EXTRACTING MANIFOLD AND FEATURE-ENHANCED MESH SURFACES FROM BINARY VOLUMES

Charlie C.L. Wang
The Chinese University of Hong Kong
Shatin, N.T., Hong Kong
cwang@mae.cuhk.edu.hk

ABSTRACT

This paper presents an approach to automatically recover mesh surfaces with sharp-edges for solids from their binary volumetric discretizations (i.e., voxel models). Our method consists of three steps. The topology singularity is first eliminated on the binary grids so that a topology correct mesh M^0 can be easily constructed. After that, the shape of M^0 is refined and its connectivity is iteratively optimized into M^n. The shape refinement is governed by the duplex distance-fields derived from the input binary volume model. However, the refined mesh surface lacks sharp edges. Therefore, we employ an error-controlled variational shape approximation (VSA) algorithm to segment M^n into nearly planar patches, and then recover sharp edges by applying a novel segmentation-enhanced bilateral filter to the surface. Using the technique presented in this paper, smooth regions and sharp edges can be automatically recovered from raw binary volume models without scalar field or Hermite data. Comparing to other related surface recovering methods on binary volume, our algorithm needs less heuristic coefficients.

KEYWORDS: surface reconstruction, binary volume model, sharp edge recovery, reverse engineering, geometric modeling.

1. INTRODUCTION

The purpose of the research presented in this paper is to develop technology for automatically converting binary volume models into B-rep mesh surfaces, so that they can be directly applied to the downstream CAD/CAM applications. Benefited from the compact and intuitive mathematical representation, many design optimization approaches employed the implicit representation to optimize the shape and topology of products [1-3]. In their approaches, the shape of optimized product is sampled on regular grids (i.e., represented by voxels or pixels). After determining the optimal structure, it is usually necessary to convert the structures into a B-rep surface model so that it can be explicitly modified and finally manufactured. This is the motivation of our work – *to develop a technique that can automatically reconstruct two-manifold mesh surface with sharp-edges from a given binary volume model.*

More specifically, a binary volume model H is a set of integral vectors in Z^3. Elements $s_{i,j,k} = (i,j,k) \in \mathrm{H} \subset Z^3$ are called *voxels* and are thought of as unit cubes centred at (i,j,k). We are going to reconstruct a two-manifold mesh surface M to approximate, $\partial\mathrm{H}$, the surface of H. The error between M and $\partial\mathrm{H}$ should be bounded, and we aim to recover the sharp-edges which are damaged by the uniform sampling in Z^3. To simplify the description of the technique, we assume the width of voxels is of unit length in the rest of this paper (the change of voxel size can be easily implemented in practice).

1.1 Related work

The reason why we do not directly apply the famous marching cubes algorithm [4] or its variants [5-17] to extract M is twofold: the two-manifold preservation and the element shape control. *Marching cubes* (MC) algorithm was first introduced by Lorensen and Cline [4] and has become the most commonly used method for isosurface extraction in scientific visualization. As first noted by Duerst [5], the original MC algorithm [4] may produce isosurfaces with holes due to topologically inconsistent decisions on the reconstruction of ambiguous faces, where the borders used by one incident cube do not match the borders of the other incident cube. Several approaches addressing this problem have been published (see [6, 7] for a review). As addressed by [7], disambiguation techniques reported so far have focused on two major concerns: topological consistency [8-12] (i.e., producing closed surfaces by proper cube polygonization), and topological correctness [13-17] (i.e., extracting a surface faithful to the geometry of the real surface). A few works attempt to recover the original topology also inside the ambiguous cubes either by using critical point analysis [18, 19] or trilinear interpolation [20, 21]. All these techniques are scalar-data-dependent and therefore cannot be applied to binary grids. In [7], the authors propose global strategies for optimizing several topological and combinatorial measures of the isosurfaces including triangle count, genus, and number of shells. However, the decisions of the measurements to be given by users are not natural, i.e., novices feel difficult to give good decision. Different from them, our algorithm follows the *Nyquist-Shannon Sampling Theorem* to reconstruct mesh surface with correct topology – no additional user input is required. The authors of [62] employed an automatically constructed lookup table to ensure the topology of resultant mesh surfaces; similarly, the approach in [17] considered the relationship between cubes to develop an extended MC table for topological guarantees. However, the implementations of these approaches are not as simple as our *CellMerge* algorithm, and they always result in many sliver polygons as other MC-like algorithms (e.g., see Figure 1).

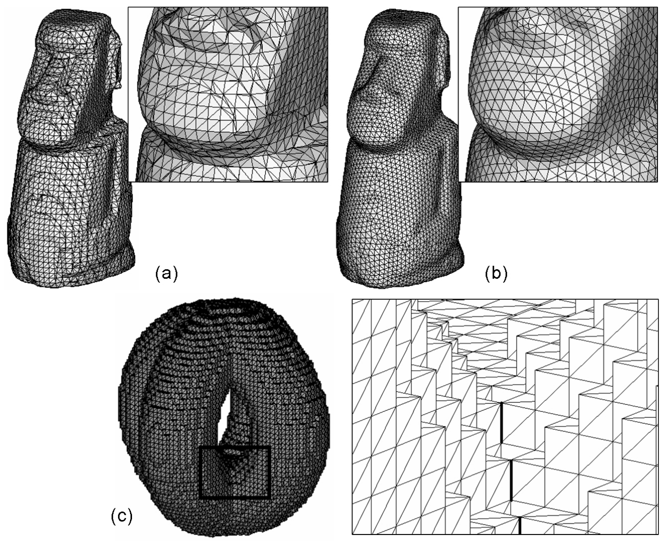


Figure 1: A model contoured using Marching Cubes algorithm [17] resulting in many triangles (a) and our approach resulting in good shape triangles (b). (c) non-manifold edges (the bolded ones are produced by using [28] to construct the connectivity of isosurface.

In MC algorithms, there is no control on the element shape that is important to many downstream applications. As shown in Figure 1(a), the resultant mesh by the algorithm in [17] has a lot of thin triangles and short edges, however the result of our approach shown in Figure 1(b) gives good element shapes. Although post-processing steps, such as triangle decimation techniques [22-24] and the re-tiling algorithm [25] can be adopted to eliminate badly shaped triangles, directly applying them to the result of MC algorithms gives no bound on the error between ∂H and the output of MC algorithms. In our approach, we develop a method using duplex distance-fields to control the error. The remeshing iteration of our algorithm borrows some idea from the triangle decimation approaches.

Another class of isosurface generation is based on the active model, where the constructed surface is deformed to approximate the underground isosurface embedded in a scalar-field. Crossno and Angel [26] employed particle systems to extract isosurfaces, where particles are programmed to attract towards a specific surface value while simultaneously repelling adjacent particles. The repulsive forces are based on the curvature of the surface at that location. The smooth shape reconstruction step of our approach is also conducted in the similar way, but is speeded up by the precomputed duplex distance-fields. The SurfaceNets algorithm developed in [27] is an alternative to MC for building globally smooth but locally accurate triangle models from binary volume data. In [28] this algorithm is further enhanced in the Kizamu system to generate mesh models from distance values sampled on an adaptive grid. During our implement of their algorithm, we find that the EdgeFace table in [28] does not guarantee to generate a manifold mesh surface (see Figure 1(c)). To solve this problem, a two-manifold preserved algorithm will be developed in section 3 to extract the connectivity of mesh surface from binary grids. A recent surface reconstruction algorithm [63] employs the similar method to generate mesh surfaces from volume data.

