Konrad-Zuse-Zentrum
für Informationstechnik Berlin

Takustraße 7
D-14195 Berlin-Dahlem
Germany

D. STALLING, M. ZÖCKLER, O. SANDER, H.C. HEGE

Weighted Labels for 3D Image Segmentation

Preprint SC 98-39 (December 1998)

Weighted Labels for 3D Image Segmentation

D. Stalling, M. Zöckler, O. Sander, H.C. Hege

December 1998

Abstract

Segmentation tools in medical imaging are either based on editing geometric curves or on the assignment of region labels to image voxels. While the first approach is well suited to describe smooth contours at subvoxel accuracy, the second approach is conceptually more simple and guarantees a unique classification of image areas. However, contours extracted from labeled images typically exhibit strong staircase artifacts and are not well suited to represent smooth tissue boundaries. In this paper we describe how this drawback can be circumvented by supplementing region labels with additional weights. We integrated our approach into an interactive segmentation system providing a well-defined set of manual and semi-automatic editing tools. All tools update both region labels as well as the corresponding weights simultaneously, thus allowing one to define segmentation results at high resolution. We applied our techniques to generate 3D polygonal models of anatomical structures.

1 Introduction

Realistic patient models are an essential prerequisite for modern computer-aided medical treatment planning, e.g., in radiation therapy, hyperthermia, or surgery. The generation of such models – either polygonal or volumetric ones – requires accurate segmentation of a stack of tomographic images. In order to represent smooth tissue boundaries or small structures like vessels, ideally segmentation results should be defined at subvoxel accuracy. In particular, this is true if large slice distances are used.

In principle, segmentation results may be represented in two ways, either using geometric primitives, e.g., polylines or splines, or by defining additional region labels for each voxel. The first approach has the advantage that smooth tissue boundaries can be defined even for low resolution input images. On the other hand, the method is more or less limited to 2D image segmentaion. In order to generate 3D models, typically separate 2D contours are defined in consecutive slices of a stack of tomographic images. These contours are then connected by polygons [2]. Problems occur, if the topology of contours in neighbouring slices changes, e.g. in case of a branching point. Morover, in general it is difficult to guarantee that individual contours don't overlap. However, this is an important requirement if consistent patient models are to be generated later on.

The second approach, labeling of image voxels, makes it easy to classify all parts of the image in an unambiguous way. A great variety of 2D and 3D segmentation algorithms directly generate region labels. In addition, region labels can be edited interactively in an intuitive way. Interactive editing plays an important role in many practical applications, since often fully automatic algorithms aren't available today. Appropriate interactive tools are well-known from traditional paint box systems. The user first selects parts of an image and then assigns these parts to certain regions or tissue types.

A severe drawback of the straight-forward labeling approach is that the accuracy of the segmentation results is limited to the resolution of the underlying image data. The tissue boundaries will inevitably show staircase artifacts. Of course, one remedy to this problem would be to super-sample the original image data. However, while the resolution of the segmentation results would be improved the staircase artifacts still remain. Thus, in many situations, for example, when surface curvature is to be computed, super-sampling is not an adequate alternative. In addition, super-sampling requires a huge amount of additional memory to store region labels at subvoxel level.

In this paper we present an alternative method to improve the resolution of segmentation results represented by region labels. In addition to the labels themselves we store additional weights loosely indicating the confidence of the assignment of a voxel to a particular region. Based on this confidence information smooth tissue boundaries can be extracted. The approach is similar to standard iso-contouring algorithms such as marching cubes or its 2D analogon [3]. Besides the probability model itself we also present tools allowing one to update region labels and weights interactively in a consistent way.

In the following we first describe the general idea of improving resolution by introducing additional weights. We then discuss how such representations can be edited interactively. Finally, we present some results obtained by applying our technique to 3D grid generation.

2 General Approach

One motivation for our work was the observation that contours or surfaces extracted from medical data sets by means of an iso-contour algorithm such as marching cubes are quite smooth and well-formed. In this section we dicuss why this is the case and how the same effect can be obtained by introducing suitable weights for each pixel or voxel.

