Constructing Smooth Non-Manifold Meshes of Multi-Labeled Volumetric Datasets

Bernhard Reitinger, Alexander Bornik, Reinhard Beichel

Institute for Computer Graphics and Vision
Graz University of Technology
Inffeldgasse 16/II
A–8010 Graz, Austria
contact: breiting@icg.tu-graz.ac.at

ABSTRACT

This paper presents a method for constructing consistent non-manifold meshes of multi-labeled volumetric datasets. This approach is different to traditional surface reconstruction algorithms which often only support extracting 2-manifold surfaces based on a binary voxel classification. However, in some – especially medical – applications, multi-labeled datasets, where up to eight differently labeled voxels can be adjacent, are subject to visualization resulting in non-manifold meshes. In addition to an efficient surface reconstruction method, a constrained geometric filter is developed which can be applied to these non-manifold meshes without producing ridges at mesh junctions.

Keywords

surface reconstruction, mesh generation, multi-labeled volume, constrained smoothing

1 INTRODUCTION

Surface reconstruction for volumetric datasets is an important method for exploring important features especially in the medical field. By segmenting stacks of 2D gray-valued images (e.g. CT, MR images), 2-manifold meshes in form of iso-surfaces can be extracted and visualized. The traditional method is the *Marching Cubes* (MC) algorithm proposed by Lorensen and Cline in [Loren87] or some of its variations like [Lewin03, Labsi02, Lopes03] generating triangular models. All of these methods are concerned about 2-manifold meshes (homeomorphic to a sphere) which only allow mesh interfaces between two different materials (one below and one above a certain threshold).

Conference proceedings ISBN 80-903100-7-9
WSCG'2005, January 31–February 4, 2005
Plzen, Czech Republic.

However, in some applications multiple coherent surfaces should be reconstructed consistently based on a non-binary classification resulted from a previous image segmentation or labeling like – for example – a full segmentation of the Visible Human Project [Acker98] where each organ is tagged with a certain label and connected to other organs. All interfaces between adjacent organs must be extracted consistently. A naive solution for this problem would be to apply the MC iteratively on each labeled region while masking out all other regions. Although all interfaces would be extracted, a lot of inconsistencies are expected even between two adjacent regions because two surfaces of one interface will be generated which might not fit together. This gets even worse if more than two materials are meeting at one *cell* (a *cell* is defined by its eight adjacent voxels). Different to the traditional *MC* which generates a 2-manifold, our algorithm inevitably produces non-manifolds if more than two materials meet at one *cell*.

This paper proposes an efficient method for extracting such non-manifold surfaces based on labeled volumetric datasets in a consistent way. The resulting mesh can either be used for visualization or as input for a volumetric mesh generator. Our algorithm neither generates holes nor cracks. Depending on the resolution of the input dataset, staircase artifacts can occur which are smoothed

by applying customized topology-invariant filters enhancing the overall mesh quality. For this reason, we extended two existing filters. After surface relaxation, triangle simplification may be applied using the quadric error metric [Garla97].

This work was motivated by a medical project which is concerned about virtual liver surgery planning [Borni03]. The presented algorithm can be used for two different tasks within this project. At first, a visualization of liver segments by their interface boundaries is provided which allows surgeons to examine the location and size of each single liver segment (see Figure 10). Secondly, the output mesh can be used as input for a consequent volumetric mesh generator by applying the algorithm proposed in [Si02]. An example for one liver dataset is shown in Figure 1. For both cases, the liver is labeled a priori using the segment classification algorithm proposed in [Beich04] where each liver voxel is tagged with a certain material value indicating one of eight different liver segments (see Figure 1(a)). Due to the liver's anatomy, more than two different segments can be adjacent. Therefore, the presented reconstruction method is necessary in order to handle multiple labels within one *cell*. The generated mesh can be classified in two different kinds of surfaces; one domain boundary surface covering the object itself and multiple interfaces which build the interior structure (surfaces) of the object.

(a) (b) (c)

Figure 1: Visualization of a liver dataset containing different liver segments. (a) A slice of the labeled input volume, (b) non-manifold extracted using the proposed algorithm, (c) a volumetric mesh based on the mesh in (b) using the TetGen library [Si02].

