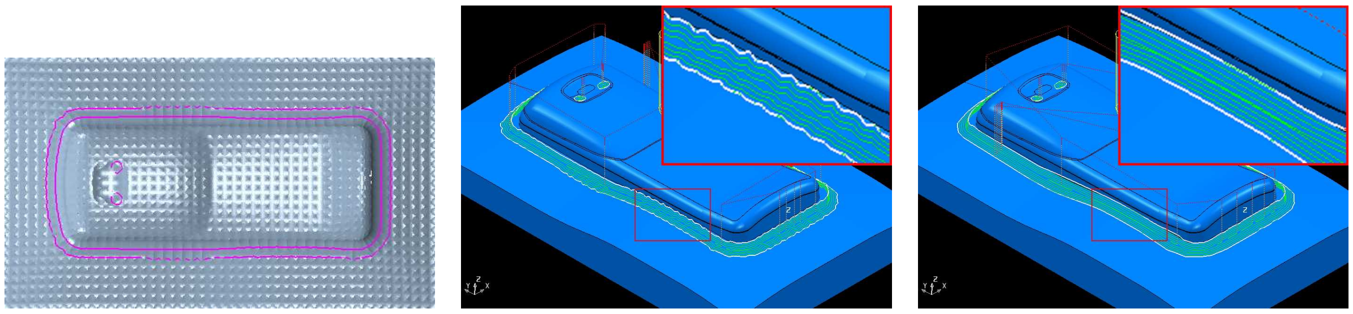
Merging and Smoothing Machining Boundaries on Cutter Location Surfaces

Weishi Li[*]
Hefei University of Technology, China

Ralph R. Martin [†]
Cardiff University, Wales, UK

Frank C. Langbein [‡]
Cardiff University, Wales, UK



(a) (b) (c)

Figure 1: *A phone mould: (a) rest model (the surface is not smooth due to a previous machining pass), and initial boundary curves; (b) initial boundary curves and derived tool paths, with close-up view; (c) smoothed boundary curves and derived tool paths, with close-up view.*

Abstract

In region machining, neighbouring regions may be close together, but disconnected. Boundary curves may also have unwanted geometric artifacts caused by approximation and discretisation. We present a strategy to improve the topology and geometry of such boundary curves, allowing the generation of better tool paths, and in turn, improved tool wear and surface quality of the machined part. We make such improvements in three steps: firstly, disconnected regions are merged where appropriate, using a method based on morphological operations from image processing. Secondly, boundary segments with undesirable geometric properties are identified and replaced by simpler segments, using a vertex deletion operation. Finally, flaws at a smaller geometric scale are removed, using a curve shortening algorithm. Experimental results are given to illustrate our algorithm.

CR Categories: I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—geometric algorithms, languages, and systems; J.6 [Computer Applications]: Computer-Aided Engineering—computer-aided manufacturing

Keywords: boundary, cutter location surface, region machining, merging, smoothing

1 Introduction

When machining free-form surfaces, the area to be machined is often divided into several cutting areas for various reasons, such as the need to remove remaining stock (rest) material, the desire to use different tool path generation strategies for particular areas, or a desire or need to use different tools or different part orientations [Choi et al. 2002; Radzevich 2005; Ren et al. 2005]. The extent of each area to be machined is typically determined by boundary curves on a cutter location (CL) surface, which contains the cutter's reference point as the cutter moves [Choi et al. 2002]. Methods used for computing such boundaries can be seen e.g. in [Park and Choi 2001]. Such an algorithm usually results in a piecewise-linear curve approximating some real, continuous, boundary on the CL-surface. For various kinds of region machining, especially rest machining, the regions may be undesirably fragmented into multiple pieces and the boundaries may also exhibit unwanted geometric artifacts, due to approximation and discretization, because of underlying surface artifacts, or simply because the correct answer has complex geometry [Flutter and Todd 2001], especially after previous tool passes—see Figure 1(a)[1].

The motivation for this work is that fragmented boundaries with poor geometry result in undesirable tool paths, both from the point of view of operating the machine tool, and in terms of quality of final surface finish. Long, smooth paths (and hence boundaries) are preferable for machining, especially high-speed machining [Hobbs 2007; Flutter and Todd 2001]—see Figure 1(c). Fragmented boundary curves should be merged, and undesirable geometric artifacts should be removed from boundaries.