The accuracy of a marching cube algorithm is mainly governed by the resolution of an underlying grid, so sharp features cannot be preserved. Over-sampling could somewhat reduce the aliasing error by taking the cost of increasing storage memory. Furthermore, as being observed by Kobbelt et al. in [29], even if an over-sampling is applied, the associated aliasing error will not be absolutely eliminated since the surface normals in the reconstructed model usually do not converge to the normal field of the original model. Therefore, the technique of recovering sharp edges on feature-insensitive sampled models is desired. Some of currently existing approaches (e.g., [29-32]) encode the original surface normals during sampling, so that a Hermite dataset is generated to reconstruct sharp features. However, no Hermite dataset can be obtained on binary volume models. The most recent sharpen and bend technique of Attene et al. [33] gives two filters that improve the quality of sampled surfaces which chamfer sharp features, so that the curved sharp edges in triangular meshes produced by feature-insensitive sampling can be recovered. However, the filters introduced in [33] can only sharpen the “chamfered” edges. For the insensitive sampled edges that are rounded (e.g., Figure 2(d)), the algorithm of [33] fails. The rounded edges are usually generated by dynamic surface extraction algorithms – e.g., ShrinkWrap [34], Skin [35], and our approach. Therefore, a new sharpening algorithm is introduced in section 5. Different from the approaches in [36] and [37], the newly developed recovery technique for sharp-edges integrates the segmentation and the bilateral filtering into a segmentation-enhanced bilateral filter so that it is more robust than the approaches based on the identification of “sharp” regions by normal variations.

1.2 Contributions

The techniques developed in this paper contribute in the following three aspects:

- We propose a new algorithm to construct the coarse shape of an isosurface. Our method automatically resolves the topology ambiguity by simple rules, whose implementation is much simpler than the marching cubes algorithm and its variants.
- The refined shape of isosurface is determined by duplex distance-fields and the smoothing operator, where the error between the reconstructed surface and the given binary volume model is bounded by the duplex distance fields.
- We integrate the segmentation into our algorithm to identify regions with sharp edges, and then recover sharp edges in these regions using a segmentation-enhanced bilateral filtering algorithm.

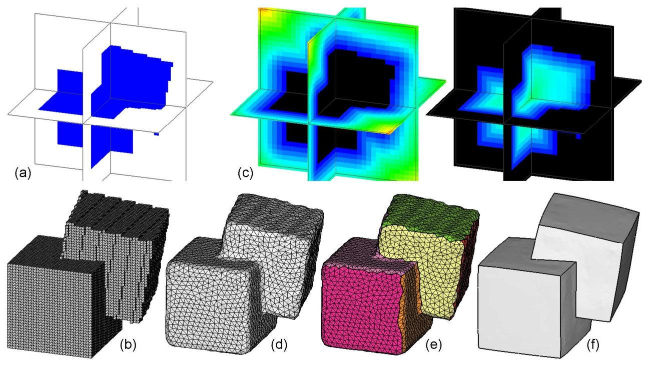


Figure 2: Step-results for an example model with two cubes to illustrate the overview of our approach: (a) the binary volume model is shown on three cross-section planes, (b) the two-manifold coarse mesh M^0 approximating ∂H with correct topology, (c) duplex distance-fields – the left one is generated by the boundary voxels in H and the right one is from the boundary voxels in $Z^3 \setminus H$ (where blank represents the point with negative distance value), (d) the resultant mesh M^n of smooth shape reconstruction, (e) the segmented patches by variational shape approximation, (f) the sharpening result from segmentation-enhanced bilateral filtering.

These techniques lead to the new function that flat regions, curved regions, and sharp edges are automatically recovered from raw binary volume models without requiring scalar field or Hermite data. Compared to our previous work presented in [36, 37, 60], the approach presented in this paper has the following advancements.

- A new *Topological Singularity Elimination* algorithm has been developed and governed by the Lemmas in section 3;
- A out-of-core extension of the *CubeMerge* algorithm (see section 4);
- A new sharp edge recovery algorithm is presented in section 5.

2. METHOD OVERVIEW

To recover the mesh surface with sharp-edges for solids from their binary volumetric discretizations, our approach consists of three steps as follows.

1) *Topology reconstruction*: The first step is to identify and eliminate the topologically singular vertices/edges on the input binary volume model H. Based on the corrected voxel set, a *CubeMerge* algorithm is developed, whose output mesh surface M^0 is guaranteed to be two-manifold and consistent to the topology of H. M^0 gives a coarse mesh approximation ∂H. We also give an out-of-core implementation of the *CubeMerge* algorithm. Figure 2(b) shows an example M^0 generated from the input binary grids in Figure 2(a).

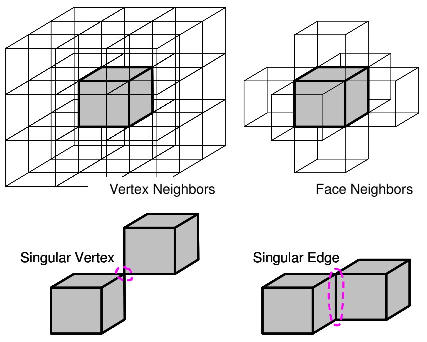


Figure 3: Illustration for the vertex neighbors, the face neighbors, the singular vertex and the singular edge.

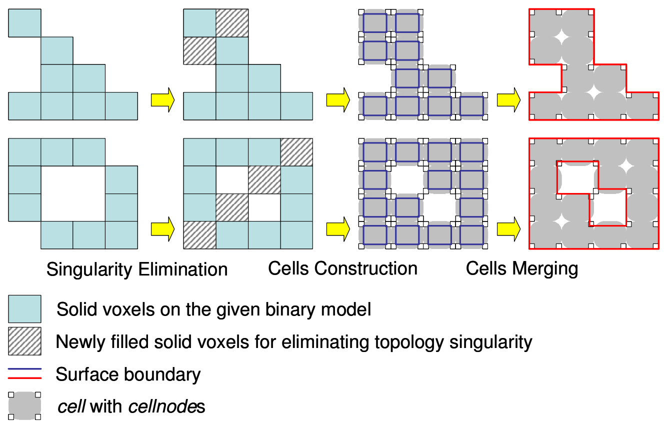

Singularity Elimination
Cells Construction
Cells Merging
Solid voxels on the given binary model
Newly filled solid voxels for eliminating topology singularity
Surface boundary
cell with cellnodes

Figure 4: Illustration for topology reconstruction.

2) *Smooth shape reconstruction*: The shape of M^0 is affected by aliasing artifacts, so the shape of M^0 will be smoothened in the second step. Meanwhile, the mesh connectivity will also be improved iteratively from M^0 to M^n (i.e., $M^0 \to M^1 \to \cdots \to M^n$). In order to bound the approximation error between M^i and ∂H, duplex distance-fields (one by the boundary voxels in H, and another by the boundary voxels in $Z^3 \setminus H$) are constructed to govern the movement of vertices on M^i. For example, Figure 2(c) shows the duplex distance-fields for the binary volume model in Figure 2(a), and Figure 2(d) gives the output mesh, M^n, of this step.

3) *Recovery of sharp edges*: No sharp edges exist on the mesh from the previous step of our approach. A novel segmentation and bilateral filtering based method is developed to recover sharp edges. An error-controlled variational shape approximation (VSA) algorithm is first employed to segment M^n into near planar patches (e.g., see Figure 2(e)). Then, the normal vectors on each triangle of M^n are filtered through a segmentation-enhanced filtering. Finally, the surface vertices are updated to let triangles follow filtered normal vectors so that sharp edges

are formed and other regions are smoothed (see Figure 2(f)).

Details of these three steps will be introduced in sections 3 to 5 consecutively. Section 6 gives the experimental results and discussion. Lastly, our paper ends with the conclusion section.

3. TOPOLOGY RECONSTRUCTION

3.1 Singularity elimination

In the binary volume model, the basic element voxel $s \in Z^3$, if $s \in H$, it is called *solid*; otherwise, called *empty* for $s \in Z^3 \setminus H$.