The rest of this paper is organized as follows: Section 2 discusses related work for existing surface reconstruction methods. Section 3 presents our method for multi-labeled surface generation. Section 4 outlines an extension to existing geometric filters which are applied on the generated surface. Section 5 presents a simplification method for decreasing the number of generated triangles. Results and screenshots are shown in Section 6 and a conclusion closes this report.

2 RELATED WORK

The traditional method for extracting iso-surfaces based on a certain threshold is the *Marching Cubes* algorithm [Loren87] which distinguishes between two different domains (above and below a certain iso-value). Given a binary classification only 2^8 (256) different configurations can occur in a *cell* which can be implemented using look-up tables. Recently, Lewiner et al. presented an extension avoiding topological errors which can occur due to uncertainties [Lewin03].

Bloomenthal and Ferguson described in [Bloom95] one of the first approaches for generating surfaces from non-binary classifications which rely on implicit surface modelling and computational solid geometry. By subdividing cubic cells into tetrahedra, a triangulation is constructed algorithmically. This approach generates a large amount of triangles which can also be degenerate and not well-shaped.

Hege et al. presented a different approach for extracting non-manifold surfaces based on a non-binary classification [Hege97]. By using probabilities assigned to each voxel, cells are subdivided producing a lot of intermediate triangles. Therefore, a post-processing patch generation must be initiated in order to reduce the large number of faces. In their approach, a look-up table was implemented supporting up to three different materials meeting at one *cell*.

Another algorithm for generating surfaces based on a non-binary classification was presented by Wu and Sullivan [Wu03]. In their work, an extension for the traditional *Marching Cubes* was developed supporting 2D and 3D datasets.

Different to these related algorithms, we present an efficient yet simple approach which supports up to eight different materials meeting at one *cell*. In the following we will describe our method in detail.

3 MESH GENERATION

The input for our algorithm is a rectilinear labeled volumetric dataset where a distinct label (material) is assigned to each voxel $\mathbf{V}$ at position (x, y, z). A zero label indicates background (outside of the domain) and all other labels unequal to zero assign material. For a binary classification, 2^8 (256) different cases can occur at one *cell*. Therefore, look-up tables exist for the *Marching Cubes* algorithm to gain performance. If the dataset con-

sists of multiple different labels (non-binary), 8^8 (16777216) cases are possible within one *cell* and a look-up table to cover all these configurations is not feasible.

3.1 The Idea in 2D

The main idea of our algorithm is based on a cell subdivision strategy and will first be explained for the 2D case. Each non-homogeneous *cell* having two or more different materials is subdivided by inserting a *cell* mid-point. Additionally, segment mid-points are generated if the labels of two adjacent voxels (nodes) are different. For each generated segment mid-point a new segment (line) to the *cell* mid-point is generated. Figure 2 shows the four possible configurations for the 2D case. By permutating these cases, all 4^4 can be captured. If using the traditional *Marching Cubes* scheme for a *cell* with three different labels, a triangular void would be generated (see Figure 3). This generated void is invalid and cannot be assigned to a material.

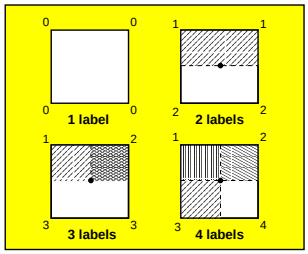
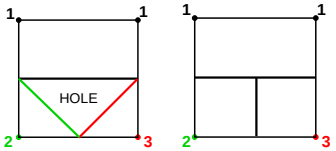


Figure 2: Four possible configurations in 2D.



(a) (b)

Figure 3: Three different materials meeting at one *cell* (quad). (a) Traditional *Marching Cubes* generates a "hole", (b) correct partitioning with three different materials.

As we can observe in Figure 2, our algorithm does produce valid T-junctions at *cell* mid-points and is therefore a conforming triangulation.

Considering a 2D *cell* with two different materials where two equal materials are opposite, the topology is preserved due to the insertion of a *cell* mid-point. However, this configuration produces singularities which must be considered for filtering in order to prevent spikes (see later).

3.2 3D Algorithm

As shown in the previous section, the 2D algorithm inserts additional *cell* mid-points in order to reconstruct interfaces correctly. The same strategy is used for the 3D case. Before going into more details of the algorithm we need to define some constraints. At first, our generated mesh is a non-manifold represented by a simplicial complex of 2-simplices. By considering this constraint, we guarantee a conforming mesh without invalid triangle intersections.