2.1 Binary Classifications and Isocontours

Imagine an arbitary scalar function f sampled on a regular 2D grid. Assume that the grid is sufficiently fine so that f can be reconstructed in each cell from its corner values using bilinear interpolation. In order to extract isocontours of this function corresponding to a certain threshold f_c we may process each grid cell independently. Whenever at the corners of a cell there are function values below *and* above f_c then the cell is intersected by an isocontour. Since a quadrilateral cell has four corners, in total there are $2^4 = 16$ different configurations, 14 of which actually correspond to an intersection. However, note that a binary classification of cell vertices alone does not uniquely determine contour topology in the interior of a cell, even if f is interpolated bilinearly. Ambiguities arise, if two values above and below the threshold are located at opposite corners of a cell. Nevertheless, inside a cell isocontours are usually approximated by straight line segments, c.f. Fig. 1. In order to find the position of an isocontour vertex on a cell's edge, the function values are interpolated linearly and a displacement coordinate u is computed using

$$u = \frac{f_c - f_1}{f_2 - f_1}. \tag{1}$$

If u equals 0.5 the edge is intersected exactly in the middle. Now, notice that the same result can be obtaind by introducing two weights w^+ and w^- per

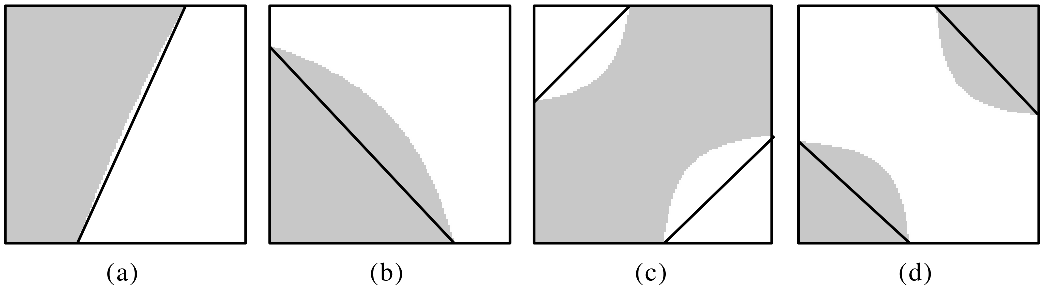


Figure 1: In order extract an iso-contour a function f is sampled on a regular 2D grid. Inside each grid cell contours are approximated by straight line segments.

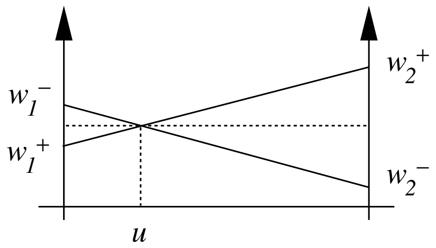
vertex, namley

$$w_i^{\pm} = c \pm d\,(f_i - f_c), \tag{2}$$

with arbitrary constants c and d. A value $w^+ > w^-$ indicates that a vertex is above the threshold, while a value $w^+ < w^-$ indicates that it is below. An isocontour just separates these two regions. In this case we can compute u as follows,

$$u = \frac{w_1^- - w_1^+}{w_1^- - w_1^+ + w_2^+ - w_2^-} = \frac{f_c - f_1}{f_2 - f_1}. \tag{3}$$

The situation is illustrated in the following figure:

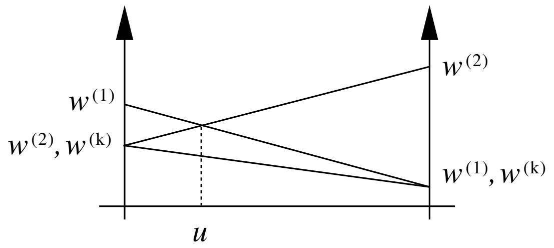


Inside a grid cell each weight is obtained in the same way as the function f, namely by means of bilinear interpolation. In this way the cell is separated into two different regions. In one regions there is $w^+ > w^-$ while in the other region there is $w^+ < w^-$. The isocontour just corresponds to points where both weights are equal. Of course, isocontours of a 3D function can be defined in the same way by introducing weights on the vertices of a 3D hexahedral grid.

2.2 Multiple Labels

The results of the previous section are easily generalized to cases where more then two different labels occur. Instead of two weights w^+ and w^- we then have n different numbers $w^{(k)}$, $k = 1...n$, one for each label or tissue type. Given some rule to interpolate these values inside a grid cell, a partioning of the cell is defined by taking the maximum value $w^{(k)}$. On tissue boundaries two of the $w^{(k)}$ are equal. In contrast to the binary case there are also points, where three or more regions join, At these points more than two weights are equal.

