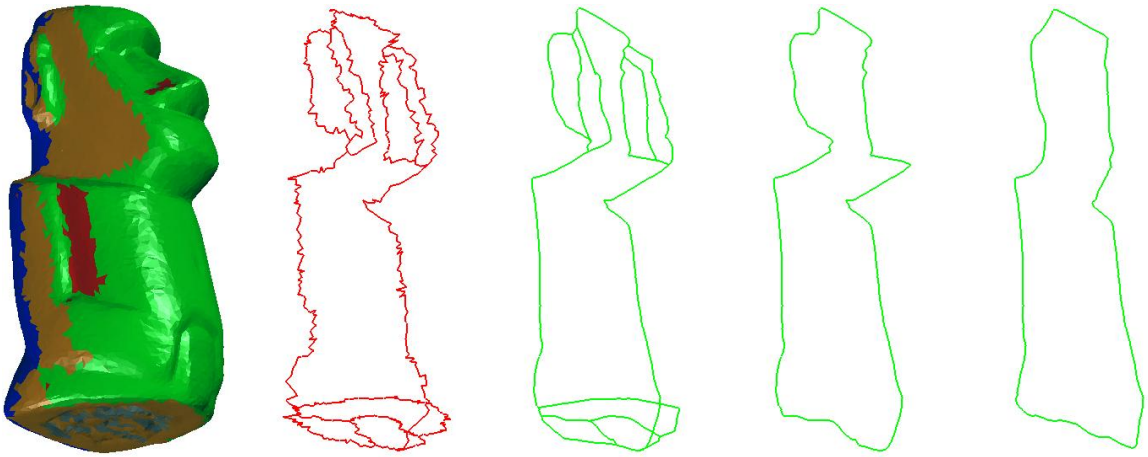
Generating Smooth Parting Lines for Mold Design for Meshes

Weishi Li* Ralph R. Martin† Frank C. Langbein ‡
Cardiff University, Wales, UK



(a) (b) (c) (d) (e)

Figure 1: *Parting line generation for moai model: (a) model and triangle band; (b) topological structure of candidate paths; (c) candidate paths; (d) chosen cycle; (e) final parting line.*

Abstract

This paper considers the mold design problem of computing a *parting line* for a complex mesh model, given a parting direction. Existing parting line algorithms are unsuitable for this case, as local variations in the orientations of the facets of such models lead to a parting line which zig-zags across the surface in an undesirable way. This paper presents a method to compute a *smooth* parting line which runs through a triangle band composed of triangles whose normals are approximately perpendicular to the parting direction. The skeleton of the triangle band is used to generate a structure representing distinct topological cycles, and to decompose the triangle band into singly-connected surface pieces, giving candidate paths. We choose a set of paths giving a good cycle; the final smooth parting line is then constructed by iteratively improving the quality of this cycle. Compliance in the physical material, or minor modifications to the surface itself, will ensure that such a parting line is appropriate for use.

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 design

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

Keywords: parting line, mesh, mold design

1 Introduction

A parting line is a curve on the surface of a model where the sections (cavity halves and cores) of a mold meet when the mold closes [Buckleitner 1995]. This paper considers the mold design problem of computing a *parting line* for a complex mesh model. Existing parting line methods are not adequate for *mesh models*, as local variations in the orientations of the facets of such models lead to a parting line which zig-zags across the surface. Although such a zig-zag parting line is theoretically correct, it is undesirable from a manufacturing point of view. A parting line that is smooth in 3D and closely approximates the theoretical parting line is more desirable for mold making [Barratt 2006; Priyadarshi and Gupta 2004]. Although this results in small undercuts, in principle meaning that the object is no longer removable from a two part mold, in practice, *either* elasticity and compliance in the materials used may still allow removal, *or* we may be prepared to slightly modify the mesh geometry locally to eliminate undercuts while leaving the final object close to the shape of the original object, yet also strictly removable from the mold. We intend to consider such mesh modification in future.

In this paper, we discuss genus-0 models only. However, our method is straightforward to extend to higher genus models—$n + 1$ curves are needed to separate a genus-n surface into two pieces [Rourke and Sanderson 1972], and thus for a genus-n model at least $n + 1$ parting lines are needed, depending on the shape of the model and the parting direction.

Given a parting direction, the parting line passes through certain points perpendicular to the viewing direction [Ravi and Srinivasan 1990]. Given a simple mesh model, and a parting direction, if the latter separates all triangles of the mesh into two contiguous regions—those visible from the parting direction, and those visible

from the opposite direction—the boundary is the parting line. For a complex mesh model, however, it may be impossible to find any parting direction which separates the triangles of the mesh into two such regions, and generation of a suitable parting line is harder.

We present a parting line generation method for complex triangle mesh models. We assume that we are provided with a closed, manifold mesh, and a desired parting direction—choice of parting direction is left as a separate problem. We do not assume the model is necessarily undercut-free along the given parting direction, but seek a *smooth* parting line which also minimises the undercut. For this we first construct a *triangle band* consisting of triangles nearly perpendicular to the parting direction, comprising an acceptable region through which the parting line may pass. We then generate the parting line within that region, taking into account that we require a smooth and flat curve, and that the parting line should also introduce as few undercuts as possible.

The process is controlled by an angular tolerance and a distance threshold. The *angular tolerance* is used in generating the triangle band: facets of the mesh which are perpendicular to the parting direction to within this angular tolerance are considered to be perpendicular, and the parting line is constrained to run through this region. A tight tolerance will result in a narrower band; a looser tolerance will give more freedom for where the parting line can go on the surface. The *distance threshold* is used during parting line generation. A tight threshold results in a parting line which is closer to the theoretically correct one; a looser threshold generally permits a smoother parting line, but possibly with more undercut. Typically a fairly large angular tolerance can be used, while the distance threshold is selected to take into account such factors as stiffness of the material being molded, desire for a smooth parting line, and desire for the final surface (after editing to agree with the new parting line) to be close to the input mesh—e.g. the detailed shape of a child's toy might be relatively unimportant. The user may also set an *area threshold* to eliminate small gaps in the triangle band e.g. caused by a noisy input mesh model. This generally improves the quality of the parting line while introducing very small local undercuts.

We next review related work in Section 2. Section 3 gives an overview of our method. An algorithm for triangle band generation is presented in Section 4. Then, we discuss the nature of the triangle band and the parting line in Section 5, and a method to determine possible topological paths for the parting line is given in Section 6. Our algorithm for computing the parting line is presented in Section 7. Experimental results are given in Section 8, and conclusions in Section 9.

2 Related Work

Parting direction and parting line computation are well-known problems in computer-aided mold design. An optimal main parting direction for the two mold cavity halves can be computed using Gauss map analysis of the model [Chen et al. 1993]. Determination of parting directions for cores is discussed by [Hui 1997]. After determining a suitable parting direction, the parting line may then be generated. However, many existing methods are designed for models bounded by planar, natural quadric, or freeform surfaces.

Parting line generation is usually based on *visible surface* detection algorithms. A point p on a model $\mathcal{M}$ is visible from a viewing direction if a ray r exists starting at p going in the opposite direction to the viewing direction such that it does not intersect any other part of the model: $r \cap \mathcal{M} = \emptyset$. We say a facet is *visible* if *every* point inside it is visible from the viewing direction, otherwise we say it is *occluded* (this includes partially occluded facets). If a facet is occluded from some direction, there is at least one other corresponding facet that is occluded from the opposite direction: the two facets occlude each other. The occluded parts of occluded facets constitute the undercuts of the model with respect to the parting direction [Ye et al. 2001]. Usually *cores* are added to enable undercut features to be molded [Hui 1997].

The connected regions of facets visible from the parting direction, and the reverse direction, can be used to construct the parting line [Fu et al. 2002]. [Tan et al. 1990] shows how to construct a parting line for a model with multiple visible regions. [Weinstein and Manoochehri 1997] gives a multi-objective optimisation method for parting line generation. Parting line complexity, draw depth, number of undercuts, number of side cores, and mold complexity are considered in the objective function. Some of these criteria depend on the choice of parting direction, which we assume is fixed in this paper. All of these methods generate a *theoretically correct* parting line as mentioned earlier.

Most recently, [Elber et al. 2005] showed how to use Gauss map analysis to compute both a parting direction and a parting line for a two-piece moldable NURBS model. [Khardekar et al. 2006] and [Chen and McMains 2006] have given graphics hardware accelerated algorithms to generate parting directions.