Definition 1 Two voxels $s, t \in Z^3$ are *vertex-neighbors* if $\|s-t\|_\infty = 1$, and are called *face-neighbors* if $\|s-t\|_1 = 1$.

To identify singular topology shown in the given binary volume model, we define some local region sets.

Definition 2 A local volume set R_v is defined as

$$R_V = \{s_{i+\alpha,j+\beta,k+\gamma} : \alpha,\beta,\gamma \in \{0,1\}\}, \tag{1}$$

and three local planar sets R_x, R_y, and R_z are defined as

$$R_x = \{s_{i,j+\beta,k+\gamma} : \beta,\gamma \in \{0,1\}\}, \tag{2}$$

$$R_y = \{s_{i+\alpha,j,k+\gamma} : \alpha,\gamma \in \{0,1\}\}, \tag{3}$$

$$R_z = \{s_{i+\alpha,j+\beta,k} : \alpha,\beta \in \{0,1\}\}. \tag{4}$$

By above two definitions, the following lemmas are derived to detect singular vertices and singular edges, where $|\cdots|$ gives the number of elements in a set.

Lemma 1 For two solid voxels $s, t \in R_V$, if $\|s-t\|_1 = 3$ and $|R_V \cap H| = 2$, *s* and *t* are linked by a *singular vertex*.

Lemma 2 For two solid voxels $s, t \in R_a$, if $\|s-t\|_1 = 2$ and $|R_a \cap H| = 2$, *s* and *t* are linked by a *singular edge* parallel to *a*-axis with *a* be *x*, *y*, or *z*.

An illustration of singular edges and singular vertices is shown in Figure 3.

Assumption 1 On the binary volume model H, if two solid voxels *s* and *t* are vertex-neighbors, the two spaces inside *s* and *t* are assumed to be connected.

Based on this assumption, and Lemma 1 and 2, we introduce the *Topological Singularity Elimination* algorithm (*TSE* in short) which detects and eliminates singularities in two runs: at first, the local volume sets containing singular vertices and the local planar sets holding singular edges are detected on the given model H; secondly, all empty voxels in both the local volume sets and the local planar sets containing singularities are changed to solid voxels (i.e., H is modified to H').

3.2 Construct meshes with correct topology

After computing the singularity eliminated H', we can construct a coarse mesh surface M^0 whose topology is consistent with ∂H under Assumption 1 in section 3.1. The basic idea of our method is under a cube-merging strategy so that it is named as the *CubeMerge* algorithm. Firstly, B-rep cubes are created for every solid voxels. The cubes, whose corresponding voxels are face-neighboured, are then merged by merging relevant nodes and removing the face in-between. The *CubeMerge* results in a two-manifold polygonal mesh surface M^0. Since we only merge polygons on those face-neighboring cubes, the two-manifold topology is preserved during the cubes merging. Detail analysis of manifold-preservation can be found in our previous publication [60]. Figure 4 gives a two-dimensional illustration for the principle of our algorithm, where polygons are first constructed on the boundary of cells and then eliminated into a two-manifold polygonal mesh M^0 during the cubes merging. Note that for models with complex topology, new singularities may be generated on H'.

Observation 1 The singular vertices or edges on H' should be separated in order to follow the topology of ∂H.

This is because that the singularities only on H' but not on H is formed by the newly added solid voxels in TSE, which originally are empty. When only merging face-neighboring cubes, the mesh surface generated by our *CubeMerge* algorithm follows the above observation. The combination of TSE and *CubeMerge* gives the following property on the resultant mesh M^0.

Property 1 Holes on the given binary volume model H, whose size are not less than two unit widths, will not be damaged.

According to the *Nyquist-Shannon Sampling Theorem*, the signal sampled into discretizations can be reconstructed only if the sampling rate is greater than two times of the highest frequency embedded in the original signal before sampling. Applying this theorem to the discretization of binary volumes, holes should be sampled into more than two voxels width. If the sampling process satisfies this rate, our topology reconstruction algorithms can reconstruct a mesh surface M^0 with the consistent topology to ∂H. Only H, not any other heuristic input, is needed here.

Implementation Detail The implementation of *CubeMerge* algorithm is based on two entities – *cell* and *cellnode*, and three operators – *cell-create*, *node-merge* and *cell-merge*. Their details are listed in Table 1 and Table 2. We first apply the *cell-create* operator to every solid voxel in H' to create cubes. After that, for each cell $c_{i,j,k}$, if it has not been merged with one of its face neighboring cell c_f, we apply the *cell-merge* operator on $c_{i,j,k}$ and c_f. Finally, polygons are constructed only on the faces of cells where there is no neighbour. An illustration for cells merging has been given in Figure 4. The implementation is simple but needs a lot of memory when the resolution of H is high.

The out-of-core extension of the above basic *CubeMerge* algorithm is conducted in a layer by layer manner – here a layer means all voxels in Z^3 with the same *z*-coordinate. In the following pseudo-code of our out-of-core implementation, we maintain two layers simultaneously. Starting from the layer with

Table 1: Entities – *cell* and *cellnode*

```
Entity cell {
    cellnode* nodes[8];      // the pointer of 8 nodes in a cell
    bool bMerged[6];         // the flag to identify whether the cell
                             // in the ith direction has been merged
}
Entity cellnode {
    float pos[3];            // the position of this node
    cell** cellList;         // the list of cells containing this node
    int num;                 // the number of cells in cellList
}
```

Table 2: Operators – *cell-create*, *node-merge* and *cell-merge*

```
Operator cell-create {
    Construct a cell, and its eight cellnodes which are positioned at
        the 8 corners of a solid voxel;
    The pointer to the cell is saved in the cellList of every node;
    The merge flags in the cell are all set to false;
}
Operator node-merge { // applied on two nodes $v_1$ and $v_2$
    Replace the pointer in every cell linked to $v_2$ by the $v_1$;
    Add all cells in the cellList of $v_2$ into the cellList of $v_1$;
    Delete $v_2$;
}
Operator cell-merge { // applied on two cells $c_1$ and $c_2$
    There are four pairs of nodes to be merged – they are merged by
        applying the operator node-merge pair by pair;
    Turn the corresponding merge flags in $c_1$ and $c_2$ to true.
}
```

the lowest z-coordinate, the 3D polygons for M^0 are progressively constructed. Note that in step 8 and 9, polygons are constructed only on the faces of cells where there is no neighbour.

Algorithm *CubeMerge*

```
1.  Sort all voxels in $Z^3$ into layers by z-coordinate;
2.  Start from the layer $z = 0$;
3.  Create an empty layer (named as bottom-layer);
4.  Repeat {
5.      Create cells for solid voxels in the layer z – this layer is
            named as top-layer;
6.      Merge cells in the top-layer;
7.      Merge cells between the top-layer and the bottom-layer;
8.      Create polygonal faces perpendicular to x- or y-axis on
            the cells in the bottom-layer;
9.      Create polygonal faces perpendicular to z-axis between
            the top-layer and the bottom-layer;
10.     Free the memory of cells in the bottom-layer;
11.     Assign top-layer to be the bottom-layer;
12.     $z = z + 1$;
13. } Until ( z has arrived the upper-bound of $Z^3$ );
14. Free the memory of cells in the top-layer;
15. Construct $\mathrm{M}^0$ by the polygonal faces.
```

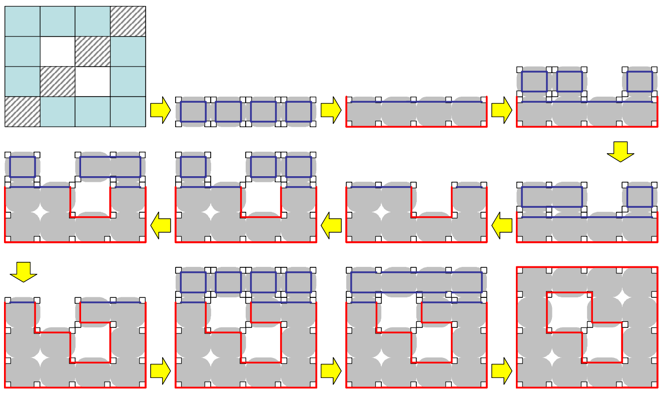
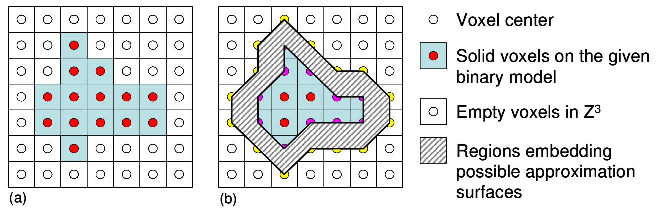


Figure 5: 2D illustration for the progressive results from the out-of-core implementation of *CubeMerge* algorithm – every 2D quadrilateral denotes a cubic cell in 3D.



