Extracting consistent and manifold interfaces from multi-valued volume data sets

Stephan Bischoff, Leif Kobbelt

Lehrstuhl für Informatik 8, RWTH Aachen, 52056 Aachen
Email: {bischoff,kobbelt}@informatik.rwth-aachen.de

Abstract. We propose an algorithm to construct a set of interfaces that separate the connected components of a multi-valued volume dataset. While each single interface is a *manifold* triangle mesh, two or more interfaces may join *consistently* along their common boundaries, i.e. there are no T-junctions or gaps. In contrast to previous work, our algorithm classifies and removes the topological ambiguities from the volume before extracting the interfaces. This not only allows for a simple and stable extraction algorithm, but also makes it possible to include user constraints.

1 Motivation and previous work

Let there be a three-dimensional voxel grid with one (tissue) label per voxel. Our goal is to extract the interfaces between all adjacent tissues as triangle meshes, i.e. the result is a set of triangle meshes each separating voxels of two different tissue types. These meshes can then be used in downstream applications, e.g. active contour models [1]. Of course, we want to guarantee certain quality properties:

1. Each triangle mesh should be a subset of a 2-manifold (in particular we allow it to have have a boundary, and the boundary can touch itself).
2. The boundaries of the triangle meshes should consistently fit together, i.e. no gaps, T-junctions or other artifacts should occur.

Unfortunately the above problem is ill-posed: Consider four voxels joining along a common edge and let there be two tissue types A and B such that the two pairs of two diagonally opposing voxels have the same tissue type. If the extraction algorithm simply extracts the interface between tissue A and B, we get a non-manifold reconstruction along the common edge. The problem becomes even worse when voxels of different tissue types join at a common vertex. This is the reason why a naive generalization of the standard Marching Cubes algorithm does not produce valid results.

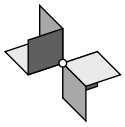
In this work we present a simple and efficient algorithm that solves the above problem. It does so by repairing the ambiguous voxels in the voxel grid before extracting the meshes. This not only leads to a very simple extraction algorithm but furthermore, the user can easily specify additional constraints, like e.g. that tissues A and B should always be separated by a layer of tissue C. Our algorithm is hence also able to incorporate a-priori knowledge and thus compensate for noise that might be present in the raw data.

Previous work There are a number of approaches that (partly) solve the above problem. Reitinger [2] uses a domain subdivision-strategy but cannot guarantee that no singularities occur. The algorithm presented by Hege [3] produces guaranteed quality output but is very complex in terms of analyzing configurations and size of lookup tables. Müller [4] presents a method based on simplices which in general produces arbitrary complexes instead of manifolds. All these methods have in common that they try to fix the extraction algorithm such that it can handle ambiguous cases. In contrast, our algorithm simply fixes the volume before extraction, so the extraction algorithm is kept simple.

2 Method

Let V be a set of voxels and let ∂V be the set of V's boundary vertices, edges and faces. We say that an edge e is *critical in* ∂V, if it is contained in four pairwise distinct faces of ∂V. Likewise a vertex v is *critical in* ∂V if it either is the end-vertex of a critical edge in ∂V or if it belongs to the singular configuration depicted to the right. Hence critical edges and vertices represent the non-manifold parts of ∂V. Now let V_i the set of voxels labelled i. Usually ∂V_i contains critical elements and so we cannot use simple Marching Cubes like algorithms to reconstruct ∂V_i. Hence, our goal is to replace the sets V_i by geometrically similar voxel sets V_i' such that $\partial V_i'$ does not contain any critical elements for all i.

critical edge



singular critical vertex

2.1 Removal of critical elements

Let us call an edge *critical* if it is critical in ∂V_i for some label i. In that case we call the smallest such i the *dominant label* at e. Critical vertices and their dominant labels are defined analogously. We can deduce solely from the voxels containing an edge or vertex whether that edge or vertex is critical or not. Thus we can efficiently collect all critical vertices and edges by iterating once over the whole volume.

The main idea of our algorithm is to remove a critical element (edge or vertex) x by *fattening* it, i.e. we replace edges by small (discrete) cylinders and vertices by small (discrete) balls. For this we subdivide the voxels that contain x and relabel the subvoxels in a neighborhood of x by x's dominant label. Figure 1a demonstrates this idea in two dimensions where a 3×3 subdivision of the pixels is sufficient to separate and resolve the complex vertices. In three dimensions we require at least a $5 \times 5 \times 5$ subdivision and we need differently sized neighborhoods for vertices and edges in order not to introduce new critical elements in the process.