The approach outlined above provides an elegant way to define a partioning into regions. However, in case of arbitrary weights complex topological configurations may occur. In particular, an edge of a cell may be intersected by more than one line segment. In order to simiplfy the situation we therefore enforce all but the maximum weight at each vertex to be equal. In this case only two different weights w occur at a vertex, and edges are intersected at most once. This is illustrated in the following figure:



Another advantage of enforcing the non-dominant weights to be equal is, that the amount of memory needed to store the segmentation results is reduced dramatically. A further simplification is achieved by assuming that the maximum weight is in range 128...255 and that the alternate weights are implicitely given by $255 - w_{max}$. In this case only one byte additional storage is needed per voxel.

Like in the binary case, region boundaries are approximated by straight line segments inside a grid cell. However, in addition to the binary configurations some more complex cases have to be considered. If the maximum weights at the corners of a cell are associated to three different labels, at most two points can occur where the corresponding regions join, provided all weights are interpolated inside a cell. However, if the two corners assigned to the same label are located along a common edge, then there is only one such point inside the cell. The position of this point can be evaluated exactly and a result as shown is Fig. 2 (a) is obtained. On the other hand, if the two labels are located at opposite corners, there is either no intersection point, c.f. Fig. 2 (b), or two of them. Two intersection points usually also occur, if every corner is assigned to a different label. For case of simplicity, configurations containing two intersection points are approximated by only two straight line segments as shown in Fig. 2 (c) and (d).

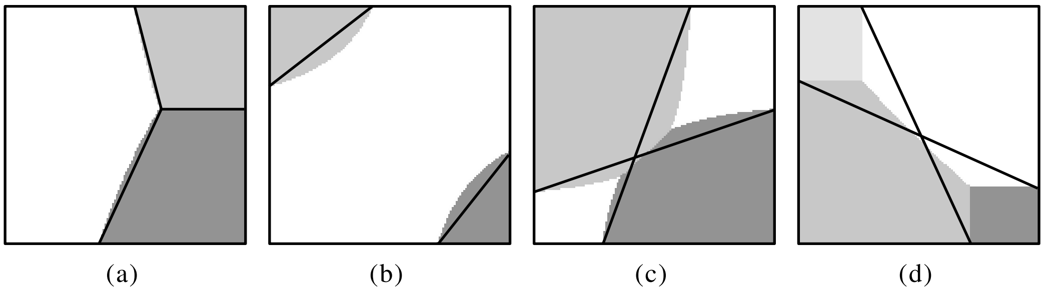


Figure 2: The figure shows how region boundaries defined by considering maximum interpolated weights are approximated by straight line segments.

3 Editing Weights

In the previous section we showed how smooth tissue boundaries can be obtained by supplementing the region labels with additional weights w. We now want to dicuss how labels and weights can be edited interactively in a consistent way. In our implementation labels are usually not modified directly. Instead, certain areas of an image are first selected using tools like the *magic wand* (region growing), the *brush* (painting), or the *lasso* (contouring). In this respect, the system resembles a traditional paint box system. Selected areas are usually visualized by a semi-transparent red layer on top of the CT or MR image to be segmented. Selected pixels may be added to a particular region or may be substracted from it. The selection is not a binary flag but is a number w_s between 0 and 255. The label of a pixel is only changed if w_s is at least 128. In this case w is set to w_s. However, if a pixel already was assigned to the current label, w is only updated if w_s is bigger than the old weight. Due to our weighting model all other labels at this pixel automatically receive a weight of $255 - w$. If the selection is less than 128, the label itself is never changed, but the weight w is set to $255 - w_s$ provided the new value is bigger than the old one. These rules ensure that boundaries remain smooth if editing operations are performed locally.

Our task is to design interactive segmentation tools which set the selection properly. In particular, w_s should be close to 255 in the interior of a selected area, but should linearly drop to 0 at its boundary. The transition area should be at least two pixels wide, so that any intersected edge completely falls into it.

3.1 Magic Wand

The magic wand implements a region growing algorithm similar to the iso-contouring algorithm discussed in Sect. 2.1. The user clicks at some point into the image. The grey value at this point together with a user-defined tolerance determines a lower bound f^- and an upper bound f^+. Starting from the seed point all connected pixels within this range are selected. The value w_s of a selected pixel is determined as follows:

$$p_s = \begin{cases} 128 + d\,(f^+ - f), & \text{if } f \geq (f^- + f^+)/2 \\ 128 + d\,(f - f^-), & \text{if } f < (f^- + f^+)/2 \end{cases} \tag{4}$$

The values are clamped to the interval 0...255. d is a constant factor determining the width of the transition region. Note, that in contrast to ordinary region growing the growing operation must be continued even if values smaller than f^- or bigger than f^+ are encountered. Only in this way a transition region containing weights smaller than 128 can be obtained. This is an essential prerequisite for defining smooth boundaries. We therefore continue region growing until a local minimum smaller than f^- or a local maximum bigger than f^+ is found. This strategy yields smooth boundary curves but also ensures that no unconnected regions arise.