We now specify the problem in more detail. We assume some algorithm has generated boundary curves B_i, $i = 0, \ldots, n$, delimiting certain regions on the CL-surface. Each boundary curve B_i is a piecewise linear curve in 3D *approximately* lying on the CL-surface, which we assume to be defined as a height field $z(x, y)$ over a bounded region of the x-y plane. Typically, while the *vertices* of each boundary curve lie *exactly* on the CL-surface, its *edges* do not. We assume the maximal *normal distance* between these piecewise linear, approximate boundary curves and the CL-surface is known (i.e. can be derived from a knowledge of upstream algorithms), and has the value δ. Any given boundary curve on the CL-surface separates the surface into two regions. The boundary curves may be nested, to any depth, with alternate regions being inside and

[*]e-mail: Weishi.Li@hotmail.com
[†]e-mail: Ralph.Martin@cs.cf.ac.uk
[‡]e-mail: F.C.Langbein@cs.cf.ac.uk

[1]In Figures 1(b,c), for clarity, the initial product model is shown rather than the rest model—the rest model may occlude the reader's view of the boundaries and tool paths.

outside the machining area; boundary curves do not intersect. We assume the outside is always on the right of each boundary curve. The CL-surface is not necessarily smooth, and may be composed of pieces which meet in sharp edges. In practice, to limit rates of vertical movement, it is desirable to treat nearly-vertical regions of the CL-surface with a slope bigger than a given threshold m as if they were vertical. We assume that the user specifies a smallest meaningful feature size d_{min} on the CL-surface—geometric detail at a smaller scale represents undesirable artifacts.

When *merging* boundaries and modifying their geometries, there are hard constraints on how much change is acceptable. When performing *boundary merging*, we assume that the user indicates which region boundaries can potentially be merged, as allowed by external considerations. The user also supplies a threshold ϵ; any two indicated regions whose minimal distance in x-y projection is less than ϵ should be merged, provided that there is no jump in height between them: boundaries should not be merged across nearly-vertical regions of the CL-surface. The spacing of vertices on newly created boundary segments should be comparable to that of the original boundaries. The shapes of newly created boundary segments should be suggested by the shapes of the original boundaries, and should merge them in a 'natural' way. The vertices of newly created boundary segments must lie on the CL-surface. The *merged* boundaries need not initially be particularly smooth, as later steps of our algorithm will subject them to boundary smoothing, as will other remaining original boundary segments.

When performing *boundary smoothing*, we assume the user provides two tolerances ε^+ and ε^- controlling allowable outward and inward motion of each boundary curve respectively. These need not be the same, and either may be set to zero. In general, the user may wish to specify the smoothing tolerances ε^+ and ε^- separately from the merging tolerance ϵ. Smoothing must keep all boundary curves free from intersections. The vertices of each final boundary curve must lie on the CL-surface. The maximal distance between the final boundary curves and the CL-surface, measured in the surface normal direction, should not exceed the input error δ for the original boundaries. The average length of any final boundary edges should be similar to that of the original boundary edges, where possible. The final boundary must not pass through nearly-vertical regions of the CL-surface. Furthermore the boundary should not go too close to the bottom of nearly-vertical regions of the CL-surface, to allow for the finite size of the cutter: in x-y projection, this distance should be no less than a distance d_v, determined by the cutter radius.

We solve this problem in the rest of the paper. Section 2 discusses related work while Section 3 summarises our method. A boundary merging algorithm is presented in Section 4, and boundary smoothing is discussed in Section 5. Practical results are shown in Section 6, and conclusions are given in Section 7.

2 Related Work

We are unaware of any existing algorithms specifically intended for improving fragmented and non-smooth boundary curves of the kind produced by region machining CAM algorithms. However, the fact that such problematic boundaries are produced is discussed in [Flutter and Todd 2001], and an expert at a leading CADCAM company has emphasized the real-world significance of this problem [Hobbs 2007]. We now discuss existing work concerning polygon merging and polygon smoothing.

In image processing, a widely used approach to merging regions is morphological closing [Soille 1999]. Unfortunately, under certain circumstances it may fail to close two regions which are less than a chosen minimum distance apart. Thus, in Section 4.2, we show how to *reliably* close *any* two such regions, using a medial axis transform (MAT) of a rasterised version of the regions.

