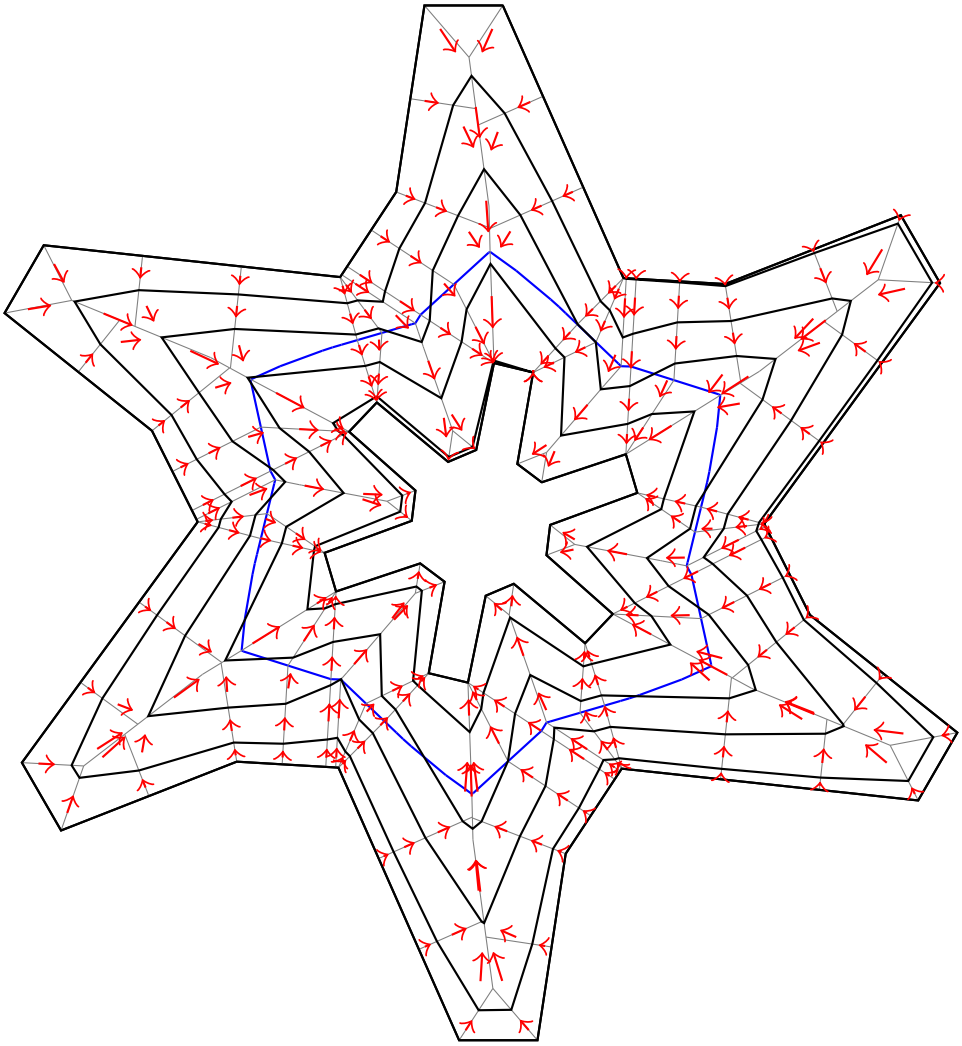


**Fig. 8.** A polyline spiral in black and red arrows from each corner $\texttt{Sp}[s]$ to its parents. The cycle $\mathcal{C}$ is in blue and the diagram $\mathcal{VD}$ is in gray.

order to do interpolation between the wavefronts later. We have $\texttt{WfLng}[i][0] = 0$ for every wavefront $i$. If $c_m$ and $c_{m+1}$ are the $m$'th and $(m+1)'$st corners on wavefront $i$, respectively, we have $\texttt{WfLng}[i][m+1] = \texttt{WfLng}[i][m] + \|c_{m+1} - c_m\|$. Notice that $c_m$ and $c_{m+1}$ can be corners in different, however neighbouring trees $\mathcal{T}_n$ and $\mathcal{T}_{n'}$ and hence stored in different arrays. $\texttt{TtlWfLng}[i]$ stores the total length of wavefront $i$.

