Fast segmentation of 3D data using an octree

J. Rodrigues[1], R.E. Loke[2] and J.M.H. du Buf[2]

1 Escola Superior Tecnologia, University of Algarve
Campus da Penha, 8000 Faro, Portugal
email: jrodrig@ualg.pt
2 Vision Laboratory, University of Algarve
Campus de Gambelas, 8000 Faro, Portugal
email: {loke,dubuf}@ualg.pt

Abstract: *The algorithm developed uses an octree pyramid in which noise is reduced at the expense of the spatial resolution. At a certain level an unsupervised clustering without spatial connectivity constraints is applied. After the classification, isolated voxels and insignificant regions are removed by assigning them to their neighbours. The spatial resolution is then increased by the downprojection of the regions, level by level. At each level the uncertainty of the boundary voxels is minimised by a dynamic selection and classification of these, using an adaptive 3D filtering. The algorithm is tested using different data sets, including NMR data.*

Keywords: *3D segmentation; boundary refinement; octree; NMR data*

1. Introduction

Image segmentation is a well-established topic in image processing. All recent text books give a good overview of the different techniques, but mainly for the 2D case although many schemes can be directly extended to three image dimensions. However, in some cases the complexity of the algorithm poses a problem in terms of CPU time when huge volumetric data sets are to be processed, in which segmentation is only a first step and further processing, at least an interactive visualisation, is necessary. In this paper we extend a well-established 2D method to 3D with the aim of obtaining a good segmentation quality while saving as much CPU time as possible.

Spann and Wilson [2] have shown that in a 2D segmentation the application of a quadtree is very useful because of the smoothing: the noise reduction improves class separation such that at a certain tree level a clustering and classification yield correct initial classes. The coarse resolution at that level is improved by a level-by-level downprojection in which the boundary pixels are reclassified on the basis of the information available at the lower tree levels. Subsequently, Schroeter and Bigün [1] experimented with different clustering algorithms and improved the boundary refinement in the downprojection by applying adaptive filtering. All clustering algorithms considered gave more or less the same results and the final segmentation quality was excellent. Hence, we studied an extension of the algorithm from 2D to 3D while trying to make it as fast as possible in view of the sizes of 3D data sets. In the following sections the algorithm will be presented in detail and experimental results will be shown.

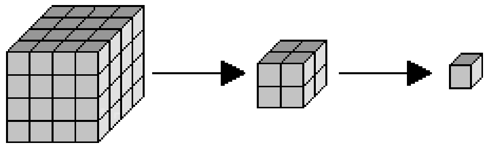

Figure 1. Octree construction by averaging nonoverlapping blocks of eight voxels.

2. Pyramid construction

The 3D pyramid used is the octree (Fig. 1), in which non-overlapping blocks of 2x2x2 voxels of the image $I(i,j,k)$ of size N_0^3, with $N_0 = 2^m$, are averaged. This results in an image of size N_1^3 with $N_1 = 2^{m-1}$ one level above the basis. We note that we use equal dimensions in x, y and z equal to a power of two, but the dimensions can be different as long as they are even. Mathematically

$$I(i,j,k;L) = \frac{1}{8}\sum_{a=0}^{1}\sum_{b=0}^{1}\sum_{c=0}^{1} I(2i+a,2j+b,2k+c;L-1),$$

where $0 < L \leq L_{\max}$ and $I(i,j,k;0) = I(i,j,k)$. In practice we do not construct the octree until the highest level but go only a few levels above the basis $L = 0$. The number of necessary levels depends on the application and can only be determined by experiment.

