

# Molds for Meshes: Computing Smooth Parting Lines and Undercut Removal

Weishi Li, Ralph R. Martin, Frank C. Langbein, *Member, IEEE*

**Abstract**—We consider the problem of computing a *parting line* for a mold for a complex mesh model, given a parting direction, and the related problem of removing small undercuts, either pre-existing, or resulting from the parting line.

Existing parting line algorithms are unsuitable for use with complex meshes: the faceted nature of such models leads to a parting line which zig-zags or wanders across the surface undesirably. Our method computes a *smooth* parting line which runs through a band of triangles whose normals are approximately perpendicular to the parting direction. We generate a skeleton of this triangle band to find its distinct topological cycles, and to decompose it into simple pieces. After selecting paths making a good cycle, we generate a final smooth parting line by iteratively improving the geometry of this cycle. Compliance in the physical material, and / or modifications to eliminate minor undercuts ensure that such a parting line is practically useful.

**Note to Practitioners**—Molding is the process of manufacturing a part by shaping a pliable raw material, such as molten plastic or metal, using a rigid mold. The material hardens and sets inside the mold, adopting its shape. To remove the part from the mold without destroying the mold, the mold must be made of multiple pieces. In the simplest case, a two-part mold is used, and these parts are separated in opposite directions. The curves on the surface of the part where the pieces of the mold meet are called ‘parting lines’. For a manufacturing perspective, a smooth parting line is more applicable than a theoretically correct parting line. We give a novel method to compute such a parting line in this paper.

**Index Terms**—parting line, mesh, mold design, undercut.

## I. INTRODUCTION

**G**IVEN an object to be manufactured in a mold, the *parting line* is that curve on the object’s surface where the sections (cavity halves and cores) of the mold meet as it closes [1]. This paper considers problem of computing a *parting line* for a *complex mesh model*. Existing parting line methods are not suited to such models, as local variations in the orientations of the model’s facets lead to a parting line which zig-zags or wanders across the surface. While such a parting line is theoretically correct, it is undesirable for manufacturing reasons: a smooth curve which closely approximates the theoretical parting line is more desirable in practice [2], [3]. However, such a curve results in *small* undercuts which in principle prevent removal of the object from the mold. Nevertheless, compliance in the material used

may still allow removal, or minor local modifications may be made to the mesh geometry to eliminate the undercuts. Such modifications are often acceptable for mesh models of e.g. ornaments and children’s toys, where exact shape is not critical. In this paper we give a method for computing smooth parting lines, and also consider undercut removal. For simplicity, we discuss genus-0 models only, although our concepts could in principle be generalized to other cases by computing multiple parting lines.

Consider a two-part mold for a simple mesh model, whose pieces are to be removed in opposite directions, which we will refer to as the *parting direction* and the *opposite direction*. If a parting direction can be found which separates the mesh triangles 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 more complex mesh, however, it may be impossible to find *any* parting direction which cleanly separates the mesh triangles into two such regions, and generation of a suitable parting line is not so straightforward.

We assume that we are given a closed, manifold mesh, and a desired parting direction—choice of suitable parting directions is addressed elsewhere in the literature [4], [5], [6]. The input may include (small) undercuts with respect to the given parting direction. Our approach is to first construct a *triangle band* consisting of triangles with normals nearly perpendicular to the parting direction, forming an acceptable region through which the parting line may pass. We then generate the parting line within that region, noting that we require a smooth, and ideally almost planar, curve [7], [8], which introduces as few additional undercuts as possible.

We provide simple control over parting line generation by an angular tolerance and a distance threshold. The *angular tolerance* is used when generating the triangle band. Mesh triangles whose normals are within this tolerance of being perpendicular to the parting direction are used to construct the triangle band. A tighter tolerance results in a narrower band; a looser tolerance gives more freedom for the location of the parting line. A smaller *distance threshold* results in a parting line which is closer to the theoretically correct one; a larger threshold allows a smoother parting line, at the possible expense of more undercut. The angular tolerance used is not critical, while the distance threshold can be adjusted to allow for e.g. compliance of the material being molded, and importance of preserving the original mesh shape. An *area threshold* is also provided to eliminate small gaps in the triangle band which can arise if the input mesh model is noisy. This generally improves the global nature of the parting line.

W. Li is with the School of Computer Science, Cardiff University, Wales, UK e-mail: Weishi.Li@cs.cf.ac.uk.

R. R. Martin is with the School of Computer Science, Cardiff University, Wales, UK e-mail: Ralph.Martin@cs.cf.ac.uk.

F. C. Frank is with the School of Computer Science, Cardiff University, Wales, UK e-mail: F.C.Langbein@cs.cf.ac.uk.

Manuscript received , ; revised , .

Section II reviews prior work. Section III summarizes our method. Generation of the triangle band is discussed in Section IV. The nature of the triangle band and the parting line are considered in Section V, and our method for topological analysis of the parting line in Section VI. Parting line computation is presented in Section VII, and undercut removal in Section VIII. Experimental results are shown in Section IX.

This paper is an extended version of [9], and in particular considers further the problem of adjusting the mesh to be compatible with the new parting line.

## II. RELATED WORK

Determination of parting directions and parting lines are well-studied problems. For a two-part mold, an optimal parting direction can be found using Gauss map analysis of the object to be molded [10]. More recent work [5], [6] has considered use of graphics hardware to accelerate generation of parting directions. Parting directions for cores are considered by [11]. Having found a suitable parting direction, a parting line can then be generated. Prior work has generally considered objects bounded by relatively few, simple surfaces (planes, natural quadrics, and splines), rather than meshes: e.g. two-piece molds for NURBS models are considered by [4].

Parting line generation methods are based on *visible surface* detection. A point  $p$  on a model  $\mathcal{M}$  is *visible* from a parting direction if the ray  $r$  starting at  $p$  going in the opposite direction does not intersect any other part of the model. Occluded regions of the surface constitute the *undercuts* for this parting direction [12]. Usually *cores* are added to enable undercut features to be molded [11].

For a mesh analysis of which facets are visible from the parting direction and the opposite direction allows construction of the parting line [13]. Tan [14] shows how to construct a parting line for a model with multiple visible regions. Weinstein [15] gives an optimisation method for parting line generation, taking into account parting line complexity, draw depth, number of undercuts, number of side cores, and mold complexity. Some of these depend on the choice of parting direction, which, however, we assume to be fixed in our algorithm. Importantly, these earlier methods all generate a *theoretically correct* parting line, however.

Using a triangle band to compute a parting line was originally proposed by [8]. However, that work only allowed convex polyhedra, whereas ours handles real-world, non-convex triangular meshes. In the convex case, the triangle band corresponds topologically to a 2-connected surface (a *belt*), while in the non-convex case, the triangle band can be more complex: it may be topologically equivalent to an  $n$ -connected surface with  $n > 2$ , or may comprise several 1- or non-1-connected surfaces connected by edges. As a result, both generating the triangle band, and computing a parting line within it, are more complicated. The method in [8] cannot be straightforwardly extended to this case.