(a)
(b)
Voxel center
Solid voxels on the given binary model
Empty voxels in Z³
Regions embedding possible approximation surfaces

Figure 6: By a given binary volume model H in the left, the region holding reasonable approximation of $\partial\mathrm{H}$ is shown in the right.

Figure 5 illustrates the progressive results from the out-of-all algorithm for the example shown in the top row of Figure 4. Since the memory of cells is released layer by layer during the bottom-up advancement, the requested memory is only linear to the cells contained in one layer.

4. SMOOTH SHAPE RECONSTRUCTION

The zigzag shape of M^0 will be processed iteratively: $\mathrm{M}^0 \rightarrow \mathrm{M}^1 \rightarrow \cdots \rightarrow \mathrm{M}^n$ so that the shape of M^n approximates $\partial\mathrm{H}$ smoothly. The element shape on a mesh surface is also optimized for the downstream applications – ideally, every element is expected to be regular.

4.1 Duplex distance-fields for surface reshaping

Different from the pure surface smoothing algorithm (e.g., [38-42]), the reference geometry in our approach is not the given mesh surface M^0. Instead, $\partial\mathrm{H}$ will be the reference surface here. However, the shape of $\partial\mathrm{H}$ is not explicitly defined on the model H. The boundary of voxels in H is just one candidate among all possible surfaces. More specifically, a good shape approximate of $\partial\mathrm{H}$ will be a surface falling in the region between the two surfaces formed by the centres of boundary voxels[1] in H and the centres of boundary voxels in $Z^3 \backslash \mathrm{H}$ (in short, \H). For example, for the model H given in

[1] For a set of solid voxels, a voxel which has any empty face neighbor is a boundary voxel of this voxel-set.

Figure 6(a), the approximation of ∂H should be in the region shown by Figure 6(b). The problem is how to effectively and efficiently control the shape of M^i in this region during the evolution of meshes. We introduce the idea of duplex distance-fields for this purpose.

Definition 3 For a given surface *S*, a signed distance-field defined on it is a function $D(x,y,z)$ assigning to every point $(x,y,z) \in \Re^3$ its distance $D(x,y,z) = dist((x,y,z),S)$ with a *positive* sign for points outside the region enclosed by *S* and *negative* for points inside *S*.

A convenient way to store the distance-field *D* in an efficient data structure is to sample *D* on a uniform spatial grid $d_{i,j,k} = (ih, jh, kh)$. For a point $p = (x,y,z)$ with

$x \in [ih,(i+1)h)$, $y \in [jh,(j+1)h)$, $z \in [kh,(k+1)h)$,

its Euclidean distance to *S* can be interpolated on the grid

$$[ih,(i+1)h) \times [jh,(j+1)h) \times [kh,(k+1)h)$$

by a tri-linear function such that we obtain a piecewise tri-linear approximation $\overline{D}(p)$ for the original distance-field $D(p)$. Meanwhile, a corresponding isosurface S^* defined by $\overline{D}(p) = 0$ gives an approximation to *S*. The smaller grid size *h* chosen, the more accurate approximation of $D(p)$ is given by $\overline{D}(p)$, but more memory is needed for storing $\overline{D}(p)$. In our approach, a balance is found when *h* is twice of the voxel width on H. The grid nodes are located at the centre of the voxels.

Two signed distance-fields are defined – one is for H (as $\overline{D}_{\mathrm{H}}$) while another for \H (as $\overline{D}_{\backslash \mathrm{H}}$). For constructing a signed distance-field $\overline{D}_{\mathrm{H}}$ for a voxel set H, the grid nodes in $\overline{D}_{\mathrm{H}}$, whose positions are coincident to the centres of boundary voxels, will be firstly detected and the distance value on them are assigned as *zero*. Then, after setting the distance values of other sample points to ∞, the vector distance transforms (VDTs) presented in [43] are applied to propagate the distance values to all grid nodes. The sign of distance at every sample point (ih, jh, kh) can be detected by whether the voxel $s_{i',j',k'}$ centred at this point is in H. If $s_{i',j',k'} \in \mathrm{H}$, the sign is negative, otherwise a positive distance value is given. The processing time of VDTs in [43] is linear to the number of grid nodes. The distance-field $\overline{D}_{\backslash \mathrm{H}}$ can be constructed similarly from the voxel set \H. $\overline{D}_{\mathrm{H}}$ and $\overline{D}_{\backslash \mathrm{H}}$ together are named as the *duplex distance-fields*, which are employed to constrain the shape of M^i during the evolution. Note that the original voxel set H (but not H') is used to generate duplex distance-fields here.

4.2 Surface remeshing and reshaping

Using the duplex distance-fields generated in section 4.1, the mesh surface M^0 is refined and optimized to give a quality approximation of $\partial\mathrm{H}$. Similar to previous explicit remeshing approaches [44-47], we iteratively equalize edge lengths and vertex valences so that it remeshes the give surface. We perform the following steps by a given target edge length $\overline{L}$ (in our implementation, we choose $\overline{L} = h$):

1. Split all edges which are longer than $2\overline{L}$ at their midpoint;
2. Collapse all edges shorter than $0.75\overline{L}$ into their midpoint;
3. Flip edges to minimize the deviation of valence from 6;
4. Relocate vertices by D_{H}, $D_{\backslash \mathrm{H}}$ and the area-gravity-weighted centroid;

After repeating these steps for several runs (about 5), we obtain a triangular mesh M^0 whose edges have length close to $\overline{L}$ and whose vertices have valence close to 6. Note that the edge collapse and edge flip operations which lead to topology degeneration will be prevented (ref. [48]).

The purposes of the 4th step in the above algorithm are to move all vertices of M^i to ∂H and relax the distribution of vertices on M^i. The surface ∂H is simulated by the isosurfaces $\overline{D}_{\mathrm{H}} = 0$ and $\overline{D}_{\backslash \mathrm{H}} = 0$. When relocating a vertex $v \in M^i$, *v* is attracted to move towards these two isosurfaces. Meanwhile, to improve the regularity, every vertex is expected to be close to its gravity-weighted centroid. The functional below governs the vertex repositioning

$$\min_{v} \left\{ \omega\left(\overline{D}^2_{\mathrm{H}}(v) + \overline{D}^2_{\backslash \mathrm{H}}(v)\right) + \lambda \left\| \left(\frac{1}{A(v)} \sum_{k=1}^{n} A(q_k) q_k \right) - v \right\|^2 \right\}. \quad (5)$$

In the second term, q_ks are the one-ring neighbours of *v*, where each vertex q_k is assigned with a gravity that equals its Voronoi area $A(q_k)$ and $A(v)$ is the sum of $A(q_k)$. This follows the area-equalization in [45, 47]. For the weights of functional terms, we choose $\omega = 0.125$ and $\lambda = 0.25$ in our implementation to balance the weights of attraction and relaxation. For relocating vertices, their positions are repeatedly updated to minimize the above functional as below with a damping factor 0.1 for about 10 iterations.

$$v' \leftarrow v + 0.1(-\nabla_v J). \quad (6)$$

where J denotes the objective function in Eq.(5).