Having a rectilinear volumetric grid, a *cell* $\mathbf{C}$ at location (x, y, z) is defined by
$\mathbf{C}_{x,y,z} = [\mathbf{V}_{x,y,z}; \mathbf{V}_{x+1,y,z}; \mathbf{V}_{x+1,y+1,z}; \mathbf{V}_{x,y+1,z}; \mathbf{V}_{x,y,z+1}; \mathbf{V}_{x+1,y,z+1}; \mathbf{V}_{x+1,y+1,z+1}; \mathbf{V}_{x,y+1,z+1}]$.

The generation algorithm is initiated by processing each single *cell* $\mathbf{C}_{x,y,z}$ of the rectilinear volume (see Algorithm 1). If a *cell* in not homogeneous (having more than one material), it is enqueued in a list Q.

Algorithm 1 Pre-processing

Input: rectilinear labeled 3D volume V
Ensure: $\mathbf{V}_{x,y,z} = \{l : l \in \mathbf{N}\}$
  for all $\mathbf{C}_{x,y,z}$ **do**
    if $\mathbf{C}_{x,y,z}$ is !homogeneous **then**
      Q.add($\mathbf{C}_{x,y,z}$)
    end if
  end for
Output: Q

In the second step, only enqueued *cells* are processed. Depending on the homogeneity of the input, the processing list can be very small. The second sweep is also called *simplex generation* because for a given node configuration, faces (triangles) are generated using a small look-up table. All generated faces and vertices are stored in a compact data structure where each vertex is unique and duplicates are avoided. This is necessary to guarantee a correct post-processing (i.e. smoothing or simplification). Algorithm 2 describes the complete second sweep and its details are explained in the following sections. The output of the *simplex generation* algorithm is an indexed face set I storing the triangulation of the whole domain and its according vertex list VL.

3.2.1 Face Generation

Our face generation strategy is very efficient. In any case of a non-homogeneous *cell*, a *cell* mid-point is generated which subdivides this *cell* into 8 sub-cells. Additionally, face mid-points are gen-

Algorithm 2 Simplex generation

Input: Q
  for all $\mathbf{C}_{x,y,z}$ in Q **do**
    subdivide $\mathbf{C}_{x,y,z}$ by creating a *cell* mid-point
    if $\mathbf{C}_{x,y,z}$ has one zero label **then**
      $\mathbf{C}_{x,y,z}$=EXTERNAL_NODE
    else
      $\mathbf{C}_{x,y,z}$=INTERNAL_NODE
    end if
    generate vertices and faces using a LUT
    classify cell nodes of $\mathbf{C}_{x,y,z}$
    for all generated faces f_i **do**
      I.add(f_i)
      VL.add(vertices of f_i)
    end for
  end for
Output: I, VL

erated if a face of $\mathbf{C}_{x,y,z}$ has non-homogeneous nodes. All interfaces between two sub-cells are investigated if two adjacent sub-cells have different materials. If a difference is found, two triangles are spanned between these two sub-cells covering the interface. For efficiency, we use a look-up (LUT) table to get the adjacencies between sub-cells. For each *cell* we need 12 look-ups to generate adjacent faces within one *cell*. Figure 4 shows some face generation examples for two, three or more materials meeting at one *cell*.

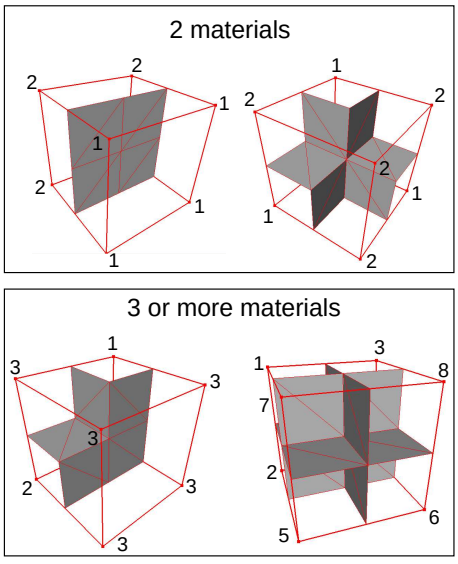


Figure 4: Examples of face generations for the 3D case

By using this generation method, adjacent *cells* are connected correctly. As the input has no quantity information but only labels, we cannot apply any interpolation scheme ad hoc. However, as the resulting mesh should be visual appealing, we need to apply a filter for smoothing the surface to reduce the staircase characteristics produced by our algorithm. However, in order to guarantee a correct smoothing of these non-manifold meshes, we have to classify the nodes in a *cell* due to a certain configuration.

