Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

HANS-CHRISTIAN HEGE MARTIN SEEBASS
DETLEV STALLING MALTE ZÖCKLER

A Generalized Marching Cubes Algorithm Based On Non-Binary Classifications

Preprint SC 97-05 (December 1997)

A Generalized Marching Cubes Algorithm Based On Non-Binary Classifications

Hans-Christian Hege, Martin Seebaß, Detlev Stalling, Malte Zöckler

Abstract

We present a new technique for generating surface meshes from a uniform set of discrete samples. Our method extends the well-known marching cubes algorithm used for computing polygonal isosurfaces. While in marching cubes each vertex of a cubic grid cell is binary classified as lying above or below an iso-surface, in our approach an arbitrary number of vertex classes can be specified. Consequently the resulting surfaces consist of patches separating volumes of two different classes each.

Similar to the marching cubes algorithm all grid cells are traversed and classified according to the number of different vertex classes involved and their arrangement. The solution for each configuration is computed on base of a model that assigns probabilities to the vertices and interpolates them. We introduce an automatic method to find a triangulation which approximates the boundary surfaces - implicitly given by our model - in a topological correct way. Look-up tables guarantee a high performance of the algorithm.

In medical applications our method can be used to extract surfaces from a 3D segmentation of tomographic images into multiple tissue types. The resulting surfaces are well suited for subsequent volumetric mesh generation, which is needed for simulation as well as visualization tasks. The proposed algorithm provides a robust and unique solution, avoiding ambiguities occuring in other methods. The method is of great significance in modeling and animation too, where it can be used for polygonalization of non-manifold implicit surfaces.

1 Introduction

The classic marching cubes algorithm proposed by Lorensen and Cline in 1987 [3] is the ancestor of a number of methods for generating surface meshes from sets of regularly sampled data [6, 7, 9, 11]. Originally designed for iso-surface generation the algorithm has been applied to a variety of related problems, ranging from polygonalization of implicit surfaces used in modeling and animation [2] to computation of complex meshes like removal envelopes [12].

In all these applications the basic assumption is that there are two distinct classes of grid vertices, e.g. vertices above and below an iso-surface or vertices inside and outside a volume of interest. Given a binary classification only (2^8) different configurations can occur in a cubic grid cell. This makes it possible to create the triangulation for a particular cell from a look-up table. In this way a very fast algorithm is obtained.

However, for many problems a simple binary classification is not suitable. Instead, one is interested in surfaces separating regions of multiple type in space. Such situations cannot be treated as simple iso-surface problems anymore. In this paper we extend the basic marching cubes algorithm by allowing multiple vertex classes. We introduce and describe

- a model to define an appropriate inner-cell space partitioning,
- an automatic method for generating topologically correct cell triangulations,
- the use of look-up tables for configurations with up to 3 vertex classes.

The surfaces created by our algorithm may contain contours and points where three or more regions join. More than two triangles are attached to an edge which is part of such a multi-region contour. In most applications only 2 or 3 vertex classes occur for the vast majority of cells. By using look-up tables for these cases we achieve a performance similar to traditional marching cubes. On the other hand, our automatic triangulation method allows us to create correct surfaces for the general case as well.

The ability to create surfaces from a non-binary space partitioning is of great significance in a number of areas. An important example for non-binary classifications are segmented tomographic images. In many medical applications for simulation of physical or physiological processes a geometric patient model is required, i.e., a volumetric mesh representing relevant tissue compartments. To be well suited for mesh generation, the surfaces extracted from segmented image data should not describe each tissue compartment as a separate object, but should be built up from surface patches each separating two regions of different type.

Traditional approaches for generating surface meshes from medical image data often rely on connecting contours in neighbouring slices [5]. In such methods ambiguities can arise, and they typically treat each compartment surface as a separate object. In contrast, our approach provides a unique, robust, and fast solution to the surface generation problem.