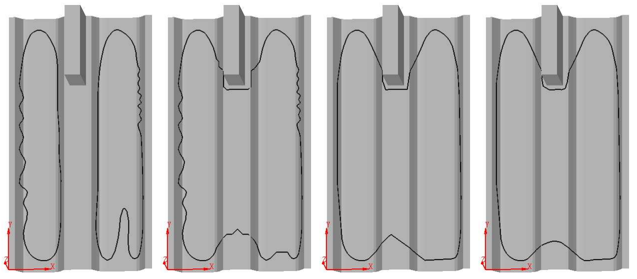


Figure 2: *Artificial example of boundaries on a CL-surface: (a) original boundaries, (b) merged boundary, (c) boundary after rough smoothing (d) boundary after final smoothing.*

Our finishing smoothing problem can be considered to be a piecewise linear curve fairing problem, with specific requirements. An algorithm for computing fair curves on surfaces was presented in [Hofer and Pottmann 2004]. It is difficult to extend such an algorithm to fair boundary curves on CL-surfaces as their height can only be evaluated pointwise, and doing so is potentially expensive.

Several algorithms have been proposed to fair *planar* piecewise linear curves, e.g. [Renz 1982; Mullineux and Robinsona 2007; Yang and Wang 2001]; fairing 3D piecewise linear curves has also been considered [Eck and Jaspert 1994]. Indeed, Choi et al's algorithm for fairing 3D piecewise linear curves [Choi et al. 2002] is intended for fairing of machining boundaries. However, we wish to smooth the boundary curves on a CL-surface while satisfying additional constraints. Furthermore, most algorithms with the exception of [Eck and Jaspert 1994] are *global* fairing algorithms, and potentially *all* vertices of the curve are adjusted. However, we wish to retain the original boundary wherever appropriate, and need a *local* fairing algorithm.

A *shortening flow* for polygons was presented in [Bruckstein et al. 1995], and is employed in this paper to fine tune the boundaries, taking into account their curvature and other properties.

3 Algorithm Overview

We now overview our strategy for improving input boundaries. Our algorithm both merges fragmented boundary curves, and smooths out geometric flaws in boundary curves. The process of merging boundaries may locally decrease smoothness, but smoothing a boundary curve certainly cannot disconnect it. Hence, merging is performed first, and then smoothing.

We call the input boundaries the *original boundaries*. After boundary merging, we have *merged boundaries*, which are then input to the boundary smoothing phase. These typically comprise some of the original boundary curve segments, plus some new curve segments generated by merging. The output boundaries after performing smoothing are the *final boundaries*. The process is illustrated using an artificial example of boundary curves on a CL-surface in Figure 2. Note the CL-surface contains a near-vertical region which the final boundary curve must avoid.

To perform boundary merging, we use a *closing operation*, based on ideas from image processing. We solve the *boundary merging* problem by treating it initially as a 2D problem in x-y projection, then consider the consequences in 3D. The 2D polygons arising from projecting the 3D boundary polygons are rasterised, and then merged using a closing operation based on morphological operations from image processing. Any pixels corresponding to nearly-vertical areas of the CL-surface are then excluded as the merged boundaries must not pass through such areas; rasterization allows us to do this in a simple manner. Finally, piecewise linear boundary curves are reconstructed and projected back onto the CL-surface to obtain the merged boundary.

In finishing smoothing, the goal is to keep maximum absolute geodesic curvature low, and also to minimise the number of zero-crossing points of signed geodesic curvature. Doing so helps to meet the goal of reducing tool acceleration and deceleration when machining the region bounded by the boundary. However, as it is time-consuming to estimate the normal of the CL-surface and hence the geodesic curvature, instead we use as a proxy the turning angle between successive projected boundary curve edges. (The CL-surface is a single-valued surface with respect to z, and furthermore the normal at any point of the CL-surface has a positive z component. Thus, the signed turning angle has the same sign as the signed geodesic curvature).

For efficiency, we use a *local* approach to smoothing the boundary curves—we only modify *flawed* parts of the boundary curves, and retain the rest. We classify these flaws into two types: local flaws and extended flaws. *Local flaws* occur at vertices where the boundary curve changes direction too suddenly, or where the turning angle of the boundary curve repeatedly changes sign from point to point. *Extended flaws* correspond to wobbles (repeated changes of sign of turning angle across several points), or deep concavities (or convexities, dependent on the level of the boundary curve and the direction in which the boundary may be modified).

We smooth each boundary curve in two steps: *rough smoothing*, which fixes issues associated with extended flaws, followed by *finishing smoothing*, which modifies local flaws.

In the rough smoothing step, *bad segments* of boundary are identified, and replaced by simpler segments which are guaranteed to both lie within the tolerance zone and avoid nearly-vertical regions of the CL-surface. In the finishing smoothing step, vertices at which local flaws exist are identified, and moved to nearby locations using a curve shortening flow (smoothing the curve), again while meeting the constraints.