The idea of computing a parting line through a triangle band, which we also use, was proposed by [Majhi et al. 1999]. However, they only consider convex polyhedra, whereas our method is designed to cope with real-world, non-convex triangular meshes. For a convex polyhedron, the triangle band is topologically equivalent to a two-connected surface, or several singly-connected surfaces connected with edges. In comparison, the triangle band for a non-convex polyhedron is more complex: it may be topologically equivalent to an n-connected surface with $n > 2$, or several non-singly-connected and/or singly-connected surfaces connected with edges. Thus, generating the triangle band, and computing a parting line lying within it, are more complicated for non-convex polyhedra. It is not straightforward to extend the method by [Majhi et al. 1999] to this case.

Majhi et al's method has also been employed by [Priyadarshi and Gupta 2004] to design multi-piece molds. Recently, [Priyadarshi and Gupta 2006] proposed a method to find a triangle band using graphics hardware for complex mesh models. However, the method is dependent on the resolution of the hardware, and it is again not straightforward to use the proposed perturbation scheme to find a triangle band which consists of several strips.

In Section 7.1, we will need to cut the triangle band into several topological disks. Any mesh model with handles can be cut along a so-called *canonical polygonal schema* to get a topological disk [Erickson and Har-Peled 2004]. A simple method for computing this schema, based on the *collapsing operation* from topology [Rourke and Sanderson 1972], is given by [Lazarus et al. 2001]. We use a similar approach to cut the triangle band, an open surface *without* handles, into several topological disks.

3 Algorithm Overview

This section states our notation, and then provides an outline of our algorithm for finding a parting line on a triangular mesh model.

A triangle mesh is defined as $\mathcal{M} = (\mathcal{K}, \mathcal{P})$ where $\mathcal{K} = \mathcal{V} \cup \mathcal{E} \cup \mathcal{T}$ is an abstract simplicial complex representing the connectivity of the mesh. Here $\mathcal{V} = \{v_i\}$, $i = 0, \ldots, n_v$, $\mathcal{E} = \{e_i\}$, $i = 0, \ldots, n_e$, $\mathcal{T} = \{t_i\}, i = 0, \ldots, n_t$ are its sets of vertices, edges and triangles, respectively. We use a directed-edge data structure [Campagna et al. 1998] to represent the mesh. $\mathcal{P} = \{p_i \in \mathcal{R}^3\}$, $i = 0, \ldots, n_v$ is a set of vertex positions defining the shape of the mesh in $\mathcal{R}^3$. We henceforth identify vertices and their positions, and refer to them

with the same name and notation. The normal of t_i is denoted as $\mathbf{n}_i$.

For simplicity, both of exposition, and in the algorithm, we initially rotate the mesh to align the parting direction $\mathbf{D}$ with the z-axis. At various stages we will wish to consider the projection of various items onto a plane perpendicular to the parting direction, the x-y plane. Henceforth when we refer to a projection, it will mean parallel projection along the z-axis onto this plane.

Given a parting direction $\mathbf{D}$, all triangles t_i can be classified with respect to the angle θ_i between $\mathbf{D}$ and $\mathbf{n}_i$. Given a user-specified angular tolerance δ, all triangles can be classified into three types:

Up: Triangles satisfying $\theta_i < 90^\circ - \delta$.

Down: Triangles satisfying $\theta_i > 90^\circ + \delta$.

Neither: Triangles satisfying $90^\circ - \delta \leq \theta_i \leq 90^\circ + \delta$.

Up and **Down** triangles can be further classified into **up visible** and **up occluded**, and **down visible** and **down occluded** triangles. **Up visible** and **down visible** triangles are visible from $-\mathbf{D}$ and $\mathbf{D}$, respectively. Groups of connected **up visible** triangles construct **up visible** *regions*; likewise **down visible** triangles. Any occluded triangles correspond to undercuts with respect to $\mathbf{D}$. We consistently colour these regions differently in the Figures in this paper: green regions are **up visible**, blue regions are **down visible**, dark yellow regions belong to triangle bands, gray regions are occluded regions within the triangle band, and other regions are filled with red. Note that all triangle bands shown in Figures in Sections 4–7 have been flattened for presentation on the 2D plane of the paper. The parting direction cannot be readily visualized in such Figures.

The classified triangles are used to construct the triangle band. With a non-zero angular tolerance the mesh triangles usually do not form just one contiguous **up visible** region adjacent to just one contiguous **down visible** region. Instead, **Neither** triangles, or undercuts, or both, generally exist between the boundaries of the visible regions. The region outside the outermost boundaries of each visible region, excluding any undercuts, is the acceptable region in which the parting line must lie: the *triangle band*. The triangle band usually has a finite width, but may degenerate locally into an edge, either where common boundaries exist between **up visible** and **down visible** regions, or where an undercut is excluded from the triangle band (see e.g. Figure 4). We call such edges *degenerate edges* in the triangle band.

If there are *multiple* **up visible** or **down visible** regions, or occluded triangles exist inside the triangle band, the triangle band is not just a two-connected surface (a strip whose ends are joined), but is n-connected, with $n > 2$ topologically distinct paths connecting any two points in the band. Thus, several topologically valid cycles may lie in the triangle band; we must choose the desired cycle from amongst them.

Having found the triangle band, we thus perform the following steps sequentially to generate the parting line; details are given in subsequent Sections:

1. *Find the connectivity of the topological paths through the triangle band, allowing us to generate all possible cycles*.
   We do this by firstly generating a a particular skeleton of the triangle band. We then remove paths passing between certain occluded regions inside the triangle band. We also add paths between certain regions visible from the same direction.

   Paths in the connectivity graph constitute *topologically valid cycles* which fully separate the **up visible** regions from the **down visible** regions. We call such paths *candidate paths*.

   Note that the connectivity graph is not just an abstract graph—it also contains geometric information. The skeleton used to generate it is designed to get lead to a final cycle that well approximates the exterior profile of the mesh, to reduce the time spent on cycle optimisation.

2. *Find geometric candidate paths in the triangle band with the desired topology*.
   We use two new, different skeletons of the triangle band to decompose it into singly-connected regions. We then compute *geometric* paths joining adjacent regions using the shortest path algorithm of [Lanthier et al. 1996]. These paths comprise some of the candidate paths for the topologically valid cycles. The *degenerate edges* are also added to them to obtain the full set of candidate paths. The connectivity of all candidate paths is topologically equivalent to the connectivity graph.

3. *Choose the best cycle by considering weighted lengths of candidate paths*.
   The geometric candidate paths allow construction of various cycles, guided by the topological information. We compute *weighted* lengths of these cycles to then select the cycle upon which to base the final parting line. The weights are used to take into account the requirements that the parting line should be smooth, and close to the theoretical parting line. For efficiency we take into account that in practice, certain paths must be present *whichever* cycle we choose, allowing us to divide cycle selection into several smaller subproblems.

4. *Improve the near-optimal cycle iteratively to get the parting line*.
   We now have an overall geometric path for the chosen cycle. We discard parts of the triangle band through which the cycle does not pass, to restrict the topology to the selected cycle. An improved path is computed iteratively in the restricted triangle band, again using the shortest path algorithm of [Lanthier et al. 1996]. The final result is a smooth parting line, whose projection closely approximates the projection of the theoretical parting line.

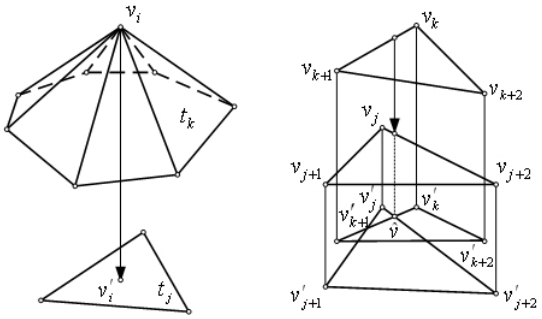
Above, we find paths of shortest *weighted* length. The weighting is done to ensure that the selected cycle, and the final parting line, take into account:

- *Planarity of the parting line*
  [Ravi and Srinivasan 1990; Majhi et al. 1999] discuss why this is desirable.

- *Smoothness of the parting line*.
  Manufacturing considerations prefer a parting line that is smooth in 3D to a zig-zag line which may be correct in theory.