3.2.2 Node Classification

Each node (or vertex in the manner of surfaces) must be classified according to its location. This guarantees a correct smoothing of non-manifold meshes and avoids ridges between the domain boundary and interface boundaries. Figure 5 shows a *cell* with its possible vertex locations. At first, the possible 27 vertices are named by their locations in the *cell*. *Corner-points* are the original voxels of the dataset each of them storing a certain label. Face mid-points are called *2D center-points* and are generated if a *cell's* face in not homogeneous. *Border-points* divide a segment between two adjacent *corner-points*. And finally, the *cell* mid-point is classified as *3D center-point*. Additionally, we define certain types of nodes as the following:

Definition 1: ***Domain Node (D)***

A node (vertex) is classified as *domain node*, if the node is part of the enclosing domain surface if one exists.

Definition 2: ***Interface Node (I)***

A node (vertex) is classified as *interface node*, if the node is part of an interior surface which separates two different interior materials.

Definition 3: ***Junction Node (J)***

A node (vertex) is classified as *junction node*, if the node is part of an interior surface and separates more than two different interior materials.

Definition 4: ***Domain Junction Node (DJ)***

A node (vertex) is classified as *domain junction node*, if the node is part of an exterior surface and separates at least two interior materials but also one exterior material.

Table 1 shows the mapping between nodes and possible types. A *corner-point* will not be classified at all because the surface will never pass

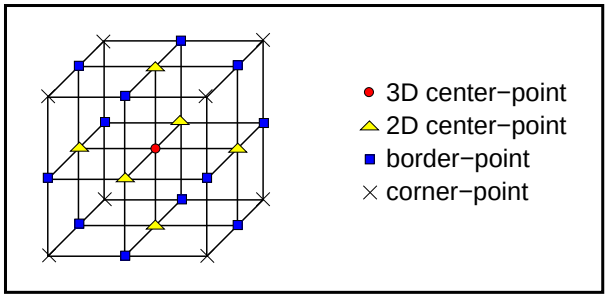


Figure 5: Classification of nodes in a *cell*.

	D	I	J	DJ
3D center-point	x	x	x	x
2D center-point	x	x	x	x
border-point	x	x		
corner-point				

Table 1: Mapping of nodes to node types.

through it. A *border-point* is a *domain node* if one of its adjacent *corner-points* has a zero material, else it will be assigned as *interface node*. A *2D center-point* is assigned as *junction node* if its corresponding face has no zero material and material count (number of different materials per face) >= 3. If a face has at least one zero material, it is assigned as *domain junction node*. If the material count for one face is < 3 and the face has no zero material it is assigned as *interface node*, else *domain node*. *3D center-points* are classified according to a *cell's* classification. If $\mathbf{C}_{x,y,z}$ is assigned EXTERNAL_NODE and material count is 2, then we classify it as *domain node*, else *domain junction node*. If $\mathbf{C}_{x,y,z}$ is an INTERNAL_NODE and material count is 2, the 3D *center-point* is assigned as *interface node*, else, *junction node*.

If this type mapping is applied, ridges are avoided and self-intersections are prevented if smoothing the surface. Figure 6 shows a comparison of two examples one without applying this classification scheme and one with these considerations.

4 SURFACE FILTERING

Surface filtering is necessary to reduce the staircase artifacts generated by our face generation method. We use surface filters which only affect geometry and do not alter topology. Beside smoothing, geometric filters are also used for improving the overall quality of the mesh. Different geometric filters exist in literature which can be applied for smoothing. The most simple but effective smoothing filter is the *Laplacian* filter [Field88]. Each vertex at position x_i is smoothed iteratively by using the following formula:

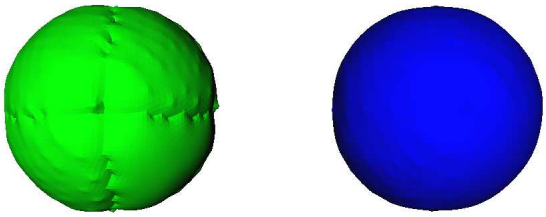


Figure 6: If node classification is not assigned, ridges occur (see left image). If the classification is applied, no ridges are generated.

