Computing the Tool Path of an Externally Monotone Polygon in Linear Time*

Prosenjit Bose† David Bremner ‡ Diane Souvaine§

1 Introduction

A Numerically-Controlled (NC) machine typically consists of a worktable and a spindle (or cutter) with several axes of freedom for positioning the tool. In this paper, we restrict our attention to machines having only translational freedom. We focus on 2D milling, which can only cut out planar objects, and $2\frac{1}{2}$D milling where only two of the axes are continuous-path controlled and the third axis is point-to-point or straight-line controlled. More than 80% of all mechanical parts to be machined can be cut by applying 2D or $2\frac{1}{2}$D for path control [11].

We study contour-parallel milling where a pocket is machined by having a cutter following paths that are equidistant offset curves from the boundary of the object. Although there are many types of cutters, the most common is the *ball-end* cutter. Such a cutter removes a disc whose radius is the radius of the ball.

We focus on the following basic problem: given an object, modelled by a simple polygon on n vertices, and the radius of the ball-end cutter, compute the boundary or outer tool path for the cutter and the complete tool path for the cutter. In Held [11], it is shown how to compute both these tool paths by using the medial axis (see [2]). The medial axis can be computed fairly easily in $O(n \log n)$ time. In the specific case of a simple polygon, it can even be computed in $O(n)$ time [6]. However, the $O(n)$ time algorithm is of theoretical interest only since it is quite complex and also uses Chazelle's [4] linear–time triangulation algorithm which in itself is extremely complex. The challenge is to simply and efficiently compute both the outer and complete tool path of a simple polygon in $O(n)$ time.

In this paper, we demonstrate a simple and efficient method to compute the outer and complete tool path of a fairly general class of polygons called *externally monotone*. A simple polygon is externally monotone if for every point inside a pocket, there is a path to the lid of the pocket that is monotone in the direction normal to the lid. To date, this is the most general class of polygons for which a simple and efficient linear time algorithm to compute the tool path is known.

In Section 2, we review notation and preliminaries related to the results. In Section 3, we show how to compute the tool path for an externally monotone polygon. Finally, we present conclusions and open problems in Section 4.

*The first two authors partially supported by NSERC Canada
†School of C.S., Carleton University, Ottawa, Canada.
‡Faculty of C.S., U. New Brunswick, Fredericton, Canada
§Dept. Computer Science, Tufts University, Medford MA

2 Preliminaries

In this section, we review some notation and preliminaries. Many of the definitions and background can be found in de Berg et al. [2] and Held [11].

A simple polygon P is defined by a set of vertices $v_1, v_2, \ldots, v_{n-1}, v_n$ in counter-clockwise order such that each pair of consecutive vertices is joined by an edge, including the pair $\{v_n, v_1\}$. The interior of the polygon P is denoted by $int(P)$, and the boundary by ∂P. The boundary is considered part of the polygon; that is, $P = int(P) \cup \partial P$.

The *convex hull*, $CH(P)$ of a simple polygon P is the smallest convex polygon containing P. The *pockets* of a simple polygon P are the areas outside P but in $CH(P)$. Each pocket is a simple polygon in itself and its boundary is formed by edges of P and one edge of the convex hull called a *lid*.

A polygonal chain is *monotone* with respect to direction d if the intersection of the chain and any line perpendicular to d is a convex set. Let $L = \{l_1, l_2, \ldots, l_n\}$ and $R = \{r_1, r_2, \ldots, r_m\}$ be two polygonal chains monotone with respect to the y-axis. The chain R is to the *right* of L provided that for every horizontal line h intersecting both R and L, the x-coordinate of the intersection point of R and h is greater than that of the x-coordinate of the intersection point of L and h.

Let $C(t) = (x(t), y(t))$ be a curve, where t is a real parameter. An offset curve $C'(t) = C(t) + r \cdot N(t)$ where r is a positive constant representing the offset distance and $N(t)$ is the unit normal to the curve. Notice that depending on the direction of the normal chosen, the offset may be to one side or the other of the given curve. We will refer to the chosen direction as the *offset* direction.

To accommodate the singularities at vertices, we slightly modify the definition of an offset curve. We define the offset of a vertex to be a circular arc centered at the vertex, and the extent of the arc is determined by the perpendiculars of the line segments that meet at the vertex (see Figure 1). The offset of a line segment is simply the translation of the given line segment by the offset distance. This definition leads to a natural definition of the offset of a polygonal chain.