5. RECOVERY OF SHARP EDGES

The mesh surface reconstructed by previous two steps of algorithm lacks sharp edges. This section introduces the method to recover sharp edges through a novel algorithm which integrates the error-controlled segmentation and a normal-based bilateral filtering.

5.1 $L^{2,1}$ planar segmentation

The given model M^n will first be segmented into nearly planar patches by the Variational Shape Approximation (VSA) algorithm [49]. We control the shape approximation error instead of the proxy number. Starting from one seed, we incrementally add more seeds into the *k*-proxy clustering algorithm until the maximal approximation error shown on all proxies is less than a given tolerance. Following [49], the $L^{2,1}$ approximation error is computed on every triangle of M^n. For a triangle T_i of area $|T_i|$, of normal n_i, and of associated proxy $\hbar$, its $L^{2,1}$ error is computed as

$$L^{2,1}(T_i, \hbar) = \|n_i - n_{\hbar}\|^2 |T_i|. \quad (7)$$

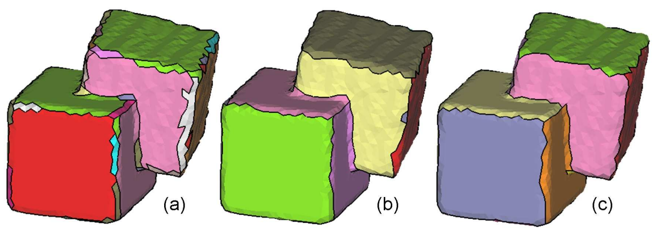


Figure 7: Results of $L^{2,1}$ planar segmentation (a) with 68 patches and the followed boundary refinements (b) with 14 patches and (c) with 12 patches on the two-cubes example.

$n_\hbar$ is the normalized vector by summing area weighted triangles normals on $\hbar$, i.e.,

$$n_\hbar = \sum_{T_j \in \hbar} |T_j| n_j \Big/ \left\| \sum_{T_j \in \hbar} |T_j| n_j \right\| . \tag{8}$$

Note that $n_\hbar$ in Eq.(8) will only be updated after the completion of clustering but not changed during clustering in each iteration. Details can be found in [49]. If $\max L^{2,1} > \varepsilon \bar{A}$ ($\bar{A}$ is the average triangle area on *H*), we increase *k* seeds into *k*+1 and apply the Lloyd algorithm [61] based clustering again. Here $\varepsilon = 0.4$ is determined by experiences. Usually, a smaller ε makes resultant proxies more planar, while a larger ε leads to smaller number of proxies but each with larger $L^{2,1}$ approximation error. The newly inserted seed is located on the triangle which gives the maximal $L^{2,1}$ error. The *k*-proxy clustering algorithm is detailed by Cohen-Steiner et al. in [49].

The error-controlled VSA results in a number of small patches on a model with aliasing error near sharp edges (e.g., Figure 7). The boundaries of patches are refined by the minimum-cut method akin to [50] so that the zigzag effects are improved. More specifically, a fuzzy area Γ around the boundaries defined above is determined. The faces in this fuzzy area Γ are converted to nodes of a weighted graph, and the edges on faces in Γ are corresponding to the arcs in the graph. The weight on the arc from an edge *e* is defined by the length of *e* weighted by

$$w_e = \frac{1}{1 + (ang(e) / avg\_ang)}, \tag{9}$$

where $ang(e)$ is the angle between normals on its adjacent two faces, and avg_ang denotes the average angle between all adjacent faces on M^n. The re-partition of triangles can be found by a maximum-flow (minimum-cut) algorithm (ref. [51]) on the weighted graph. The refinement is repeated for a few times (2 or 3) to remove the small/narrow patches. Results of the two-cubes example are shown in Figure 7. This planar segmentation actually provides the region that may embed sharp edges – i.e., regions near the boundary of each patch. We define the *potential sharp region* as follows.

Definition 4 An edge is named as *boundary edge* if its left and right faces belonging to different proxies. When centering at a vertex on a boundary edge, all its two-ring neighbours are called *sharp region vertices* and the regions occupied by them are defined as the *potential sharp regions*.

Faces in potential sharp regions and the rest of surface will be processed separately in the segmentation-enhanced bilateral filtering below.

5.2 Bilateral recovering of sharp edges

Based on the planar segmentation result, we apply a normal-based bilateral filtering to process the normal vectors of each triangle and then reposition vertices to follow the normals on the adjacent triangles. Using this process the sharp edges can be recovered.

The segmentation-enhanced bilateral filter is extended from the bilateral filter on 2D image, which is a nonlinear feature-preserved image filter proposed separately by Smith and Brady [52] and Tomasi and Manduchi [53]. Recent research [54, 55] shows that this filter has close connections with the robust estimation and anisotropic diffusion. For a mesh face *f* with the unit surface normal n_f and centered at c_f, the filtered normal $\bar{n}_f$ at the face *f* is computed by

$$\bar{n}_f = \frac{1}{k(f)} \sum_{q \in N(f)} W_p(f,q) W_c(\|c_f - c_q\|) W_s(I(f,q)) n_q , \tag{10}$$

where

$$k(f) = \sum_{q \in N(f)} W_p(f,q) W_c(\|c_f - c_q\|) W_s(I(f,q)),$$

N(*f*) is the neighbor of *f* and defined to be the set of triangles

$$N(f) = \{ q : \|c_q - c_f\| < 2\sigma_c \} . \tag{11}$$

$$W_c(t) = e^{-t^2/2\sigma_c^2} , \tag{12}$$

is the standard Gaussian filter with parameter σ_c, and

$$W_s(t) = e^{-t^2/2\sigma_s^2} , \tag{13}$$

is a similarity weight function for feature-preserving with parameter σ_s that penalizes large variation in face normals. $I(f,q)$ defines the projection of the normal difference on the face normal n_f as

$$I(f,q) = n_f \cdot (n_f - n_q) . \tag{14}$$

Our filter is different from the bilateral filters in [36, 56-58] for 3D models. It has one more step function $W_p(f,q)$ with parameter σ_p to reduce the blurred effect near sharp features

$$W_p(f,q) = \begin{cases} 0 & (n_{\hbar(f)} \cdot n_{\hbar(g)} < \sigma_p) \\ 1 & (n_{\hbar(f)} \cdot n_{\hbar(g)} \geq \sigma_p) \end{cases}, \tag{15}$$

where $\hbar(f)$ returns the proxy that the face *f* belongs to, and $n_{\hbar(f)}$ is the proxy normal vector computed by Eq.(8) but not the face normal. Note that this step function seldom produces unwanted sharp-edges on smooth regions since many small pieces are segmented on curved smooth regions. As the segmented regions are small, the variation between the proxy normals should be always smaller than σ_p.

The normal processing is repeatedly applied to all triangles for several runs (about five in our implementation). Then, the vertices are repositioned to follow the processed face normal

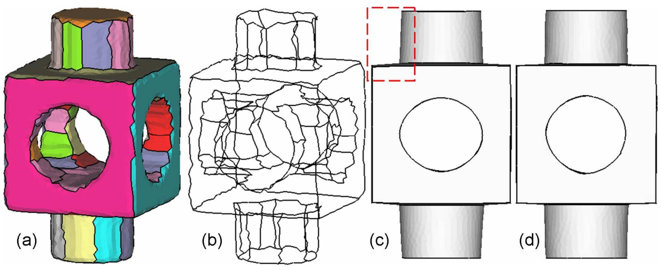


Figure 8: Comparison of position update all together (c) vs. separately for smooth region and shape region (d), where unwanted slopes are generated in (c). (a) the segmentation result, and (b) all patch boundary edges.