Bloomenthal and Ferguson [1] describe a different method for generating surfaces from non-binary space partitionings, they refer to as non-manifold surfaces. Their paper mainly addresses modeling via implicit surfaces and computational solid geometry applications. In contrast to our table-based approach they propose a continuation method. A cubic cell is propagated across the surface and decomposed into tetrahedra. For each tetrahedron a plausible triangulation is constructed algorithmically. Although decomposition into tetrahedra simplifies the polygonalization problem, it produces an excessive number of triangles, many of them not well-shaped. This is also the case with marching tetrahedra algorithm by Nielson and Franke [10]. These drawbacks are efficiently avoided by our approach.

2 The Algorithm

In the following sections we will outline the ideas of the surface meshing algorithm. We will first describe how to determine a unique region type for every point in space. Then a method is presented to compute a consistent approximate triangulation. Finally we show how the meshing algorithm can be accelerated via look-up tables and compare the method with standard marching cubes algorithms.

2.1 Space Partitioning

Our goal is to compute non-manifold surfaces which accurately separate regions of different type in space. For example in a medical application some parts of a volume

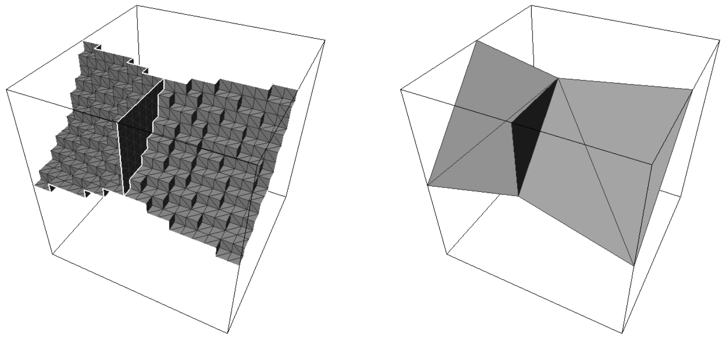


Figure 1: The left image shows the surface in a cell before the simplification step. The surface in the right image has been simplified by keeping only branching points and points on edges.

may be classified as bone, while others are classified as muscle or fat. We denote the total number of region types by n.

In the following we assume the region types to be defined on a discrete uniform grid. In case of surface modelling via generalized implicit surfaces such a labelling can be obtained by evaluating an analytical expression at each grid node. In medical applications usually a segmentation algorithm like thresholding or region growing is applied to a stack of tomographic images, e.g. CT or MR images.

Let us consider a cubic grid cell defined by eight vertices. In the general case it is not quite obvious how to subdivide the grid cell into regions of different types, since the type information is located at the vertices. To constitute a simple and unique space partitioning strategy we define a set of n probabilities p_i at each vertex. If a vertex is assigned to material k then all p_i are set to zero except p_k which is set to 1.

The probabilities for all types are interpolated independently within a cell. To classify an inner point $\boldsymbol{x}$ we simply choose the region of maximal probability, i.e.,

$$\text{region}(\boldsymbol{x}) = \{i \mid p_i(\boldsymbol{x}) \text{ maximal}\}, \tag{1}$$

where $p_i(\boldsymbol{x})$ denotes an interpolated probability at $\boldsymbol{x}$. For the triangulation algorithm described in the following section we use trilinear interpolation to compute $p_i(\boldsymbol{x})$. Other interpolation methods can be used as well, but usually they will result in more complex surface topologies.

It should be mentioned that a classification according to maximal probability does not require p_i to be 0 or 1 at the vertices of a cell. However, the case of distinct region labels naturally corresponds to such integer vertex probabilities. In addition, integer probabilities make it easy to distinguish between topologically different configurations. Therefore cell triangulations can be stored in look-up tables using a simple indexing scheme. An algorithm for the general case of non-integer probabilites will be presented in a subsequent paper.