### 3.4. Interpolating between wavefronts

We interpolate between two wavefronts $i-1$ and $i$ in each tree $\mathcal{T}_n$ separately, but using the same technique as in Section 2.6. If $i \leqslant r_n$, we interpolate between the wavefront fragments stored in $\texttt{IslWf}[n][j]$ and $\texttt{IslWf}[n][j+1]$, where $j = r_n - i + 1$, using the values of the length of wavefront $i-1$ stored in $\texttt{WfLng}[i-1]$. If $i > r_n + 1$, we interpolate between $\texttt{BndWf}[n][j-1]$ and $\texttt{BndWf}[n][j]$, where $j = i - r_n$, using the values stored in $\texttt{WfLng}[i]$. A special case occurs when $i = r_n + 1$, i.e., when we are interpolating between the first wavefront on each side of the root node $n$. In that case, let $t = (i-1)\Delta + \frac{\texttt{WfLng}[i][m]}{\texttt{TtlWfLng}[i]}\Delta$, when $\texttt{BndWf}[n][1][0]$ is the $m$'th corner on wavefront $i$. If $t \leqslant \texttt{TmNd}[n]$, we interpolate between $\texttt{IslWf}[n][0]$ and $\texttt{IslWf}[n][1]$. Otherwise, we interpolate on the other side of $\mathcal{C}$, that is, between $\texttt{BndWf}[n][0]$ and $\texttt{BndWf}[n][1]$. The convexification process described in Section 2.6 can be used in each tree $\mathcal{T}_n$ separately.

We may store the interpolated spiral in a one-dimensional array $\texttt{Sp}$. Before we add the first interpolated revolution to $\texttt{Sp}$, i.e., the one between wavefront 0 and 1, we add wavefront 0 to $\texttt{Sp}$, that is, all the corners of $\mathcal{I}$. Likewise, after the final revolution between wavefronts $r-1$ and $r$, we add wavefront $r$, which is all the corners of $\mathcal{P}$. These are used to ensure that the distance from the first and last revolution to the boundaries of $\mathcal{I}$ and $\mathcal{P}$, respectively, does respect the stepover when rounding the spiral.

For every corner $\texttt{Sp}[s]$ we have a pointer $\texttt{Pa}[s]$ such that $\texttt{Sp}[\texttt{Pa}[s]]$ is the first spiral corner we meet when traveling from $\texttt{Sp}[s]$ to the root $n$ of the tree $\mathcal{T}_n$ containing $\texttt{Sp}[s]$. The parents are not defined for the spiral corners closest to the root node $n$. Therefore, make parent dependencies across $\mathcal{C}$ such that the parent nodes are on the same side of $\mathcal{C}$ as $\mathcal{I}$. We want all the parent pointers to be towards the island $\mathcal{I}$. Therefore, we reverse all the pointers between pairs of corners in each tree $\mathcal{IT}_n$. Now, the parent pointers are defined for all spiral corners except for the ones on the boundary of $\mathcal{I}$. See Fig. 8 for an example of a polyline spiral around an island and red arrows indicating the parent pointers. Notice that a corner can have multiple parents, but the parents are consecutive and can thus be stored using two indices.