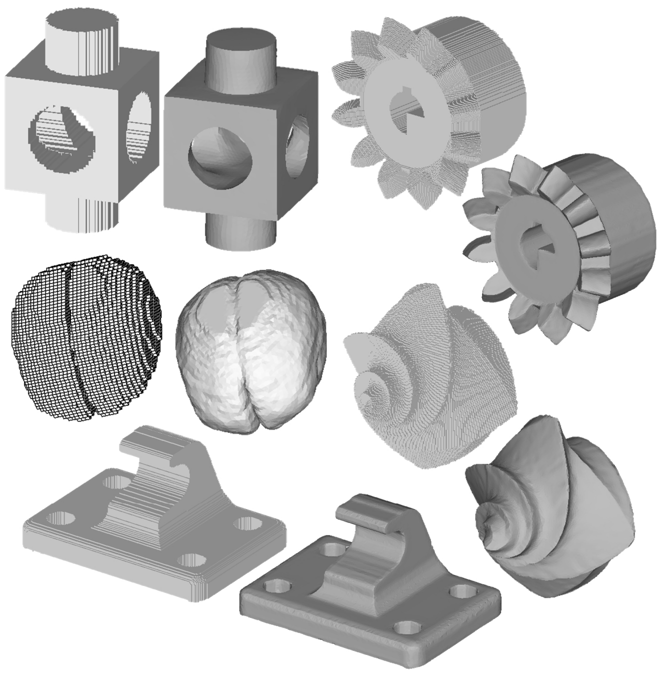


Figure 9: Results of our approach on various models.

vectors by minimizing the following least square error (LSE) defined on the faces

$$E(v_i) = \sum_{j \in star(i)} \sum_{f \in F_{ij}} \left(\bar{n}_f \cdot (v_i - v_j) \right)^2 , \tag{16}$$

where $star(i)$ denotes the 1-ring neighboring vertices of v_i, and F_{ij} represents the two faces that are adjacent to the edge $v_i v_j$. $E(v_i)$ can be iteratively minimized by the update given in [59]

$$v_i' \leftarrow v_i + \tau \sum_{j \in star(i)} \sum_{f \in F_{ij}} \bar{n}_f \bar{n}_f^T (v_j - v_i) . \tag{17}$$

In implementation, we choose the parameters: $\sigma_c = 2\bar{L}$ and $\sigma_s = \sigma_p = 0.25$ to process normal vectors. To speed up the search of $N(f)$, we only search triangles locally starting from f by using the connectivity on the given mesh surface. Once all the newly added triangles are with the distance to c_f greater than $2\sigma_c$, the search stops. The positions of vertices in the potential sharp region and the rest of surface (smooth regions) are updated separately. The positions of vertices in smooth regions are updated by Eq.(17) in 30 runs with each run using $\tau = 0.1$ as the damping factor of update. The sharp region vertices are re-positioned also in 30 runs with $\tau = 0.1$. This separation prevents the shrink effect which is in general shown on diffusion procedures. When processing the smooth region, the sharp regions act as a keel to prevent surface shrinkage. During the sharpening, smooth regions act as keels to prevent shrinkage. As shown in Figure 8, if all vertices are updated together (see the circled region in Figure 8(c)), the surface is slightly blurred – so that the cylinder is sloped. By separating the two steps, the sloping effect is reduced (see Figure 8(d)).

6. RESULTS AND DISCUSSION

We applied the approach presented in this paper to several models, and successfully generated two-manifold mesh surface with sharp edges (Figure 9). The input binary volume models are also listed in Figure 9. We implemented the approach presented in this paper on a PC with PIV 3.0GHz CPU and 1GB RAM. The processing of all examples shown in this paper can be finished in tens to a few hundred seconds. Computing time of the topology-extraction step is comparable to the MC algorithms – can be completed very fast. The smooth shape reconstruction step usually takes about few seconds. The most time-consuming step is the $L^{2,1}$ planar segmentation, which takes about 60%-70% of the processing time.

An interesting test is illustrated in Figure 10, where the meshes are extracted for the same model but with different resolution of binary volume inputs. It can be found that our surface reconstruction algorithm converges while increasing the sampling rate (i.e., generates more and more accurate surfaces with increasing the resolution of input). When the sampling rate is increased, the results after smooth reconstruction move closer to the results after sharp edges recovery.

The last test is conducted to illustrate the functionality of our novel bilateral filtering method for sharp edges extraction. As Eq.(10), with $W_p(f,q) \equiv 1$, our filter degenerates to the bilateral filter in [58]. Figure 11 compares the results from our segmentation-enhanced bilateral filter, the normal-based bilateral filter in [58], and the position-based bilateral filter of [57] on the mesh generated from the $92 \times 92 \times 92$ binary volume model. Our filter gives the best result, the normal-based bilateral filter damages the original shape, and the position-based bilateral filter accumulates edges around the sharp features so that the element shape becomes worse. Figure 12 gives similar results on the anchor plate model. However, the proposed sharpening filter does not work very well on the freeform models like the flower model in Figure 13. It is because that the VSA algorithm cannot give clear boundary on highly-curved freeform objects, which is the major drawback of our segmentation-enhanced bilateral filter.

7. CONCLUSION

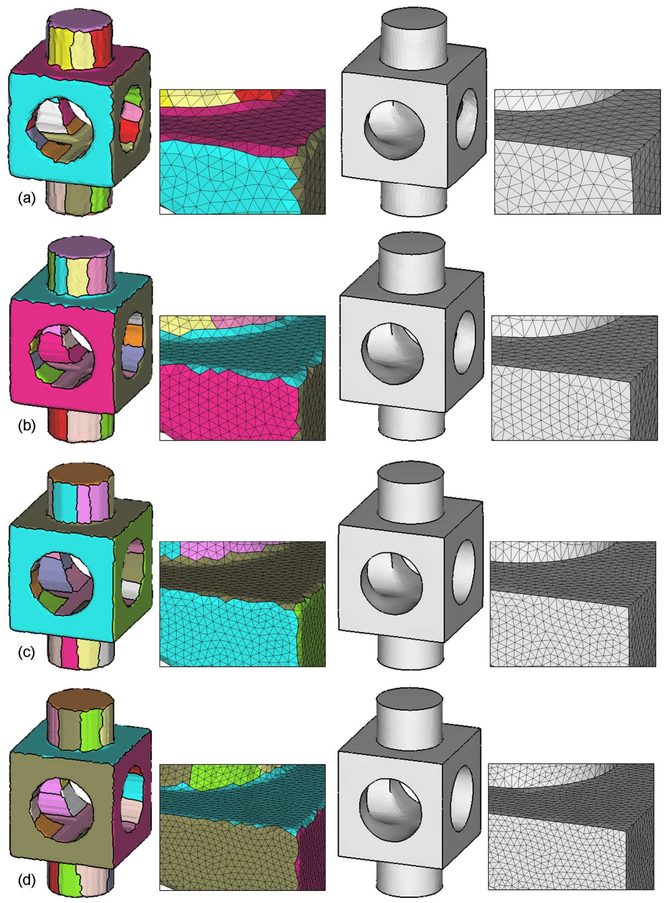


Figure 10: The output of our approach gives more and more accurate result while increasing the sampling rate on input binary models with (a) $92\times92\times92$, (b) $128\times128\times128$, (c) $160\times160\times160$ and (d) $192\times192\times192$ voxels. The left column shows the results from smooth surface reconstruction (step 2) and the right column gives the result after recovering sharp edges (step 3).