2.2 Cell Triangulation

The interpolation model defines a unique set of separating surfaces inside a cell. On these surfaces the two largest probabilities are equal. Since trilinear interpolation is a non-linear transformation the inner surfaces are not planar. Our goal is to construct a triangular approximation of these non-planar surface patches. Such a triangulation has

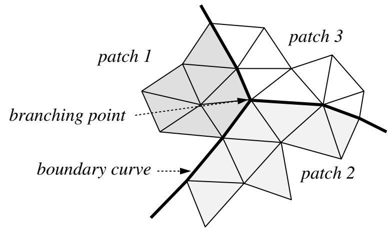
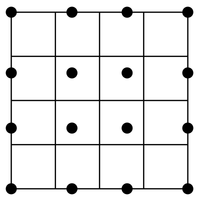


Figure 2: Parts of three different patches delimited by boundary curves (dark lines). At a branching point multiple boundary curves join.

to be consistent between adjacent grid cells to avoid holes in the resulting mesh. It should also preserve the topology inside a cell.

In the case of two different region types 256 configurations are possible which can be divided into 14 different topological classes. For three region types there are 44 additional topological classes, see Figure 4. To be able to handle any case with up to eight different region types we would like to have a method which is capable of generating the triangulation for any given configuration automatically. Such a method can be obtained by a straight-forward subdivision approach.

Subdivision. To obtain a representation of the inner structure of a grid cell we start by subdividing the cell into a number of smaller sub-cells. For each sub-cell the probabilities p_i at a representative point $\boldsymbol{x}$ are interpolated. We evaluate the probabilities not at the centers of a sub-cell, but at slightly translated locations as depicted in the following diagram:



Points are distributed equi-distantly. Notice, that they are located on the cell's faces for sub-cells at the boundaries. This way we can guarantee consistency between adjacent grid cells.

The sub-cells are classified according to Eq. (1). Whenever two adjacent sub-cells are of a different class their common face is added to an intermediate triangulation. An example of a resulting sub-cell triangulation is shown in Fig. 1. It turns out that in case of integer vertex probabilities 6^3 sub-cells are sufficient to give a topologically correct representation of the implicitly defined separating surfaces.

From the high resolution sub-cell triangulation we compute the final cell triangulation using a mesh simplification step.

Finding Patches. Simplification of the sub-cell triangulation first requires analysis of the surface topology. All connected triangles separating the same vertex classes are grouped into patches. Then the boundary curves surrounding each patch are extracted. For each vertex of the triangulation the number of boundary curves it belongs to is determined. For inner points of a patch this number is zero. Vertices which belong to more than two boundary curves will be referred to as branching points. The notion of surface patches, boundary curves, and branching points is illustrated in Fig. 2.

Simplifying Boundary Curves. The key observation for the following simplification step is that only few vertices of the current triangulation are of real importance.

Among these vertices are the branching points because they reflect the inner topology. Also vertices lying on the cell's edges are important, since these are referenced by multiple cells. All other points will merely introduce complexity on sub-cell level, which is not necessary to create a topologically correct triangulation. Therefore one can thin out the boundary curves by keeping important points only. After this procedure the boundary curve of a patch often consists of just three or four vertices. However, there are also patches with five or more vertices.

Retiling Patches. After simplifying the boundary curves the patches are retiled with triangles again. If a patch is planar this can be done in a straight-forward way, for example using an anchor point strategy.

More attention has to be paid in cases where non-planar patches occur. Then the simplification of the bounding curve in conjunction with an unfavorable triangulation can lead to patches which penetrate each other. We avoided all penetration problems by inserting an additional center vertex into patches with 5 or more boundary points and choosing a fan-type triangulation scheme. However, in many cases it is also possible to find a suitable triangulation without introducing an additional vertex.