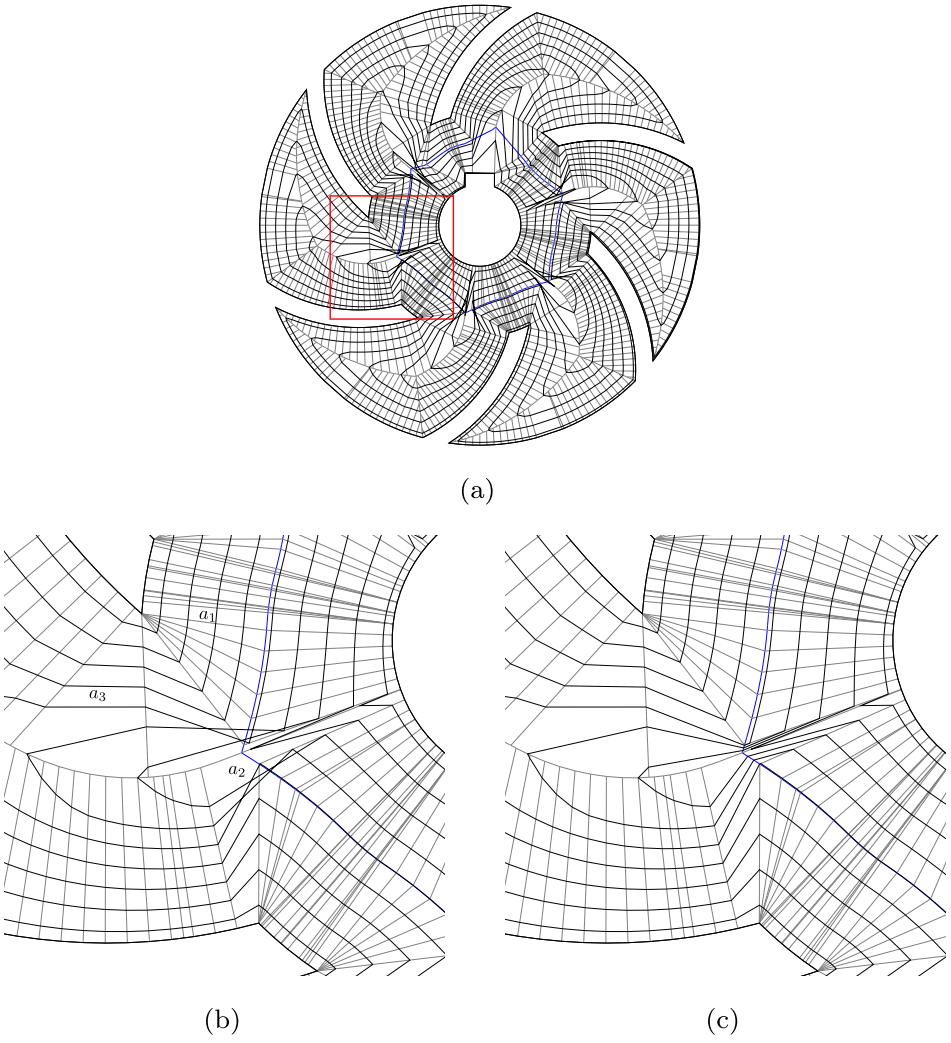



**Fig. 9.** The same pocket and island as in Fig. 7. The cycle $\mathcal{C}$ is blue, the other edges of $\mathcal{VD}$ are gray. The interpolated spiral is black, and we have not introduced extra spiral corners on $\mathcal{C}$ to avoid self-intersections. In order to make the intersection problems appear, we have not smoothed the times and speeds around $\mathcal{C}$ as described in Section 3.2, but merely used the preferred time and speed of each node. (b) is a close-up of (a) of the area in the red rectangle. In (c), we have introduced new corners on $\mathcal{C}$ when the spiral jumps from one side of $\mathcal{C}$ to the other when the union of the two faces on each side of $\mathcal{C}$ is not convex.