A three-step algorithm has been presented in this paper to automatically reconstruct a two-manifold mesh surface with sharp edges from a binary volume model. Smooth regions and sharp edges can be automatically recovered from raw binary volume models without scalar field or Hermite data. Comparing to other related surface recovering methods on binary volume input, our algorithm needs less heuristic coefficients – only 4 coefficients: ε , σ_c , σ_s , and σ_p are needed. In short, our technical contributions are

- an algorithm to construct the connectivity of isosurface with consistent topology to H;
- using duplex distance-fields to give error bound on reconstructed smooth surfaces;
- a segmentation-enhanced sharpening filter for recovering sharp edges.

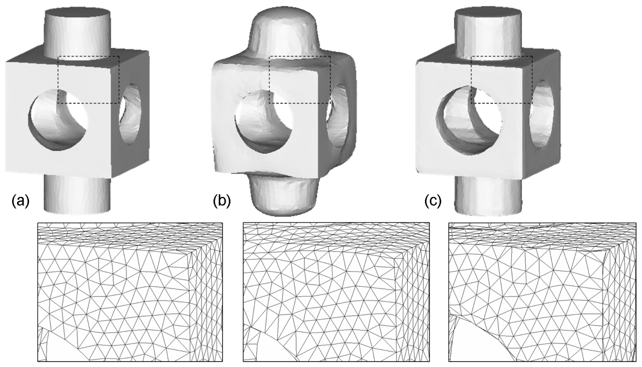


Figure 11: Results of sharpened mechanical part from (a) our segmentation-enhanced bilateral filter, (b) the normal-based bilateral filter [58], and (c) the position-based bilateral filter [57] with the same σ_c and σ_s .

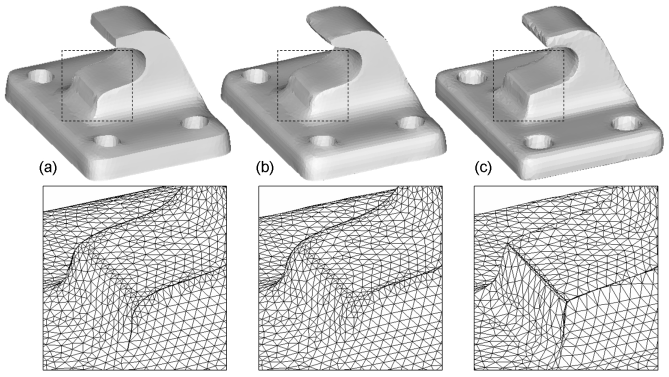


Figure 12: Results of the sharpened anchor plate from (a) our filter, (b) the normal-based bilateral filter [58], and (c) the position-based bilateral filter [57] with the same σ_c and σ_s .

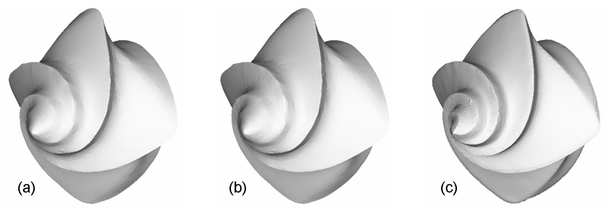


Figure 13: Results of the sharpened flower model from (a) our filter, (b) the normal-based bilateral filter [58], and (c) the position-based bilateral filter [57] with the same σ_c and σ_s .

Our future research will focus on how to construct faces with less memory in the topology construction step and how to extract more semantic design features from the mesh surface generated by this approach.

ACKNOWLEDGMENTS

The authors would like to acknowledge the valuable comments by the reviewers and the support by CUHK Direct Research Grant CUHK/2050374 and Hong Kong RGC/CERG Grant CUHK/416307.

REFERENCES

[1] Wang M.Y. and Wang X, 2004, "A level-set based variational method for design and optimization of heterogeneous objects," Computer-Aided Design, vol.37, pp.321-337.

[2] Habbal A., Petersson J., and Thellner M., 2004, "Multidisciplinary topology optimization solved by a Nash game," International Journal for Numerical Methods in Engineering, vol.61, pp.949-963.

[3] Cappello F. and Mancuso A., 2003, "A genetic algorithm for combined topology and shape optimizations," Computer-Aided Design, vol.35, pp.761-769.

[4] Lorensen W. and Cline H., 1987, "Marching cubes: a high resolution 3D surface construction algorithm," Computer Graphics, vol.21, no.4, pp.163-169.

[5] Duerst M. J., 1988, "Letters: Additional reference to marching cubes," Computer Graphics, vol.22, no.2, pp.72-73.

[6] Ning P. and Bloomenthal J., 1993, "An evaluation of implicit surface tillers," IEEE Computer Graphics and Applications, vol.13, no.6, pp.33-41.

[7] Andujar C., Brunet P., Chica A., Navazo I., Rossignac J., and Vinacua A., 2004, "Optimizing the topological and combinatorial complexity of isosurfaces," Computer-Aided Design, vol.37, no.8, pp.847-857.

[8] Lachaud J.-O., 1996, "Topologically defined iso-surfaces," In Proc. 6th Discrete Geometry for Computer Imagery (DGCI'96), Lyon, France, pp.245-256. Springer-Verlag, Berlin.

[9] Montani C., Scateni R., and Scopigno R., 1994, "A modified look-up table for implicit disambiguation of marching cubes," The Visual Computer, vol.10, no.6, pp.353-355.

[10] Bloomenthal J., 1994, "An implicit surface polygonizer," In Paul S. Heckbert, editor, Graphics Gems IV, pp.324-349.

[11] Zahlten C., 1992, "Piecewise linear approximation of isovalued surfaces," In F. H. Post and A. J. S. Hin, editors, Advances in Scientific Visualization, pp.105-118. Springer-Verlag.

[12] Nielson G.M., Foley T.A., Hamann B., and Lane D., 1991, "Visualizing and modeling scattered multivariate data," IEEE Computer Graphics and Applications, vol.11, no.3, pp.47-55.

[13] Wallin A., 1991, "Constructing isosurfaces from CT data," IEEE Computer Graphics and Applications, vol.11, no.6, pp.28-33.

[14] Nielson G.M. and Hamann B., 1991, "The asymptotic decider: Resolving the ambiguity in marching cubes," In Proc. of IEEE Visualization 91, pp.83-91.

[15] Wilhelms J. and Van Gelder A., 1990, "Topological considerations in isosurface generation," Computer Graphics, vol.24, no.5, pp.79-86.

[16] Wyvill G., McPheeters C., and Wyvill B., 1986, "Data structures for soft objects," The Visual Computer, vol.2, no.4, pp.227-234.

[17] Lewiner T., Lopes H., Vieira A.W., and Tavares G., 2003, "Efficient Implementation of Marching Cubes' Cases with Topological Guarantees," Journal of Graphics Tools, vol.8, no.2, pp.1-15.

[18] Weber G.H., Scheuermann G., Hagen H., and Hamann B., 2002, "Exploring scalar fields using critical isovalues," In Proc. of IEEE Visualization 2002, pp. 171-178.

[19] Stander B.T. and Hart J.C., 1997, "Guaranteeing the topology of an implicit surface polygonization for interactive modeling," In Proceedings of SIGGRAPH 97, pp.279-286.

[20] Nielson G., 2003, "On marching cubes," IEEE Trans. on Visualization and Computer Graphics, vol.9, no.3, pp.283-297.

[21] Cignoni P., Ganovelli F., Montani C., and Scopigno R., 2000, "Reconstruction of topologically correct and adaptive trilinear isosurfaces," Computers & Graphics, vol.24, no.3, pp.399-418.

[22] Schroeder W., Zarge J., and Lorensen W., 1992, "Decimation of triangle meshes. Computer Graphics," vol.26, no.2, pp.65-70.