- *Undercuts introduced by the parting line*.
  Undercuts are of two kinds: certain undercuts are due to choice of parting direction, and cannot be avoided. Other undercuts are due to deviation of the chosen parting line from the exact parting line. In projection, the exact parting line is identical to the exterior profile of the mesh. We use deviation from the exterior profile to measure the amount of undercut introduced by a particular parting line.

Our method takes the above criteria into account implicitly rather than explicitly by using a weighted shortest path algorithm. Locally, on a surface, the *shortest* path is the smoothest and flattest path, while *weighted* distances, measuring deviation of the parting line from the exterior profile, penalise undercut.



(a) (b)

Figure 2: *Visible triangle detection.*

4 Triangle Band Generation

This section gives our algorithm for generating the *triangle band*, the region that the parting line must run through, given a mesh, a parting direction, and an angular tolerance δ. First we detect *visible* triangles, and from these we identify upper and lower boundaries of the triangle band. Then, region growing is used to determine the triangles inside the triangle band; we also identify any areas where the triangle band degenerates to edges.

4.1 Visible Triangle Detection

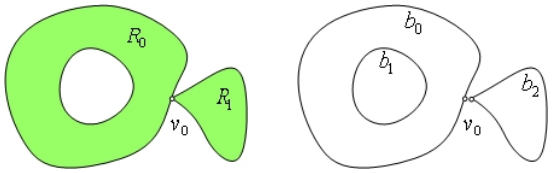
We first classify each triangle as **Up**, **Down**, or **Neither** as defined previously. We next decide which **Up** and **Down** triangles are **occluded**, as explained next. All non-occluded triangles are **visible**. Note that *partially* occluded triangles are marked as occluded.

The topological relationships between the vertices v_i, edges e_i, and triangles t_i, when projected onto the x-y plane, is employed to detect visible triangles. We must consider two cases:

1. In projection, if v_i is inside t_j, then t_j and those triangles incident to t_i and facing t_j are (mutually) **occluded**. E.g., in Figure 2(a), v_i' is the projection of v_i, lying inside t_j in projection. t_k is a triangle incident to v_i. If $\mathbf{n}_j \cdot \mathbf{n}_k < 0$ and $\mathbf{n}_j \cdot v_i v_i' < 0$, t_j and t_k face each other, and occlude each other.

2. Triangles near the boundaries of visible regions may occlude each other even if neither has vertices inside them in projection—they may may overlap edge-to-edge. Thus, we consider the triangles adjacent to pairs of edges intersecting in projection. If two such triangles, one adjacent to each edge, face each other, they are (mutually) **occluded**. E.g. see Figure 2(b). Projections $v_k' v_{k+1}' v_{k+2}'$ and $v_j' v_{j+1}' v_{j+2}'$ of $v_k v_{k+1} v_{k+2}$ and $v_j v_{j+1} v_{j+2}$ intersect, e.g. at $\hat{v}$ on $v_k' v_{k+1}'$ and $v_j' v_{j+2}'$. If $v_k v_{k+1} v_{k+2}$ faces $v_j v_{j+1} v_{j+2}$, they are both **occluded** although neither contains the vertices of the other.

To efficiently carry out the above tests, we store the projected vertices in a kD-tree, and use bounding boxes of the projected triangles to quickly find those vertices which might have an inclusion relationship with each projected triangle. The Bentley-Ottmann sweepline algorithm [O'Rourke 2001] is further used to improve the efficiency of determining intersections between edges.



(a) (b)

Figure 3: *Boundary tracing.*

4.2 Triangle Band Boundary Identification

From the **Up** and **Down** visible triangles, it is straightforward to find the boundaries of each **Up** and **Down** visible region. We can then identify the boundaries of the triangle band.

Any edge whose adjacent triangles are of different kinds is a *boundary* edge. If one triangle is a **down visible** triangle, and the other triangle is not, the edge is a boundary edge of a **down visible** region; similar logic applies to **up visible** regions. Note that some edges may be boundaries of both **Up** and **Down** visible regions.

By stepping along edges belonging to such boundaries, we can find the complete boundary of each visible region. Note that more than two boundary edges may be incident to a boundary vertex (e.g. see vertex v_0 in Figure 3(a)). We start tracing at any boundary edge, remembering the start vertex, and on which side of the boundary edge the visible triangle lies. If just one other edge runs from the end vertex of the current edge, we step to the next edge. If there is a choice of next edge, we step to the next clockwise boundary edge if the visible triangle is on the left of the current edge, or the next counter-clockwise edge if the visible triangle is on the right. Thus, visible triangles always lie on the same side of the boundary, guaranteeing that in projection the boundary is a simple polygon. We continue tracing until we return to the start point. Figure 3(b) shows the boundaries of R_0 and R_1: b_0 and b_1 are the boundaries of R_0, and b_2 is the boundary of R_1. The boundaries are displaced by a small distance for visual clarity, and hence v_0 in Figure 3(b) is drawn twice.

The boundary of the triangle band can be identified using a point-in-polygon test for simple polygons [O'Rourke 2001]. The boundaries of separate **up visible** (and **down visible**) regions do not intersect each other, and are disconnected except possibly for isolated points, such as v_0 in Figure 3: no common edge exists between any two boundaries. The projection of each boundary is a simple polygon. Thus, considering two boundaries b_i and b_j, in projection, if the midpoint of any edge of b_i is inside b_j, b_i is inside b_j; otherwise, b_i is outside b_j. By deciding the relationships between boundaries pairwise, the *outmoster* boundaries can be identified (in Figure 3, b_0 and b_2 are outermost boundaries). The outermost boundaries of the **up visible** regions give the upper boundaries of the triangle band, while the outermost boundaries of the **down visible** regions give the lower boundaries of the triangle band.

4.3 Finding the Triangle Band

A region growing method is used for the last step of triangle band generation. **Neither** triangles that are adjacent to the edges of the boundaries are selected as seeds, from which we grow to find all **Neither** triangles between the boundaries. Note that we must grow from all boundaries, as the triangles of the triangle band may not be contiguous if the triangle band contains degenerate edges (see Figure 4). This may occur either where there are common edges between the boundaries of **up visible** and **down visible** regions, or

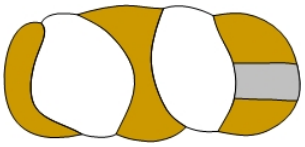


Figure 4: *A triangle band with degenerate edges.*

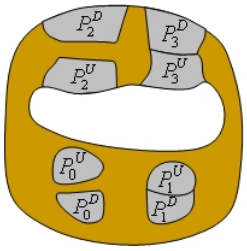


Figure 5: *A triangle band with occluded regions.*

where the boundaries of occluded regions contact the boundaries of the triangle band (see Figure 4). Note that such edges must be identified and considered as parts of the triangle band.

The topological structure of the triangle band can be simplified. If the mesh model is noisy, some **up visible**, **down visible** or **occluded** regions may have a very small area in projection. By labelling triangles in such areas as **Neither**, according to a user-selected parameter as mentioned earlier, we can decrease the complexity of the triangle band. Simplifying the topology of the triangle band both reduces the amount of computation needed later, and generally improves the quality of the final parting line at the expense of introducing small undercuts. An alternative method to simplify the triangle band might be to make a minor adjustment to the parting direction, but we have not tried doing so as it would seemingly involve an expensive search.

5 The Triangle Band and the Parting Line

In general, the triangle band cannot be used simply and directly to compute the parting line, for the reason that the triangle band may not be a two-connected surface, for two possible reasons.

Firstly, **Up** and **Down** occluded regions may exist in pairs within the triangle band. The parting line may not pass through the region between (i.e. separate) any pair of mutually occluded regions if the molded part is to be removable from the mold. This is true, regardless of whether these regions meet along a common boundary (see Figure 5, U_1^U–P_1^D and U_3^U–P_3^D; the yellow region is the triangle band, and the gray regions are occluded regions within it), or not (see U_0^U–P_0^D and U_2^U–P_2^D). As a separate issue, we note that the boundaries of occluded regions may also meet the upper or lower boundaries of the triangle band (see U_2^U–P_2^D and U_3^U–P_3^D); here the triangle band simply locally degenerates into an edge.