3. Clustering and classification

The second step is the application of a clustering algorithm at a certain pyramid level $L_{\max}$, using in our case a local centroid clustering directly based on the voxel values; see [2]. The convergence of this algorithm is difficult to prove, but in practice it has been observed that it converges with a small number of iterations (typically 10-20). This algorithm allows to find the class centres. The sensitivity of the clustering depends on the peaks in the original histogram and the window size utilised. Sensitivity is bigger for small windows and consequently more classes can be found, but as we increase the window size, sensitivity decreases, the algorithm becomes more robust to the noise, allowing only to find the significant

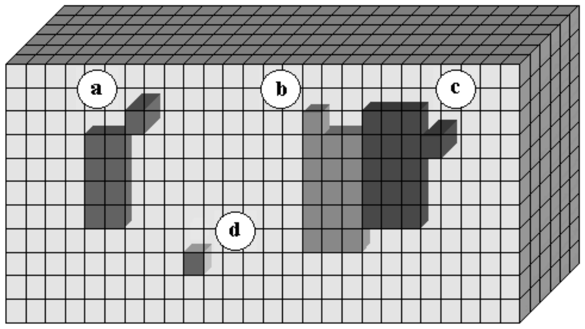

Figure 2. An isolated voxel (d), an isolated insignificant region (a) and two connected insignificant regions (b,c).

classes. This dependence can be removed by applying the algorithm with a set of increasing window sizes until the number of classes stabilises. For the creation of the label image at $L_{\max}$ a minimum-distance classification is applied, which results in the label image $I\gamma(i,j,k;L_{\max})$. This method is simple and fast but it can create single-voxel and insignificant regions that must be corrected in the restoration of the spatial connectivity.

4. Restoration of the spatial connectivity

Because of the application of a minimum-distance classification without any spatial connectivity constraints, there is the possibility of creating isolated voxels and sets of voxels which form insignificant regions within significant ones. A voxel at position (i,j,k) is considered to be isolated (Fig. 2d) if in $I_\gamma(i,j,k;L_{\max})$ its label is different from all the labels in its N_{26} neighbourhood. An isolated voxel is assigned the class most representative from all classes in its N_{26}. If there are classes with an equal number of voxels in N_{26}, the voxel is assigned the class that has the minimum label distance from its own label.

A region is considered to be insignificant (Fig. 2a,b,c), when it does not count a sufficient number of connected voxels in the 26-connected sense. The voxels within such regions are assigned the neighbouring classes and/or assigned between themselves (Fig. 2b,c) by a deterministic process in a single pass through the volume. This scanning process starts at the first coordinate of the volume (upper-left corner in Fig. 2), it passes through all voxels and ends at the last coordinate (lower-right corner). Once a region is determined to be insignificant, like R_a with n_a voxels in Fig. 2, its voxels will be reclassified following the scanning order, as if they were isolated voxels.

In the case of region b in Fig. 2, which is adjacent to region c, there exists the possibility of assigning its voxels the label of region c. After finishing the reclassification of all voxels belonging to region b, if region c proves to be insignificant it will be processed in the same way as were regions a and b. This method can provoke a gain of voxels in adjacent insignificant regions, which could become significant, but an alternative solution (multi-pass method) without this problem would be more expensive in terms of CPU time. Such multi-pass solutions must be considered in applications where no or smaller systematic biases towards the right or left etc must be achieved, and can be easily implemented.

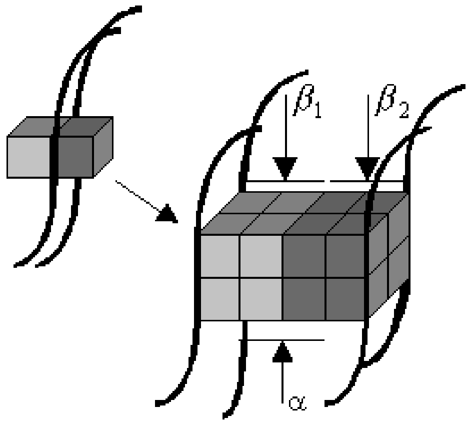

Figure 3. Two boundary voxels at level L with the definition of the boundary regions β_1, β_2 and α after the downprojection to $L-1$.

5. Boundary refinement

The last step, boundary refinement, consists of mapping the label image I_γ from the level $L_{\max}$ to level 0. This is accomplished by the downprojection of the labels within the regions to level $L_{\max}-1$, in combination with the analysis of the voxels around the boundaries in the original data tree I available at level $L_{\max}-1$. This process is repeated level by level until $L=0$. Assume that we are at level L and are going to level $L-1$.

(A) In $I\gamma(i,j,k;L)$ the boundary voxels are determined. A voxel is considered to be a boundary voxel if there is at least one different label in its N_{26} neighbourhood.