We now make the above ideas mathematically precise: Each voxel that intersects a critical element x is (conceptually) subdivided into $5 \times 5 \times 5$ subvoxels and each subvoxel inherits the label of its parent. We define the k-neighborhood $N^k(x)$ of x as $N^0(x) = \{x\}$ and

$$N^k(x) = \{\text{all subvoxels that intersect } N^{k-1}(x)\}.$$

A simple two-step algorithm then removes all critical elements (Fig. 1b): First, we resolve all critical edges e by relabeling all subvoxels in $N^1(e)$ to e's dominant label. Second, we resolve all critical vertices v by relabeling all subvoxels in $N^2(v)$ to v's dominant label.

The reason for choosing a larger neighborhood for the critical vertices than for the critical edges is that edges meeting at a common vertex must be well separated in order not to introduce new critical elements (Fig. 2).

Note that in an implementation the subdivision and relabeling need not be actually carried out, but can be easily computed on the fly. Furthermore, the new voxel sets V_i' (now containing regular-sized voxels as well as subvoxels) are geometrically close to the original sets V_i, i.e. they are not more than $1/2$ voxel apart.

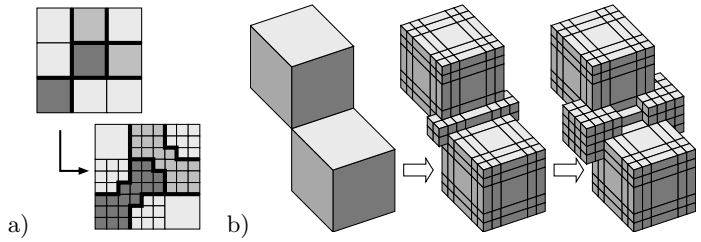


Fig. 1. Critical elements are resolved by subdividing and relabeling the adjacent voxels appropriately. In two dimensions a 3×3 subdivision is sufficient (a), in three dimensions a $5 \times 5 \times 5$ subdivision is needed (b, subvoxels are shown distorted).

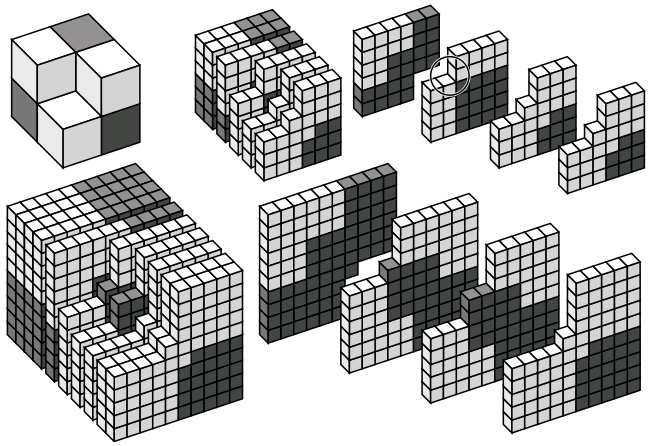


Fig. 2. The top row shows a symmetrical voxel configuration where six critical edges meet in a common vertex. Fattening this configuration on a 3^3 subdivision by 1-neighborhoods introduces new critical edges. The same configuration and its proposed fattening on a 5^3 subdivision using 2-neighborhoods for vertices and 1-neighborhoods for edges is shown in the bottom row. In this case no new critical elements are created.

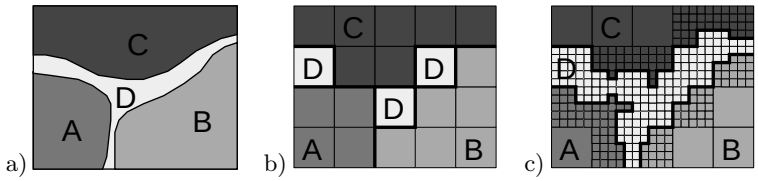


Fig. 3. a) Three tissues A, B, C are separated by a thin layer of tissue D. b) The finite grid resolution leads to discretization artifacts. c) Incorporating a-priori knowledge yields the correct reconstruction.

Incorporating user constraints Suppose we have three tissues A, B, C which we know to be separated by a thin layer of tissue D (Fig. 3a). Due to noise or limited resolution, it might not be possible to accurately capture D on a coarse voxel grid (Fig. 3b). However, it is easy to modify our algorithm such that it takes this a-priori knowledge into account. For this we define a face f to be *critical,* if f separates label A from label B (or B from C, or C from A), and the dominant label of f is set to D.

Then we proceed analogously to the removal of critical edges and critical vertices, this time however, we need a $7 \times 7 \times 7$ subdivision (5×5 in the two-dimensional case), in order to separate adjacent critical faces. To be more precise, we subdivide all voxels that contain a critical element and then resolve all critical faces/edges/vertices x (in that order) by relabeling all subvoxels in $N^1(x)$, $N^2(x)$, $N^3(x)$, to x's dominant label (Fig. 3c).