Secondly, there may be multiple visible regions, both **Up** and **Down**. The parting line is required to separate *all* **up visible** regions from *all* **down visible** regions, again so that the model can be removed from the mold (using side cores if necessary). If multiple **up visible** and **down visible** regions exist, there are *several* topologically different cycles which correctly separate the **up visible** and **down visible** regions. See, for example, Figure 6. There are two **up visible** regions, P_0^U and P_1^U (note that P_0^U is *outside* the triangle band in this *flattened* representation), and three **down visible** regions, P_0^D, P_1^D and P_2^D, leading to three topologically valid cycles separating **up visible** regions from **down visible** regions, as shown by dashed lines in Figures 6(b)–(d).

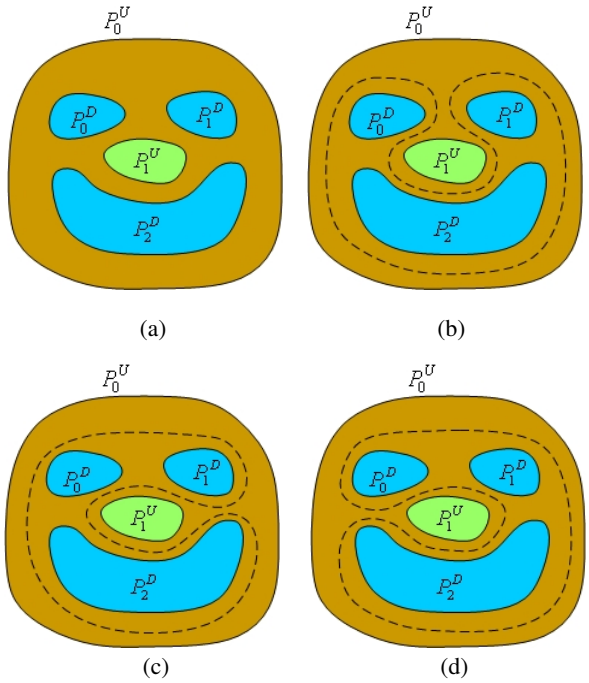


Figure 6: *Multiple **up visible** and **down visible** regions, allowing topologically different parting lines.*

We must take both of the above issues into account when determining the parting line. To do so, we generate a topological candidate path structure from the skeleton of the triangle band and use it to construct all valid topologically distinct cycles which separate the **up visible** and **down visible** regions. The skeleton is also used in Section 7.1 to decompose the triangle band into singly-connected regions to compute the the geometric candidate paths for constructing the cycles.

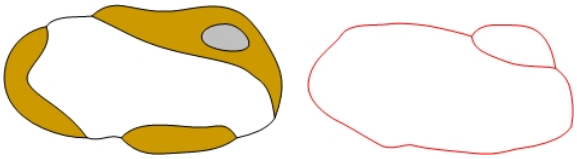
6 Topological Paths for the Parting Line

We now describe an approach for generating skeletons of the triangle band, and how we use a particular skeleton to determine possible topological paths for the parting line.

6.1 Triangle Band Skeleton Generation

The skeleton of the triangle band is a collection of edges that represents the connectivity of those parts of the triangle band through which the parting line may pass. The triangle band is an open triangle mesh, possibly including degenerate edges. A triangle band with degenerate edges and its skeleton are illustrated in Figure 7. We use a method similar to the *canonical polygonal schema* generation method of [Lazarus et al. 2001]. However, our method can generate the skeleton of an open mesh with degenerate edges directly, while degenerate edges have to be considered separately by the method of [Lazarus et al. 2001].

Skeleton generation is performed using a simple *erosion* operation on the triangle band's triangles. The idea is to successively remove triangles from the triangle band which are adjacent to non-triangle-

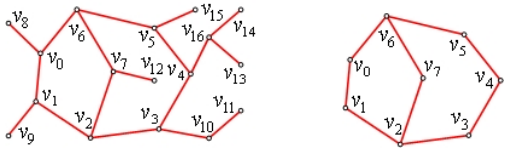


(a) (b)

Figure 7: *A triangle band and its skeleton.*

(a) (b)

Figure 8: *The skeleton generating operation.*



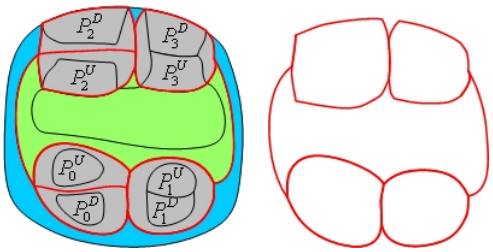
(a) (b)

Figure 9: *Removing side branches of the skeleton.*

band triangles. Suppose two triangles, t_j (belonging to the triangle band) and t_i (belonging to some other *particular* kind of region, such as **up visible**), meet in e_k. We call a triangle like t_i an *erosion triangle*. We label t_j with yellow, and its edges with red; t_i has its edges other than e_k labelled black: see Figure 8(a). Initially, the red edges are the edges of the triangle band, and the black edges are other edges of the mesh. An erosion step is performed as follows: if a red edge is adjacent to a yellow triangle and an erosion triangle, we change the label of the edge and the yellow triangle to the label of the erosion triangle. Note that after this operation, the vertices of the yellow triangle are still connected by red edges. See Figure 8(b). In this way we may gradually relabel triangle band triangles until the skeleton is found.

The generation of the skeleton of a triangle band is carried out by eroding the triangle band using a *particular* kind of triangle. We can choose various possibilities: **up visible** triangles, **down visible** triangles, etc. Doing so results in different skeletons, but in each case they have the *same connectivity* as the triangle band. We first initialise the labels of all edges and triangles: blue for **down visible** triangles; green for **up visible** triangles; yellow for triangles in the triangle band; red for all edges of the triangle band (including degenerate edges) and black for all other edges. Applying erosion steps repeatedly to the edges and triangles of the triangle band until no further change is possible, the end result is that all red edges constitute a skeleton of the triangle band. Erosion is performed one ring of triangles at a time from the starting triangles.

After the above process, the skeleton will generally have side branches—see edges connecting v_k, $k = 8, 9, 11$–15 in Figure 9(a). Removing such side branches does not change the cycle structure of the skeleton, but simplifies the skeleton. Side branches are removed iteratively: after the removal of $v_{13}v_{16}$ and $v_{14}v_{16}$, v_4v_{16} becomes a side branch, and is also removed. After doing so, the skeleton in Figure 9(a) is simplified to the one in Figure 9(b).



(a) (b)

Figure 10: *Path breaking between pairs of occluded regions.*

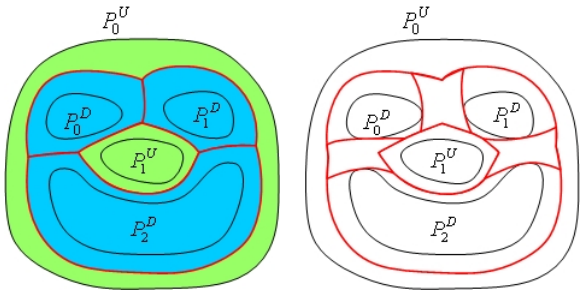
6.2 Topological Structure Generation for Paths

The skeleton of the triangle band corresponds to a *topological structure* representing all possible paths running through the triangle band. This takes the form of a connectivity graph, where each edge corresponds to some polyline on the mesh. To produce this graph, we first generate a skeleton using **up visible**, **down visible** and **occluded** triangles as erosion triangles simultaneously. This particular choice of erosion triangles produces a skeleton which, in projection, generally approximates the exterior profile better than skeletons generated using other triangle choices (such as the two skeletons used later in Section 7.1). This is important as we also later use the *geometric* information in the connectivity graph to determine a good initial cycle which well approximates the exterior profile, as a basis for generating the final parting line.

Although this skeleton represents the connectivity of the triangle band, we must make adjustments to it to determine the candidate paths because (i) the parting line is forbidden from passing between each pair of mutually occluded regions, and (ii) if multiple **up visible** and multiple **down visible** regions exist, we must ensure that a cycle can be found using skeleton segments which separates the **up visible** regions from the **down visible** regions. Thus, in general we must add and remove paths from the skeleton before we can generate the topological structure of the candidate paths.

To solve the first problem, we remove any segment of the skeleton passing between a pair of occluded regions. After generating the skeleton using the erosion operation, we can easily identify segments of the skeleton lying between a pair of occluded regions: triangles adjacent to such segments are labelled gray on either side, and have grown from the same pair of **occluded** regions. The skeleton of the triangle band in Figure 5 is shown in red in Figure 10(a), supposing the region inside the triangle band is **up visible** . After removal of the segments between P_0^D–P_0^U and P_2^D–P_2^U, the skeleton is reduced to the structure shown in Figure 10(b).

