Scallop hull and its offset

Shuo-Yan Chou, Tony C Woo*, Lin-Lin Chen†, Kai Tang‡ and Sung Yong Shin§

A linear-time algorithm that computes the envelope 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 that of the convex-hull construction algorithm. The offset of the scallop hull, which yields the desired envelope, can then be computed in linear time from the scallop hull, giving a tool path.

Keywords: offsets, machining, inspection

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 centre 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 centre 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 path: 'direction-parallel'[1] paths, and 'contour-parallel' paths. This paper examines the former by taking the following view. Suppose that the representation is a graph surface $z=f_1(x, y)$. (A parametric surface $\mathscr{P}(u, v)$, 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 zigzag 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. Also, 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[2-11] deal with sculptured surfaces, and are concerned with tool collision (owing to the possible self intersection of the offset). This paper offers a novel concept, the *scallop hull* (a generalization of the familiar convex hull[12]), and applies it to the computation of a collision-free tool path, all in linear time, i.e. in time that is linearly proportional to the number of segments in the input curve. So that this novel idea is unencumbered 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, the 3-axis machine, is adopted. Because of the limited number of degrees of freedom, the chain must be *monotone*, i.e. single-valued in the z direction[13].

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 2a*. 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[1,14,15] of the chain to identify self intersections of the initial offset, and the other uses an algorithm developed by Hershberger[16] for constructing the envelope for a set of line segments. Both methods require $O(n \log n)$ time[17].

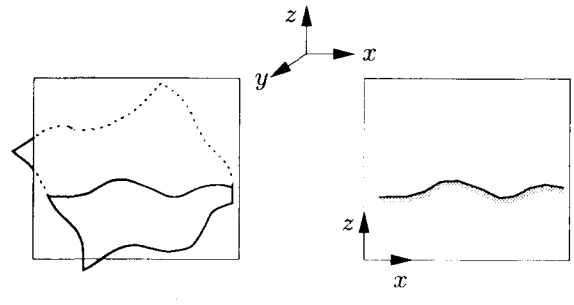


Figure 1 Sectioning a surface and simplifying its geometry

Department of Industrial Management, National Taiwan Institute of Technology, Taiwan
*Department of Industrial and Operations Engineering, University of Michigan, 10E Building, 1205 Beal Avenue, Ann Arbor, MI 48109-2117, USA
†Graduate School of Engineering Technology, National Taiwan Institute of Technology, Taiwan
‡CAD/CAM Division, Schlumberger Technologies, Ann Arbor, MI, USA
§Department of Computer Science, Korea Advanced Institute of Science & Technology, 373-1 Kusong-dong, Yusung-gu, Taejon 305-701, Korea
Paper received: 6 December 1993. Revised: 14 February 1994