3.2 Brush and Lasso

The brush and the lasso allow the user to select pixels by painting and contouring. Subvoxel accuracy, i.e., weights somewhere between 0 and 255, can only be achieved if the image is magnified on the screen. In this case only parts of a pixel may be covered by a brush or by a hand-drawn contour. It turns out that the coverage itself is not suited to compute proper values w_s. Instead, we incorporate a transition region into a brush. Within this region the selection linearly varies from 0 to 255. In order to determine w_s for a given pixel, the brush is sampled at the center of that pixel. The transition region is chosen two pixels wide. For the lasso we proceed similarly. Given a hand-drawn polyline we compute the normal vectors of the contour in order to define a proper transition region. Again this region is sampled at the pixel centers. Ideally, instead of direct sampling a fast scan-conversion algorithm similar to those used for drawing anti-aliased polygons should be applied.

Instead of being defined manually contours may also be obtained from active contour models or snakes algorithms. Such algorithms are well suited to modify a hand-drawn contour so that it fits the underlying image data. Obviously, contours generated automatically can be processed in the same way as hand-drawn contours.

3.3 Smoothing

Filters provide an alternative way for editing weights. We distinguish two variants, smoothing and resampling. For smoothing we generate n intermediate intermediate images, one for each region. These images are zero everywhere except for pixels belonging to the current region. At these pixels the weights of the input image are copied. Then all n images are blurred using a Gaussian filter. Each pixel is assigned to the region corresponding to the maximum blurred intensity, c.f. Fig. 3. The maximum intensity itself determines the weight for that pixel. This filter operation not only produces smooth boundaries, but also removes isolated (noisy) pixels.

Of course, smoothing can also be performed for 3D images. This is often useful in order to obtain smooth segmentation results in case of large slice distances. Instead of performing 2D blurs, for every label we average three

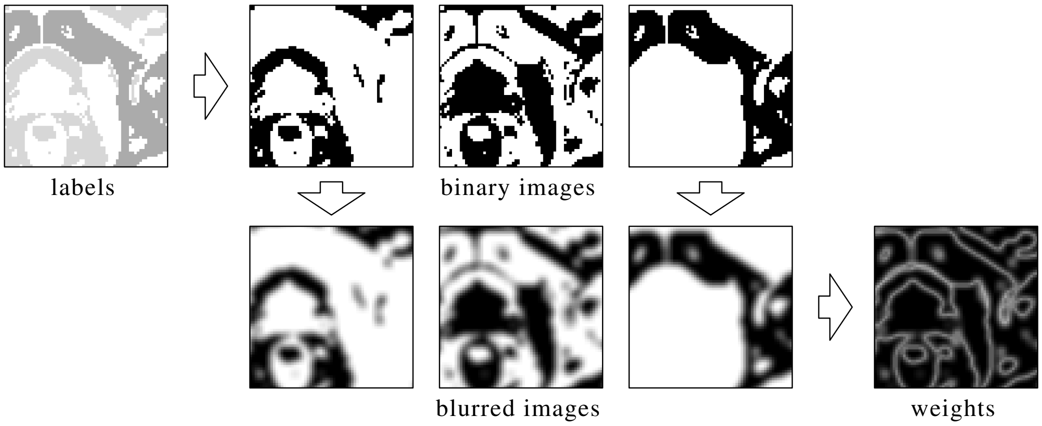


Figure 3: A smoothing filter for weighted labels can be implemented by smoothing intermediate images for each label. Determing the maximum intensity of all intermediate images yields new labels with associated weights.

binary images corresponding to neighbouring slices. If a whole 3D volume is to be processed and if many labels occur, performance issues become relevant. In order to avoid redundant computations we proceed as follows.

The main algorithm is straightforward. We take the first three slices, split them into $3n$ intermediate images as described above and buffer the results. After computing slice 0 as the weighted average of only the first two buffers, this allows us to compute slice 1. The last buffer can now be used to hold the intermediate images of the next slice. This process is repeated for the whole 3D image. Unfortunately, while this is pretty fast, large images with lots of different regions need a lot of buffer space. More precisely, we need three buffers containing 2D images for every region that appears anywhere in the whole dataset. This can easily amount to several dozens of megabytes. Therefore we have implemented a second algorithm which doesn't buffer intermediate images for all regions but computes them on the fly. In this case only space for three single intermediate images must be allocated in addition to the final slices itself. However, since a smoothed intermediate image is needed in the computation of three resulting slices, Gaussian filtering is performed three times for each region instead of just once. Nevertheless, this strategy is advisable in situations where memory is short.