$$x_{i+1} = x_i + \lambda \Delta_i \tag{1}$$

where Δ_i is defined as:

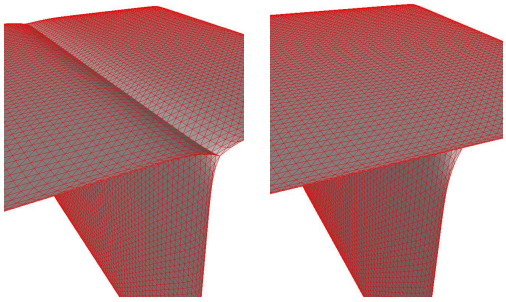
$$\Delta_i = \sum_{j=1}^{N} w_{ij}(x_j - x_i) \tag{2}$$

where w_{ij} specifies the weight between x_i and its neighboring x_j and $\sum w_{ij} = 1$. A good choice for w_{ij} is $\frac{1}{N}$, where N is the number of adjacent vertices. The scale factor λ $(0 < \lambda < 1)$ influences the degree of smoothness and is defined equally for all vertices. Intrinsically, this filter also improves the overall mesh quality. The big advantage of this filter is its simplicity, however, it produces shrinkage if applied many times.

Therefore, an extension to the *Laplacian* filter was presented by Taubin [Taubi95] which avoids shrinkage. The main idea is to apply two consecutive *Laplacian* steps: at first, using a positive scale factor λ and then using a negative scale factor μ, greater in magnitude than λ $(0 < \lambda < -\mu)$.

Applying the standard filter for our produced mesh, ridges on the domain boundary can occur as shown in Figure 7(a). This is because vertices on the domain boundary which are adjacent to interface nodes are attracted by these nodes and produce valleys. Therefore, we have performed the node classification which will be considered now for the filter. The $(x_j - x_i)$ expression in Equation 2 will only be calculated under certain constraints. If the vertex at location x_i is assigned as *domain node* or *interface node*, all adjacent vertices at x_j are used for calculating Δ_i. However, if a vertex at location x_i is a *domain junction node*,

only vertices at x_j with classification *domain junction node* and *domain node* are considered. Similarly, if vertices at x_i is a *junction node*, only vertices at x_j of type *junction node* or *domain junction node* are used. The result can be seen in Figure 7(b).



(a) (b)

Figure 7: By applying the constrained *Laplacian* (Taubin) filter, ridges are avoided successfully as shown in this figure.

We observed that singularities can occur if voxels are only 26-connected for 3D datasets. This gets a problem if the surface is filtered. Two different material regions are only connected through one vertex and filtering would generate unwanted spikes. In the current implementation we assign a freeze label to these vertices which means that they are not moved during smoothing and avoids therefore these spikes.

5 MESH SIMPLIFICATION

In order to reduce the triangle count a simplification algorithm is applied which is invariant to the surface's topology. We are using the quadric error metric which was introduced by Garland and Heckbert [Garla97]. The algorithm works with an iterative contraction of vertex pairs to simplify models and maintains surface error approximations using quadric matrices. The advantage of this algorithm is that it supports non-manifold meshes. Therefore, it can be applied to our data structure without any modifications.

6 RESULTS

After explaining all different components of our mesh generation method, some results are presented. The test input datasets are either generated by a prior segmentation and labeling of a CT scan, or artificially generated by using the hardware-accelerated voxelization presented in [Reiti03]. Table 2 shows a summary of the input datasets with different resolution (size). The **#vtx** and **#f** indicate the generated vertices and triangles after the *simplex generation*. If the input resolution is high, Algorithm 1 is the dominant factor in the measured time. If a lot of non-homogeneous *cells* are found, Algorithm 2 takes more time. We have measured, that – on average – Algorithm 1 takes 83% and Algorithm 2 17% of time for the tested datasets.