0010-4485/94/07/0537-06

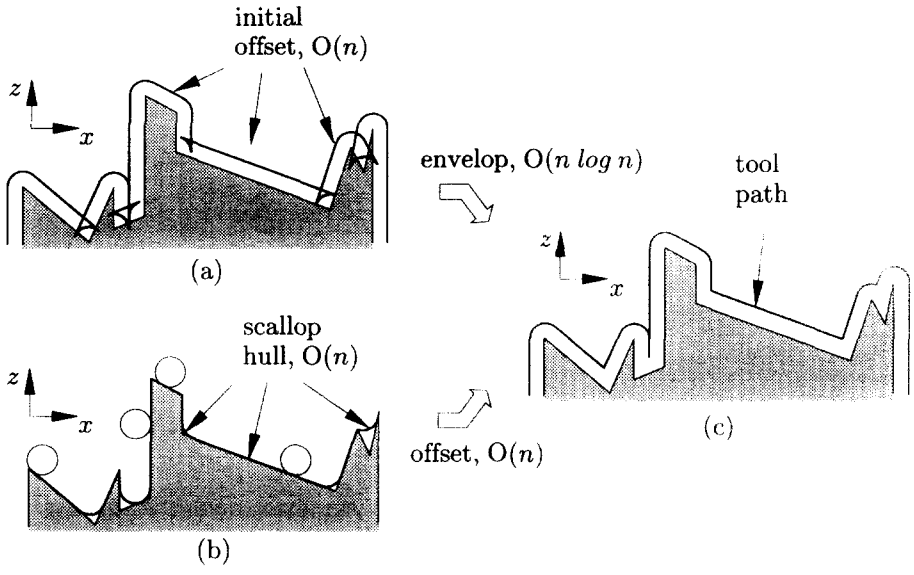


Figure 2 Intersection-free scallop hull

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 the α hull defined by Edelsbrunner, Kirkpatrick and Seidel[18] for capturing the 'shape' of a set of points. The parameter α is a real number. When it is positive, the α hull of a set of points is defined as the intersection of all the discs that have radii of $1/\alpha$, and contain all the points in the set. When α is negative, the α hull is the intersection of all the 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 α hull of a set of n points can be computed in $O(n \log n)$ time[18].

The *scallop hull* of a monotone chain is computed by sliding a circle against the boundary of the chain from its nonmaterial side (see *Figure 2b*). The monotonicity of the chain facilitates the amortized complexity for backtracking, hence allowing the development of a linear-time algorithm. The scallop hull has three kinds of element: convex vertices, circular arcs substituting for 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 centre 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 a storage cost, as shown in *Figure 4*. This takes $O(n)$ time, since there is no self intersection in the scallop hull. The total time is $O(n)+O(n)=O(n)$.

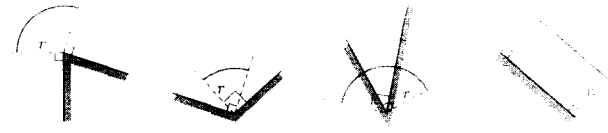


Figure 3 Initial offset of vertex and edge

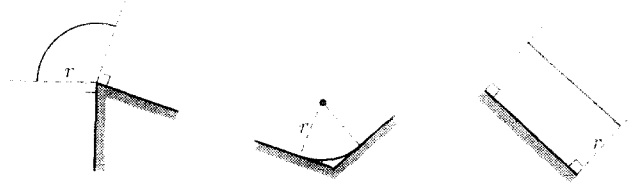


Figure 4 Offset of scallop hull

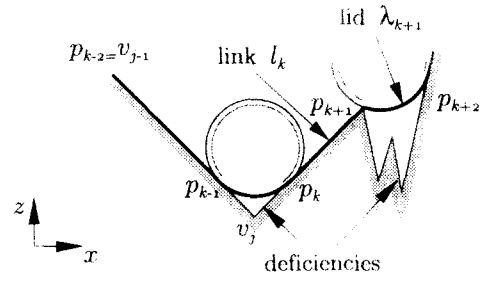


Figure 5 Examples of defined terminologies

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 it is denoted by e_i. Assume that e_i is directed such that $x_i \leqslant x_{i+1}$, for all $1 \leqslant i \leqslant n-1$. 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-hand side (the nonmaterial side) of C makes contacts at points p_k. The portion of C that does not come into 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 it 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 C, only the lower half of the circle may come into contact with C. Let the lower semicircle containing the lid λ_{k+1} be denoted as π_{k+1}. An edge or a partial edge of C which comes into contact with the circle between p_k and p_{k+1} (supporting points or vertices of C) is called a *link*, and it 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 nonmaterial sides of these two lines and let it 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*. For a slope–slope pair, as shown in *Figure 6a*, 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 centre and the two endpoints of the lid thus defined can then be computed. For a peak–peak pair, as shown in *Figure 6b*, the centre 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. For a peak–slope pair, as shown in *Figure 6c*, the centre 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 centre 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 the testing of 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, that is, 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 the 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 enumeration of all the seven possible pairs, the combination of the supporting points can be verified, and the centre and the endpoints of the lid can then be computed.

