SCALLOP HULL AND ITS OFFSET

Shou-Yan Chou
Dept of Industrial Management
National Taiwan Institute of Technology

Tony C. Woo
Department of Industrial and Operations Engineering
The University of Michigan

Lin-Lin Chen
Graduate School of Engineering Technology
National Taiwan Institute of Technology

Kai Tang
Schlumberger Technologies CAD/CAM Division
Ann Arbor, Michigan

Sung Yong Shin
Department of Computer Science
Korea Advanced Institute of Science and Technology

Technical Report 94-9

March 1994

Abstract

A linear time algorithm that computes the envelop of the offset of a monotone chain is presented. The *scallop hull* – an extended notion of the convex hull – of the monotone chain is first computed by using an approach similar to the convex hull construction algorithm. The offset of the scallop hull, which yields the desired envelop, can then be computed in linear time from the scallop hull, giving a tool path.

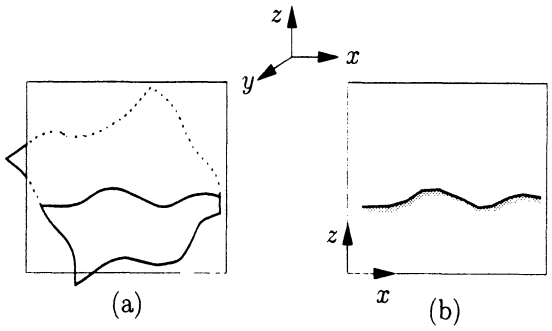

Figure 1: Sectioning a surface and simplifying its geometry.

1 Introduction

The problem of finding an *offset* arises in the generation of tool paths for machining a surface on a numerically controlled (NC) machine. When a hemispherical (ball-end) cutter is used to create the surface of a workpiece, the center of the hemispherical tip of the cutter and the resulting surface maintain a constant distance, which equals the radius of the ball-end cutter. The tool path (taken by the center of the cutter) can therefore be obtained by computing an offset of the desired locus of contact points on the surface of the workpiece.

There are in general two types of NC tool paths:"direction-parallel" [13] and "contour-parallel". This paper examines the former by taking the following view. Suppose the representation is a graph surface $z = f_1(x, y)$. (A parametric surface $\mathcal{P}(u, v)$, i.e. $x = F(u, v)$, $y = G(u, v)$, $z = H(u, v)$ can be implicitized as $f_2(x, y, z) = 0$; hence $z = f_1(x, y)$.) Sectioning the surface with a plane parallel to, say, the xz-plane, gives a curve of intersection. See Figure 1. The offset of this curve yields one of the zig-zag tool paths; the subsequent ones are obtained similarly by taking further parallel sections. Note that the tool paths thus generated do not have to correspond to the

constant parametric curves. And, overcut and undercut may occur since the surface normals are not necessarily in the section plane. This when cast in the framework of a coordinate measuring machine (CMM), seems not unreasonable. The probe of a CMM is spherical. The manner in which it acquires data is by contact, via small deviations from the computed tool path.

In the literature on NC tool path generation, most researchers [2, 3, 5, 7, 6, 11, 10, 15, 16, 17] deal with sculptured surfaces and are concerned with tool collision (due to the possible self-intersection of the offset). This paper offers a novel concept, the *scallop hull* — a generalization of the familiar convex hull [19], and applies it to the computation of a collision-free tool path, all in linear time, i.e. in time linearly proportional to the number of segments in the input curve. So that this novel idea is un-encumbered by analytic or numerical techniques, the input is assumed to be composed of n line segments, a *chain.* While there is a selection of 3-, 4-, and 5-axis NC machines, the simplest of them, 3-axis, is adopted. Because of the limited degrees of freedom, the chain must be *monotone*, i.e. it is single-valued in the z direction [4].