To solve the second problem above, we duplicate segments of the skeleton between adjacent **up visible** regions, and adjacent **down visible** regions, the basic idea being to provide a *return path*. We can easily identify segments of the skeleton lying between two regions visible from the same direction: the labels of triangles on either side of such segments are the same (either blue or green). Such segments are removed from the skeleton. Next, we identify the **Neither** triangles that are vertex-adjacent to the boundaries between visible regions with the same label, to produce a region between the two visible regions. Two segments of the boundary of this region exist and are not currently included in the skeleton. These are added to the skeleton, and provide a path and the corresponding return path between the two visible regions. We illustrate the idea using the triangle band from Figure 6(a); its skeleton is shown in red in Figure 11(a). It is not difficult to see that no cycle can



(a) (b)

Figure 11: *Path duplication for regions visible from the same direction.*

be constructed simply using the segments of the skeleton in Figure 11(a) which separates the **up visible** and **down visible** regions. After duplicating those segments of the skeleton separating adjacent blue regions, we get the red structure shown in Figure 11(b). We can construct all possible cycles separating the **up visible** and **down visible** regions, as given in Figures 6(b)–(d). Apparently, if the cycle enters the area between two regions visible from the *same direction*, the cycle then must go back along the return path, i.e. if a path between two regions visible from the same orientation is included in a cycle, the corresponding return path must also be: otherwise, the cycle may be unable separate **up visible** regions from **down visible** regions.

We call the structure generated after adjusting the skeleton as above the *connectivity graph* of the candidate paths. All topologically valid cycles can be constructed by selecting edges from this structure. We next select a particular cycle from the structure, and its geometric realization; it is then optimised to produce the parting line.

7 Parting Line Generation

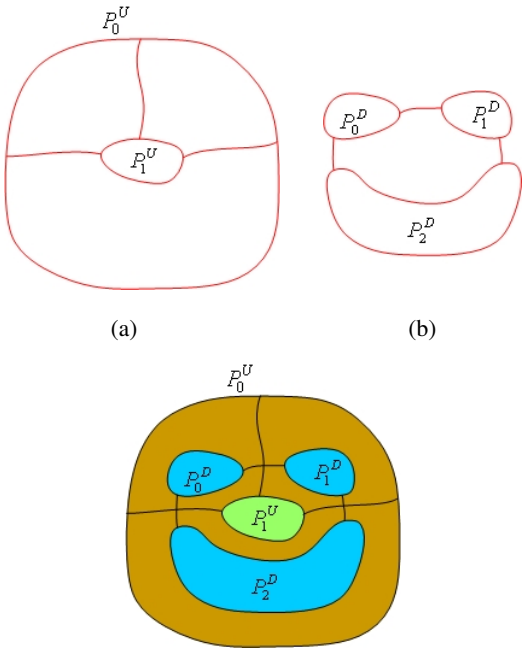
We now explain how to generate the parting line. Note that in general, the triangle band is an n-connected open surface ($n \geq 2$), so we must solve both a topological and a geometric problem.

We must first compute actual geometric paths whose connectivity is topologically equivalent to the connectivity graph. This is achieved by decomposing the triangle band into singly-connected regions. We compute suitable geometric paths between adjacent regions using a shortest path algorithm, using the connectivity graph to tell us adjacency. Secondly, these paths can be linked in various different cycles; we search for a near-optimal cycle. Thirdly, we improve this cycle to produce the final smooth parting line.

7.1 Triangle Band Decomposition

We first discuss how to decompose the triangle band into singly-connected regions; these are subsequently used to compute geometric paths with the same connectivity as the connectivity graph. This decomposition is most easily done using *two different* skeletons of the triangle band as we now explain. Note that this information cannot readily be deduced otherwise from the skeleton we have already computed.

Firstly, we select **down visible** triangles as erosion triangles and find a skeleton. This skeleton partially coincides with the boundaries of **up visible** regions and occluded regions inside the triangle band. This skeleton decomposes a triangle band with m lower



(a) (b)

(c)

Figure 12: *Triangle band decomposition using skeletons.*

boundaries into m two-connected regions. We then select **up visible** triangles as erosion triangles and find another skeleton. The skeleton partially coincides with the boundaries of **down visible** regions and occluded regions inside the triangle band. This skeleton decomposes a triangle band with n upper boundaries into n two-connected regions. Simultaneously overlaying the two skeletons onto the triangle band decomposes it into a set of singly-connected regions. Consider the triangle band shown in Figure 6(a). The two skeletons generated using **down visible** and **up visible** erosion triangles are shown in Figures 12(a) and (b), respectively. Overlaying both skeletons on the triangle band simultaneously decomposes it into six regions, all singly-connected, as shown in Figure 12(c).

In practice, the problem may be more complex, in two ways. Skeleton edges may exist which have the same region on either side, and must be handled specially—if we do not do this it is more difficult to compute geometric paths that are topologically equivalent to the connectivity graph. Secondly, the above approach does not decompose two-connected triangle band into two singly-connected regions, and a further step is needed.

Consider the case where skeleton edges have the same region on both sides: see Figure 13. There are two **up visible** regions P_0^U and P_1^U, and one **down visible** region P_0^D (outside the triangle band in the flattened figure). The skeletons generated using **down visible** erosion triangles and **up visible** erosion triangles are shown in Figures 13(b) and 13(c). The two skeletons decompose the triangle band into two regions, shown in turquoise and lavender in Figure 13(d). Note that two boundary segments, shown in red, have the same region on either side, in one case the turquoise region, and in the other, the lavender region. These must be further decomposed, so that each piece of boundary has different regions on either side. This decomposition is done by making seed regions on either side of the problematic edge, and growing them away from the boundary until they meet elsewhere on the region (the erosion principle is again used for this purpose). This gives a new boundary which decomposes the region into two singly-connected regions. For ex-

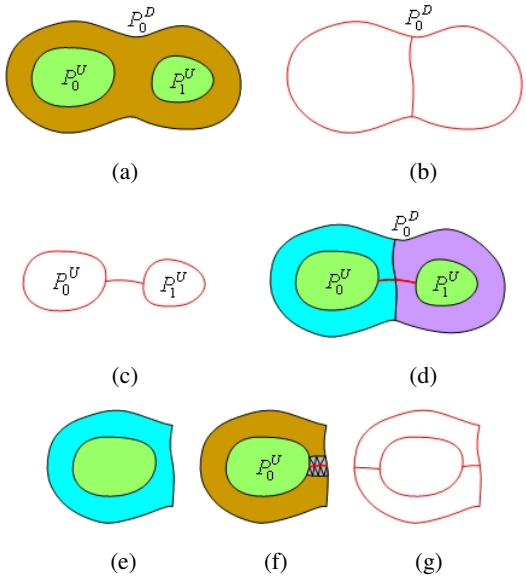


Figure 13: *Decomposing a pseudo-singly-connected region.*

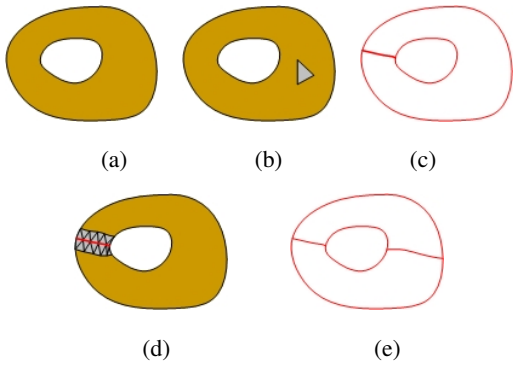


Figure 14: *Decomposing a two-connected triangle band.*

ample, for the turquoise region in Figure 13(e), we make seeds incident to the single-region-edges as shown in Figure 13(f), and then grow outwards until they meet, giving the new edge to be added to the skeleton shown in Figure 13(g). This decomposes the region into two singly-connected regions.

We now consider the case where the triangle band forms a 2-connected surface, as in Figure 14(a). This happens when the triangle band only has a single upper boundary and a single lower boundary, with no common edge, and no occluded triangles in the triangle band. We first randomly select a triangle in the triangle band—see Figure 14(b), and grow outwards from it until the growing region meets itself elsewhere in the band, as shown in Figure 14(c). This results in an edge with the same region on both sides, which we then decompose as in the previous case: see Figures 14(d) and (e).