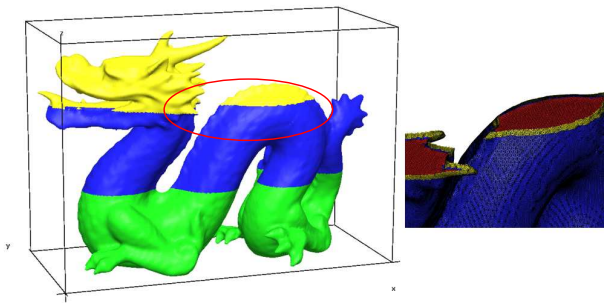


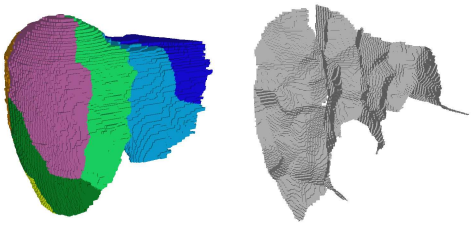
Figure 8: Example of the dragon model which was generated using our method using the *Laplaican* filter with $\lambda = 0.8$ and 30 iterations. The right image shows a zoom-in where the boundary interface can be seen clearly.

	size	**#vtx**	**#f**	**time**
Dragon	256^3	453921	910400	25.7
Horse	128^3	63147	127960	2.6
Liver	$256 \times 256 \times 174$	632272	1272088	18.5
Lung	$128 \times 128 \times 148$	331934	668624	6.1

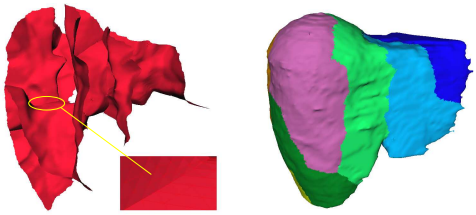
Table 2: Table showing results of different datasets. The time is measured in seconds.

Figure 8 shows the dragon model where the dataset is divided into multiple layers simulating interface boundaries. The right image shows a zoom-in showing three bounded materials (yellow, blue, red). Similarly, we also sub-divided the horse dataset into different labels. Figure 9(a) shows the raw output before smoothing, and Figure 9(b) displays the smoothed result using the constrained *Laplacian* filter.

We have also applied our algorithm on medical datasets which are generated using a segmentation and classification as described in [Beich04]. Figure 10 shows a trajectory of different smoothing levels performed on the liver dataset, starting from no smoothing up to 30 iterations. Similar to

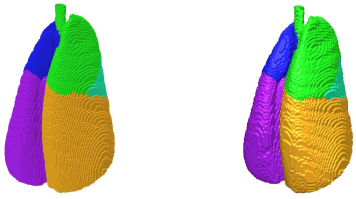


(a) raw (b) raw

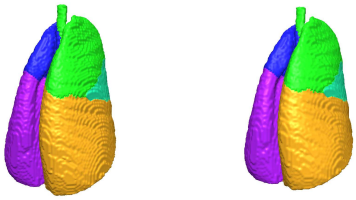


(c) 30 iterations (d) 30 iterations

Figure 10: Construction of liver segment boundaries. Left image visualizes the liver with domain boundaries without smoothing. Image (b) shows the interior structure. Images (c) and (d) display a smoothed liver with $\lambda = 0.88$ and 30 iterations. In (c) the ridge-free interface between multi-material regions can be seen.

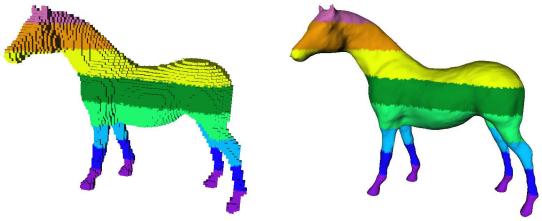


(a) raw (b) 10 iterations



(c) 30 iterations (d) 50 iterations

Figure 11: The sheep dataset with different levels of filtering. Left image is the raw output of the *simplex generation*. The consecutive images show the result of smoothing with the Taubin filter using $\lambda = 0.88$ and $\mu = -0.9$.



(a) (b)

Figure 9: Example of the horse model. Left image shows the raw output of the surface generation. The right image is the result of a *Laplacian* smoothing using $\lambda = 0.7$ and 20 iterations.

the human liver, we also applied our algorithm on a sheep lung which is shown in Figure 11. For this dataset we used the Taubin filter with different levels of smoothness.

As the resulting triangle counts are quite large for interactive usage, we applied a the simplification algorithm as explained in Section 5. Figure 12 shows two different datasets where the left model was simplified using the quadric error metric to a target face count of 70000 faces which decreases the number of triangles to about 10% of the original surface.