CONSTRUCTION OF SCALLOP HULL

The algorithm for constructing the scallop hull is analogous, in spirit, to the *Graham-scan* algorithm[19] for computing the convex hull of a set of points, though it differs 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,

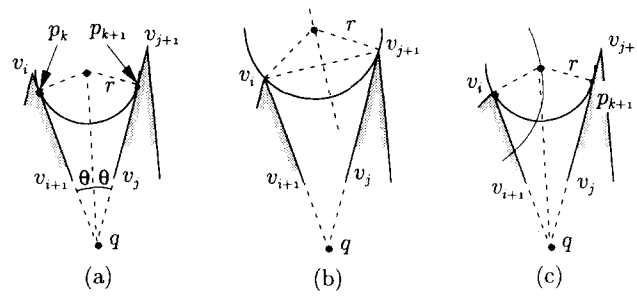


Figure 6 Pairs of supports; (a) slope–slope, (b) peak–peak, (c) peak–slope

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, 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 Forward_Inclusion procedure first determines whether 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 7a*, the two edges form a support. A lid thus computed is called the *active lid*. The Backward_Exclusion procedure is then invoked to determine whether the active lid agrees with its preceding lids and links, and to 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 *Figures 7b–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 7b*, 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 7c*), is pushed into the stack as a link l_k. When $p_k = v_i$, as shown in *Figure 7d*, no link is generated. This corresponds to Step 2.2.2. However, if π_{k-1} intersects e_i, as shown in *Figure 7e*, λ_{k-1}, the lid at the top of the stack, needs to be updated, and the Backward_Exclusion procedure is then invoked with the updated λ_{k-1} as the active lid.

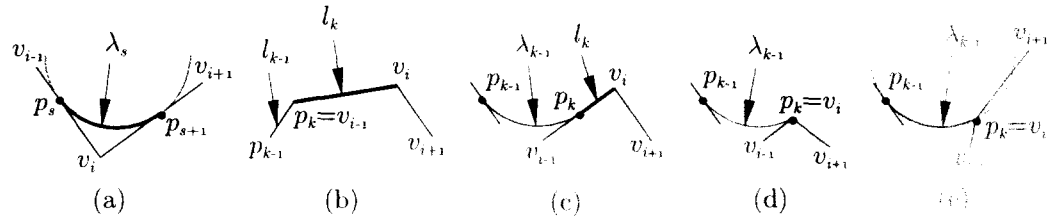


Figure 7 Cases in Forward_Inclusion procedure

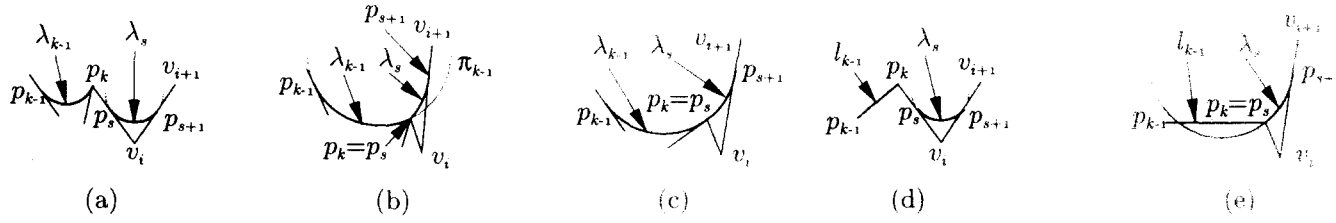


Figure 8 Some cases illustrating Backward_Exclusion procedure

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** $\angle v_{i-1}v_iv_{i+1} < 180^\circ$ **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** [$\angle v_{i-1}v_iv_{i+1} \geq 180^\circ$]

(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) = $Lid = \lambda_{k-1}$]

(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 $\pi_{k-1} \cap e_i \neq \varnothing$ **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)

The Backward_Exclusion procedure is responsible for checking whether 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 part of the edge e_{i-1} between λ_{k-1} and the active lid, as shown in *Figure 8a*, 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 semicircle π_{k-1} and the edge e_i connecting v_i and v_{i+1} is not empty, as shown in *Figure 8b*, 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. The Backward_Exclusion procedure continues with the new active lid. However, if π_{k-1} and the active lid coincide, as shown in *Figure 8c*, the lid λ_{k-1} is extended to p_{s+1} and the Backward_Exclusion procedure terminates.