Using the above method, we can decompose any triangle band into a set of singly-connected regions.

7.2 Path Computation

We next compute geometric candidate paths based on the connectivity graph, and the singly-connected regions just generated. These

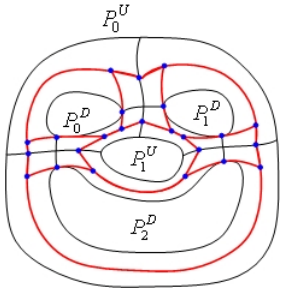


Figure 15: *End points for geometric candidate paths.*

paths are used as the basis for finding the cycle chosen as a basis for the parting line. First we identify suitable end points for these paths, and then generate suitable paths between these end points using a weighted shortest path algorithm.

We use end points of two kinds. We first collect all junctions of the connectivity graph where more than two edges meet. We then add all intersections between the connectivity graph and the boundaries of the singly-connected regions. The latter are necessary as we want each path to lie entirely within a single region, to ensure that each path can only pass through an appropriate part of the mesh. The end points for the triangle band shown in Figure 6(a) are shown in blue in Figure 15. The connectivity graph is next used to identify between which pairs of end points paths are needed.

We next use the algorithm in [Lanthier et al. 1996] to compute the weighted shortest path between each appropriate pair of end points. In general, the shortest path between two points on a surface is locally the smoothest and flattest curve between them. By using a weighted distance we can also ensure that the path is not too far in projection from the exact parting line.

We define the *weighted distance* between any two points p_i and p_j of an edge of the path to be

$$D(p_i, p_j) = w_i w_j \|p_i p_j\|. \tag{1}$$

where $\|p_i p_j\|$ measures distance on the mesh and the weights are defined by

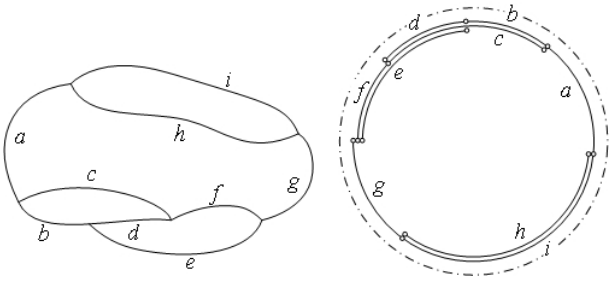
$$w_i = \begin{cases} 1 & \text{if } d_i \leq \varepsilon, \\ e^{(d_i - \varepsilon)/\varepsilon} & \text{otherwise.} \end{cases} \tag{2}$$

Here d_i is the distance of p_i to the exterior profile, in projection, and ε is a positive user-defined threshold. The smaller the value of ε, the better the approximation of the path to the theoretical parting line, while the larger the value of ε, the smoother the path. If we do not wish to follow small features of a certain size in the mesh, we can set ε to be larger than their size in projection.

Note that it is unrealistic for the user to choose a ε which is smaller than the size of the triangles in the mesh. Discretisation errors arising through the use of a triangulated model are of the order of the triangle size, and setting ε smaller than this asks for a parting line with a smaller error than the discretisation error.

An alternative method to compute smooth curves on meshes is presented by [Hertzmann and Zorin 2000]. However, it could be difficult with this approach to constrain the topology of the paths and to carry out cycle optimisation.

As well as the paths computed above, we must also add as geometric candidate paths any paths in the connectivity graph consisting entirely of degenerate edges. Overall, the connectivity of all geometric candidate paths is topologically equivalent to the connectivity graph. By selecting the shortest cycle in the triangle band



(a) (b)

Figure 16: *Paths and their parameterisation.*

amongst these paths, we can find a good cycle, which is then optimised to give the final parting line.

7.3 Cycle Choice

We now wish to find the best cycle using the geometric candidate paths which fully separates the **up visible** regions from the **down visible** regions. In principle, we seek the weighted-shortest cycle: as explained in Section 6.2, this provides the desired optimal parting line. The weighted-length of a cycle is simply the sum of the weighted-lengths of the paths making it up. However, it is well known that finding all cycles in a graph is an NP-hard problem, so in this paper, we use a geometric heuristic to quickly find a near-optimal cycle.

First, however, we subdivide the problem into smaller subproblems, if possible. For many meshes (and, indeed, all we have encountered in practice), we can identify certain paths that the cycle *must* include. If we parameterise the start and end point of each path with respect to the exterior profile of the model (see Figure 16, for example), it is clear that *any* cycle *must* include any path which is not overlapped by other paths for *some* part of its parameter range. We call paths of this kind *must-include* paths. Paths a and g are of this kind. Such paths divide the other paths into several groups, allowing us to separately search for an optimal path for each group. The overall optimal cycle comprises any must-include paths and the optimal path within each group terminated by must-include paths. As we first find the must-include paths, and we also require that a path and the corresponding return path if any must appear together in a cycle, all cycles we get fully separate **up visible** regions from **down visible** regions. Thus, local small cycles are excluded by our algorithm.

The optimal path connecting two adjacent must-include paths can be found using triply-linked trees [Knuth 1997]. Tracing from the leaf nodes to the root node, we can find all paths connecting the two adjacent must-include paths. During construction of the triply-linked tree, we use the heuristic that the path should always proceed in the same sense parametrically (as defined above), and not in the reverse direction—intuitively, the shortest path is unlikely to double back on itself. For example, in Figure 16(a), the path starting from a should proceed after c to f, and we ignore d. (For practical reasons, we permit the succeeding path to proceed in a reverse direction for a small distance, but not a large distance). If *all* succeeding paths proceed in a reverse direction, we select the one whose parameter interval is the smallest as the next path. This heuristic seems to work well in practice to choose a near-optimal cycle.

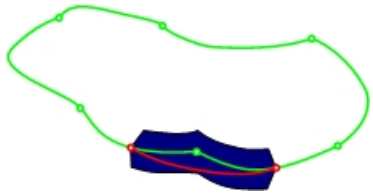


Figure 17: *Cycle smoothing.*

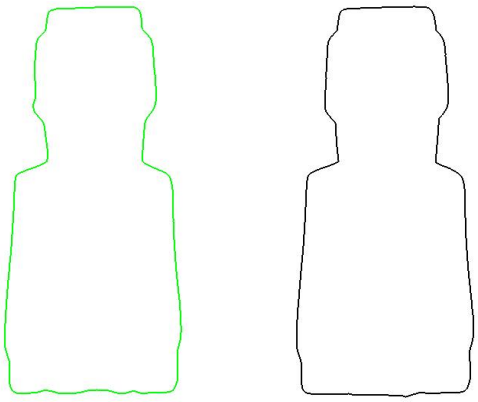
7.4 Cycle Optimisation

Having found a good cycle within the triangle band, we now geometrically optimise that cycle to generate the final parting line, again using Lanthier et al's algorithm. However, for a degenerate path that is adjacent to invisible triangles, we do not weight the distance in computing the shortest path because such path does not approximate the exterior profile in projection. The basic idea is to compute the shortest path between the mid-points of each pair of adjacent geometric paths to locally optimise the cycle; this is repeated several times. This optimisation is carried out within a triangle strip next to each geometric path. See Figure 17. The dark blue region is part of the triangle strip through which the updated geometric path passes. The green cycle is the initial cycle, and the red path is the path after one optimisation step.

The cycle may contain two kinds of candidate paths: *shortest paths* computed as in Section 7.2, and *degenerate paths* composed of degenerate edges. For *degenerate paths*, the triangle strip consists of taking the union of neighbourhoods of triangles with edges on the path, such that each neighbourhood is a topological disk, and all vertices of all triangles in the neighbourhood are within the distance threshold ε of the exterior profile, in projection. (See our earlier remarks on how small ε can be. Typically, for satisfactory results, we might need ε to be at least two or three times larger than the triangle size for this step to work). However, we must exclude from these any triangles which belong to or touch the triangle band except the triangles incident to the junctions of the paths, to prevent optimisation from potentially changing the topology of the final parting line. For *shortest paths*, the triangle strip consists of the **Neither** triangles in the triangle band next to the path.