In Section 2.6, we stated that since there are no intersections between different wavefronts, the polyline spiral has no self-intersections when there is no island in $\mathcal{P}$. The wavefronts do not intersect in that case due to the convexity of the faces into which the diagram $\mathcal{VD}(\mathcal{P})$ subdivides $\mathcal{P}$. When there is an island $\mathcal{I}$ in $\mathcal{P}$, there are two kinds of faces into which $\mathcal{VD}(\mathcal{P} \setminus \mathcal{I})$ subdivides $\mathcal{P} \setminus \mathcal{I}$, see Fig. 9. Some faces, like $a_3$ in the figure, are bounded by edges of one tree $\mathcal{T}_n$ while some are between two trees, like $a_1$ and $a_2$. The latter kind is bounded by edges of two neighboring trees $\mathcal{T}_n$ and $\mathcal{T}_m$ and an edge $e$ from $n$ to $m$ on $\mathcal{C}$, where $n$ and $m$ are neighboring nodes on $\mathcal{C}$. The first kind of faces is similar to the faces in Section 2, so here we do not worry about self-intersections of the spiral. The second, however, can lead to wavefronts crossing each other and therefore also a self-intersecting spiral as is the case in the figure when the spiral jumps over $\mathcal{C}$ and crosses $a_2$. If the union of the faces on each side of $e$ is convex, like $a_1$ and its neighboring face on the other side of $\mathcal{C}$, there is no problem. It can easily be tested if the union of the faces is convex by considering the angles of the union at nodes $n$ and $m$. If it is not, we introduce a new corner on the edge on $\mathcal{C}$ whenever the spiral jumps from one side of $\mathcal{C}$ to the other. The new corner is an interpolation of nodes $n$ and $m$ using the time of the spiral in the last corner in tree $\mathcal{T}_n$. Fig. 9(c) shows the result of introducing the extra corners. Our experience is that these intersection problems occur very rarely when the times and speeds have been appropriately smoothed around $\mathcal{C}$. Our experience is that these intersection problems occur very rarely if the time and speed of the wave have been smoothed around $\mathcal{C}$ as described in Section 3.2.

## 4. The skeleton method for a spiral in a pocket without islands

The method from Section 2 is mainly applicable if the polygon $\mathcal{P}$ is not too far from being a circle. If $\mathcal{P}$ is very elongated or branched, the distance between neighboring revolutions will often be much less than the maximum stepover. Therefore, the tool path will be unnecessarily long and the cutting width will vary a lot. That leads to long machining time and an uneven finish of the part. See Fig. 10 for an example. In such cases, we construct a *skeleton* in $\mathcal{P}$, which is

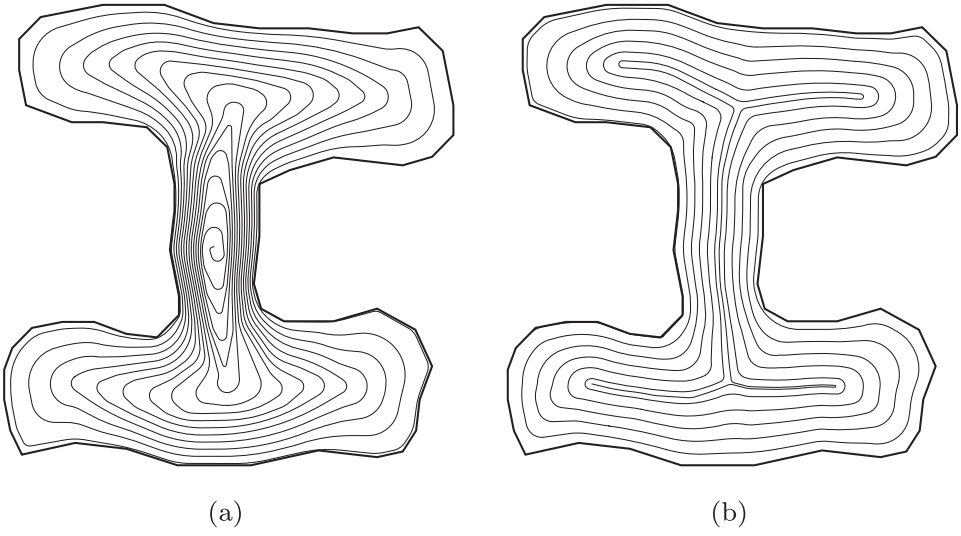



**Fig. 10.** Comparison of the basic spiral method from Section 2, figure (a), with the improved skeleton method from Section 4, figure (b). Note that the spiral obtained from the skeleton method is significantly shorter and that the distance between neighboring revolutions is varying much less than when using the basic method.