During rough and finishing smoothing we must prevent the final boundary from passing too close to the bottom of nearly-vertical regions of the CL-surface; otherwise, the tool paths derived from these boundaries may cause interference between the tool and the surface of the given model. This is controlled by the given distance d_v. Our approach to meeting this requirement is to use a 2D *fat edge* [Duguet and Drettakis 2002] corresponding to each projected boundary curve edge. This is a closed 2D curve formed by offsetting each boundary curved segment all round by d_v . For example, the boundary shown in Figure 2(d) was obtained using fat edges with $d_v = 5$, where the size of the CL-surface shown is about 110×160 units.

We now detail in turn the boundary merging, rough smoothing, and finishing smoothing steps.

4 Boundary Merging

We now present our algorithm for merging boundaries. In region machining, some areas may only be expanded, while other areas may only be contracted, due to the requirements to cut away certain parts of the stock and leave other parts intact. Clearly, as merging *adds* extra parts of the CL-surface to the area to be machined, it

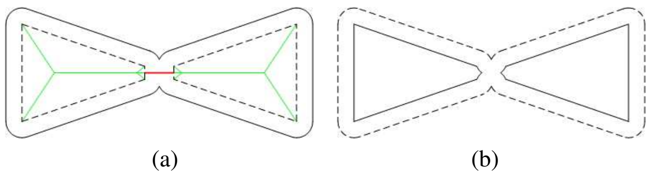


Figure 3: *Shortcoming of the standard closing operation: (a) dilation result; also showing its medial axis; (b) final closing result.*

is only applicable to regions which may be expanded. Two such selected regions should be merged if their boundaries lie within a distance ϵ of each other; the choice of this merging tolerance is determined by external, technological criteria [Hobbs 2007].

We initially consider merging the set of original boundary curves as a 2D problem, after projecting these curves into the x-y plane, and generate a tentative solution which is then checked in 3D. We use image processing methods to merge the boundaries instead of merging the polygons directly, as a raster makes it easier to provide guarantees concerning the requirement that nearly-vertical regions should be excluded.

The first task is to generate a gridded, or raster, representation of the 2D polygons. This is done using a standard computer graphics scan conversion algorithm.

4.1 Scan Conversion

To rasterise the polygon, a suitable grid interval d must be determined. By Nyquist's Theorem, d should satisfy $d \leq d_{\text{min}}/2$. Also, d should satisfy $d \leq l_e$ as stated in Section 1. Let $\bar{d} = \min(d_{\text{min}}/2, l_e)$. In subsequent processing we offset the polygons according to the tolerance ϵ. We can do so most accurately, at least for axis aligned rectangles, by ensuring the grid interval d is an exact fraction of ϵ, while also satisfying $d \leq \bar{d}$. Thus we set $d = \epsilon/n$, where n is the smallest integer greater than or equal to $\epsilon/\bar{d}$).

The merging process needs a little extra working space outside the polygon boundaries. The rasterised image is formed by taking a bounding box $\{(x_{\min}, y_{\min}), (x_{\max}, y_{\max})\}$ enclosing all polygons to be merged, and extending it outwards by a distance $e = \epsilon/2 + 2d$ in both positive and negative x- and y-directions. The lower-left corner of the grid is placed at $(x_{\min} - e, y_{\min} - e)$ and the grid size in pixels is thus $\lceil (x_{\max} - x_{\min} + 2e)/d \rceil \times \lceil (y_{\max} - y_{\min} + 2e)/d \rceil$.

Having chosen a suitable spacing and size for the grid, we classify each grid square as either (at least partly) *covered-by-some-polygon* or *not-covered-by-any-polygon*.

4.2 Image Closing

Our approach is based on morphological closing operations which are explained in [Soille 1999]. We use a *disc* as the structuring element because the merging distance should be the same in all directions. Its radius is $r = \epsilon/(2d)$ to meet the requirement that two polygons whose minimum distance is less than or equal to ϵ should be connected by the closing operation.

A problem with the standard image closing operation is that *erosion* may re-open some areas connected by *dilation*. An example is shown in Figure 3. In Figure 3(a), the dashed lines delimit the initial polygons, while the solid line indicates the boundary of the *single* region remaining after dilation. As the minimum distance between the two initial polygons is less than $2r$, the dilated result connects