To further speed up either algorithm, in a preprocessing step we determine for each label in each slice whether it is present at all, and if so the smallest bounding axially parallel rectangle. All further convolving and averaging operations can then be restricted to that rectangle, which reduces processing time immensely.

3.4 Resampling

Often it is necessary to reduce the resolution of a labeled image for subsequent operations. In particular, this is true if 3D polygonal surfaces are to be extracted. Ordinary resampling filters cannot be applied to labeled images, since labels cannot be interpolated. For example, if two regions indicated by labels 0 and 2 have a common boundary, the labels must not be averaged to 1 since this may denote a completely different material.

However, if the zoom factor is an integer number, i.e., if $n \times n$ pixels are combined into a new one, then it is easy to update labels and weights in a consistent way. We simply take the label with maximum accumulated weights. The accumulated weight divided by the number of pixels being averaged yields the weight for the new pixel.

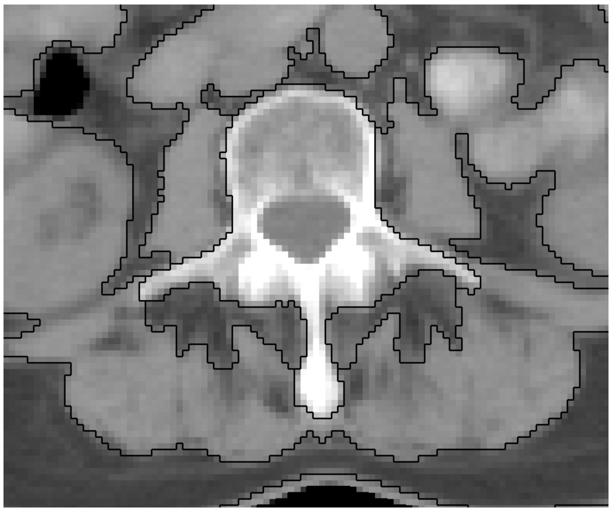
4 Application

We applied the techniques described above to treatment planning in regional hyperthermia. In this application a 3D polygonal grid of the abdominal region of a patient has to be produced. The patient model is generated using a stack of about 60 CT images, typically with a distance of one centimeter. At least fatty tissue, muscles, bones, and the tumor need to be identified. The main requirement for segmentation is not to achieve accuracy at a diagnostic level. Instead, smooth contours should be generated in order to ease further processing.

4.1 Segmentation of 2D-Slices

The first step in the segmentation process is thresholding. In this way fatty tissue, muscles, and bones are identified. Then usually the smoothing filter described in Sect. 3.3 is applied. Additional corrections can be performed using the brush or lasso. These tools are also used to separate the tumor itself. The boundary curves between different tissue types can be displayed either with weights being ignored or being considered correctly. Switching between both display styles is a useful feature and gives an idea of how the weights are set. An example is shown in Fig. 4. The complete segmentation process including definition of inner organs takes about 1-2 hours for 40-60 slices.

(a)



(b)

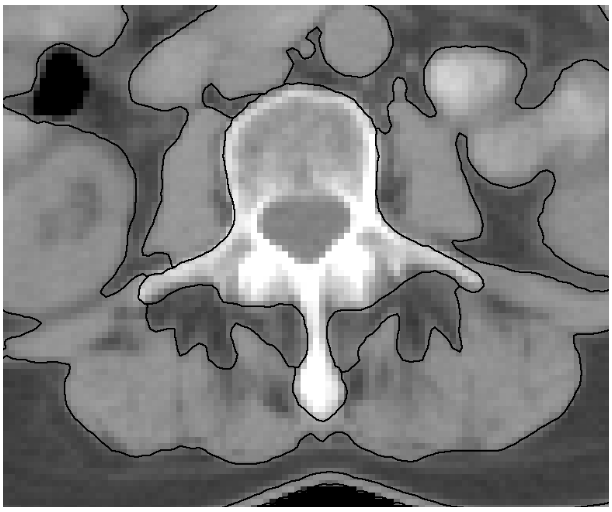


Figure 4: Contours extracted from a labeled CT image. In (a) pixel boundaries are shown and no weights are considered. Taking weights into account yields much smoother results (b).