[23] Hoppe H., DeRose T., Duchamp T., McDonald J., and Stuetzle W., 1993, "Mesh optimization," In Proc. of SIGGRAPH 1993, pp.19-26.

[24] Kalvin A. and Taylor R., 1996, "Superfaces: polygonal mesh simplification with bounded error," IEEE Computer Graphics and Applications, vol.16, no.3, pp.64-77.

[25] Turk G., 1992, "Re-tiling polygonal surfaces. Computer Graphics," vol.26, no.2, pp.55-64.

[26] Crossno P. and Angel E., 1997, "Isosurface extraction using particle systems," In Proc. of IEEE Visualization 97, pp.495-498.

[27] Gibson S., 1998, "Using distance maps for smooth surface representation in sampled volumes," In Proc. of 1998 IEEE Volume Visualization Symposium, pp.23-30.

[28] Perry R.N. and Frisken S.F., 2001, "Kizamu: a system for sculpting digital characters," In Proc. of ACM SIGGRAPH 2001, pp.47-56.

[29] Kobbelt L.P., Botsch M., Schwanecke U., and Seidel H.-P., 2001, "Feature sensitive surface extraction from volume data," In Proc. of SIGGRAPH 2001, pp.57-66.

[30] Ju T., Losasso F., Schaefer S., and Warren J., 2002, "Dual contouring of Hermite data," ACM Trans. on Graphics, vol.21, no.3, pp.339-346.

[31] Ohtake Y. and Belyaev A., 2003, "Dual-prime mesh optimization for polygo-nized implicit surfaces with sharp features," In Proc. of ACM Solid Modeling Symposium 2003, pp. 171-178.

[32] Ohtake Y., Belyaev A., and Pasko A., 2003, "Dynamic mesh optimization for polygonized implicit surfaces with sharp features," The Visual Computer, vol.19, pp.115-126.

[33] Attene M., Falcidino B., Spagnuolo M., and Rossignac J., 2005, "Sharpen&Bend: Recovering curved edges in triangle meshes produced by feature-insensitive sampling," IEEE Trans. on Visualization and Computer Graphics, vol.11, no.2, pp.181-192.

[34] van Overveld K. and Wyvill B., 2004, "Shrinkwrap: An efficient adap-tive algorithm for triangulating an iso-surface," The Visual Computer, vol.20, no.6, pp.362-379.

[35] Markosian L., Cohen J.M., Crulli T., and Hughes J., 1999, "Skin: a constructive approach to modeling free-form shapes," Proceedings of SIGGRAPH 99, pp.393-400.

[36] Wang C.C.L., 2006, "Bilateral recovering of sharp edges on feature-insensitive sampled meshes," IEEE Trans. on Visualization and Computer Graphics, vol.12, no.4, pp.629-639.

[37] Wang C.C.L., 2006, "Incremental reconstruction of sharp edges on mesh surfaces," Computer-Aided Design, vol.38, no.6, pp.689-702.

[38] Taubin G., 1995, "A signal processing approach to fair surface design," In Proceedings of SIGGRAPH 95, pp.351-358.

[39] Desbrun M., Meyer M., Schröder P., and Barr A.H., 1999, "Implicit fairing of irregular meshes using diffusion and curvature flow," Proceedings of SIGGRAPH 99, pp.317-324.

[40] Bajaj C.L. and Xu G., 2003, "Anisotropic diffusion of surfaces and functions on surfaces," ACM Trans. on Graphics, vol.22, no.1, pp.4-32.

[41] Hildebrandt K. and Polthier K., 2004, "Anisotropic filtering of non-linear surface features", Computer Graphics Forum, vol.23, no.3.

[42] Meyer M., Desbrun M., Schroder P., and Barr A.H., 2002, "Discrete differential-geometry operators for triangulated 2-manifolds," Proceeding of Visualization and Mathematics.

[43] Jones M.W. and Satherley R.A., 2001, "Shape representation using space filled sub-voxel distance fields," Proceedings of International Conference on Shape Modeling and Applications 2001, pp.316-325.

[44] Surazhsky V., Alliez P., and Gotsman C., 2003, "Isotropic remeshing of surfaces: a local parameterization approach," In Proc. of 12th International Meshing Roundtable.

[45] Surazhsky V. and Gotsman C., 2003, "Explicit surface remeshing," In Proc. of the Eurographics/ACM SIGGRAPH Symposium on Geometry Processing, pp.20-30.

[46] Vorsatz J., Rössl C., and Seidel H.-P., 2003, "Dynamic remeshing and applications," In Proc. of Solid Modeling and Applications, pp.167-175.

[47] Botsch M. and Kobbelt L., 2004, "A remeshing approach to multiresolution modeling," In Proc. of Eurographics /ACM SIGGRAPH Symposium on Geometry Processing, pp/185-192.

[48] Hoppe H., DeRose T., Duchamp T., McDonald J., Stuetzle W., 1993, "Mesh optimization," Extended TR UW CSE 1993-01-01, http://research.microsoft.com/~hoppe/.

[49] Cohen-Steiner D., Alliez P., and Desbrun M., 2004, "Variational shape approximation," ACM Trans. Graphics, SIGGRAPH 2004, vol.23, pp. 905-914.

[50] Katz S. and Tal A., 2003, "Hierarchical mesh decomposition using fuzzy clustering and cuts," ACM Trans. Graphics, Proc. of SIGGRAPH 2003, vol.22, no. 3, pp. 954-961.

[51] Cormen T.H., Leiserson C.E., Rivest R.L., and Stein C., *Introduction to Algorithms* (2nd ed.), MIT Press, 2001.

[52] Smith S.M. and Brady J.M., 1997, "SUSAN – a new approach to low level image processing", International Journal of Computer Vision, vol.23, pp.45-78.

[53] Tomasi C. and Manduchi R., 1998, "Bilateral filtering for gray and color images," Proc. of IEEE International Conference on Computer Vision, pp.836-846.

[54] Barash D., 2002, "A fundamental relationship between bilateral filtering, adaptive smoothing and the nonlinear diffusion equation," IEEE Trans. on Pattern Analysis and Machine Intelligence, vol.24, no.6.

[55] Black M.J., Sapiro G., Marimont D.H., and Heeger D., 1998, "Robust anisotropic diffusion," IEEE Trans. on Image Processing, vol.7, no.3, pp.421-432.

[56] Fleishman S., Drori I., Cohen-Or D., 2002, "Bilateral mesh denoising," ACM Trans. on Graphics, vol.22, no.3, pp. 950-953.

[57] Jones T.R., Durand F., and Desbrun M., 2003, "Non-iterative, feature-preserving mesh smoothing," ACM Trans. on Graphics, vol.22, no.3, pp. 943-949.

[58] Lee L.-W. and Wang W.-P., 2005, "Feature-preserving mesh denoising via bilateral normal filtering," Proc. of 9th International Conference on Computer Aided Design and Computer Graphics, pp.275-280.

[59] Taubin G., 2001, "Linear anisotropic mesh filtering," Technical Report of IBM Research, TR-RC2213.

[60] Wang C.C.L., 2006, "Direct extraction of surface meshes from implicitly represented heterogeneous volumes," Computer-Aided Design, vol.39, no.1, pp.35-50.

[61] Lloyd S., 1982, "Least square quantization in PCM," IEEE Trans. on Inform. Theory, vol.28, pp.129-137.

[62] Ju T., Schaefer S., and Warren J., 2003, "Convex contouring of volumetric data," The Visual Computer, vol.19, pp.513-525.

[63] Azernikov S. and Fischer A., 2006, "A new volume warping method for surface reconstruction," Journal of Computing and Information Science in Engineering, ASME Transactions, vol.6, no.4, pp.355-363.