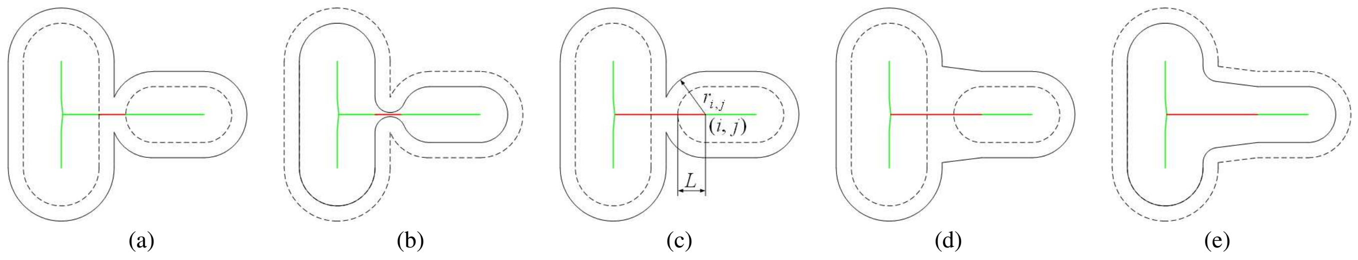


Figure 4: *Closing two regions with local MAT adjustment: (a) dilation result and its medial axis; (b) closing result using traditional morphological algorithm; (c) extension of the connection segment; (d) modified dilated shape reconstructed from the adjusted MAT; (e) final result after erosion.*

them. However, erosion *re-opens* the two regions as shown in Figure 3(b), where the dashed line is the dilation result and the solid lines denote the final boundaries.

This unwanted behaviour can be explained by considering the medial axis transform (MAT). In the dilated region, the associated radius of the MAT must be bigger than r for each medial axis pixel inside the initial polygons, but this is not necessarily true for any medial axis pixels outside the initial regions. If there are pixels on the MAT with associated radius less than or equal to r, the dilated region will be re-opened by the erosion operation. See Figure 3(a). The green and red segments give the medial axis of the dilated result; pixels coloured red may have associated radius less than r.

The medial axis segments of the dilated shape that lie outside the initial regions, connecting different regions, are referred to as *connection segments*. We examine the radii associated with each pixel of such connection segments to identify areas where the connected region may become disconnected by erosion. If the minimum associated radius at any such pixel is less than r, re-opening would occur during erosion if nothing were done to prevent it—see Figure 3. Furthermore, if the radius were less than $r + l$, where l is the length of the connection segment, the final result would include undesirably narrow shapes connecting the regions—see Figure 4(b). We thus adjust the associated radius function of connection segments of the MAT where the associated radius is less than $r + l$.

Before adjusting the radii, each identified connection segment is extended in both directions along the medial axis—see Figure 4(c). Doing so allows us to give the closing region a *natural shape*: the associated inscribing circles inside the original regions allow us to determine new radii for each pixel of the segment in a suitable way. Each segment is extended to whichever comes first: either the nearest medial axis branch pixel—see the left side of the connection segment illustrated in Figure 4(c), or the first pixel (i, j) along the medial axis which satisfies $L \geq r_{i,j} - r$, where $r_{i,j}$ is the radius associated with pixel (i, j), and L is the distance from (i, j) to the corresponding initial endpoint of the segment—see the right side of the connection segment illustrated in Figure 4(c).

A new associated radius is assigned to each pixel of each extended connection segment of the MAT, by linearly interpolating the MAT radii at the segment's endpoints, subject to the constraint that the difference between the old and the new associated radius is not more than r: this provides a smoothly varying shape, while ensuring that the distance between the merged boundary and the original boundaries can not exceed ϵ. After such adjustment, we have a new MAT $\bar{M}(D_S(f))$, which is in places the same as the MAT of $D_S(f)$. Our reliable closing operation is formally defined by

$$\hat{C}_S(f) = E_S(M^{-1}(\bar{M}(D_S(f)))). \tag{1}$$

The maximal distance between a pixel in $M^{-1}(\bar{M}(D_S(f)))$ and $D_S(f)$ is not greater than r and so the distance from $\hat{C}_S(f)$ to f is not greater than $\epsilon/2$, and erosion can no longer disconnect the dilated shape. The new region provides a *natural* connection between the initial regions because of the way we have interpolated appropriate MAT radii from these regions.

After the closing operation, the pixels in the new regions may undesirably cover nearly-vertical parts of the CL-surface, and these must be excluded for machining [Hobbs 2007]. This is done by checking each newly added pixels and removing it again if necessary. If multiple pixels are removed so as to re-disconnect two newly connected regions, all newly added pixels are removed between them, as these no longer make a useful contribution. Slightly more complex rules are need in the case of multiple newly connected regions.

4.3 Polygon Reconstruction

Having appropriately merged the raster representations of the original regions, polygons are now reconstructed from the result using a simple aproach. Where possible, the original boundary curves are kept, and new polyline segments are generated only where *new* regions are adjacent to the background.