Majhi's approach has also been employed by [3] to design multi-piece molds. Recently, [16] gave a method to find a triangle band for complex mesh models using graphics hardware. However, the method is limited by the resolution

of the hardware, and again, it does not readily to generalize to triangle bands with complex topology.

In our method, we cut the triangle band into a set of adjacent topological disks. Any mesh model with handles can be opened into a topological disk along a *canonical polygonal schema*, which can be computed by use of topological collapsing operations [17]: see [18]. We use a similar approach to cut the triangle band, an open surface *without* handles, into topological disks.

## III. PARTING LINE ALGORITHM OVERVIEW

This Section explains our notation and provides an overview of how we find a parting line on a triangular mesh model.

We define a triangle mesh  $\mathcal{M}$  to be  $\mathcal{M} = (\mathcal{K}, \mathcal{P})$  where  $\mathcal{K} = \mathcal{V} \cup \mathcal{E} \cup \mathcal{T}$  is a simplicial complex representing the connectivity of the mesh, and  $\mathcal{P}$  gives the vertex positions. Here  $\mathcal{V} = \{v_i\}$ ,  $i = 0, \dots, n_v$  is the set of vertices,  $\mathcal{E} = \{e_i\}$ ,  $i = 0, \dots, n_e$  is the set of edges, and  $\mathcal{T} = \{t_i\}$ ,  $i = 0, \dots, n_t$  is the set of triangles.  $\mathcal{P} = \{\mathbf{p}_i \in \mathcal{R}^3\}$  gives the position of each vertex  $v_i$ . For brevity, we identify vertices and their positions, and use the same notation for both. We use  $\mathbf{n}_i$  to denote the normal to triangle  $t_i$ .

We use a coordinate system chosen such that the given parting direction  $\mathbf{D}$  coincides with the  $z$ -axis. Later, when we refer to projection, we will always mean parallel projection in the  $z$  direction onto the  $x$ - $y$  plane.

Given the parting direction  $\mathbf{D}$ , and a user-chosen angular tolerance  $\delta$ , all triangles  $t_i$  can be classified according to the angle  $\theta_i$  between  $\mathbf{D}$  and the triangle's normal,  $\mathbf{n}_i$ . Triangles are divided into three types:

- up:** triangles for which  $\theta_i < 90^\circ - \delta$ ,
- down:** triangles for which  $\theta_i > 90^\circ + \delta$ ,
- neither:** triangles for which  $90^\circ - \delta \leq \theta_i \leq 90^\circ + \delta$ .

**Up** and **down** triangles can be further classified as **visible** or **occluded**. **Down visible** and **up visible** triangles which are entirely visible from  $\mathbf{D}$  or  $-\mathbf{D}$  respectively; other triangles are **occluded**. Contiguous **visible** triangles form **up** and **down visible regions**. Occluded triangles indicate undercut regions. To aid understanding of the figures in this paper, we colour regions as follows: **up visible:** green, **down visible:** blue, the triangle band: dark yellow, occluded regions within the triangle band: gray, other regions: red (see e.g. Figure 1(a)).

Figures in Sections IV–VII show triangle bands which have been flattened from 3D onto the 2D plane of the paper to illustrate their topology.

The triangle band is constructed from the triangle classification. Given an angular tolerance greater than zero, **neither** triangles and undercuts generally exist between the visible regions. The region outside the outermost boundary of each visible region, excluding any undercuts, is the *triangle band*: this is the region which the parting line must pass through. The triangle band usually has a non-zero width, although it may degenerate locally into an edge where common boundaries exist between **up visible** and **down visible** regions, or where an undercut is excluded from the triangle band.

Often, there are *multiple* **up visible** or **down visible** regions, or occluded triangles within the triangle band, so the triangle

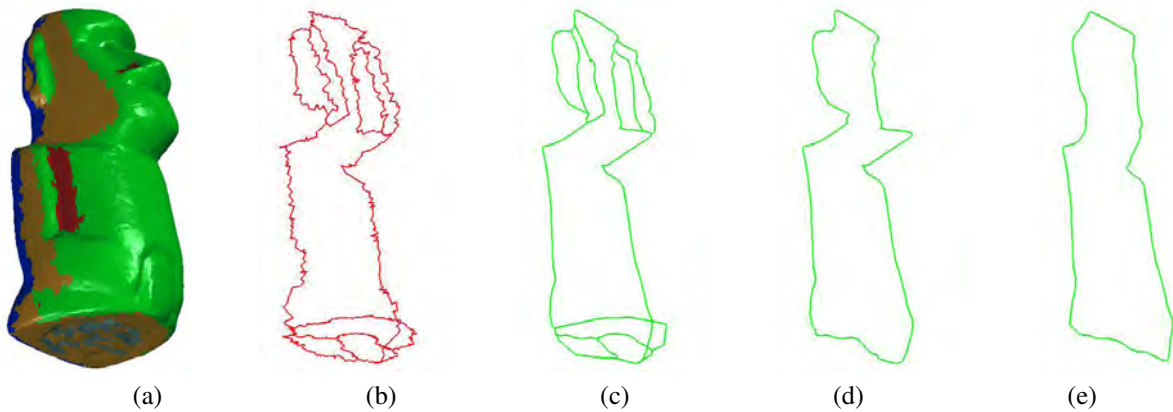


Fig. 1. Moai model: (a) model and triangle band; (b) topological structure of candidate paths; (c) candidate paths; (d) chosen cycle; (e) final parting line.

band is not just a 2-connected surface (a belt encircling the mesh), but is  $n$ -connected, with  $n > 2$  topologically distinct paths connecting any two points in the band. Clearly, in such cases, multiple topologically distinct cycles exist in the triangle band; we must choose one as the basis for the parting line.

Thus, after finding the triangle band, the following steps are used to generate the parting line:

- 1) Find the topological structure of the triangle band (see e.g. Figure 1(b)).

We reduce the triangle band to a skeleton, giving a connectivity graph. We then remove disallowed paths between certain occluded regions inside the triangle band. For technical reasons, we also add paths between certain regions visible from the same direction. We then seek valid *candidate paths* within the connectivity graph, which are *topological cycles* fully separating the **up visible** regions from the **down visible** regions.

- 2) Find geometric candidate paths in the triangle band from the topological structure (see e.g. Figure 1(c)).

We next decompose the triangle band to into 1-connected (disk-like) regions and compute shortest *geometric paths* joining adjacent regions. We add to these any *degenerate edges* where the triangle band has locally collapsed to an edge. The overall connectivity of these geometric paths is topologically equivalent to the connectivity graph.

- 3) Choose the best cycle by considering weighted lengths of candidate paths (see e.g. Figure 1(d)).

*Weighted* lengths of valid topological cycles formed by the geometric candidate paths are computed. The shortest gives the cycle used to determine the final parting line. The weights take into account the needs for the parting line to be smooth, and close to the theoretical parting line.

- 4) Improve the geometry of the chosen cycle to give the parting line (see e.g. Figure 1(e)).

The triangle band is now restricted to those parts containing the chosen cycle, to avoid unwanted topological changes during optimization. An improved path is then computed iteratively, again using a shortest path algorithm, to give the final smooth parting line.