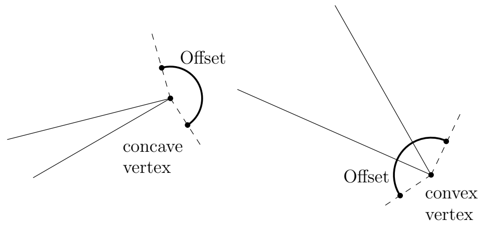

Figure 1: Offset of a convex and concave vertex shown in bold.

The offset of a polygonal chain is the concatenation of the offset of its edges and vertices. The offset of a polygonal chain may be to one side or the other of the chain depending on the offset direction chosen for the offset distance. While traversing a polygonal chain C in its given order, if the offset of an edge is to the right of the edge it is a *right offset* and symmetrically, if the offset of an edge is to the left of the edge it is a *left offset*.

The offset of a polygonal chain may be self-intersecting (see Figure 2). The *tool path* of a polygonal chain consists of the portion of the offset that can be followed by the cutter without removing any part of the chain. More formally, a point p is on the tool path provided that p is on the offset and a circle of radius r centered at p contains no part of the polygonal chain in its interior. The tool path of a simple polygon is simply the tool path of its boundary.

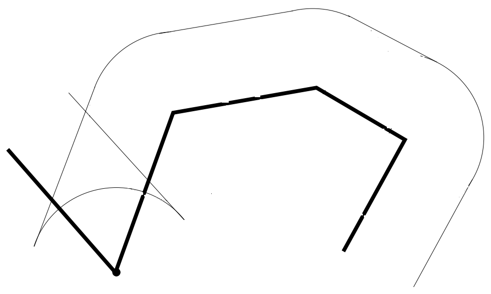

Figure 2: Offset of a polygonal chain may be self-intersecting.

The tool path of a polygonal chain is not necessarily a single chain but may consist of several chains (see Figure 3). One of these chains is of particular interest, namely the *outer tool path*. A point p is on the outer tool path provided that p is on the tool path and a circle of radius r centered at p can be moved to infinity without ever intersecting the chain in its interior (see Figure 3). However, the tool path of a monotone chain is a single chain consisting of straight edges and circular arcs. This property of the tool path of monotone chains is vital to the algorithm.

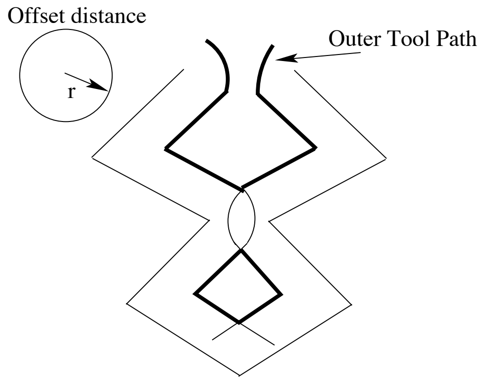

Figure 3: Example of a chain having two components to the tool path. Tool path is shown in bold.

3 Computing the Tool Paths for an Externally Monotone Polygons

Computing the tool path of a convex polygon is trivial. The difficulty in computing the tool path of a simple polygon lies in computing the tool path of each of its pockets. We begin by looking at a special case.

3.1 Tool Path of a special type of pocket

Before studying the general problem, we first examine a special case that will shed some light on the more general problem. Suppose polygon P has a pocket whose boundary consists of a lid (edge $[l_1, r_1]$) and two monotone chains $L = \{l_1, l_2, \ldots, l_n\}$ and $R = \{r_1, r_2, \ldots, r_m\}$ with R being to the right of L and the two chains share their ends, i.e. $l_n = r_m$. Without loss of generality, assume that the lid is horizontal and the two chains are monotone with respect to the y-axis.

There are two steps involved in computing the tool path. Given an offset distance r, the first step is to compute the tool path of the right offset of chain L and the left offset of chain R. Chou et al. [5] present a simple and elegant algorithm that computes the tool path of a monotone chain in $O(k)$ time where k is the size of the chain. The algorithm is similar in spirit to Graham's scan [10] and the only data structure used is a stack.

Let L' and R' be the tool path of the right offset of L and left offset of R, respectively, as computed by the algorithm of Chou et al. [5]. The next step in computing the tool path for the pocket is to find the intersection points $(i_1, \ldots, i_k)$ between L' and R'. There must be at least one intersection point since R and L share the vertex $l_n = r_m$. Since both L' and R' are monotone chains, the intersections can be computed in a manner similar to the merging of two sorted lists (see [7]).

Note that the intersection points divide both L' and R' into $k+1$ pieces. Let $L'_1, \ldots, L'_{k+1}$ and $R'_1, \ldots, R'_{k+1}$ denote the $k+1$ pieces of L' and R' respectively.