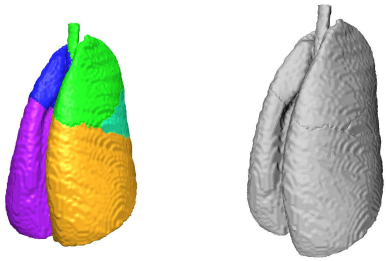
7 CONCLUSION

This paper presented an algorithm for generating non-manifold meshes based on a non-binary classification of volumetric datasets. The generation method consists of two main components, one preprocessing step and a consecutive *simplex generation*. As the raw output produces staircase artifacts constrained surface filters based on a vertex labeling scheme are applied. This prevents from generating ridges at interface boundaries and provides a high-quality mesh.

As the output surface is a conforming mesh, and therefore guarantees consistency, it can be used for a further volumetric mesh generation as presented in [Si02].

ACKNOWLEDGMENTS

This work was supported by the Austrian Science Foundation (FWF) under grant P17066-N04.



(a) (b)

Figure 12: The sheep lung before and after simplification using the quadric error metric. Target face size is 70000.

References

[Acker98] M.J. Ackerman. The visible human project. *Proc. of the IEEE*, 86(3):504–511, 1998.

[Beich04] R. Beichel, T. Pock, Ch. Janko, R. Zotter, B. Reitinger, A. Bornik, K. Palagyi, E. Sorantin, G. Werkgartner, H. Bischof, and M. Sonka. Liver segment approximation in CT data for surgical resection planning. In *In SPIE Medical Imaging '04*, San Diego, 2004. in print.

[Bloom95] J. Bloomenthal and K. Ferguson. Polygonization of non-manifold implicit surfaces. In *Proc. of SIGGRAPH '95*, pages 309–316, 1995.

[Borni03] A. Bornik, R. Beichel, B. Reitinger, G. Gotschuli, E. Sorantin, F. Leberl, and M. Sonka. Computer aided liver surgery planning: An augmented reality approach. In R.L. Galloway, editor, *Medical Imaging 2003, Proceedings of SPIE*, volume 5029. SPIE Press, May 2003.

[Field88] D.A. Field. Laplacian smoothing and delaunay triangulations. *Communications in Applied Numerical Methods*, 4:709–712, 1988.

[Garla97] M. Garland and P.S. Heckbert. Surface simplification using quadric error metrics. In *Proc. of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216, 1997.

[Hege97] H.-C. Hege, M. Seebaß, D. Stalling, and M. Zöckler. A generalized marching cubes algorithm based on non-binary classifications. Technical report, Konrad-Zuse-Zentrum (ZIB), 1997. SC 97-05.

[Labsi02] U. Labsik, K. Hormann, M. Meister, and G. Greiner. Hierarchical iso-surface extraction. *Journal of Computing and Information Science in Engineering*, 2(4):323–329, December 2002.

[Lewin03] T. Lewiner, H. Lopes, A. Wilson Vieira, and G. Tavares. Efficient implementation of marching cubes: Cases with topological guarantees. *Journal of Graphics Tools*, 8(2):1–15, 2003.

[Lopes03] A. Lopes and K. Brodlie. Improving the robustness and accuracy of the marching cubes algorithm for isosurfacing. *IEEE Transactions on Visualization and Computer Graphics*, 9(1):16–29, 2003.

[Loren87] W.E. Lorensen and H.E. Cline. Marching cubes: A high resolution 3d surface construction algorithm. *Computer Graphics*, 21(4):163–169, 1987.

[Reiti03] B. Reitinger, A. Bornik, and R. Beichel. Efficient volume measurement using voxelization. In *Proc. of the Spring Conference on Computer Graphics 2003*, pages 57–64, Budmerice, April 2003. Comenius University, Bratislava.

[Si02] H. Si. *TetGen: A 3D Delaunay Tetrahedral Mesh Generator, Version 1.2 User Manual.* Weierstrass Institute for Apply Analysis and Stochastics, 2002. No. 4.

[Taubi95] G. Taubin. Curve and surface smoothing without shrinkage. In *Proc. of the Fifth International Conference on Computer Vision*, pages 852–857. IEEE Computer Society, 1995.

[Wu03] J. Wu and J.M. Sullivan. Multiple material marching cubes algorithm. *Journal for Numerical Methods in Engineering*, 2003.