#### IV. COMPUTING THE TRIANGLE BAND

We now explain in detail how to generate the *triangle band*, given the mesh, the parting direction, and the angular tolerance  $\delta$ .

We first find *visible* triangles. Each triangle is initially classified (see earlier) as **up**, **down**, or **neither**. We next determine which **up** and **down** triangles are **occluded** (including partially occluded). Non-occluded **up** and **down** triangles are **visible**.

Any mesh edge whose adjacent triangles have different classifications is a *boundary edge*. If one triangle of a boundary edge is a **down visible** triangle, the edge belongs to a boundary of a **down visible** region, and similarly for **up visible** regions. An edge may belong simultaneously to the boundary of an **up visible** and a **down visible** region. By tracing neighbors of boundary edges, complete boundaries of visible regions can be found. The *outermost* boundaries of the **up visible** regions are the upper boundaries of the triangle band, and its lower boundaries can be found similarly.

Region growing is used to complete triangle band generation. A **neither** triangle adjacent to each piece of boundary is selected as a seed, from which growing is performed to find all **neither** triangles between the boundaries. The triangle band may degenerate locally into an edge where common boundaries exist between **up visible** and **down visible** regions, or where an undercut is excluded from the triangle band. Such degenerate edges are included as parts of the triangle band.

If the mesh model is noisy, small **visible** or **occluded** regions may exist within the triangle band. For efficiency, and to improve the quality of the results, the user may specify that such regions whose projected area is below a threshold are ignored—they are simply relabeled as **neither** triangles, simplifying the topological structure of the triangle band. Doing so introduces small extra undercuts; we consider later how these might be removed.

#### V. THE TRIANGLE BAND AND THE PARTING LINE

As noted, the triangle band may not be a 2-connected surface, but may have more complex topology. Two considerations must be taken into account when deciding where the parting line may go within the triangle band.

Firstly, **up occluded** and **down occluded** regions may exist, as pairs, within the triangle band. The parting line should not

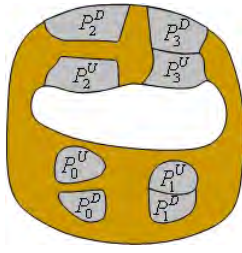
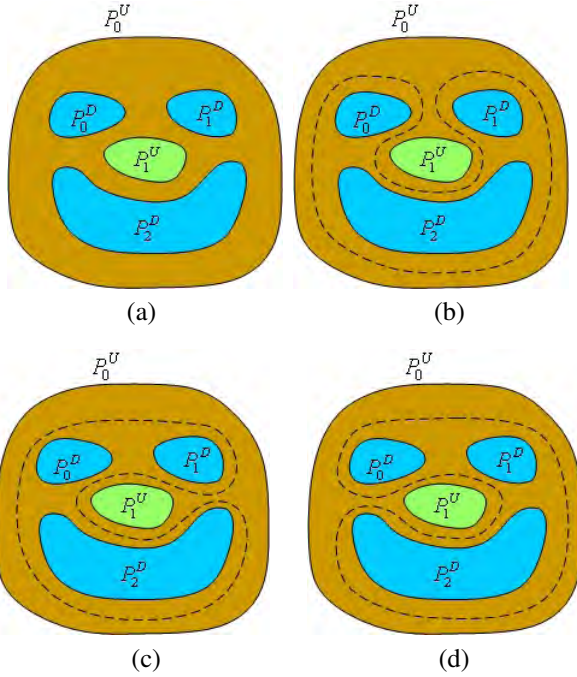


Fig. 2. A triangle band containing occluded regions.


 Fig. 3. Multiple **up visible** and **down visible** regions, allowing topologically different parting lines.

separate any pair of mutually occluded regions, otherwise the molded part will not be removable from the mold. This is true whether such pairs of regions touch—see Figure 2,  $U_1^U-P_1^D$  and  $U_3^U-P_3^D$ , or not—see  $U_0^U-P_0^D$  and  $U_2^U-P_2^D$ . (Note again that this and other Figures show a *flattened* view of the triangle band; in this case outside the band is **up visible** and inside is **down visible**).

Secondly, there may be multiple **up visible** and/or **down visible** regions. Again, for removability reasons (using side cores if needed), the parting line must separate *all* **up visible** regions from *all* **down visible** regions. Nevertheless, multiple alternative topological cycles may exist which separate the **up visible** regions from the **down visible** ones. Figure 3(a) shows an example with two **up visible** regions,  $P_0^U$  and  $P_1^U$  ( $P_0^U$  is *outside* the triangle band in this flattened representation), and three **down visible** regions,  $P_0^D$ ,  $P_1^D$  and  $P_2^D$ . Three topologically distinct cycles separating **up visible** regions from **down visible** regions are shown in Figures 3(b)–(d).

To perform topological analysis, we reduce the triangle band to a skeleton, and then construct all topologically distinct cycles which meet the above requirements.

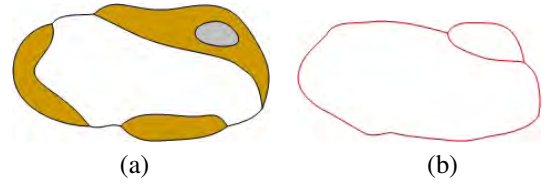


Fig. 4. A triangle band and its skeleton.



Fig. 5. The skeleton generation operation.

## VI. TOPOLOGICAL ANALYSIS OF THE TRIANGLE BAND

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

### A. Skeleton Generation

The triangle band is an open triangle mesh, possibly including degenerate edges. Its skeleton is a collection of edges representing the connectivity of those parts of the triangle band through which the parting line may pass—see Figure 4. To compute the skeleton, we use a method based on the *canonical polygonal schema* idea of [18]. However, our method can directly handle an open mesh with degenerate edges; the latter must be treated separately in [18].

Skeleton generation is performed by *eroding* the triangle band. We repeatedly remove triangles from the triangle band which are adjacent to triangles outside the band. We first initialize labels for all triangles, and edges: blue for **down visible** triangles, green for **up visible** triangles, gray for **occluded** triangles, yellow for triangles in the triangle band; red for edges of the triangle band (including degenerate ones), and black for all other edges. Let triangle  $t_j$  belong to the triangle band, and  $t_i$  belong to some other *particular* kind of region, e.g. an **up visible** region. Triangle  $t_i$  is thus an *erosion triangle*. Suppose these triangles meet in edge  $e_k$ . The label for  $t_j$  is yellow, and its edges are red; the edges of  $t_i$  other than  $e_k$  are black: see Figure 5(a). An erosion step does the following: where a red edge is adjacent to an erosion triangle and a yellow triangle, it is relabeled as black, and the yellow triangle is given the label of the erosion triangle. Other edges of the yellow triangle remain red. See Figure 5(b). Erosion is performed one ring of triangles at a time from the boundaries of the triangle band inwards. Erosion is applied repeatedly until no further change occurs, at which point the remaining red edges are the desired skeleton.