The tool path for a monotone chain can be obtained by first computing an *initial offset* and then eliminating the portions that cause gouging. The initial offset is the concatenation of the offsets of every individual line segment and vertex in the chain, as shown in Figure 2 (a). The offset of a line segment is simply a parallel line segment at a given distance. The direction is understood to be away from the material side. A vertex can be thought of as a degenerate line segment, a constant-distance offset from which is a circular arc. The extent of the arc is determined by the perpendiculars of the line segments that meet at the vertex. (See Figure 3.) To eliminate self-intersection in the initial offset, two methods are applicable: one utilizes the closest points *Voronoi* diagram [13, 18, 20] of the chain to identify

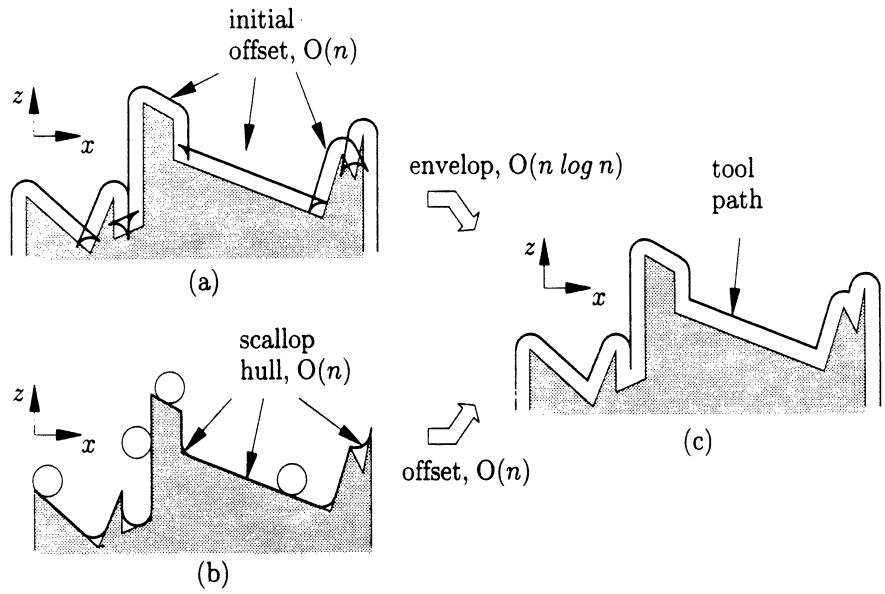

Figure 2: The intersection-free scallop hull.

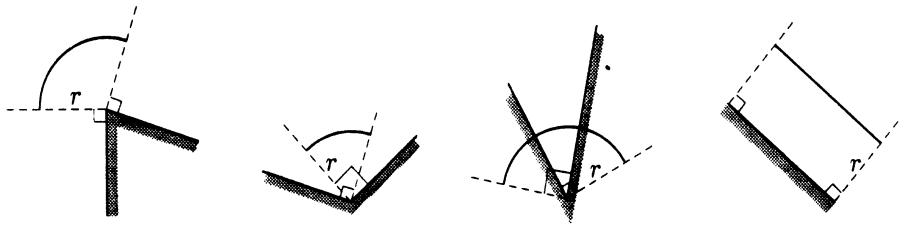

Figure 3: Initial offset of a vertex and of an edge.

self-intersections of the initial offset, and the other utilizes an algorithm developed by Hershberger [14] for constructing the envelop for a set of line segments. Both methods require $O(n \log n)$ time [1]. In this paper, rather than dealing with the initial offset, the tool path is computed by offsetting the *scallop hull* of the monotone chain, which is free of self-intersection. The scallop hull is similar to $\alpha-$hull defined by Edelsbrunner, Kirkpatrick, and Seidel [9] for capturing the "shape" of a set of points. The parameter α is a real number. When it is positive, the $\alpha-$hull of a set of points is defined as the intersection of all the discs that have radii $1/\alpha$ and contain all the points in the set. When α is negative, the $\alpha-$hull is the intersection of all closed complements of the discs (with radii equal to $-1/\alpha$) that contain all the points in the set. The case of $\alpha < 0$ is analogous to a scallop hull for a set of points. The $\alpha-$hull of a set of n points can be computed in $O(n \log n)$ time [9].