5 Boundary Smoothing

Boundary smoothing is performed in two steps: rough smoothing, then finishing smoothing. The input may include both original boundary curve segments on the CL-surface, and segments of boundary curves arising from merging. The raster used for merging is also passed to the boundary smoothing process, as a basis for detecting nearly-vertical regions.

5.1 Rough Smoothing

The aim of rough smoothing is to remove *extended* flaws as far as possible while respecting the constraints: each smoothed boundary must lie within a tolerance ε^+ outwards, and ε^- inwards, of the input boundary, and at the same time, it must not cross any nearly-vertical regions. It is too time consuming to iteratively remove extended flaws using curve smoothing methods of the kind used for finishing smoothing. Instead, a quicker approach is used to fix extended, more significant problems of this kind: we simply *delete* certain boundary curve vertices identified as belonging to bad segments, which has the effect of replacing the latter by a smoother curve.

We first identify and label all *potentially deletable vertices* of each boundary curve, taking into account the *nature* of the tolerances ε^+ and ε^-. Contiguous spans of such vertices are marked as *candidate segments*. We then assess each candidate segment: if it represents a *deep concavity* or is a *wobble* in the boundary curve, it is marked as a *bad segment*. Here, a 'wobble' is a piece of curve whose curvature changes sign unnecessarily, bending this way and that. *Bad* segments are replaced by simpler pieces of boundary curve, using a *vertex deletion* operation; other candidate segments are left unchanged.

This rough smoothing process may introduce sharp corners into the boundary, but these will be removed during finishing smoothing.

5.1.1 Bad Segment Identification

We start by finding a set of *potentially deletable vertices*. If $\varepsilon^+ \geq \varepsilon^-$, we only attempt to delete concave vertices; otherwise we only attempt to delete convex vertices adjacent to concave vertices. Thus if a boundary curve can only be expanded, we may only delete concave vertices, as deletion of convex vertices would make the boundary contract. On the other hand, if a boundary curve can only contract, we may only delete convex vertices adjacent to concave vertices if we wish to retain sections of the original boundary. For a boundary curve that can be modified in either direction, we may attempt to delete vertices using either approach above, but again note that we wish to retain sections of the original boundary where it is feasible to do so.

The whole algorithm iteratively tests whether vertices could be deleted with respect to the tolerance ε^+ and ε^- until no further change might occur, as deleting a vertex of one boundary may allow some vertex of another boundary to also be subsequently deleted. For example, after a vertex is deleted, the vertices formerly adjacent to it may change from convex to concave, or vice versa. Each segment that corresponds to a set of contiguously deletable vertices of an original polygon P_i is a *candidate segment*.

We now identify which of the candidate segments are considered to be *bad segments* of the boundary curves. To assess wobbles, we compute the relative magnitude of total turning angle to total absolute turning angle:

$$m_\alpha(i,j) = \left|\sum_{k=i}^{j} \alpha_i\right| / \sum_{k=i}^{j} |\alpha_i|. \tag{2}$$

where α_i is the turning angle at boundary point p_i. The smaller m_α, the more severe the wobble in terms of oscillation of direction, and magnitude of changes in direction.

To identify deep concavities or long convexities, we compute:

$$m_c(i,j) = \max_{k=i,\cdots,j} \left(\frac{\|p_k - p_{i-1}\| + \|p_k - p_{j+1}\|}{\|p_{i-1} - p_{j+1}\|} \right). \tag{3}$$

Distances here are measured in 3D. For a concavity, the bigger m_c, the deeper the concavity.

If either tolerance exceeds a threshold, $m_\alpha < \kappa$ or $m_c > \lambda$, we flag the segment as a *bad segment*. Suitable values for κ and λ are dependent on the dynamic performance of the machine tool and the specific object to be machined.

5.1.2 Vertex Deletion

Vertex deletion is a simple operation which we now consider. Deleting a vertex from a polygon modifies the polygon at the scale of the triangle formed by the vertex and its two adjacent vertices. The input polygons are simple, and do not intersect each other. Thus, when we delete a vertex, as long as there are no vertices of the current, or any other, polygon *inside* (or on) the triangle formed by that vertex and its two neighbours in the x-y projection, the operation does not cause any polygons to *intersect* (or touch).

5.1.3 Bad Segment Simplification

Having determined the *bad segments*, we replace each one with a simpler segment, using the *vertex deletion* algorithm, while constraining the modified boundary to avoid nearly-vertical regions of the CL-surface.