Note that skeleton generation is carried out by using a *particular* kind of triangle in erosion steps, e.g. **up visible** triangles, or **down visible** triangles, etc. Different choices result in different skeletons, but in each case, the result has the *same connectivity* as the triangle band.

The above method results in a skeleton which generally has side branches. These are removed, iteratively, to simplify the

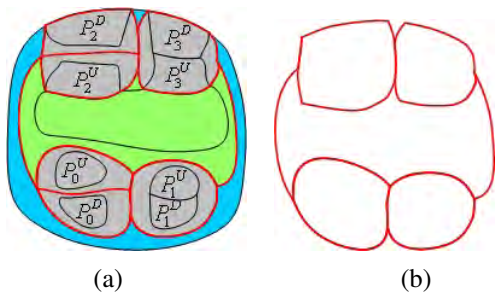


Fig. 6. Path breaking between pairs of occluded regions.

skeleton without affecting its cycle structure.

### B. Connectivity Graph Generation

The skeleton of the triangle band is a graph which encapsulates the *topology* of the triangle band. We now use it to generate the connectivity graph of all valid topological cycles, which separate the **up visible** regions from the **down visible** regions. The edges of this graph correspond to polylines on the mesh. To obtain this graph, we generate a skeleton based on simultaneous use of **up visible**, **down visible** and **occluded** triangles as erosion triangles. This particular choice is carefully selected to give a skeleton whose projection is a good approximation to the exterior profile of the mesh. This means that the *geometric* information in the connectivity graph can also be used later to determine a good initial cycle which well approximates the exterior profile. This cycle is then smoothed to give the final parting line.

While this skeleton represents the connectivity of the triangle band, we must modify it before use, to meet the requirement that the parting line must not pass between pairs of mutually occluded regions, and to ensure that it is possible to generate cycles which separate all **up visible** regions from all **down visible** regions. Various paths must be added to and removed from the skeleton to do so.

The first requirement is met by *removing* any segment of the skeleton passing between a pair of mutually occluded regions. Such segments can easily be identified, at the end of the growing process used to construct the skeleton: triangles adjacent to such segments are labelled gray on either side, and on either side have been grown from **occluded** regions belonging to the same pair. Consider the triangle band in Figure 2. Its skeleton is shown in red in Figure 6(a). After removal of the segments between  $P_0^D-P_0^U$  and  $P_2^D-P_2^U$ , the skeleton is altered to the one shown in Figure 6(b).

The second requirement is met by *duplicating* segments of the skeleton between adjacent **up visible** regions, and adjacent **down visible** regions. This is done to provide a *return path* between such regions—a path may need to go between the regions, then around a region of the opposite type, and back again: see the return path used between the top two blue regions in Figure 3(b). Identifying segments of the skeleton lying between two regions visible from the same direction is easy: the triangles on either side of such segments have the same label (blue or green). Such segments are removed from the skeleton, and are replaced by two new segments, as

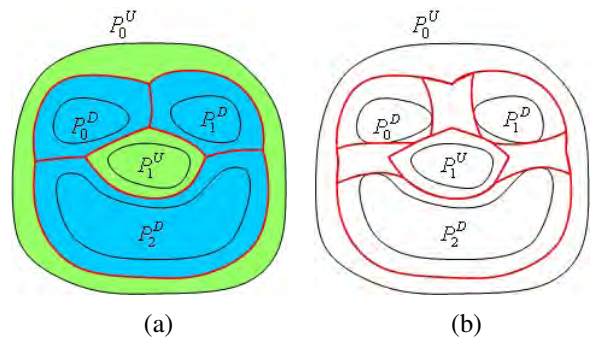


Fig. 7. Path duplication for regions visible from the same direction.

follows. We find the **neither** triangles that are vertex-adjacent to the skeleton segment between visible regions with the same label. These are used to produce a new region between the two visible regions. The boundary of this region includes two segments which are not currently part of the skeleton. These segments are added to the skeleton, providing a path and a return path between the two visible regions. Again, see Figure 3(a); its skeleton is in red in Figure 7(a). Note that there is only one skeleton edge between the two upper blue regions. After the process above, the skeleton is modified to give the red structure in Figure 7(b), which now has a path and a return path between the upper blue regions (paths are also duplicated between other adjacent pairs of blue regions).

Using this structure, we can now construct all topologically valid cycles separating **up visible** regions from **down visible** regions (see Figures 3(b)–(d)). We call this structure generated by modifying the skeleton the *connectivity graph*. We next discuss how to select a particular cycle from this structure, by also taking geometric considerations into account. This cycle is then optimised to give the final parting line.

## VII. PARTING LINE GENERATION

We now explain how to generate the parting line. As the triangle band is in general not 2-connected, we must determine an appropriate topology as well as its geometry.

To choose the most suitable topological cycle, we must take geometry into consideration. We thus first determine geometric paths whose connectivity corresponds to that of the connectivity graph. We do this by decomposing the triangle band into 1-connected regions (disk-like pieces), and computing appropriate geometric paths in each region using a shortest path algorithm. These paths are then analyzed to choose the most desirable cycle. Lastly, we improve the geometry of this cycle to give the final smooth parting line.

### A. Triangle Band Decomposition

We start by decomposing the triangle band into disk-like pieces. We can do so simply by computing two further, *different*, skeletons of the triangle band. (The information obtained is not readily available from the skeleton already computed earlier).

We first compute a skeleton using **down visible** triangles as erosion triangles. This skeleton splits a triangle band with

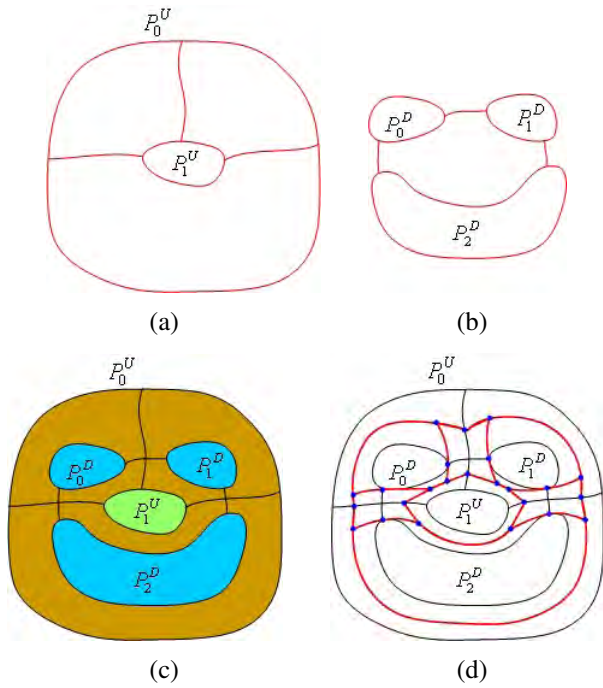


Fig. 8. Triangle band decomposition using skeletons and end points for geometric candidate paths.

$m$  lower boundaries into  $m$  separate 2-connected regions. We then do the same using **up visible** triangles as erosion triangles, splitting a triangle band with  $n$  upper boundaries into  $n$  2-connected regions. By overlaying the two skeletons, the triangle band is cut into a set of 1-connected regions. For example, given the triangle band in Figure 3(a), the skeletons generated using **down visible** and **up visible** erosion triangles are shown in Figures 8(a) and (b), respectively. Together, these decompose the triangle band (dark yellow) into six 1-connected regions as shown in Figure 8(c).