The *scallop hull* of a monotone chain is computed by sliding a circle against the boundary of the chain from its non-material side. See Figure 2 (b). The monotonicity of the chain facilitates the amortized complexity for backtracking hence enabling the development of a linear time algorithm. The scallop hull has three kinds of elements: convex vertices, circular arcs substituting the portions of the chain that cannot be reached by the circle, and the portions of the chain that are in contact with the circle. Notice that there is no concave vertex in a scallop hull, which gives rise to a self-intersecting offset. When the radius of the circle is infinite, the scallop hull of the chain becomes its convex hull.

The *tool path* is simply the locus of the center of the circle, which can be stored during the process of constructing the scallop hull. This takes $O(n)$ storage, where n is the number of segments. In this paper, the tool path is computed by offsetting the scallop hull, without costing in storage, as illus-

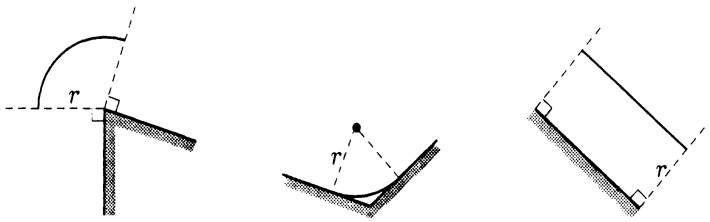

Figure 4: Offset of a scallop hull.

trated in Figure 4. This takes $O(n)$ time since there is no self-intersection in the scallop hull. The total time is $O(n) + O(n) = O(n)$.

2 Preliminaries

Let $C = \langle v_1, v_2, \ldots, v_n \rangle$ be a chain that is monotone with respect to the x-axis and let C be represented by an array of its n vertices. The line segment joining two successive vertices v_i and v_{i+1} is called an *edge* of C and is denoted by e_i. Assume that e_i is directed such that $x_i \leq x_{i+1}$, for all $1 \leq i \leq n-1$. Two downward vertical rays with the endpoints at v_1 and v_n are added to the chain C so as to permit the offset of the first and the last vertices.

Sliding a circle of radius r against the left side (the non-material side) of C makes contacts at points p_k. The portion of C that does not come in contact with the circle is called a *deficiency*. The two contact points p_{k-1} and p_k, bounding the deficiency $\langle p_{k-1}, v_j, p_k \rangle$, shown in Figure 5, are the *supporting points* of the circle. A supporting point can also be at a vertex in the chain, such as p_{k+1} shown in Figure 5. The concave upward portion of the circle between two supporting points p_{k+1} and p_{k+2} is called the *lid* of the deficiency, and is denoted by λ_{k+1}. Since C is monotone with respect to the x-axis and since the circle slides against the upper side of

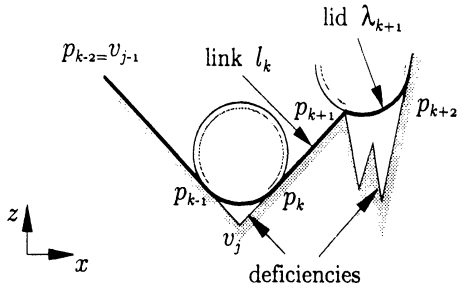

Figure 5: Examples of the defined terminologies.

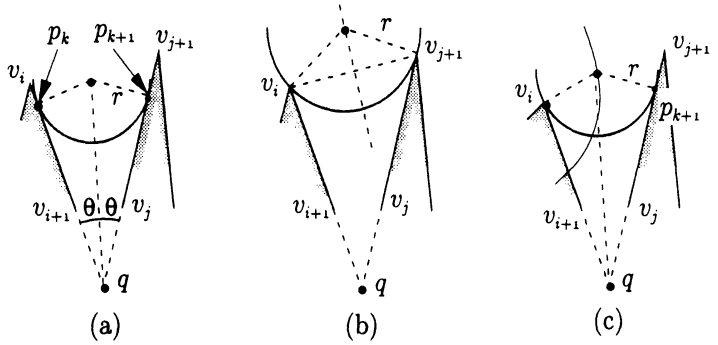

Figure 6: Pairs of supports: (a) slope-slope, (b) peak-peak, (c) peak-slope.