2.2 Extraction of the patches

To extract the patches we consider each (sub-)voxel face in turn. If the face separates two differently labelled voxels we extract a triangulated quadrangle for that face. Faces that are adjacent to subvoxel faces are triangulated by connecting its barycenter to the surrounding vertices by a triangle fan. We then apply a simple remeshing algorithm that is aware of common patch boundaries to remesh the extracted patches to a user-prescribed target edge length [5]. This algorithm also smoothes the patches and removes the typical staircase-artifacts that are due to the discrete nature of the voxels.

3 Results

We tested our algorithm on synthetic as well as on a number of real data sets of human hearts. As there are usually very few critical elements (typically less than 20 singular critical vertices and less than 300 critical edges) the overhead in memory consumption, running time and output complexity is very low. A typical $256 \times 256 \times 256$ volume containing eight different tissues takes less than 2 minutes to extract and remesh to a target edge length of 2mm. Figure 4 shows reconstructions at different target edge lengths while Figure 5 shows the different stages of the algorithm and the effect of incorporating a-priori knowledge.

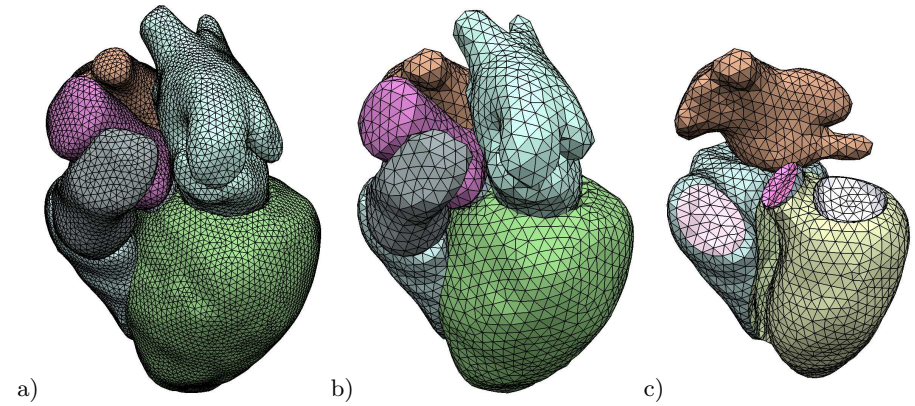


Fig. 4. Reconstruction of a human heart at a target edge length of 2mm (a) and 4mm (b). By hiding some of the boundary patches, the interior patches become visible (c).

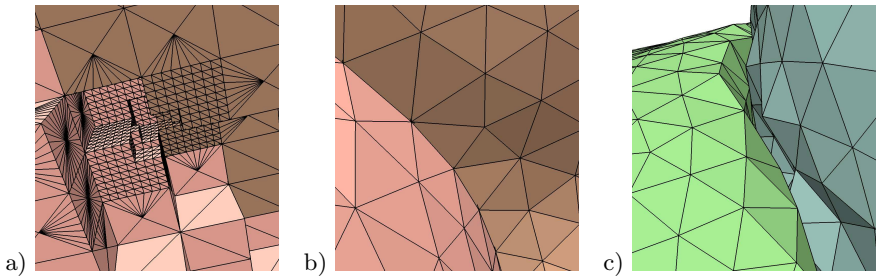


Fig. 5. a) A closeup on the reconstruction before remeshing clearly shows the subdivided parts. b) The same part after remeshing to a target edge length of 2mm. c) A reconstruction where two tissues are automatically separated due to user constraints.

Acknowledgments We want to thank O. Ecabert, J. Peters and J. Weese, Philips Research Aachen, for helpful discussions and for providing the datasets.

References

1. Ecabert, O., Peters, J., Lorenz, C., von Berg, J., Vembar, M., Subramanyan, K., Lavi, G., Weese, J.: Towards automatic full heart segmentation in computed-tomography images. In: Computers in Cardiology. (2005)
2. Reitinger, B., Bornik, A., Beichel, R.: Consistent mesh generation for non-binary medical datasets. In: Bildverarbeitung für die Medizin. (2005) 183–187
3. Hege, H.C., Seebass, M., Stalling, D., Zoeckler, M.: A generalized marching cubes algorithm based on non-binary classifications (1997)
4. Müller, H.: Boundary extraction for rasterized motion planning. In: Modelling and Planning for Sensor Based Intelligent Robot Systems. (1994) 41–50
5. Botsch, M., Kobbelt, L.: A remeshing approach to multiresolution modeling. In: Symp. Geometry Processing. (2004) 189–196