Two special cases must be taken into account. Skeleton edges may be present with the same region on either side. To ensure 1-connectedness, these must be further decomposed so that each edge has different regions on either side. Secondly, the above procedure does not decompose a 2-connected triangle band into two 1-connected regions, so such a cut must be made explicitly. Procedures based on growing from seed triangles may be used to add the extra edges needed to resolve these issues: see [9].

### B. Geometric Candidate Path Computation

We next compute geometric paths based on the connectivity graph, using the above 1-connected regions. We refer to these as *geometric candidate paths*, as we select from amongst these paths to determine the most desirable cycle. We first choose suitable end points for these paths, and then compute weighted shortest paths between these end points.

End points of two kinds are chosen to encapsulate the topological information. First, we need all points in the connectivity graph where more than two edges meet. Secondly, we add all intersections between the connectivity graph and the boundaries of the 1-connected regions. Adding the latter

ensures that each individual geometric path is restricted to a single 1-connected region. These end points are shown in blue in Figure 8(d) for the triangle band in Figure 3(a).

The weighted shortest path between each appropriate pair of end points is computed using Lanthier's method [19]. This path has the desirable properties that it is locally smooth and has no unwanted geodesic curvature between the endpoints. The weighting is used to ensure that the path is not too far in projection from the exact parting line. These considerations lead to geometric paths which are suitable starting points for optimization leading to the final parting line. We define the *weighted distance* between any two points  $p_i$  and  $p_j$  of an edge of the path to be

$$D(\mathbf{p}_i, \mathbf{p}_j) = w_i w_j \|\mathbf{p}_i \mathbf{p}_j\|, \quad (1)$$

where  $\|\mathbf{p}_i \mathbf{p}_j\|$  measures unweighted distance on the mesh, and the weighting function is given by

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

where  $d_i$  is the distance in projection of  $\mathbf{p}_i$  to the exterior profile, and  $\varepsilon$  is a (positive) user-defined threshold. The smaller  $\varepsilon$ , the more nearly the path will follow the theoretical parting line, while the larger the value of  $\varepsilon$ , the smoother the path: if features in the mesh less than a certain size are unimportant, we may set  $\varepsilon$  to be larger than this size. Note that it is not meaningful to set  $\varepsilon$  smaller than the typical triangle size.

In addition to the paths computed above, we must also add as geometric candidate paths any paths in the connectivity graph consisting entirely of degenerate edges.

We now choose a cycle amongst these geometric candidate paths, and optimise it to give the final parting line.

### C. Cycle Selection

We now wish to find the best (i.e. weighted-shortest) cycle amongst the geometric candidate paths which fully separates the **up visible** regions from the **down visible** regions. However, finding all cycles in a graph is an NP-hard problem, so instead, we use a simple heuristic to quickly find a good cycle rather than the best cycle.

We start by subdividing the problem into smaller subproblems, if possible. For many meshes (indeed, all we have encountered in practice), there are certain paths that the cycle *must* include. We project the geometric candidate paths into the  $x$ - $y$  plane, which allows us to place each path in correspondence with an arc of a circle (see Figure 9, for example). The exact way in which correspondence is made is unimportant; the key point is that any path whose arc does not overlap other arcs *must* be included *whichever* cycle we construct. We call such paths *must-include* paths; see paths  $a$  and  $g$  in Figure 9. As a result, the remaining paths can be divided into several groups. The overall optimal cycle is formed from any must-include paths, and the optimal path for each group between the must-include paths.

By first finding the must-include paths, and by requiring that a path and the corresponding return path if any must appear together in a cycle, all cycles we compute must fully separate

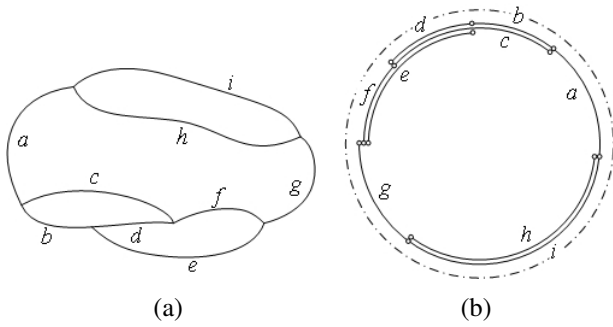


Fig. 9. Paths and their parameterisation.

**up visible** regions from **down visible** regions, guaranteeing that our algorithm does not just find a local loop.

The problem of finding the shortest path connecting two adjacent must-include paths can be solved by using triply-linked trees [20]. Tracing from the leaf nodes to the root node gives all paths between the ends. To speed up the search, we use the simple heuristic that the path should always proceed in the same sense around the circle mentioned above and should not take an arc which goes backwards—the shortest path is unlikely to double back on itself. (In practice, we permit a succeeding path to proceed in a reverse direction for a small distance, but not a large distance). For example, in Figure 9(a), proceeding from  $a$ , after considering  $c$ , we choose  $f$ , and ignore  $d$ . If *all* succeeding paths proceed in a reverse direction, we select the shortest one. This heuristic rapidly chooses a near-optimal cycle.

#### D. Cycle Optimisation

Having chosen a cycle within the triangle band, we now optimize its geometry to give the final parting line, again using Lanthier’s algorithm. While still using weighted distances for most paths, we use unweighted distances in computing the shortest path for degenerate paths, because they do not approximate the exterior profile in projection.

The approach used is to iteratively recompute the shortest path between the *mid-points* of each pair of adjacent geometric paths to locally find new optimal paths. This optimisation is done within the triangle strip adjacent to each geometric path.

For most paths, the triangle strip consists of the **neither** triangles in the triangle band next to the path. For degenerate paths, the triangle strip is formed by taking the union of neighbourhoods of triangles with edges on the path, such that each neighbourhood is a topological disk. However, we must exclude from these any triangles which belong to or touch the triangle band except for triangles incident to junctions of the paths. This is necessary to prevent the possibility of optimization changing the topology of the parting line.

This local optimisation process is iterated until the maximum distance between the current cycle and the previous cycle falls below a threshold. During shortest path computation, each triangle edge is subdivided into several segments [19]; this threshold is simply set to half the average length of all such edge segments. This optimization procedure quickly produces a *tight* cycle: the path between any two points of the cycle is the shortest path on the surface.

#### E. Parting Line Compatibility

Many applications require that the final parting line is *compatible* with the mesh, in the sense that all vertices and edges of the parting line are also vertices and edges of the mesh. To do this, all triangles that the parting line touches or crosses are identified. If more than one edge of the parting line lies inside or on the boundary of any triangle, a section of the parting line is locally replaced by an edge connecting the two endpoints of the section, and any triangle containing an edge of the parting line is subdivided.