C, only the lower half of the circle may come in contact with C. Let the lower semi-circle containing the lid λ_{k+1} be denoted as π_{k+1}. An edge or a partial edge of C which comes in contact with the circle between p_k and p_{k+1} (supporting points or vertices of C) is called a *link* and is denoted by l_k. The concatenation of lids λ_i and links l_j forms the scallop hull of C.

The supporting points of the circle on a pair of edges can be computed analytically. Consider the edges e_i and e_j, as shown in Figure 6. Let the extensions of e_i and e_j intersect at q, and let $\angle v_i q v_{j+1}$ denote the angle between the non-material sides of these two lines and equal 2θ, where $\theta < 90°$. Let the distance between q and v_i be denoted as d_i, and likewise for distances to the other vertices. There are three combinations of supporting

points — slope-slope, peak-peak, and peak-slope — as shown in Figure 6. When it is a slope-slope pair, as shown in Figure 6 (a), the supporting points p_k and p_{k+1} are two interior points of e_i and e_j such that $d_k = d_{k+1} = r/\tan\theta$. The center and the two endpoints of the lid thus defined can then be computed. When it is a peak-peak pair, as shown in Figure 6 (b), the center of the lid lies on the perpendicular bisector of v_i and v_{j+1} such that its distances from v_i and v_{j+1} equal r. When, it is a peak-slope pair, as shown in Figure 6 (c), the center of the lid lies on the parabolic curve defined with v_i as the focus and edge e_j as the directrix such that its distance from v_i equals r. With the center of the lid identified, the supporting point p_{k+1} on e_j can then be computed.

However, the determination of the combination of supporting points requires testing all the possible cases. When $(d_{i+1} \cdot \tan\theta) < r < (d_i \cdot \tan\theta)$ and $(d_j \cdot \tan\theta) < r < (d_{j+1} \cdot \tan\theta)$, only a slope-slope pair of supporting points can be realized. Otherwise, i.e., if r and the distances of q to the four vertices of e_i and e_j do not satisfy the above relationship, whether it is a peak-peak or peak-slope pair can be determined by checking the validity for the rest of possible cases. Note that vertices v_{i+1} and v_j may also be the peak. Additionally, in the proposed algorithm, if e_i and e_j are not successive edges, supporting points are computed only when the two edges are detected to support the sliding circle. Therefore, by enumerating all the seven possible pairs, the combination of the supporting points can be verified, and the center and the endpoints of the lid can then be computed.

3 Construction of a scallop hull

The algorithm for constructing the scallop hull is analogous, in spirit, to the *Graham scan* algorithm [12] for computing the convex hull of a set of

points, though they differ in details. The Graham scan starts by sorting the given set of points lexicographically by their polar angles with respect to an arbitrary point in the interior of the convex hull of the set. Interior points are eliminated one by one by comparing three successive points. For the scallop hull, the deficiencies are equivalent to the "interior points". They are eliminated by comparing successive edges of the given monotone chain, and therefore no sorting is required.

Algorithmically, the monotone chain is traversed and, during the process, lids and links of the scallop hull are constructed and/or updated. A *current* scallop hull is maintained by a stack. Two procedures — *forward inclusion* and *backward exclusion* — are executed during the traversal. As suggested by their names, forward inclusion identifies segments (lids and links) that may be pushed into the stack, whereas backward exclusion removes or updates the segments in the stack. The lids and links remaining in the stack when the algorithm terminates constitute the scallop hull of the chain.

Algorithm (Scallop_Hull).

$\langle v_1, v_2, \ldots, v_n \rangle$: a monotone chain
S: the stack maintaining the segments in the current scallop hull
$k \leftarrow 1$, $p_1 \leftarrow v_1$, TOP(S) $\leftarrow$ *Link*
for $i = 2$ to $n - 1$ **do**
    Forward_Inclusion(S,e_{i-1},e_i)
OUTPUT(S).