2.3 Look-Up Tables

Subdividing grid cells, computing an intermediate sub-cell triangulation, and simplifying the resulting surfaces is a pretty time-consuming procedure and is definitely not suited to handle ten-thousands of grid cells occuring in common data sets. Fortunately the overall algorithm can be accelerated very much by making use of look-up tables. In contrast to standard marching cubes algorithms in our case not only points on edges have to be created, but also inner points and points on faces. The look-up table has to contain the point types in addition to the triangle information for each cell configuration.

As already mentioned in the case of two vertex classes only 256 different configurations are possible. For three vertex types this number increases to 6561 (3^8), although there are only 58 different topological situations (including the 14 standard cases). These 58 configurations are shown in Figure 4 in the Appendix, p. 9.

If the full triangulation for all 6561 configurations is tabulated, about 70 000 triangles have to be stored. The exact number depends on how many inner points are inserted during the patch retiling step. For every triangle three one-byte point indices have to be stored. For every vertex an additional byte is necessary to encode the point type (edge vertex, face vertex, or inner vertex). Consequently the table can easily be kept smaller than 1 MB. While this is still a handable size for a look-up table, a full table gets far too large in case of four vertex classes or more. For four classes 65536 (4^8) configurations are possible, but only 124 of them are topologically distinct (including the 58 cases from Figure 4). There are two ways out of this dilemma. The first way is to store only the topological different cases for the 'more than 3' configurations and to perform a mapping to a base configuration before the table look-up. Then the vertices read from the table have to be rotated and mirrored accordingly. Of course this method could also be applied to the 3-type case. However, here the use of a full table is feasible and easier to implement.

The second alternative to handle configurations involving four and more vertex classes arises from the observation that in most applications these cases are very rare. This makes it feasible to compute the triangulations for these cases on-the-fly using the comparatively expensive subdivison technique described above.

2.4 Comparison to Marching Cubes

Our method resembles the standard marching cubes algorithm [3] and its variations in respect of the cell-by-cell traversal and the use of look-up tables to create triangular surface patches. However, there are also some important differences.

First of all, checkerboard cases are handled differently than in most isosurface algorithms. Such cases occur if adjacent vertices on a face are of different type, while the vertices located at opposite corners have the same type. In this case the original marching cubes algorithm chooses a triangulation which does not preserve the symmetry of the configuration index. Special care has to be taken to ensure consistency between neighbouring cells. Otherwise the resulting surface would exhibit holes [8, 4, 9, 11]. The classification model described in Section 2.1 avoids such problems, prescribing a symmetric triangulation, i.e., a triangulation which has a branching point at the center of a face. Such cell triangulations are depicted in Figure 4. The approach avoids any ambiguities at the expense of an increased topological complexity of the resulting surface.

Another difference to the original marching cubes algorithm is that we have integer probabilities at the vertices instead of fractional values. As a consequence, points on edges are always located at the edge midpoint. In case of a binary vertex classification a similar approach is known as discretized marching cubes [7]. In contrast to the general case edge bisection has the effect that only a limited set of triangle orientations occur. This makes it possible to combine multiple triangles with equal orientation. Triangle decimation was the reason why the discretized marching cubes algorithm has been developed. A similar decimation strategy may be applied to our results as well.

3 Applications

In the following Figures 5 and 6 we additionally applied vertex shifting to account for non-integer probabilities. In this way smooth surfaces can be obtained. Details will be described in a subsequent paper. In [13] it is described how non-integer probabilities can be generated during the segmentation process.

3.1 Mesh Generation in Medicine and Biology

An important application of our algorithm is generation of geometric patient models for medical simulations. Typically generation of volumetric meshes is based on segmented tomographic image data. Such meshes are frequently needed for finite-element methods and other simulations. The proposed algorithm converts segmentation results into a form well suited as input for volumetric mesh generation. The surface data provided by the algorithm include a description of boundary curves and branching points which have to be reproduced in the volumetric meshes.

Another application area is biology, where image data from confocal microscopy has to be analysed and converted into surfaces separating compartments of different functionality.