### VIII. UNDERCUT REMOVAL

Undercuts are normally considered to be regions of the mesh that are neither visible from  $D$  nor  $-D$ . However, here, another kind of undercut arises: regions that are outside the parting line in projection, arising due to the difference between the computed parting line and the theoretical parting line. Manufacturability is improved by removing some or all of the undercuts, particularly any new ones of the latter kind. Side cores may still be needed for larger undercut regions, but the fewer that remain, the better. We thus now consider how to remove undercuts.

Related work has considered how to impose a draft angle on a geometric model for mold design [21], [22]. However, we are not aware of any method which can remove undercuts.

Our approach works in three steps. We first identify undercut regions to be modified. We next adjust the normal for each triangle inside the undercut region, giving it a new  $z$ -component value; these normals are chosen to be consistent with a surface with no undercut. Finally, we update the positions of the vertices inside the undercut region to give these triangles the desired normals, using a method originally designed for mesh filtering [23]. We now detail these steps.

#### A. Region Identification

For simplicity, we assume that if the projected area of an undercut region is smaller than a threshold, it should be removed. Such regions are then extended several triangles outwards to give some freedom when modified and to allow smooth connections to neighboring parts of the surface. Henceforth, we will mean these extended undercut regions when we refer to undercut regions.

Note that the parting line may pass through a given undercut region. If it does, we divide it into two separate regions along the parting line. We can now classify every region to be modified as lying above, or below, the parting line. We will later process regions which touch the parting line slightly differently from those which do not.

#### B. Normal Assignment

Suppose a parting line separates a simple genus-0 model into two contiguous regions—those visible from the parting direction  $\mathbf{D}$ , and those visible from the opposite direction—the model is undercut-free. The normal of a triangle inside a region visible from  $\mathbf{D}$  must have a negative  $z$ -component, and vice versa for the opposite direction. We thus adjust the

normals of all triangles inside an undercut region adjacent to a region visible from  $\mathbf{D}$  to have a negative  $z$ -component, causing the region to become visible from  $\mathbf{D}$  (after vertex adjustment), thus removing the undercut, and vice versa.

To adjust the normal of a triangle  $t_i$  which is inside such a region above or below the parting line, we rotate its normal  $\mathbf{n}_i$  around  $\mathbf{n}_i \times \mathbf{D}$  so that the angle between  $\mathbf{n}_i$  and  $\mathbf{D}$  or  $-\mathbf{D}$  respectively has a prescribed value  $\theta$ . We require that  $\theta \leq \delta$ , the angular tolerance used to classify the triangles. Using a large value for  $\theta$  will change the model too much, as well as taking a long time to compute. The new normals are then smoothed, following [24], to reduce the impact of noise present in the original mesh, and to help blend the region blend smoothly with the rest of the original mesh. We must ensure that the sign of the  $z$ -component of the normal is not changed by filtering, which we do as follows: let  $\mathcal{T}^*(t_i)$  be those neighbors of  $t_i$  having the same sign of  $z$ -component. The filtered normal of  $t_i$  is computed as

$$\mathbf{n}'_i = \mathbf{n}/|\mathbf{n}| \quad (3)$$

where

$$\mathbf{n} = \sum_{t_j \in \mathcal{T}^*(t_i)} \mathbf{n}_j e^{(\mathbf{n}_i \cdot \mathbf{n}_j)^2 / 2\sigma^2} \quad (4)$$

and  $0 < \sigma < 1$  controls the strength of filtering. The smaller  $\sigma$ , the smoother the new normals. For a smooth mesh without sharp features,  $\sigma = 0.4$  is a good choice, whereas for a mesh with sharp features,  $\sigma = 0.2$  works better.

### C. Ordered Vertex Updating

Having allocated new normals for triangles inside the undercut region, we now update the corresponding vertex positions to be as compatible as possible with the new normals, by optimizing an error measuring lack of perpendicularity of the edges of each triangle to its normal.

In mesh filtering, triangles are usually updated in an arbitrary order [23], but such this does not work well for our problem. Instead, we sort the vertices to be updated in each region. If the region touches the parting line, we find all region triangles which are 1-ring neighbors of the the parting line. The vertices of such triangles, except for vertices on the parting line, are numbered sequentially first. We then proceed to number vertices of their neighboring triangles inside the region, again sequentially, and so on. If the region does not meet the parting line, we simply number the vertices of all triangles adjacent to the region boundary consecutively, and then iteratively pass to their neighbors, and so on.

The vertex coordinates  $\mathbf{p}_i$  are updated using:

$$\mathbf{p}'_i = \mathbf{p}_i + \frac{1}{|\mathcal{T}(v_i)|} \mathbf{n}_i \left( \mathbf{n}_i \cdot \sum_{t_k \in \mathcal{T}(v_i)} \mathbf{n}'_k (\mathbf{n}'_k \cdot (\mathbf{c}_k - \mathbf{p}_i)) \right) \quad (5)$$

which is slightly different from the version in [23]. Here  $\mathbf{c}_k$  is the center of triangle  $t_k$ ,  $\mathcal{T}(v_i)$  is a half-ring of triangles neighboring  $v_i$ , and  $|\mathcal{T}(v_i)|$  is the number of the triangles. The vertex is displaced along its normal to prevent vertex drift in its tangent plane.

We update the vertices using the ordering in a forward and backward direction on alternate iterations, as this provides

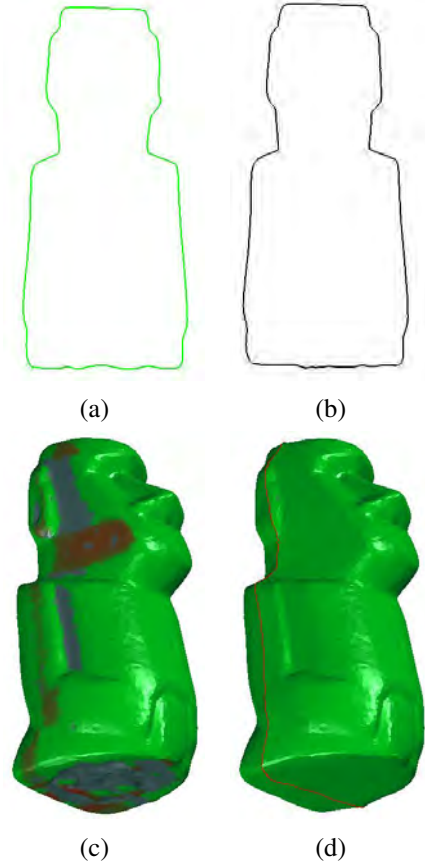


Fig. 10. Moai model: (a) the final parting line, in projection; (b) the exterior profile, (c) all undercuts; (d) the modified model.

faster convergence. In the forward direction, the visited half-ring triangles neighboring  $v_i$  are used, and in the backward direction, the unvisited half-ring triangles neighboring  $v_i$  are used. Updating is iterative until no undercut remains, or a maximum number of iterations is reached.

### D. Restarting

Although extending the undercut regions gives extra freedom to this mesh modification process, such freedom may still not be enough—after several iterations, the vertices may still not produce the desired normals, and hence may not fully remove the undercut. In such cases, we restart, and once again identify undercut regions to be modified, which are usually much smaller than before. Generally this process quickly removes undercut, although at times a few *very* small areas of undercut may remain.