Once both these steps are complete, both the outer tool path and the complete tool path can be computed. The outer tool path consists of $L'_1 \cup R'_1$. In fact, when computing the outer tool path, one can stop the second step after the first intersection point has been found. The complete tool path is the set of paths $L'_{2t+1} \cup R'_{2t+1}$ for all $0 \le t \le k/2$.

3.2 Outer and Complete Tool Path

We now consider the more general problem of computing the complete tool path of an externally monotone polygon P.

Recall that a polygon P is externally monotone if for every point inside a pocket, there is a path to the lid of the pocket that is monotone in the direction normal to the lid. We assume for simplicity that our polygon contains no horizontal edges other than the lid. A polygon Q is *internally monotone* from a *root* e provided there is a path from every point in Q to the edge e that is monotone in the direction normal to e. A polygon is externally monotone provided that each of its pockets is internally monotone from its lid.

Given an internally monotone polygon, we can always re-orient it so that its root is horizontal; henceforth we assume that all internally monotone polygons are re-oriented such that their root is horizontal. A vertex v is called *critical* if it is a reflex vertex tangent to a horizontal line. Critical vertices may be classified as *upward*, where ∂P is below the tangent line in the open neighbourhood of v, and *downward* otherwise. Without loss of generality, assume that lid of our pocket is horizontal and strictly above every non-lid vertex of the pocket. The following lemma follows from e.g. Lemma 2 in [3].

Lemma 1 *A pocket is internally monotone from its lid if and only if it has no downward critical vertices.*

Lemma 1 provides a very simple test to determine if a pocket is internally monotone from its lid, and to decompose the boundary of a pocket into maximal *left* (of the polygon interior) and *right* monotone chains. Let $C = \{c_1, c_2, \ldots, c_p\}$ be the sequence of vertices of the boundary of a pocket. Assume the lid is the horizontal edge $[c_1, c_p]$. The first maximal left monotone chain is the sequence $\{c_1, \ldots, c_j\}$ where c_j is the first local y-minimum. The first maximal right chain is the sequence $\{c_j, \ldots, c_k\}$ where c_k is the first local y-maximum after c_j. The local maxima play a special role in this decomposition. We will refer to them as *split vertices* for reasons which will be clear in the sequel.

Let $[L_1, \ldots, L_j]$ and $[R_1, \ldots, R_j]$ be the sequence of maximal monotone left and right chains, respectively, in the order that they occur along the boundary (which is identical to the order in which they are computed by the above method). Note that L_i and R_i share a vertex that is a local minimum and R_i and L_{i+1} share a vertex that is a local maximum. To compute the tool path of the pocket, we first compute the right tool path of the L_i's, denoted by L'_i, using the algorithm of Chou et al. [5] and similarly the left tool path of the R_i's, denoted by R'_i.

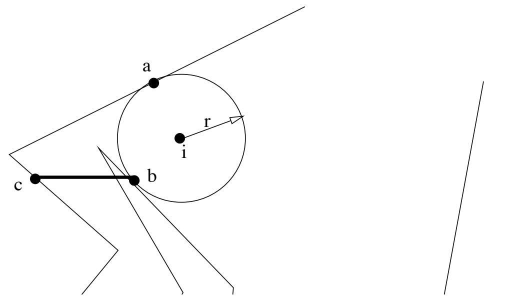

Figure 4: Illustration of proof of Lemma 2.

Although L'_i and R'_i can intersect a linear number of times, it turns out two left tool paths (or two right tool paths) can only feasibly intersect once. An intersection point i between two tool paths is feasible provided that a circle C of radius r centered at i does not contain any part of ∂P in its interior. In the special case of two left tool paths intersecting at i_j, let a_j (repectively b_j) denote the intersection of C and the leftmost (respectively rightmost) of the pair of tool paths (see Figure 4). It follows from monotonicity that a_j and b_j are both in the left half circle of C. We classify intersection points i_j as *upper* (resp. *lower*) if a_j is above (resp. below) b_j. We omit the proof of the following lemma in this abstract.

Lemma 2 ***(a)*** *The right tool paths of two maximal left monotone chains have at most one feasible upper intersection.* ***(b)*** *If i_j is a lower intersection point, then the polygon P has a lower critical vertex.*

We now discuss how to compute the intersection points of the right tool paths of maximal monotone left chains. The computation is symmetric for the left tool paths of the right monotone chains. Along with the previous lemma, a key ingredient which allows one to to merge a set of left tool paths in constant amortized time per tool path is the following lemma.