The optimisation process is iterated until the maximum distance between the optimised cycle and the previous cycle is less than a distance threshold. During shortest path computation, each edge of the triangle is subdivided into several segments to compute the approximate shortest path [Lanthier et al. 1996]. The distance threshold is simply set to be half of the average length of all edge segments. The optimised cycle converges, in practice, to a tight cycle, i.e. the path between any two points of the cycle is the shortest path on the surface. A small error in geometric positioning of the final parting line is traded off for smoothness of the final parting line.

A minor difficulty with the above approach exists in theory, although we have not observed it in practice. As the selected cycle is optimised within a narrow strip through which the cycle passes, theoretically the parting line is not guaranteed to be simple in projection. If this happens where there are significant undercuts, this is unimportant, as side cores will be needed in such places anyway, and when the part is removed from the mold, any side cores are removed first along a direction other than the parting direction. In other cases, a possible solution is to find the self-intersection of the parting line in projection, and remove the small loop formed, replacing it by a straight line along the parting direction to connect the corresponding pair of points on the parting line. (Although this introduces a sharp corner in the parting line, corners of this type are



(a) (b)

Figure 18: *Moai model: (a) the final parting line, in projection; (b) the exterior profile.*

Model	Size	δ	Area	ε	MaxErr	AveErr
Moai	10	5°	0.15	0.2	0.141	0.017
Alligator	100	5°	4	2	6.365	0.664
Dolphin	400	5°	10	0.9	2.540	0.308

Table 1: *Parameters used for experiments, and errors.*

acceptable from a manufacturing point of view).

8 Experimental Results and Discussion

We have tested our algorithms with several models, and give three typical examples to show the performance of our algorithms.

For the moai model shown in Figure 1, all paths were optimised simultaneously. The degenerate paths in this model are very short, and so smoothing all paths simultaneously leads to optimised parting line close to the theoretically correct one. The projection of the final parting line and the exterior profile of the model are shown in Figure 18.

The alligator model shown in Figure 19(a) has a very complex triangle band, mainly because several **occluded** regions exist within it. These **occluded** regions cause the visible differences between the exterior profile and the projection of the optimised cycle.

The last example shows the parting line generated for the dolphin model in Figure 20(a). The triangle band around the head of the dolphin is again very complex. Figure 20(f) shows the detail of the optimised cycle around the head.

The approximate size of the largest dimension of the model, the parameters used for these examples and the maximum and average deviation of the smoothed cycle from the exterior profile are summarised in Table 1. The maximum errors of the cycles for the alligator and dolphin models exceed ε because there are degenerated paths adjacent to invisible triangles. Such degenerate paths can be used to design the parting line between the cavity halves and side cores.

For the experiments we used an AMD Athlon-64 FX-55 PC running at 2.61GHz with 2GB RAM. The times for generating the triangle band, computing the paths, choosing the best cycle, and optimising the cycle are summarised in Table 2. The total times taken are of the order of several minutes, which seems acceptable. We also give the number of triangles in each model in Table 2.

Model	Triangles	Band	Paths	Cycle	Optimize
Moai	20000	7s	29s	1s	81s
Alligator	48758	37s	159s	1s	88s
Dolphin	26286	8s	39s	1s	43s

Table 2: *Triangle counts of models and computation times of different phases.*

9 Conclusions and Future Work

We have given a novel parting line generation method specially designed for complex mesh models. We first find an acceptable region, the so called *triangle band*, within which the parting line should approximately lie, and show how to generate a smooth parting line with respect to certain criteria within this region. As the triangle band is generally not two-connected, we analyse possible cycles within the triangle band to choose a near-optimal cycle, which is then optimised to find the final parting line. Experiments have demonstrated the success of our approach.

Although the parting line generated using our method deviates from the theoretical parting line, and in principle this means that the molded part is no longer removable from the mold, often compliance in the material being molded will mean that our solution is acceptable as the deviation is small. However, an alternative which we also intend to investigate is to make minor modifications to the mesh itself so that it remains strictly removable from the mold.

Acknowledgements

The authors wish to acknowledge the support of Delcam plc, including many helpful discussions with Richard Barratt, and the support of EPSRC grant GR/T24579/01, for this work. We thank J.-R. Sack for providing us with the code for shortest path computation. We also thank the anonymous reviewers for their comments.

References

BARRATT, R., 2006. Personal communication.

BUCKLEITNER, E. L. 1995. *Dubois and Pribble's Plastics Mold Engineering Handbook*. Springer.

CAMPAGNA, S., KOBBELT, L., AND SEIDEL, H.-P. 1998. Directed edges—a scalable representation for triangle meshes. *Journal of Graphics Tools 3*, 4, 1–12.

CHEN, X., AND MCMAINS, S. 2006. Finding all undercut-free parting directions for extrusions. In *Proceedings of Geometric Modeling and Processing (GMP) 2006, Lecture Notes in Computer Science (LNCS) 4077*, Springer-Verlag, M.-S. Kim and K. Shimada, Eds., Geometric Modeling and Processing Conference, IEEE, 514–527.

CHEN, L.-L., CHOU, S.-Y., AND WOO, T. C. 1993. Parting directions for mould and die design. *Computer-Aided Design 25*, 12, 762–768.

ELBER, G., CHEN, X., AND COHEN, E. 2005. Mold accessibility via gauss map analysis. *Transactions of the ASME: Journal of Computing and Information Science in Engineering 5*, 2, 79–85.

ERICKSON, J., AND HAR-PELED, S. 2004. Optimally cutting a surface into a disk. *Discrete & Computational Geometry 31*, 1, 37–59.

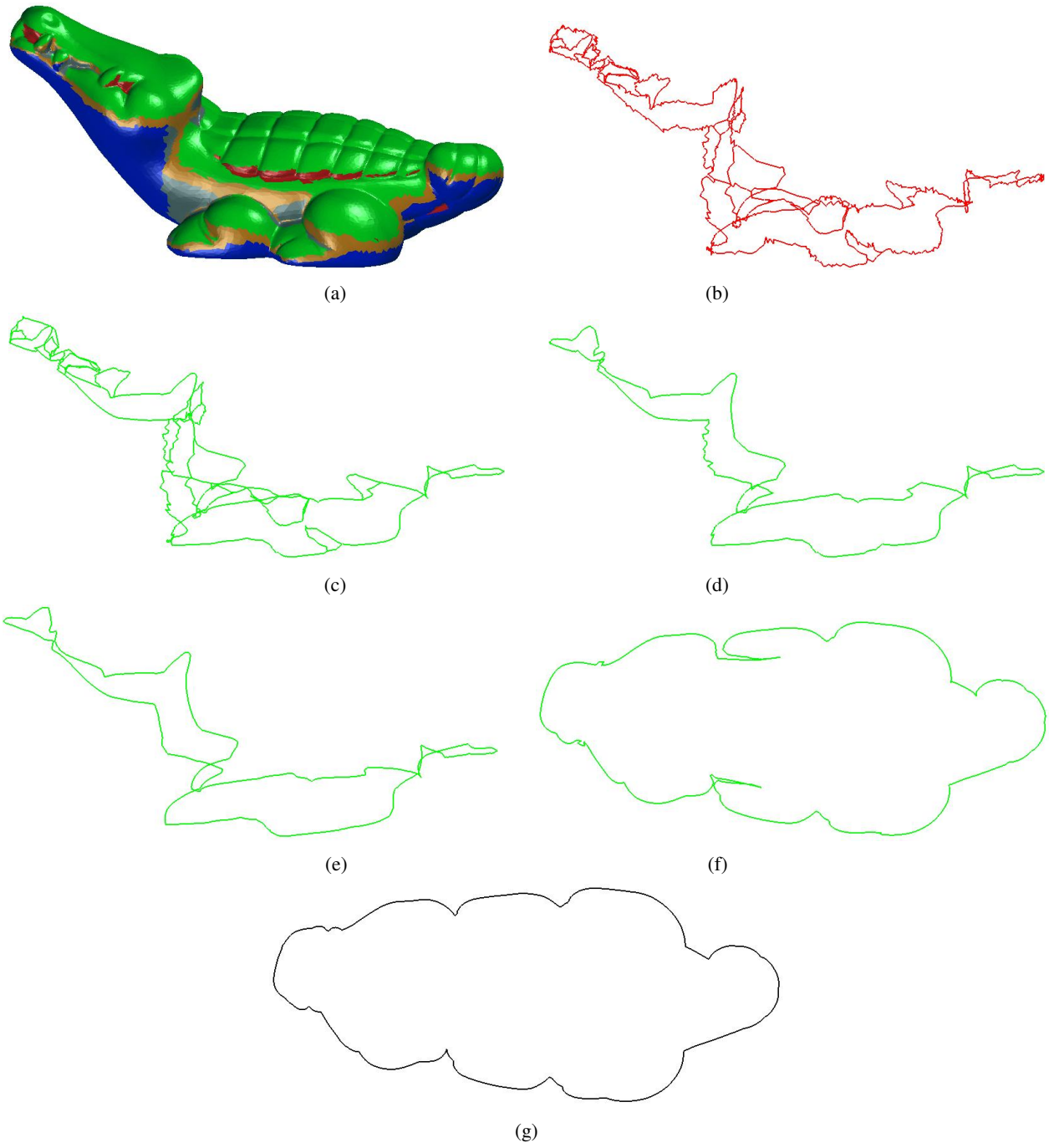


Figure 19: *Alligator model: (a) the model and triangle band; (b) topological structure of candidate paths; (c) candidate paths; (d) chosen cycle; (e) optimised cycle; (f) projection of the optimised cycle; (g) exterior profile.*