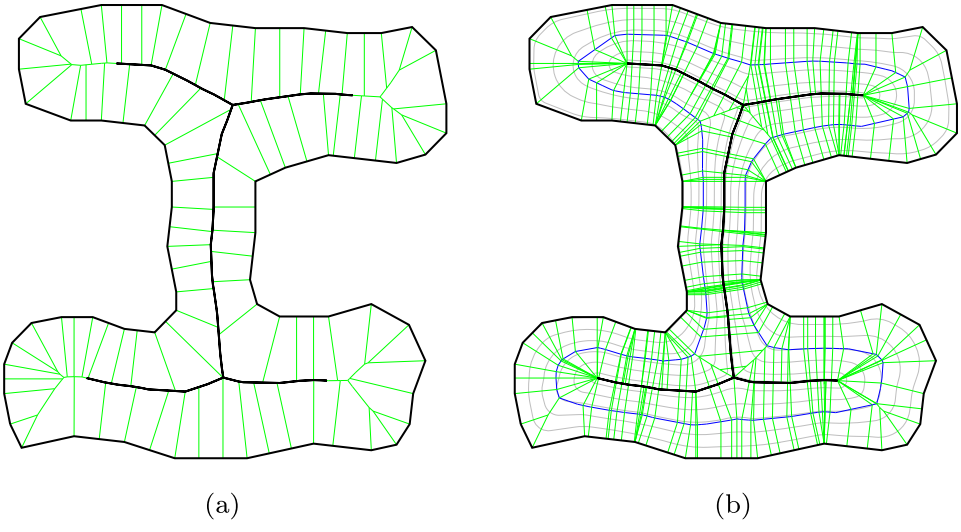



**Fig. 11.** (a) The diagram $\mathcal{VD}(\mathcal{P})$ of the polygon $\mathcal{P}$ from Fig. 10 in green, where the edges chosen for the skeleton are black. (b) The diagram $\mathcal{VD}(\mathcal{P} \setminus \mathcal{I})$ of $\mathcal{P}$ with the skeleton considered an island $\mathcal{I}$. The cycle $\mathcal{C}$ is blue and the remaining edges are green. The resulting spiral from Fig. 10(b) is included in gray.

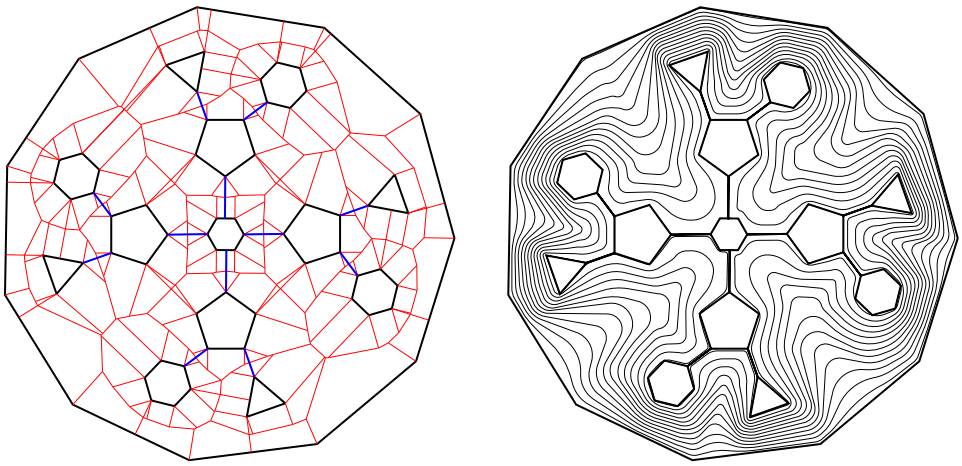



**Fig. 12.** (a) A polygon with 13 islands. The bridges chosen among the Voronoi edges to connect the islands are blue. The remaining Voronoi edges are red. (b) The resulting spiral around the islands.

an island $\mathcal{I}$ with zero area. We then use the method from Section 3 to make a spiral from the island to the boundary. It does not matter for the construction of the spiral that the island $\mathcal{I}$ has zero area. See Fig. 10 for a comparison between the basic method from Section 2 with the skeleton method described here when applied to the same polygon.