## IX. EXPERIMENTAL RESULTS AND DISCUSSION

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

The Moai model in Figure 1 contains only very short degenerate paths, so the optimised parting line lies close to the theoretically correct one. Note the smoothness of the final parting line in Figure 1(e) compared to the candidate paths shown in Figure 1(c). We have attempted to remove *all*

TABLE I  
PARAMETERS USED FOR EXPERIMENTS, AND ERRORS.

Model	Size	$\delta$	Area	$\epsilon$	MaxErr	AveErr	$\theta$
Moai	10	5°	0.15	0.2	0.141	0.017	3°
Fish	130	8°	10	0.5	1.521	0.352	3°
Alligator	100	5°	4	2	6.365	0.664	—

undercuts: both those caused by the parting line, i.e. the red regions in Figure 10(c), and any original undercuts determined by the parting direction, i.e. the gray regions in Figure 10(c). Some of these undercuts overlap. The modified model is shown in Figure 10(d); the red curve is the parting line. The computation took about 8 minutes. A few very minor undercut regions still remain at the bottom of the model (too small to be seen in the Figure), which are parts of the original undercuts.

The Fish model in Figure 11 is symmetric about a plane through the dorsal fin and tail. The parting direction is almost perpendicular to this plane, but tipped slightly towards the viewer. Its triangle band has a very simple topological structure, shown in Figure 11(b). However, it has an order of magnitude more triangles than the Moai and Alligator models: as can be seen in Table II, the times spent computing paths and optimizing the chosen cycle are longer than for the two other models. Path computation takes time  $\mathcal{O}(n \log n)$ , where  $n$  is the number of triangles in the region containing the path [19]. For this model, we only removed the undercuts caused by the parting line, i.e. the red regions in Figure 11(c). The original undercuts, i.e. the gray regions between the two ventral fins, can not be removed, needing a side core is needed. The modified model is shown in Figure 11(d); the red curve is the parting line. Undercut removal took about 2 hours; there is scope to significantly improve this time by use of more efficient methods for undercut identification than used for our proof-of-concept implementation.

The Alligator shown in Figure 12(a) has a very complex triangle band, in part because it contains several **occluded** regions. These **occluded** regions cause noticeable differences between the exterior profile and the projection of the optimised cycle. The computed parting line for this model is significantly different from the theoretical one, causing large undercuts which overlap with undercuts determined by the parting direction. Our undercut removal method is not appropriate for the modification of the such big undercuts.

The approximate size of the largest dimension of each model, in arbitrary units, the parameters used, and the maximum and average deviation of the smoothed cycle from the exterior profile are summarised in Table I. The maximum error between the theoretical and computed parting line for the alligator model exceeds  $\epsilon$  because of the presence of degenerate paths, where side cores are required. The maximum error between the theoretical and computed parting lines for the fish model also exceeds  $\epsilon$  because some small visible regions have been removed in computing the triangle band (see Section IV).

Our experiments used a 2.61GHz AMD Athlon-64 FX-55 PC with 2GB RAM. Numbers of triangles in each model, as

TABLE II  
TRIANGLE COUNTS AND COMPUTATION TIMES OF DIFFERENT PHASES.

Model	Triangles	Band	Paths	Cycle	Optimize	Undercut
Moai	20000	7s	29s	1s	81s	8 mins
Fish	436794	280s	390s	1s	305s	2 hours
Alligator	48758	37s	159s	1s	88s	—

well as times taken to generate the triangle band, compute the geometric candidate paths, choose the best cycle, and optimise the parting line are given in Table II. The total times taken for parting line generation are a few minutes, and although undercut removal can take rather longer, we believe the algorithm's speed is acceptable for industrial use.

#### ACKNOWLEDGEMENTS

The authors wish to acknowledge the support of EPSRC grant GR/T24579/01, and Delcam plc. We also thank J.-R. Sack for providing us with code for shortest path computation.

#### REFERENCES

- [1] E. L. Buckleitner, *Dubois and Pribble's Plastics Mold Engineering Handbook*. New York: Chapman & Hall, 1995.
- [2] R. Barratt, private communication, 2006.
- [3] A. K. Priyadarshi and S. K. Gupta, "Geometric algorithms for automated design of multi-piece permanent molds," *Computer-Aided Design*, vol. 36, pp. 241–260, Mar. 2004.
- [4] G. Elber, X. Chen, and E. Cohen, "Mold accessibility via gauss map analysis," *Transactions of the ASME: Journal of Computing and Information Science in Engineering*, vol. 5, pp. 79–85, Apr. 2005.
- [5] R. Khardekar, G. Burton, and S. McMains, "Finding feasible mold parting directions using graphics hardware," *Computer-Aided Design*, vol. 38, pp. 327–341, Apr. 2006.
- [6] X. Chen and S. McMains, "Finding all undercut-free parting directions for extrusions," in *Proceedings of GMP 2006, Lecture Notes in Computer Science (LNCS) 4077*, Pittsburgh, Pennsylvania, U.S.A., Jun. 2006, pp. 514–527.
- [7] B. Ravi and M. N. Srinivasan, "Decision criteria for computer-aided parting surface design," *Computer-Aided Design*, vol. 22, pp. 11–18, Jan. 1990.
- [8] J. Majhi, P. Gupta, and R. Janardan, "Computing a flattest, undercut-free parting line for a convex polyhedron, with application to mold design," *Computational Geometry: Theory and Applications*, vol. 13, pp. 229–252, Dec. 1999.
- [9] W. Li, R. R. Martin, and F. C. Langbein, "Generating smooth parting lines for mold design for meshes," in *SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling*, Beijing, China, Jun. 2007, pp. 193–204.
- [10] L.-L. Chen, S.-Y. Chou, and T. C. Woo, "Parting directions for mould and die design," *Computer-Aided Design*, vol. 25, pp. 762–768, Dec. 1993.
- [11] K. C. Hui, "Geometric aspects of the mouldability of parts," *Computer-Aided Design*, vol. 29, pp. 197–208, Mar. 1997.
- [12] X. G. Ye, J. Y. H. Fuh, and K. S. Lee, "A hybrid method for recognition of undercut features from moulded parts," *Computer-Aided Design*, vol. 33, pp. 1023–1034, Dec. 2001.
- [13] M. W. Fu, A. Y. C. Nee, and J. Y. H. Fuh, "The application of surface visibility and moldability to parting line generation," *Computer-Aided Design*, vol. 34, pp. 469–480, Jun. 2002.
- [14] S. T. Tan, M. F. Yuen, W. S. Sze, and K. W. Kwong, "Parting lines and parting surfaces of injection moulded parts," *Proc. IMechE Part B: Journal of Engineering Manufacture*, vol. 204, pp. 211–221, Sep. 1990.
- [15] M. Weinstein and S. Manoochchri, "Optimal parting line design of molded and cast parts for manufacturability," *Journal of Manufacturing Systems*, vol. 16, pp. 1–12, Feb. 1997.
- [16] A. K. Priyadarshi and S. K. Gupta, "Finding mold-piece regions using computer graphics hardware," in *Proceedings of GMP 2006, Lecture Notes in Computer Science (LNCS) 4077*, Pittsburgh, Pennsylvania, U.S.A., Jul. 2006, pp. 655–662.