FU, M. W., NEE, A. Y. C., AND FUH, J. Y. H. 2002. The application of surface visibility and moldability to parting line generation. *Computer-Aided Design 34*, 6, 469–480.

HERTZMANN, A., AND ZORIN, D. 2000. Illustrating smooth surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., International Conference on Computer Graphics and Interactive Techniques, ACM SIGGRAPH, 517–526.

HUI, K. C. 1997. Geometric aspects of the mouldability of parts. *Computer-Aided Design 29*, 3, 197–208.

KHARDEKAR, R., BURTON, G., AND MCMAINS, S. 2006. Finding feasible mold parting directions using graphics hardware. *Computer-Aided Design 38*, 4, 327–341.

KNUTH, D. E. 1997. *The Art of Computer Programming—Volume 1 (3rd Edition)*. Addison-Wesley.

LANTHIER, M., MAHESHWARI, A., AND SACK, J.-R., 1996.

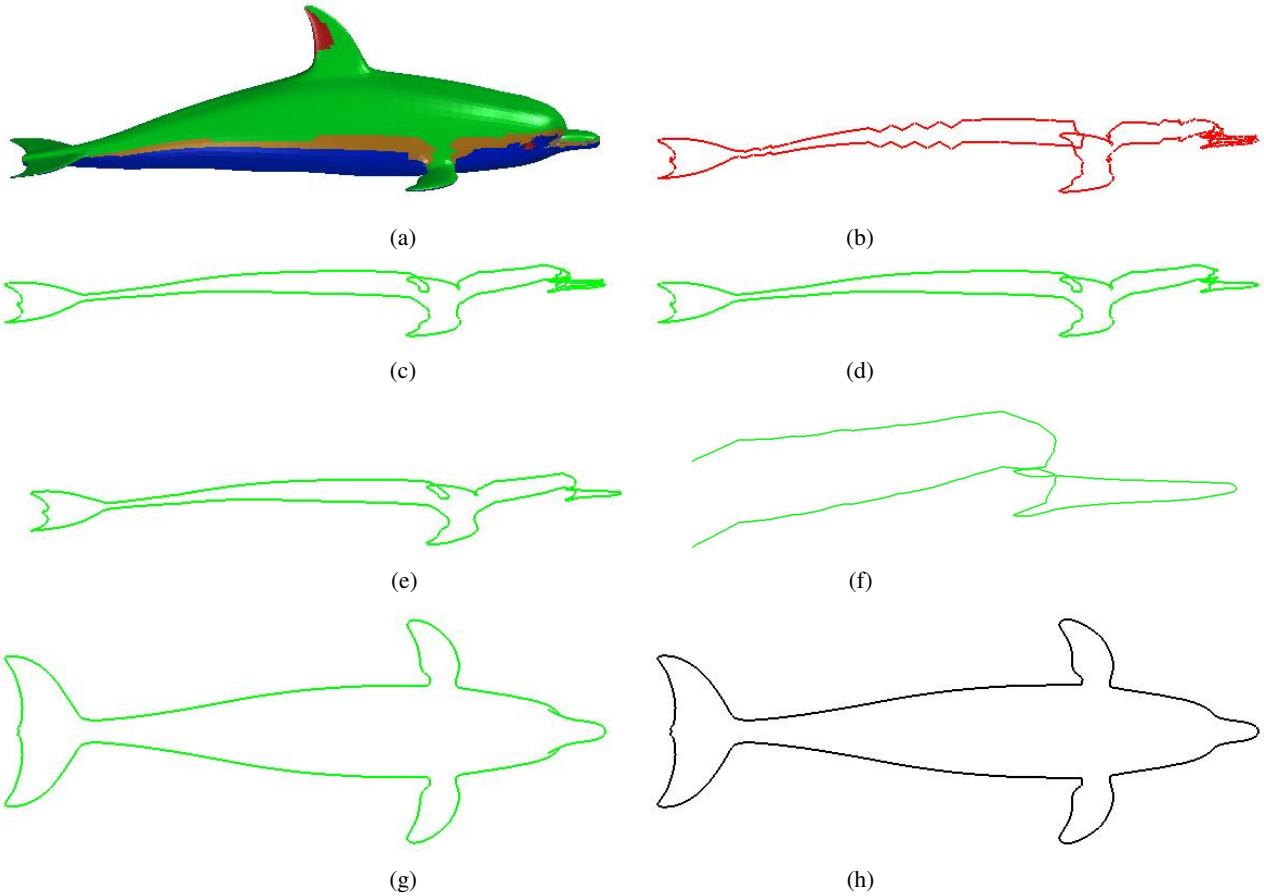


Figure 20: *Dolphin model: (a) the model and triangle band; (b) topological structure of candidate paths; (c) candidate paths; (d) chosen cycle; (e) optimised cycle; (f) detail around head; (g) projection of the optimised cycle; (h) exterior profile.*

Approximating weighted shortest paths on polyhedral surfaces. Tech Report TR–96–32, Carleton University, December.

LAZARUS, F., POCCHIOLA, M., VEGTER, G., AND VERROUST, A. 2001. Computing a canonical polygonal schema of an orientable triangulated surface. In *Proceedings of the seventeenth annual symposium on Computational geometry*, ACM Press, Annual Symposium on Computational Geometry, ACM SIGACT and SIGGRAPH, 80–89.

MAJHI, J., GUPTA, P., AND JANARDAN, R. 1999. Computing a flattest, undercut-free parting line for a convex polyhedron, with application to mold design. *Computational Geometry: Theory and Applications 13*, 4, 229–252.

O'ROURKE, J. 2001. *Computational Geometry in C*. Cambridge University Press.

PRIYADARSHI, A. K., AND GUPTA, S. K. 2004. Geometric algorithms for automated design of multi-piece permanent molds. *Computer-Aided Design 36*, 3, 241–260.

PRIYADARSHI, A. K., AND GUPTA, S. K. 2006. Finding mold-piece regions using computer graphics hardware. In *Proceedings of Geometric Modeling and Processing (GMP) 2006, Lecture Notes in Computer Science (LNCS) 4077*, Springer-Verlag, M.-S. Kim and K. Shimada, Eds., Geometric Modeling and Processing Conference, IEEE, 655–662.

RAVI, B., AND SRINIVASAN, M. N. 1990. Decision criteria for computer-aided parting surface design. *Computer-Aided Design 22*, 1, 11–18.

ROURKE, C., AND SANDERSON, B. 1972. *Introduction to Piecewise-Linear Topology*. Springer-Verlag.

TAN, S. T., YUEN, M. F., SZE, W. S., AND KWONG, K. W. 1990. Parting lines and parting surfaces of injection moulded parts. *Proc. IMechE Part B: Journal of Engineering Manufacure 204*, 4, 211–221.

WEINSTEIN, M., AND MANOOCHEHRI, S. 1997. Optimal parting line design of molded and cast parts for manufacturability. *Journal of Manufacturing Systems 16*, 1, 1–12.

YE, X. G., FUH, J. Y. H., AND LEE, K. S. 2001. A hybrid method for recoginition of undercut features from moulded parts. *Computer-Aided Design 33*, 14, 1023–1034.