We choose the skeleton as a connected subset of the edges of the diagram $\mathcal{VD} = \mathcal{VD}(\mathcal{P})$. We traverse $\mathcal{VD}$ once starting at the root and decide for each edge whether to include it in the skeleton. If an edge from node $n$ to $m$ is not included, we do not include anything from the sub-tree $\mathcal{VD}[m]$. For any node $n$, let $d(n)$ be the length of the shortest path from $n$ to a leaf in $\mathcal{VD}[n]$, and let $D = \max_{n \in \mathcal{VD}} d(n)$. We have found that the following criteria for including an edge $e$ from node $n$ to $m$ gives good results. We require all the criteria to be satisfied.

1. The longest path from $n$ to a leaf in $\mathcal{VD}[n]$ goes through $m$ or $\ell[e] + \mathtt{Hgt}[m] \geqslant 1.5 \cdot D$, where $\ell[e]$ is the length of edge $e$.
2. The length of the spanned boundary[3] of $m$ is larger than $2 \cdot D$.
3. $\mathtt{Hgt}[m] \geqslant D$.

Criterion 1 is to avoid getting a skeleton that branches into many short paths. Therefore, we only make a branch which is not following the longest path from $n$ if it seems to become at least $0.5 \cdot D$ long (taking criterion 3 into account). When criterion 2 fails, it seems to be a good indicator that an edge is not a significant, central edge in $\mathcal{VD}$, but merely one going straight to the boundary. Criterion 3 ensures that we do not get too close to the boundary. If we did, we would get very short distances between the neighboring revolutions there. If criterion 3 is the only failing criterion, we find the point $p$ on $e$ such that $\mathtt{Hgt}[m] + \|p - \mathtt{Pt}[m]\| = D$ and include the edge from node $n$ to $p$ in the skeleton.

If the polygon is close to being a circle, the method described here results in a very small skeleton, and we get a better spiral using the basic method from Section 2 in that case. This can be tested automatically by falling back to the basic method if the circumference of the skeleton is less than, say, 5% of the circumference of $\mathcal{P}$.

[3] The *spanned boundary* of a leaf $l$ of $\mathcal{VD}$ is half of the sum of the lengths of the edges of $\mathcal{P}$ incident to $l$. The spanned boundary of a node $m$ of $\mathcal{VD}$ is the sum of the spanned boundaries of the leafs in $\mathcal{VD}[m]$.

## 5. Computing a spiral in a pocket with multiple islands

The method from Section 3 works only for polygons with a single island. If there are many islands $\mathcal{I}_0, \ldots, \mathcal{I}_{h-1}$ in a polygon $\mathcal{P}$, we may connect them with bridges in a tree structure to form one big connected island, see Fig. 12. The basic idea of reducing the number of islands by connecting them is also used by Chuang and Yang (2007) and Held and Spielberger (2014). Our method is a variation of the minimum spanning tree algorithm of Dijkstra (1959). We choose the bridges as edges in the Voronoi diagram of the area $\mathcal{P} \setminus \bigcup_{i=0}^{h-1} \mathcal{I}_i$. The algorithm creates an array of the edges to use as bridges. We keep a growing set $s$ of the nodes of the Voronoi diagram that we have connected by bridges so far. We first find one central node $n_0$ and $s$ is only containing $n_0$ in the beginning. We use Dijkstra's algorithm to make all shortest paths from nodes in $s$. When we reach an island $\mathcal{I}_i$ whose corners are not in $s$, we use the shortest path to that island as a bridge and add the nodes on the shortest path and the corners on $\mathcal{I}_i$ to $s$. We implement the algorithm so that bridges starting at nodes close to $n_0$ are preferred over nodes far from $n_0$. That makes the bridges grow from the center node $n_0$ out in every direction. If we did not choose the bridges in this careful way, the resulting connected island would possibly make unnecessarily long dead ends that would require many revolutions to fill out by the spiral.

## 6. Conclusion

We have described methods for the computation of spiral tool paths suitable for many shapes of pockets for which no previously described algorithms yield equally good results. Our main contribution is the new possibility of making a spiral morphing an island in a pocket to the boundary of the pocket.

Our algorithms works under the assumption that the input is polygonal. An obvious improvement is to generalize to input consisting of line segments and circular arcs, as done by Held and Spielberger (2009), using ArcVRONI by Held and Huber (2009) to compute the Voronoi diagrams for such input.