(B) The image $I_\gamma(i,j,k;L-1)$ is obtained by the downprojection of the label of each parent voxel, that is not a boundary voxel, to its 8 children, i.e. $I_\gamma(i,j,k;L-1) = I_\gamma(i/2,j/2,k/2;L)$, with / being an integer division.

(C) The children that belong to the projected boundary form the region β at level $L-1$ (Fig. 3) will be reclassified using 3D butterfly filters in several orientations.

The boundary region β is divided into two: β_1 that corresponds to the 8 voxels originating from the left voxel at level L and β_2 originating from the right voxel at level L. We also define the centre boundary region α that contains only those voxels from β that connect β_1 with β_2. In order not to process all 16 voxels belonging to the β region, we process only the 8 α voxels; the other 8 voxels are assigned their two parent's labels to save CPU time. This is a simple example because the boundary orientation corresponds with an axis of the 3D volume. In practice the boundary can be in any orientation and a neighbourhood can be more complex (vertices between three or more regions). Hence, we extract all candidate orientations at level L (Fig. 4a) to select the combinations of α voxels at

Table 1. The list of voxels belonging to region α at tree level $L-1$ as a function of the boundary positions at level L.

Boundary position at L	Voxels in α at $L-1$
1	A
2	A+B
3	B
4	A+C
5	A+B+C+D
6	B+D
7	C
8	C+D
9	D
10	A+E
11	A+B+E+F
12	B+F
13	A+C+E+G
14	B+D+F+H
15	C+G
16	C+D+G+H
17	D+H
18	E
19	E+F
20	F
21	E+G
22	E+F+G+H
23	F+H
24	G
25	G+H
26	H

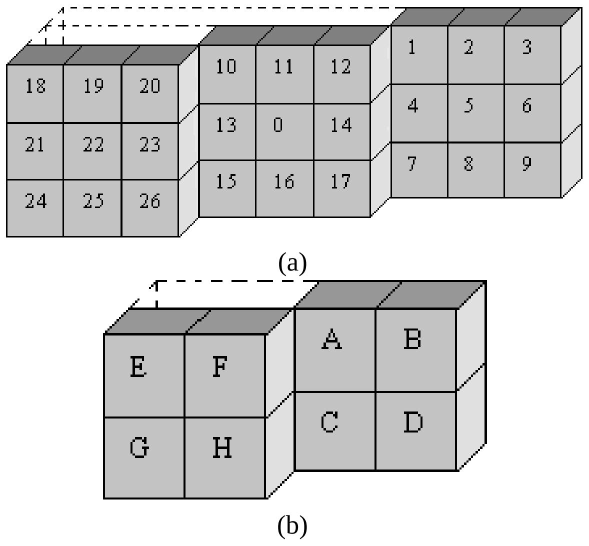

Figure 4. The N_{26} neighbourhood at level L (a) and the voxels in the boundary region α at level $L-1$ (b).

level $L-1$ (Fig. 4b) by using Table 1.

The next step is to reclassify the α voxels by applying a 3D butterfly filter in all candidate orientations of these, but applying the (rotated) filter(s) to the data tree at level $L-1$. The filter format and weights are given by the 3x3x3 mask in the y-axis orientation as shown in Fig. 5 (a). For other orientations the filter weights are determined by a rotation and redistribution. This filter is an extension of the 2D one presented in [1]. We consider the 13 orientations a to n as shown in Fig. 5 (b). The weights r and rr of the filter masks are given by $rr = (1-r)/n$ and $r = 0$ for $d \leq 0$, $r = (d-2)/6$ for $2 < d < 8$, and $r = 1$ for $d \geq 8$, with the dissimilarity given by $d = |\mu_1 - \mu_2| / \sqrt{\sigma_1^2 - \sigma_2^2}$. Here n equals the number of weights differing from 0 and r, and μ_i and σ_i^2 are the local mean and variance in the image $I(i,j,k;L)$ of the two most representative classes in the 26-vicinity of the boundary voxel in $I\gamma(i,j,k;L)$.

Table 2. The filter orientations as a function of the voxel in the boundary region.