We modify the bad segments by deleting one vertex at a time, evaluating the surface height $z(x, y)$ at appropriate locations to ensure avoidance of nearly-vertical regions. To do so, we use a discrete scheme with a regular 2D grid $\mathcal{G}$. The grid for boundary merging is reused here. (If the boundary merging step was deemed unnecessary and omitted, we sample the bounding box of the boundaries in x-y projection of the bad segment with a regular new 2D grid $\mathcal{G}$ with spacing $d_{\text{min}}/2$). When p_j is considered for deletion, the heights of all grid points inside $\triangle p_{j-1}p_jp_{j+1}$ (and inside the fat edge $p_{j-1}p_{j+1}$ on the CL-surface if $d_v > 0$) are evaluated using $z(x, y)$. If the slope between any two such adjacent grid points, or the slope between $p_{j-1}/p_j/p_{j+1}$ and the corresponding nearest grid point whose height has been determined, is bigger than m, $\triangle p_{j-1}p_jp_{j+1}$ at least partially overlaps a nearly-vertical region in projection, so p_j cannot be deleted. We use *fat edges* instead of edges of zero width if $d_v > 0$.

The vertex deletion process generally introduces new long edges into the boundary curves. Points are sampled on the CL-surface along each such long edge and inserted into it, to make the average 3D length of the output edges similar to L_e, the average 3D edge length of the original boundary curves.

5.2 Finishing Smoothing

The rough smoothing process may introduce sharp corners. These, together with any remaining *original* pointwise flaws in the boundaries, are removed by the final finishing smoothing scheme, while still ensuring all constraints remain satisfied.

A point p_j is considered bad if the turning angle satisfies either of the following conditions: $|\alpha_j| > \alpha_{\text{max}}$, or $\alpha_{j-1}\alpha_j < 0$ and $\alpha_j\alpha_{j+1} < 0$ and $(|\alpha_j - \alpha_{j-1}| > \alpha'_{\text{max}}$ or $|\alpha_{j+1} - \alpha_j| > \alpha'_{\text{max}})$. The smaller the values of α_{max} and α'_{max}, the smoother the curve obtained. Suitable values are dependent on the dynamic performance of the machine tool. Generally the higher the speed of machining, the smaller the values should be.

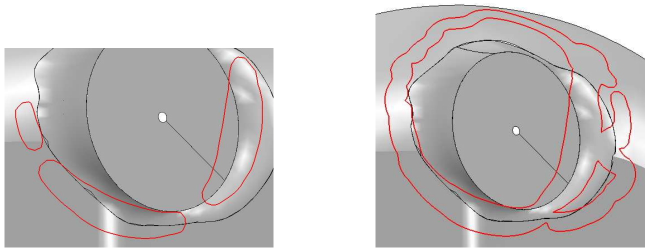
For each bad point p_j, we compute a new 2D position q_j using the discrete version of curve shortening flow given by Bruckstein [Bruckstein et al. 1995]:

$$q_j = p_j + \frac{c}{2}(p_{j+1} - 2p_j + p_{j-1}), \tag{4}$$

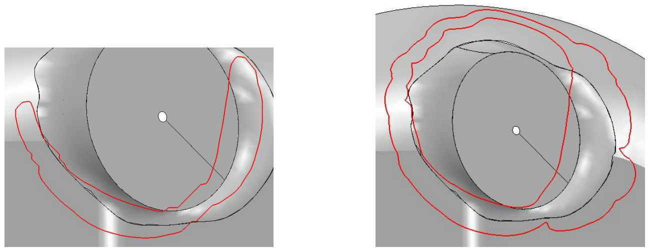
where c is a tuning parameter; $0 < c \leq 2/3$ results in a smooth curve [Bruckstein et al. 1995].

After computing q_j we take the constraints into account to ensure that we do not move the point to an new unacceptable location. If q_j *would* place p_j in an unacceptable location, instead we try moving its neighbouring vertices in the opposite direction where possible.

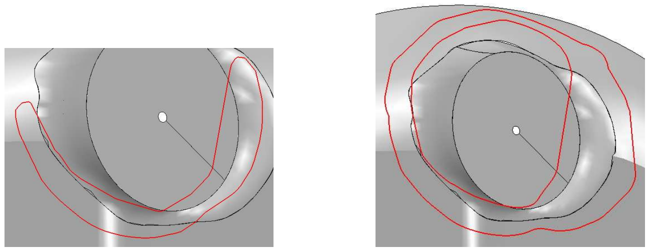
The new position of q_j of p_j must be inside the tolerance zone defined by ε^+ and ε^-, and updating the position of p_j to q_j should not result in any boundary curves intersecting (including self-intersections), or cutting across a nearly-vertical region, or passing too close to the bottom of a nearly-vertical region of the CL-surface when $d_v > 0$. The above are satisfied using the same general ideas as in Section 5.1.3. We must also ensure that the *edges* of the final smoothed curve lie within a user-specified distance δ of the true CL-surface. Smoothing is iterated until no point needs further modification.