Figure 3 shows a surface generated using our algorithm applied to a segmentation of kidney and liver. Note how topological information is revealed correctly by the algorithm. Bones and some more organs are shown in the upper part of Figure 5 in the Appendix at page 10. In the upper part surfaces are colored according to the tissue type of the enclosed volume, in the lower part according to the adjacent one.

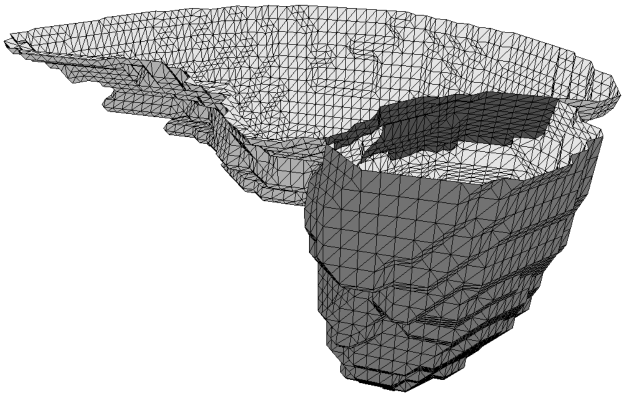


Figure 3: The image shows a surface from a segmentation of kidney and liver. The darker patch represent the common interface between both organs.

3.2 Modelling

Computational solid geometry is a common way to generate computer graphics models. This is done by using geometric primitve bodies like spheres, cylinders, or cones to add or remove volumes from the model. The geometric primitives are often defined by analytical functions. To actually render the model many algorithms and especially graphics hardware require a polygonal representation of the bounding surfaces. In the traditional approaches the primitive bodies are added or subtracted sequentially. Often this approach is too limited, since only two different region types are distinguished: inside and outside the model. If for example a boat swimming in the water is to be modelled, at least three different regions, namely water, air, and the boat are needed, to distinguish between the water surface and the boat surface, that definitely should look differently. This has been discussed in [1], where a cubic cell which is propagated across the surfaces, is divided into tetrahedra to construct the triangulation.

With our method surface models can be generated from samples on a discrete grid of appropriate resolution, taking advantage of the high-performance look-up tables. The generated surfaces are guaranteed to be consistent, e.g. to have no holes, even if the resolution is choosen too coarse to represent all details of the model. A simple geometric example is shown in Figure 6. There non-integer probabilities have been considered by moving the vertices on edges of the cubes. This leads to much smoother surfaces.

4 Conclusions

We have presented a generalization of the well known marching cubes algorithm, which can treat non-binary space partitionings instead of only distinguishing between above and below an isosurface. Using look-up tables, the new method achieves high performance, comparable to the standard marching cubes algorithm. It offers new possibilities for generating volumetric meshes as needed in medical applications, as well as for surface mesh generation in computer graphics modeling.

References

[1] J. Bloomenthal, K. Ferguson, *Polygonization of Non-Manifold Implicit Surfaces*, Proceedings of SIGGRAPH '95 (Los Angeles, California, August 6-11, 1995). In *Computer Graphics* Annual Conference Series, 1995, ACM SIGGRAPH, pp. 309-316.

[2] T.A. Galyean, J.F. Hughes, *Sculpting: An interactive volumetric modeling technique*, Computer Graphics (SIGGRAPH '91 Proceedings), 25:4, pp. 267-274, 1991.

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

[4] W.E. Lorensen, *Extracting Surfaces from Medical Volumes*, In Visualization'94 Course Notes: Volume Visualization Algorithms and Applications, pp. 26-45, 1994.

[5] D. Meyers, S. Skinner, K. Sloan, *Surfaces from contours*, ACM Trans. Graphics, 11(3), pp. 228-258, July 1992.

[6] D. Moore, J. Warren, *Mesh Displacement: An Improved Contouring Method for Trivariate Data*, Technical Report TR-91-166, Rice University, Department of Computer Science, 1991.