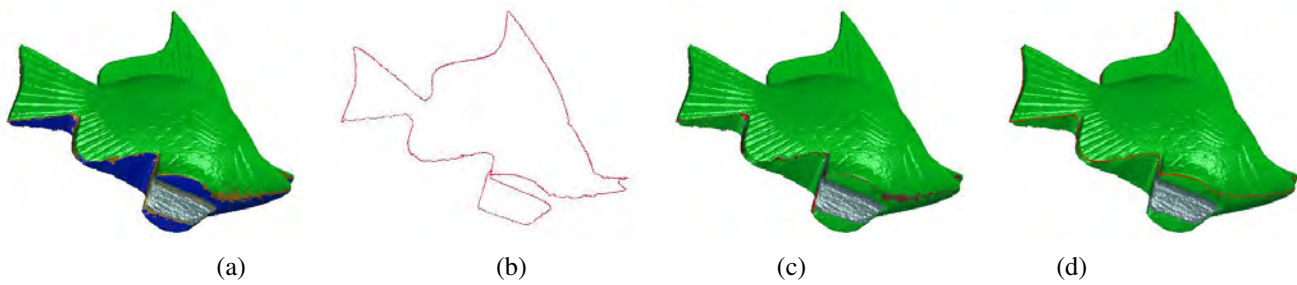


Fig. 11. Fish model: (a) model and triangle band; (b) topological structure of triangle band, (c) all undercuts; (d) the modified model.

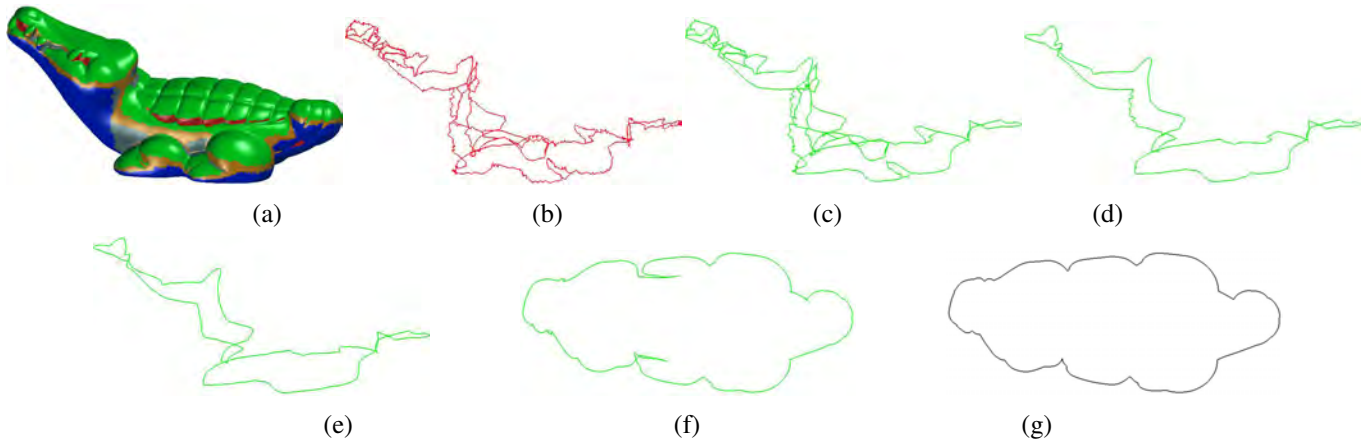


Fig. 12. Alligator model: (a) the model and triangle band; (b) topological structure of candidate paths; (c) candidate paths; (d) chosen cycle; (e) optimised cycle; (f) projection of the optimised cycle; (g) exterior profile.

- [17] C. Rourke and B. Sanderson, *Introduction to Piecewise-Linear Topology*. New York, USA: Springer-Verlag, 1972.
- [18] F. Lazarus, M. Pocchiola, G. Vegter, and A. Verroust, "Computing a canonical polygonal schema of an orientable triangulated surface," in *Proceedings of the seventeenth annual symposium on Computational geometry*, New York, NY, USA, Jun. 2001, pp. 80–89.
- [19] M. Lanthier, A. Maheshwari, and J.-R. Sack, "Approximating weighted shortest paths on polyhedral surfaces," Carleton University, Ottawa, ON, Canada, Tech. Rep. TR-96-32, Dec. 1996.
- [20] D. E. Knuth, *The Art of Computer Programming—Volume 1 (3rd Edition)*. Reading, MA: Addison-Wesley, 1997.
- [21] I. J. W. Gunnink, "High speed milling by using STL-technology," in *Rapid Prototyping & Manufacturing 2000*, Rosemont, Illinois, Apr. 2000.
- [22] T. Berger, B. A. Payne, and W. C. S. III, "Apparatus and methods for modifying a model of an object to enforce compliance with a manufacturing constraint," U.S. Patent 7 149 596, Dec. 12, 2006.
- [23] X. Sun, P. L. Rosin, R. R. Martin, and F. C. Langbein, "Fast and effective feature-preserving mesh denoising," *IEEE Transactions on Visualization and Computer Graphics*, vol. 13, pp. 925–938, Sep-Oct 2007.
- [24] Y. Shen and K. Barner, "Fuzzy vector median-based surface smoothing," *IEEE Transactions on Visualization and Computer Graphics*, vol. 10, pp. 252–265, May-June 2004.



**Weishi Li** is a research associate at Cardiff University. He obtained his PhD in 2002 from Zhejiang University. Before joining Cardiff University, he had been with Shanghai Jiao Tong University, China as a lecturer and Institute of High Performance Computing, Singapore as a research fellow. He has published about 20 papers on reverse engineering, geometric modelling and computing.



**Ralph R. Martin** has been working in the field of geometric modelling since 1979. He obtained his PhD in 1983 from Cambridge University for a dissertation on "Principal Patches", and since then has progressed from Lecturer to Professor at Cardiff University. His publications include over 170 papers and 10 books covering such topics as solid modelling, surface modelling, intelligent sketch input, vision based geometric inspection, geometric reasoning and reverse engineering. He is a Fellow of the Institute of Mathematics and its Applications, and a Member of the British Computer Society. He is on the editorial boards of "Computer Aided Design", the "International Journal of Shape Modelling", "CAD and Applications", and the "International Journal of CAD/CAM".



**Frank C. Langbein** received a Diploma in Mathematics from the University of Stuttgart in 1998. He obtained his PhD in 2003 from Cardiff University for a dissertation on "Beautification of Reverse Engineered Geometric Models". Since then he has been a Lecturer in Computer Science at Cardiff University. His research interests in geometric modelling include detection and representation of design intent in geometric models, geometric constraints, mesh processing, free-form surfaces and reverse engineering. He is also interested in quantum modelling and quantum information and works on simulation, identification and control of quantum systems. He is a member of the American Mathematical Society and the IEEE.