Voxel in α	Filter orientations						
A	a	b	d	e	j	l	n
B	b	c	e	f	l	m	n
C	d	e	g	h	l	m	n
D	e	f	h	i	j	l	n
E	e	f	h	i	j	l	n
F	d	e	g	h	l	m	n
G	b	c	e	f	l	m	n
H	a	b	d	e	j	l	n

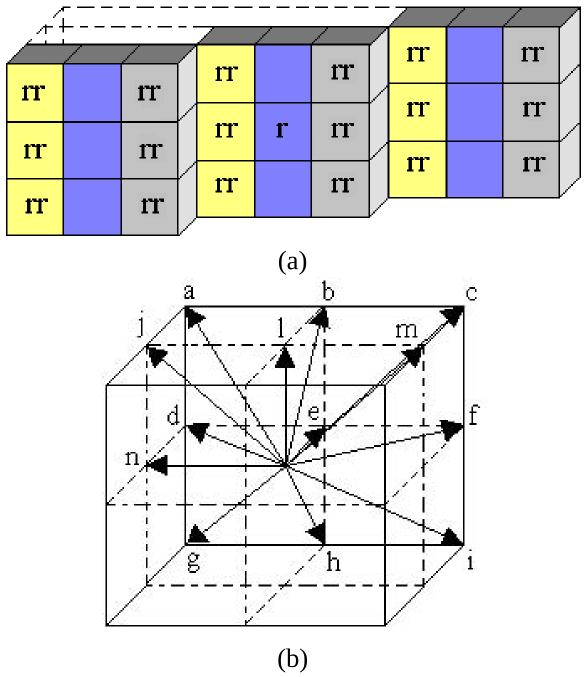

Figure 5. Filter mask coefficients (a) and filter orientations a to n (b).

For each α voxel, all the 13 filter orientatons can be applied simultaneously to the image $I(i,j,k;L-1)$. The left and right halves of the filter masks as shown in Fig. 5 (a) are applied separately, which prevents from smoothing across the boundary, yielding the two convolution sums $S_1(i,j,k;L-1)$ and $S_2(i,j,k;L-1)$ for each orientation and each mask; this results in a set of values S_i, $i = 13N_c(N_c-1)$ with N_c being the number of representative classes. We then calculate all the Euclidean distances between the S_i values and the mean μ_n of the representative classes, i.e., $||\mu_n - S_i||$. Finally, each boundary voxel in $I\gamma(i,j,k;L-1)$ receives the label from the representative class that has the minimum Euclidean distance. We then decrease the value of L by 1 and repeat the procedure until we reach the bottom of the pyramid.

A further speedup is based on the geometry of the masks and the 13 filter orientations. The orientations that point away from a voxel to be reclassified have no or little effect to that voxel. Hence, we consider only the filter orientations that are adjacent to the boundary orientations. Therefore only 7 filter orientations according to Table 2, which leads to a total of $i = 7N_c(N_c-1)$ filter sums, need to be processed.