The details in the two procedures are now discussed individually. The procedure **Forward_Inclusion** first determines if two successive edges, e_{i-1} and e_i, support a circle. At step 1, if the exterior angle of v_i, formed by e_{i-1} and e_i, is less than 180°, as shown in Figure 7 (a), the two edges

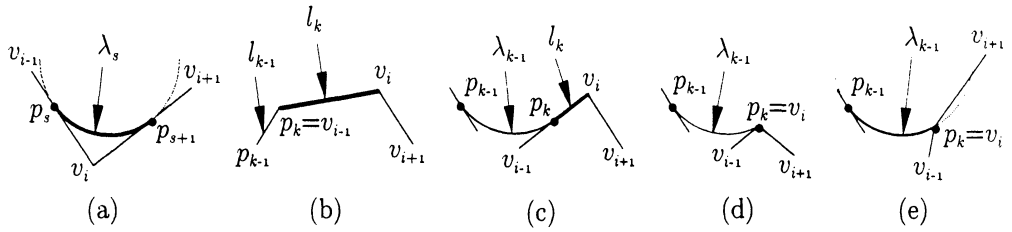

Figure 7: Cases in procedure **Forward_Inclusion.**

form a support. A lid thus computed is called the *active lid.* Procedure **Backward_Exclusion** is then invoked to determine if the active lid agrees with its preceding lids and links, and resolve the disagreement if there is any.

If the exterior angle of v_i, formed by e_{i-1} and e_i, is greater than or equal to 180°, as shown in Figure 7 (b), (c), and (d), e_{i-1} and e_i *do not* support the circle. Step 2 begins. If $p_k \neq v_i$, either the entire edge e_{i-1} (which occurs when the segment obtained in the previous iteration is a link, say l_{k-1}, as shown in Figure 7 (b)) or the part of e_{i-1} between v_i and the supporting point p_k (which occurs when the segment obtained in the previous iteration is a lid, say λ_{k-1}, as shown in Figure 7 (c)) is pushed into the stack as a link l_k. When $p_k = v_i$, as shown in Figure 7 (d), no link is generated. This corresponds to step 2.2.2. However, if π_{k-1} intersects e_i, as shown in Figure 7 (e), λ_{k-1}, the lid at the top of the stack, needs to be updated, and procedure **Backward_Exclusion** is then invoked with the updated λ_{k-1} as the active lid.

Procedure (Forward_Inclusion(S,e_{i-1},e_i)).

S: the stack maintaining the segments in the current scallop hull

λ_s: the active lid

1. **if** $\angle v_{i-1}v_iv_{i+1} < 180°$ **then**

Compute λ_s, such that $p_s \in e_{i-1}$ and $p_{s+1} \in e_i$

Backward_Exclusion(S,λ_s)

2. **else** [$\angle v_{i-1}v_iv_{i+1} \geq 180°$]

2.1. **if** TOP(S) = *Link* **then**

$p_k \leftarrow v_{i-1}$, $p_{k+1} \leftarrow v_i$

PUSH(l_k, S)

$k \leftarrow k + 1$

2.2. **else** [TOP(S) = *Lid* = λ_{k-1}]

2.2.1. **if** $p_k \neq v_i$ **then**

$p_{k+1} \leftarrow v_i$

PUSH(l_k, S)

$k \leftarrow k + 1$

2.2.2. **else** [$p_k = v_i$]

if $\pi_{k-1} \cap e_i \neq \emptyset$ **then**

UPDATE(λ_{k-1}), such that $p_k \in e_i$

$p_s \leftarrow p_{k-1}$, $p_{s+1} \leftarrow p_k$

$k \leftarrow k - 1$

Backward_Exclusion(S,λ_s).

Procedure Backward_Exclusion is responsible for checking if the active **lid λ_s, supported** by e_{i-1} and e_i, agrees with the segment at the top of the **stack.** TOP(S) can only be a lid or a link. If the segment at the top of the stack is a lid λ_{k-1}, then the intersection between π_{k-1} and e_i is checked. If the intersection is empty, the active lid is pushed into the stack; if there exists a part of the edge e_{i-1} between λ_{k-1} and the active lid, as shown in Figure 8 (a), this partial edge p_kp_s is pushed into the stack as a link before the active lid is. This corresponds to step 1.1. If, on the other hand, the intersection between semi-circle π_{k-1} and the edge e_i connecting v_i and v_{i+1}