If the segment at the top of the stack is a link l_{k-1}, the side of the active lid on which l_{k-1} lies is determined. At Step 2.1, if l_{k-1} lies on the *concave side* of the lower semicircle 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 8d*, this partial edge p_kp_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 on the *convex side* of the active lid, as shown in *Figure 8e*, 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. The Backward_Exclusion procedure 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) = $Lid = \lambda_{k-1}$ **then**

(1.1) **if** $\pi_{k-1} \cap e_i = \varnothing$ **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** [$\pi_{k-1} \cap e_i \neq \varnothing$]
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) $\neq \varnothing$]
Compute λ_s such that $p_s \in l_{k-1}$ and $p_{s+1} \in e_i$
UPDATE(l_{k-1})

The fact 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 7e*, 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 8b*. 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 on the convex side of the active lid, as shown in *Figure 8e*. A new active lid is computed in both cases. The following lemma formalizes the fact that, if neither of these two cases occurs, there will be no conflict between the active lid and any lids or links already in the stack.

Lemma: Suppose that an active lid λ_s is identified in the Forward_Inclusion procedure. If the segment at the top of the stack is (a) a lid λ_{k-1} such that π_{k-1} does not intersect e_i, or (b) a link l_{k-1} such that it lies on the concave side of λ_s, then $e_i \cap \pi_{j-1} = \varnothing$ and $\pi_k \cap l_{j-1} = \varnothing$, for all $j < k$.

Proof: (a) Since C is monotonic with respect to the x axis, the x coordinates of the centres 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 on the concave side of π_{k-1}, as shown in *Figure 9a*, cannot intersect e_i either. On the other hand, by induction, π_{k-1} does not intersect l_{j-1}, for $j < k-1$, because, otherwise, λ_{k-1} would have been updated in an earlier iteration. Moreover, since $x_j \leqslant x_{k-1} \leqslant x_s$ (as shown in *Figure 9b*), for $j < k-1$, it is not possible for π_s to intersect l_{j-1}. (b) This part of the lemma can be established using reasoning similar to that for part a. □

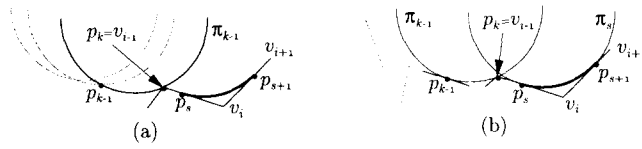


Figure 9 Termination conditions for Backward_Exclusion procedure

The lemma establishes the stopping condition for the Backward_Exclusion procedure. When the stopping condition is met, the active lid is 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 nongouging tool path for a monotone chain is now analysed. 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 in 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 centre, and it 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 created by offsetting a monotone chain of n vertices can be computed in O(n) time.

SUMMARY AND CONCLUSIONS

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.

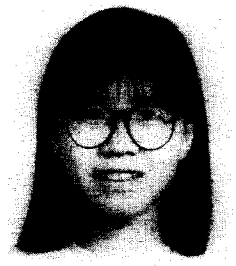
Applications for the reported work includes NC machining and CMM inspection. That the mechanisms are 3-axis is underscored by the assumption of the monotonicity of the 2D data. In 3D, monotonicity is ramified as a *terrain surface*[20], for which each point (x, y, z) on the surface is visible from 'above', e.g. from $z = \infty$. Computing the offset for such a terrain surface has obvious use in geographical information systems as well.

REFERENCES

1 Held, M *On the Computational Geometry of Pocket Machining* Springer-Verlag (1991)
2 Anderson, R O 'Detecting and eliminating collison in NC machining' *Comput.-Aided Des.* Vol 10 (1978) pp 231–237
3 Chappel, I T 'The use of vectors to simulate material removed by numerical controlled milling' *Comput.-Aided Design* Vol 15 (1983) pp 156–158
4 Chen, Y J and Ravani, B 'Offset surface generation and contouring