Further work could be focused on developing a hybrid method of the skeleton method and the method for handling one or more islands. Also, one could work on automatic subdivision of complex pockets into pockets more suitable for spiral machining. Given the new techniques described in this paper, this problem seems

different from the one handled by Held and Spielberger (2014) and Patel and Lalwani (2017).

## Conflict of interest

None.

## Acknowledgments

I would like to thank my colleague at Autodesk, Niels Woo-Sang Kjærsgaard, for numerous helpful discussions while I did the development of the spiral algorithms and for many useful comments on this paper. My thanks also go to Autodesk in general for letting me do this research and publish the result.

## References

Banerjee, A., Feng, H.-Y., & Bordatchev, E. (2012). Process planning for floor machining of $2\frac{1}{2}$D pockets based on a morphed spiral tool path pattern. *Computers & Industrial Engineering, 63*(4), 971–979.

Bieterman, M., & Sandstrom, D. (2003). A curvilinear tool-path method for pocket machining. *Journal of Manufacturing Science and Engineering, Transactions of the ASME, 125*(4), 709–715.

Chuang, J.-J., & Yang, D. (2007). A laplace-based spiral contouring method for general pocket machining. *The International Journal of Advanced Manufacturing Technology, 34*(7–8), 714–723.

Cormen, T., Leiserson, C., Rivest, R., & Stein, C. (2009). *Introduction to algorithms*. MIT Press.

Dijkstra, E. (1959). A note on two problems in connexion with graphs. *Numerische Mathematik, 1*(1), 269–271.

Graham, R., & Yao, F. (1983). Finding the convex hull of a simple polygon. *Journal of Algorithms, 4*(4), 324–331.

Handler, G. (1973). Minimax location of a facility in an undirected tree graph. *Transportation Science, 7*(3), 287–293.

Held, M. (2001). VRONI: An engineering approach to the reliable and efficient computation of Voronoi diagrams of points and line segments. *Computational Geometry, 18*(2), 95–123.

Held, M., & de Lorenzo, S. (2018). On the generation of spiral-like paths within planar shapes. *Journal of Computational Design and Engineering, 5*(3), 348–357.

Held, M., & Huber, S. (2009). Topology-oriented incremental computation of Voronoi diagrams of circular arcs and straight-line segments. *Computer-Aided Design, 41*(5), 327–338.

Held, M., & Spielberger, C. (2009). A smooth spiral tool path for high speed machining of 2D pockets. *Computer-Aided Design, 41*(7), 539–550.

Held, M., & Spielberger, C. (2014). Improved spiral high-speed machining of multiply-connected pockets. *Computer-Aided Design and Applications, 11*(3), 346–357.

Huang, N., Lynn, R., & Kurfess, T. (2017). Aggressive spiral tool paths for pocket machining based on medial axis transformation. *Journal of Manufacturing Science and Engineering, 139*(5).

Huertas-Talón, J., García-Hernández, C., Berges-Muro, L., & Gella-Marín, R. (2014). Obtaining a spiral path for machining STL surfaces using non-deterministic techniques and spherical tool. *Computer-Aided Design, 50*, 41–50.

Patel, D., & Lalwani, D. (2017). Quantitative comparison of pocket geometry and pocket decomposition to obtain improved spiral tool path: A novel approach. *Journal of Manufacturing Science and Engineering, 139*(3).

Romero-Carrillo, P., Torres-Jimenez, E., Dorado, R., & Díaz-Garrido, F. (2015). Analytic construction and analysis of spiral pocketing via linear morphing. *Computer-Aided Design, 69*, 1–10.

Xu, J., Sun, Y., & Zhang, X. (2013). A mapping-based spiral cutting strategy for pocket machining. *The International Journal of Advanced Manufacturing Technology, 67*(9–12), 2489–2500.

Zhou, B., Zhao, J., & Li, L. (2015). CNC double spiral tool-path generation based on parametric surface mapping. *Computer-Aided Design, 67*, 87–106.