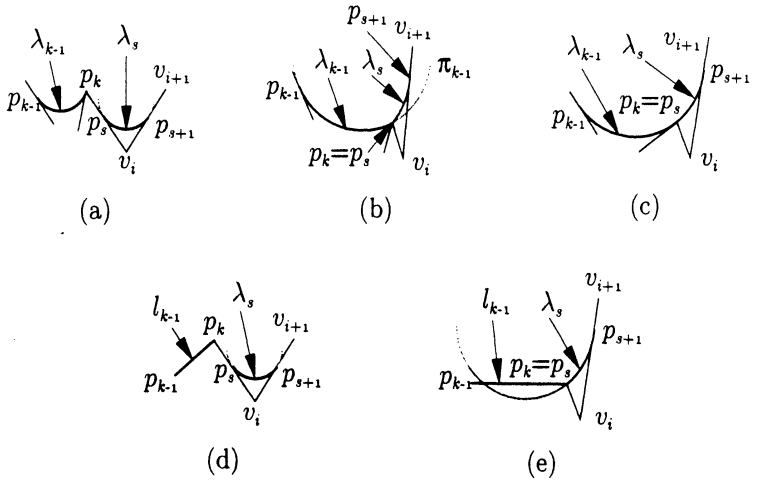

Figure 8: Some cases illustrating procedure **Backward_Exclusion.**

is not empty, as shown in Figure 8 (b), a new active lid supported by the edge containing the support p_{k-1} and e_i is computed. This corresponds to step 1.2. Procedure **Backward_Exclusion** continues with the new active lid. However, if π_{k-1} and the active lid coincide, as shown in Figure 8 (c), the lid λ_{k-1} is extended to p_{s+1}, and procedure **Backward_Exclusion terminates.**

If the segment at the top of the stack is a link l_{k-1}, the side of the active **lid to which** l_{k-1} lies is determined. At step 2.1, if l_{k-1} lies to the *concave* ***side*** **of the** lower semi-circle containing the active lid, the active lid is pushed into the stack. Again, if there exists a part of the edge e_{i-1} between l_{k-1} and the active lid, as shown in Figure 8 (d), this partial edge $p_k p_s$ is pushed into the stack as a link before the active lid is. At step 2.2., if, on the other hand, a portion of l_{k-1} lies to the *convex side* of the active lid, as shown in Figure 8 (e), a new active lid supported by the edges containing p_{k-1} and e_i is computed, and the link l_{k-1} is updated accordingly. Procedure **Backward_Exclusion** then continues with the new active lid.

Procedure (Backward_Exclusion(S,λ_s)).

S: the stack maintaining the segments in the current scallop hull

λ_s: the active lid

while TOP(S) $\neq$ *nil* **do**

1. **if** TOP(S) = *Lid* = λ_{k-1} **then**

1.1. **if** $\pi_{k-1} \cap e_i = \emptyset$ **then**

if $p_k \neq p_s$ **then**

$p_{k+1} \leftarrow p_s$

PUSH(l_k, S)

$k \leftarrow k + 1$

$p_k \leftarrow p_s$, $p_{k+1} \leftarrow p_{s+1}$

PUSH(λ_k, S)

$k \leftarrow k + 1$

EXIT

1.2. **else** [$\pi_{k-1} \cap e_i \neq \emptyset$]

if COINCIDE(π_{k-1},π_s) **then**

$\lambda_{k-1} \leftarrow \lambda_{k-1} \cup \lambda_s$

EXIT

else

UPDATE(λ_{k-1}, S), such that $p_k \in e_i$

$p_s \leftarrow p_{k-1}$, $p_{s+1} \leftarrow p_k$

$k \leftarrow k - 1$

2. **else** [TOP(S) = *Link* = l_{k-1}]

2.1. **if** $l_{k-1} \subset$ CONCAVE(π_s) **then**

if $p_k \neq p_s$ **then**

$p_{k+1} \leftarrow p_s$

PUSH(l_k, S)

$k \leftarrow k + 1$

$p_k \leftarrow p_s$, $p_{k+1} \leftarrow p_{s+1}$

PUSH(λ_k, S)

$k \leftarrow k + 1$

EXIT

2.2. **else** [$l_{k-1} \cap$ CONVEX(π_s) $\neq \emptyset$]

Compute λ_s, such that $p_s \in l_{k-1}$ and $p_{s+1} \in e_i$

UPDATE(l_{k-1}).