in computer-aided design' *ASME Trans. J. Mech. Transmissions & Automat. Des.* Vol 109 (1987) pp 133–142
5 Choi, B K, Lee, C S, Hwang, J S and Jun, C S 'Compound surface modelling and machining' *Comput.-Aided Design* Vol 20 (1988) pp 127–136
6 Choi, B K and Jun, C S 'Ball-end cutter interference avoidance in NC machining of sculptured surfaces' *Comput.-Aided Design* Vol 21 (1989) pp 371–378
7 Farouki, R T and Chastung, J A 'Curves and surfaces in geometrical optics' *in Mathematics Methods in CAGD II* (1992)
8 Farouki, R T 'Exact offset procedures for simple solids' *Comput. Aided Geom. Des.* Vol 3 (1985) pp 257–279
9 Hwang, J S 'Interference-free tool-path generation in the NC machining of parametric compound surfaces' *Comput.-Aided Des.* Vol 24 (1992) pp 667–676
10 Kimmel, R and Bruckstein, A M 'Shape offset via level sets' *Comput.-Aided Des.* Vol 25 (1993) pp 154–162
11 Kuragaus, T 'Fresdam system for design of aesthetically pleasing free-form objects and generation of collision-free tool paths' *Comput.-Aided Des.* Vol 24 (1992) pp 573–581
12 Preparata, F P and Shamos, M I *Computational Geometry: An Introduction* Springer-Verlag (1985)
13 Chen, L L and Woo, T C 'Computational geometry on the sphere for automated machining' *ASME J. Mech. Des.* Vol 114 No 2 (1992) pp 288–295
14 Lee, D T 'Medial axis transformation of a planar shape' *IEEE Trans. Pattern Anal. & Mach. Intell.* Vol PAMI-4 No 4 (1982) pp 363–369
15 Yang, T C, Shin, S Y and Chwa, K Y 'Rolling discs and their applications' *J. Des. & Manuf.* Vol 2 (1992) pp 71–82
16 Hershberger, J 'Finding the upper envelope of n line segments in $O(n \log n)$ time' *Inf. Proc. Lett.* Vol 33 No 4 (1989) pp 169–174
17 Aggarwal, A, Edelsbrunner, H, Raghavan, P and Tiwari, P 'Optimal time bounds for some proximity problems in the plane' *Inf. Proc. Lett.* Vol 42 No 2 (1992) pp 55–60
18 Edelsbrunner, H, Kirkpatrick, D G and Seidel, R 'On the shape of a set of points in the plane' *IEEE Trans. Inf. Theor.* Vol IT-29 No 4 (1983) pp 551–559
19 Graham, R L 'An efficient algorithm for determining the convex hull of a finite planar set' *Inf. Proc. Lett.* Vol 1 (1972) pp 132–133
20 Cole, R and Sharir, M 'Visibility problems for polyhedral terrains' *J. Symbolic Comput.* Vol 7 No 1 (1989) pp 11–30

Tony C Woo is a professor of industrial and operations engineering and a professor of mechanical engineering and applied mechanics at the University of Michigan, USA. He is active in CAD/CAM research.



Lin-Lin Chen is an associate professor of engineering and technology. Her research interests are in design for manufacturing, computational geometry, solid modelling, and geometric tolerancing. She received a BS in industrial design from Cheng-Kung University, Taiwan, in 1984, and a PhD in industrial and operations engineering from the University of Michigan, USA, in 1992. She worked as a design engineer at Sampo Corporation, Taiwan, from 1984 to 1986, and as a part-time computer programmer in the computer-aided engineering department of the Ford Motor Company from 1987 to 1989.

Kai Tang received a PhD in computer science from the University of Michigan, USA, in 1990. He also has a BSE in mechanical engineering. Since 1991, he has been with Applicon (formerly Schlumberger CAD/CAM) as a software specialist. His research interests include geometric algorithms in CAD/CAM, efficient tool-path generation and verification for complex sculptured surfaces in numerical-control machining, domain decomposition and partitioning in solid modelling, and spherical algorithms. His most recent interest is in the efficient computation of swept volumes under various geometric constraints.

Shuo-Yan Chou is an associate professor of industrial management. His research interests are in computational geometry and its applications in design, process planning, manufacturing, and inspection. He received a BBA in industrial management from Cheng-Kung University, Taiwan, in 1983, and an MS and a PhD in industrial and operations engineering from the University of Michigan, USA, in 1987 and 1992.

Sung Yong Shin received a BS in industrial engineering in 1970 from Hanyang University, Korea, and an MS and a PhD in industrial and operations engineering from the University of Michigan, USA, in 1983 and 1986. He is an associate professor in the Department of Computer Science at the Korea Advanced Institute of Science & Technology. His research interests include computer graphics and computational geometry.