[7] C.Montani, R. Scateni, R. Scopigno, *Discretized Marching Cubes*, In Proceedings of Visualization '94, IEEE Computer Society Press, pp. 281-287, 1994.

[8] C. Montani, R. Scateni, R. Scopigno, *A modified look-up table for implicit disambiguation of Marching Cubes*, The Visual Computer, 10(6), pp. 353-355, 1994.

[9] B.K. Natarajan, *On generating topologically consistent isosurfaces from uniform samples*, The Visual Computer, 11(1), pp. 52-62, 1994.

[10] G.M. Nielson, Richard Franke, *Computing the Separating Surface for Segmented Data*, IEEE Visualization '97, Oct., pp. 229-233, 1997.

[11] S. Röll, A. Haase, M. von Kienlin, *Fast Generation of Leakproof Surfaces from Well-Defined Objects by a modified Marching Cubes Algorithm*, Computer Graphics Forum, 14(2), pp. 127-138, 1995.

[12] W. Schroeder, W. Lorensen, S. Linthicum, *Implicit Modeling of Swept Surfaces and Volumes*, In Proceedings of Visualization '94, IEEE Computer Society Press, pp. 40-45, 1994.

[13] D. Stalling, M. Zöckler, H.C. Hege, *Interactive Segmentation of 3D Medical Images with Subvoxel Accuracy*, to appear in Proceedings of CAR'98 Computer Assisted Radiology.

Appendix

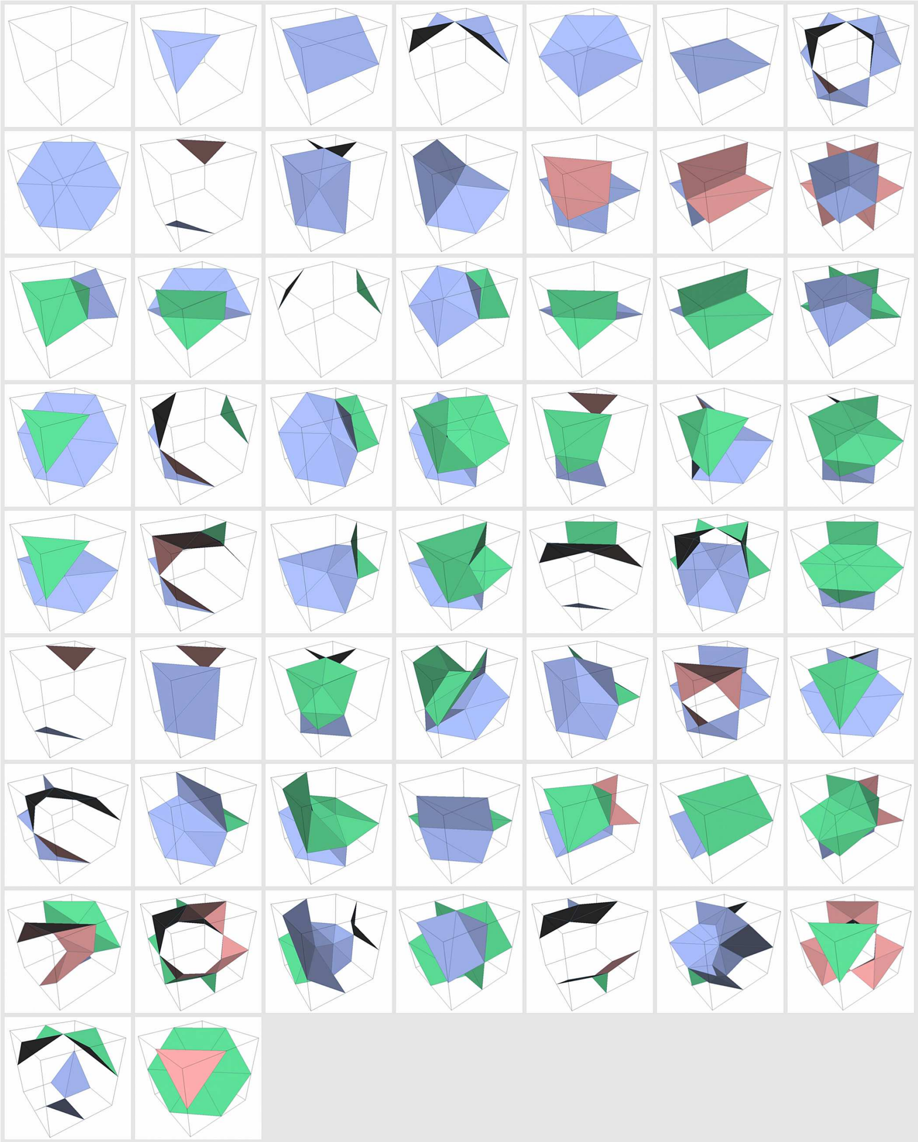
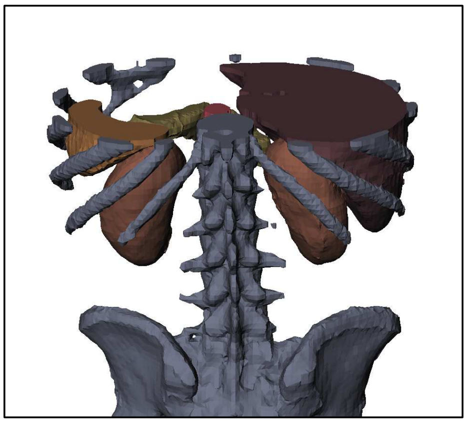