That the algorithm gives a correct scallop hull is now discussed. Forward inclusion computes, for all pairs of successive edges, circular arcs and line segments which may become the lids and links in the scallop hull. When a newly created link conflicts with the lid at the top of the stack (the only possible conflict between a link and the current scallop hull), as shown in Figure 7 (e), a new lid is computed as the active lid, which reduces this case to the case of having a newly created lid. Therefore, backward exclusion only needs to ensure that there is no conflict between an active lid and the current scallop hull. There are two cases of possible conflict. If the top of the stack is a lid λ_{k-1}, a conflict occurs when π_{k-1} intersects e_i, the second of the pair of successive edges in question, as shown in Figure 8 (b). On the other hand, if the top of the stack is a link l_{k-1}, a conflict occurs when a portion of l_{k-1} lies to the convex side of the active lid, as shown in Figure 8 (e). A new active lid is computed in both cases. The following lemma formalizes that if neither of these two cases occurs, there will not be any conflict between the active lid and any lids or links already in the stack.

Lemma *Suppose an active lid λ_s is identified in Forward_Inclusion. If the segment at the top of the stack is*

(i) a lid λ_{k-1} such that π_{k-1} does not intersect e_i, or

(ii) a link l_{k-1} such that it lies to the concave side of λ_s,

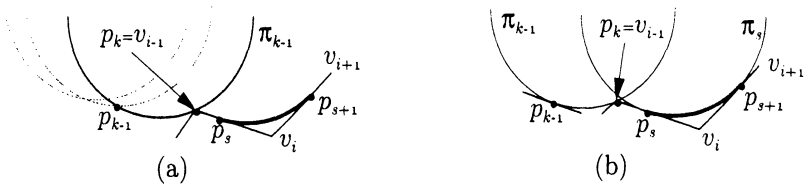

Figure 9: Termination conditions for Backward_Exclusion.

then, $e_i \cap \pi_{j-1} = \emptyset$ *and* $\pi_k \cap l_{j-1} = \emptyset$, *for all* $j < k$.

Proof. (i) Since C is monotonic with respect to the x-axis, the x coordinates of the centers of the tentative lids are ordered with respect to the x-axis. If π_{k-1} does not intersect e_i, then π_{j-1}, for $j < k$, which is to the concave side of π_{k-1} as shown in Figure 9 (a), cannot intersect e_i either. On the other hand, by induction, π_{k-1} does not intersect l_{j-1}, for $j < k-1$, for, otherwise, λ_{k-1} would have been updated in an earlier iteration. Moreover, since $x_j \leq x_{k-1} \leq x_s$ (as shown in Figure 9 (b)), for $j < k-1$, it is not possible for π_s to intersect l_{j-1}.

(ii) This part of the lemma can be established with reasoning similar to those for (i). □

The lemma establishes the stopping condition for Backward_Exclusion. When the stopping condition is met, the active lid will be a lid of the scallop hull for the chain from the beginning of the chain to e_i. By induction, the **Scallop_Hull** algorithm described above gives the scallop hull of the monotone chain C.

The time complexity of the algorithm for computing the non-gouging tool path for a monotone chain is now analyzed. There are only $(n+1)$ iterations for forward inclusion since there are only $(n+1)$ successive pairs of edges (including the two downward vertical rays). Updating the stack, which is the main task for backward exclusion, takes $O(n)$ time in total,

since there are at most $(n-1)$ lids and $(n+2)$ links in the scallop hull and each segment is stacked or unstacked at most once. Therefore, the time complexity for computing the scallop hull is bounded linearly by the total number of vertices in the chain. The offset of a scallop hull can be computed by concatenating the offset of the links and the vertices in the scallop hull. The offset of a lid is its center and can therefore be ignored. Since the offset of a scallop hull can also be computed easily in $O(n)$ time, the tool path for a monotone chain can be constructed in linear time. This establishes the following theorem:

Theorem *A non-self-intersecting tool path by offsetting a monotone chain of n vertices can be computed in $O(n)$ time.*

4 Summary and Conclusion

The scallop hull of radius r is a generalization of the convex hull (for which r equals infinity). An $O(n)$ time algorithm for computing the scallop hull of a monotone chain of n edges has been given; the algorithm is therefore optimal.