Lemma 3 *Let L_i, L_j and L_k be three maximal left chains such that $i < j < k$. If L'_k intersects L'_j it does so no higher than any upper intersection of L'_i and L'_j and no lower than any lower intersection of L'_i and L'_j.*

We now turn our attention to split points (or local maxima). Consider Figure 5. Notice that a split point cleanly divides the problem of computing a tool path into two portions. No tool path of polygonal chains in C_1 can properly intersect tool paths of chains in C_2.

This gives rise to a simple recursive algorithm to compute the tool path.

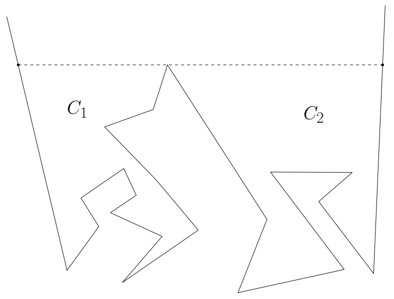

Figure 5: Example of property of split vertices.

Before outlining the algorithm for computing the outer tool path of a pocket, we discuss how to compute for each split vertex S, a pointer to the topmost split vertex to its left and to its right. If we had such a structure, it is precisely a binary tree rooted at S. It is well known that one can reconstruct a binary tree on n nodes in $O(n)$ time given a sequence representing the pre-order or post-order traversal of its nodes. We omit here the details of computing the pre-order traversal.

We now outline the algorithm to compute the outer tool path. The additional details related to computing the complete tool path are omitted here. After computing the intersection (effectively merging) the set of left tool paths and the set of right tool paths, we proceed by an implicit plane sweep following the (merged) left and right tool paths downward until a split vertex is reached. We then proceed recursively on the pairs of left and right tool paths adjacent to the split vertex (i.e. using the previously computed vertices adjacent to split vertices).

There are only three cases of recursive calls. The three cases depend on where the split vertex appears with respect to the left and right tool path currently under consideration. If the split vertex appears in between them, then both sides of the split vertex are feasible which accounts for the two recursive calls. If the split vertex appears on the left of the left tool path or the right of the right tool path, one of the two sides is no longer feasible and therefore only one recursive call is required. All of the pre-processing prior to the invocation of the algorithm takes linear time. Also, the algorithm itself takes linear time, which can seen from the fact that each vertex is visited at most once.

4 Conclusions

We have presented a simple and efficient linear time algorithm for computing the tool path of an externally monotone polygon. This algorithm is relatively easy to implement: a proof-of-concept implementation was the subject of Zavlin's master's project [13]. An open question is whether computing the outer tool path of a simple polygon in linear time requires sophisticated techniques like the medial axis and linear–time triangulation.

References

[1] A. Aggarwal, L.J. Guibas, J. Saxe, and P.W. Shor. A Linear-time Algorithm for Computing the Voronoi Diagram of a Convex Polygon. *Disc. & Comp. Geom.*, 4:591-604, 1989.

[2] M. de Berg, M. van Kreveld, M. Overmars, O. Schwarzkopf. Computational Geometry: Algorithms and Applications Springer-Verlag, 1998.

[3] D. Bremner and T. Shermer. Point visibility graphs and $\mathcal{O}$-convex cover. *Int. J. of Comp. Geom. and Appl.*, 10(1):55–71, 2000.

[4] B. Chazelle. Triangulating a Simple Polygon in Linear Time. *Disc. & Comp. Geom.*, 6:485–524, 1991.

[5] S-Y. Chou, T.C. Woo, L-L. Chen, K. Tang, and S.Y. Shin. Scallop Hull and Its Offset. *CAD*, 26(7):537-542, 1994.

[6] F. Chin, J. Snoeyink, C. Wang. Finding the Medial Axis of a Simple Polygon in Linear Time. *Disc. & Comp Geom.*, 21:405-420, 1999.

[7] T. Cormen, C. Leiserson, and R. Rivest. Introduction to Algorithms. MIT Press, 1990.

[8] H. Edelsbrunner, D. Kirkpatrick, and R. Seidel. On the Shape of a Set of Points in the Plane. *IEEE Trans. Inf. Theor.*, IT-29(4):11-30, 1989.

[9] M.R. Garey, D.S. Johnson, F.P. Preparata, and R.E. Tarjan. Triangulating a Simple Polygon. *Inf. Proc. Letters*, 7(4):175-179, 1978.

[10] R. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Inf. Proc. Letters*, 1:132-133, 1972.

[11] M. Held. On the Computational Geometry of Pocket Machining. *LNCS*, vol. 500, Springer-Verlag, 1992.

[12] A.A. Melkman. On-Line Construction of the Convex Hull of a Simple Polyline. *Inf. Proc. Letters*, 25(1):11-12, 1987.

[13] B. Zavlin. Computing tools paths of externally monotone polygons. Rutgers University Master's Essay. 1998.