4.2 3D Grid Generation

In order to produce a 3D polygonal mesh describing the tissue interfaces we proceed very similar to the 2D iso-contouring algorithm outlined in Sect. 2.1. Instead of generating line segments for 2D pixels we create triangles for 3D voxels. The triangle vertices are shifted according to the weights defined at the cell corners. The method generalizes the standard marching cubes algorithm and uses lookup tables, too [1]. A resulting model is shown in Fig. 5. The smoothness of the surfaces ensures that in a subsequent step standard techniques can be applied to reduce the number of triangles of the model.

5 Summary

We presented a weighting model which allows us to represent segmentation results using region labels at significantly improved resolution. The method is able to describe smooth contours in 2D and 3D. We described how weights associated to a pixel can be computed using simple thresholding or region growing, and how this information can be edited interactively using tools like brush or lasso. Smoothing and resampling filters provide another way to modify the weights. Our method can be used to generate realistic 3D polygonal models for medical treatment planning.

References

[1] H.-C. Hege, M. Seebaß, D. Stalling, M. Zöckler, A Generalized Marching Cubes Algorithm Based on Non-Binary Classifications, ZIB Preprint SC 07-05, Berlin, 1997.

[2] B. Geiger, Three-dimensional modeling of human organs and its application to diagnosis and surgical planning. Technical Report 2105, Institut National de Recherche en Informatique et Automatique, Dec. 1993.

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

(a)

(b)

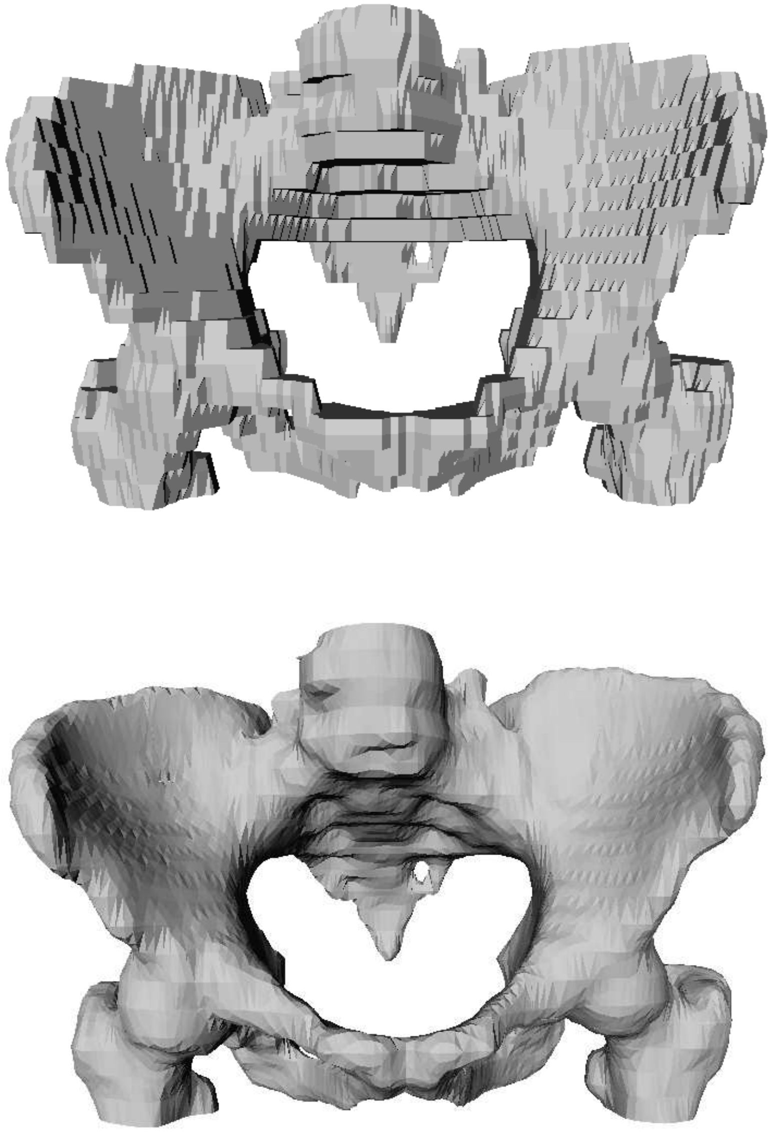


Figure 5: Polygonal model of a human pelvis. For the model at the top only region labels have been considered, while for the one at the bottom weights were taken into account.