(a)



(b)



(c)

Figure 5: *A combustion chamber(Left: three fragmented rest boundary curves; right: three noisy nested boundary curves): (a) original boundary curves, (b) merged boundary curves, (c) final boundary curves.*

6 Practical Results

All algorithms presented in this paper have been implemented as a DLL using C++, and integrated into Delcam's PowerMILL software for testing and evaluation. Space is only available for one further example. The model was provided by Delcam plc. For simplicity and ease of visualization, we do not compute and show the CL-surface itself, but only the product model from which the CL-surface is derived.

Two sets of initial boundary curves for a combustion chamber model are shown in Figure 5(a). For the left set, it is desirable to merge them into a single boundary curve. The boundary curves of the right set are nested, and bound a single region. Self-merging occurs in this example, erasing the pocket on the upper right. Merging also removes the smaller inner loop. The flaws introduced into each set of boundary curves during the boundary merging process are removed (locally, at least) by the rough smoothing and finishing smoothing.

7 Conclusions

Various algorithms are used to generate boundaries for region machining on a cutter location surface (CL-surface). Typically the boundaries output by these algorithms may be fragmented and exhibit geometric flaws, for various reasons. This paper has presented a strategy for improving such boundaries, while constraining the tool path to lie in certain areas, and avoid others. We can merge them appropriately, and remove unwanted artifacts. We are unaware of any existing algorithms specifically intended to do this.

Experiments have demonstrated to the satisfaction of a leading CADCAM company the success of our approach in improving boundaries, and consequently the tool paths generated from them, with positive benefits to machining including increased tool life, faster machining, and better surface finish.

Acknowledgements

The authors wish to acknowledge the support of Delcam plc, including many helpful discussions with Dr Steve Hobbs, and the assistance of Dr Chris Bainbridge when integrating our code with PowerMILL. This work was supported by EPSRC grant GR/T24579/01.

References

Bruckstein, A. M., Sapiro, G., and Shaked, D. 1995. Evolutions of planar polygons. *International Journal of Pattern Recognition and Artificial Intelligence 9*, 6, 991–1014.

Choi, B. K., Kim, B. H., and Jerard, R. B. 2002. Sculptured surface nc machining. In *Handbook of Computer Aided Geometric Design*, Elsevier, G. Farin, J. Hoschek, and M.-S. Kim, Eds., 543–574.

Duguet, F., and Drettakis, G. 2002. Robust epsilon visibility. *ACM Transactions on Graphics 21*, 3, 567–575.

Eck, M., and Jaspert, R. 1994. Automatic fairing of point sets. In *Designing fair curves and surfaces*, SIAM, N. S. Sapidis, Ed., 44–60.

Flutter, A., and Todd, J. 2001. A machining strategy for toolmaking. *Computer-Aided Design 33*, 13, 1009–1022.

Hobbs, S., 2007. Personal communication.

Hofer, M., and Pottmann, H. 2004. Energy-minimizing splines in manifolds. *ACM Transactions on Graphics 23*, 3, 284–293.

Mullineux, G., and Robinsona, S. T. 2007. Fairing point sets using curvature. *Computer-Aided Design 39*, 1, 27–34.

Park, S. C., and Choi, B. K. 2001. Boundary extraction algorithm for cutting area detection. *Computer-Aided Design 33*, 8, 571–579.

Radzevich, S. P. 2005. A cutting-tool-dependent approach for partitioning of sculptured surface. *Computer-Aided Design 37*, 7, 767–778.

Ren, Y. F., Zhu, W. H., and Le, Y.-S. 2005. Material side tracing and curve refinement for pencil-cut machining of complex polyhedral models. *Computer-Aided Design 37*, 10, 1015–1026.

Renz, W. 1982. Interactive smoothing of digitized point data. *Computer-Aided Design 14*, 5, 267–269.

Soille, P. 1999. *Morphological Image Analysis: Principles and Applications*. Springer.

Yang, X. N., and Wang, G. Z. 2001. Planar point set fairing and fitting by arc splines. *Computer-Aided Design 33*, 1, 35–43.