Figure 4: The image shows the look-up table for up to three different vertex classes. The first two rows contain the cases with only two classes. For some cases the triangulation is different from the original marching cubes.



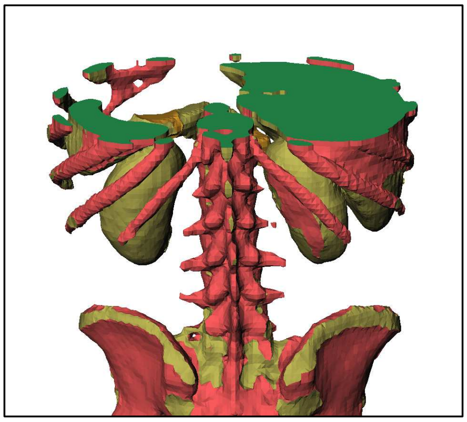


Figure 5: Compartments resulting from segmented CT data. Top: bone and some of the organs, bottom: same, but surfaces are colored according to the adjacent tissue type.

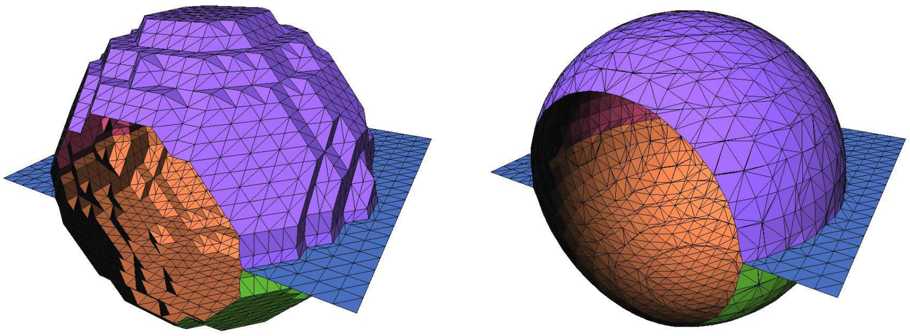


Figure 6: Three different regions are separated by the surface: inside the sphere, below the plane, and above the plane. Left: triangular surfaces with a finite number of normal directions due to integer probabilities $\{0, 1\}$ only, right: smooth surface due to utilization of non-integer probabilites.