Application for the reported work includes NC machining and CMM inspection. That the mechanisms are 3-axis is underscored by the assumption of the monotonicity of the two-dimensional data. In three dimensions, monotonicity ramifies as a *terrain surface* [8], for which each point (x, y, z) on the surface is visible from "above", e.g., from $z = \infty$. Computing the offset for such a terrain surface has obvious utility in geographical information systems as well.

References

[1] A. Aggarwal, H. Edelsbrunner, P. Raghavan, and P. Tiwari. Optimal time bounds for some proximity problems in the plane. *Information Processing Letters*, 42(2):55–60, 1992.

[2] R. O. Anderson. Detecting and elliminating collision in nc machining. *Computer-Aided Design*, 10:231–237, 1978.

[3] I. T. Chappel. The use of vectors to simulate material removed by numerical controlled milling. *Computer-Aided Design*, 15:156–158, 1983.

[4] L.L. Chen and T.C. Woo. Computational geometry on the sphere for automated machining. *ASME Journal of Mechanical Design*, 114(2):288–295, 1992.

[5] Y. J. Chen and B. Ravani. Offset surface generation and contouring in computer-aided design. *ASME Trans., J. Mechanisms, Transmissions and Automation in Design*, 109:133–142, 1987.

[6] B. K. Choi and C. S. Jun. Ball-end cutter interference avoidance in nc machining of sculptured surfaces. *Computer-Aided Design*, 21:371–378, 1989.

[7] B. K. Choi, C. S. Lee, J. S. Hwang, and C. S. Jun. Compound surface modelling and machining. *Computer-Aided Design*, 20:127–136, 1988.

[8] Richard Cole and Micha Sharir. Visibility problems for polyhedral terrains. *J. Symbolic Comput.*, 7(1):11–30, 1989.

[9] H. Edelsbrunner, D. G. Kirkpatrick, and R. Seidel. On the shape of a set of points in the plane. *IEEE Trans. Information Theory*, IT-29(4):551–559, 1983.

[10] R. T. Farouki. Exact offset procedures for simple solids. *Computer-Aided Geometric Design*, 3:257–279, 1985.

[11] R. T. Farouki and J. A. Chastung. Curves and surfaces in geometrical optics. *Mathematics methods in CAGD II*, 1992.

[12] R.L. Graham. An efficient algorithm for determining the convex hull of a finite planar set. *Information Processing Letters*, 1:132–133, 1972.

[13] M. Held. *On the Computational Geometry of Pocket Machining.* Springer-Verlag, 1991.

[14] J. Hershberger. Finding the upper envelope of n line segments in $o(n \log n)$ time. *Information Processing Letters*, 33(4):169–174, 1989.

[15] J. S. Hwang. Interference-free tool-path generation in the nc machining of parametric compound surfaces. *Computer-Aided Design*, 24:667–676, 1992.

[16] R. Kimmel and A. M. Bruckstein. Shape offset via level sets. *Computer-Aided Design*, 25:154–162, 1993.

[17] T. Kuragaus. Fresdam system for design of aesthetically pleasing free-form objects and generation of collision-free tool paths. *Computer-Aided Design*, 24:573–581, 1992.

[18] D. T. Lee. Medial axis transformation of a planar shape. *IEEE Trans. Pattern Analysis and Machine Intelligence*, PAMI–4(4):363–369, 1982.

[19] F.P. Preparata and M.I. Shamos. *Computational Geometry: An Introduction.* Springer Verlag, 1985.

[20] T. C. Yang, S. Y. Shin, and K. Y. Chwa. Rolling discs and their applications. *Journal of Design and Manufacturing*, 2:71–82, 1992.