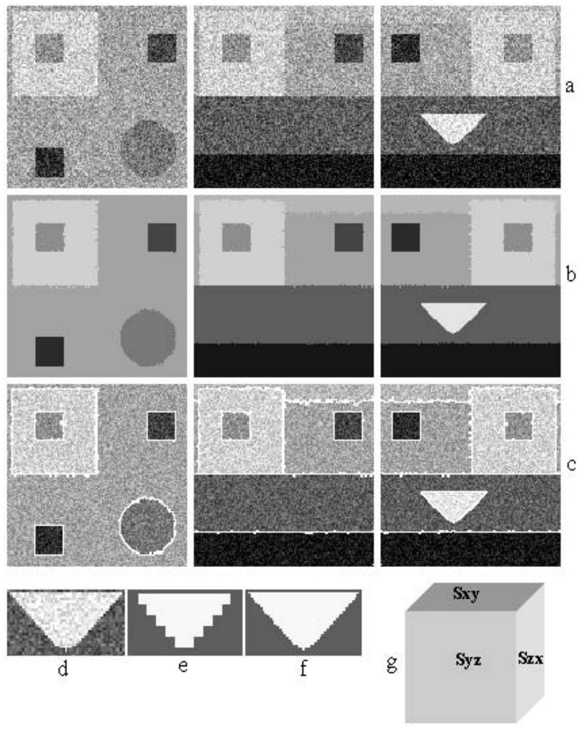

Figure 6. Planar sections through the artificial volume: (a) data with noise, (b) after the segmentation and (c) segmentation boundaries in the original data. The left, middle and right columns correspond to sections *Sxy*, *Syz* and *Szx*, respectively. Sections d, e and f show a central section through the cone in the original data, at tree level $L = 2$ and the reconstruction at level $L = 0$, respectively.

6. Experimental results

The algorithm was tested using different data sets: an artificial volume as well as NMR data. We created a volume of 128×128×128 voxels, the voxel values ranging from 0 to 255, which was composed of 10 regions comprising 4 parallelepipeds that divide the volume in layers, 4 cubes in the different layers, plus 1 sphere and 1 cone. The difference of the voxel values between the different regions is 25. Figure 6 shows three orthogonal sections of the volume: (a) with $\sigma_{\text{noise}} = 25$, (b) the final segmentation and (c) the segmentation boundaries in the noisy volume. Figure 6 also shows a central section through the noisy cone (d), the lower resolution at $L = 2$ (e), and the restored resolution after the segmentation (f). As can be seen, the boundary quality is quite good and only a few voxels at the cone tip have been lost due to the noise.

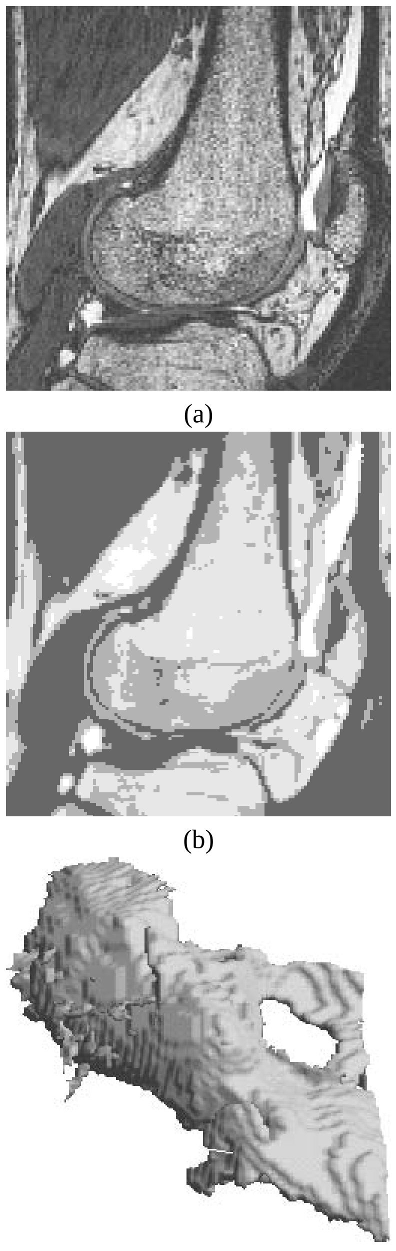

Figure 7. A slice through 3D NMR data (a), the segmentation result (b), and a selected structure visualised in 3D (c).

Results obtained on a NMR data volume of size $128\times140\times88$ are shown in Fig. 7. Figure 7a is one plane in the original data set of a knee, Fig. 7b shows the corresponding segmentation result. The white area to the right is called *suprapatellar bursa.* This "object" was selected by connected component labelling and visualised with OpenGL using Gouraud shading (Fig. 7c). This object shows a hole and we verified by displaying the original data by means of a movie that this hole is part of the data and not caused by the segmentation algorithm (e.g. by eliminating very thin structures). The computation time on an SGI Origin 200 server is 0.700 seconds for the pyramid construction, 0.290 for the clustering and classification and 60.950 for the boundary refinement. For the refinement all 4 available CPUs were used.

The results presented in this paper show that our method is very robust, not only in the presence of outliers, but also in the presence of other types of noise. We can observe that in some occasions inaccuracies appear at the region boundaries, but these are at most 2 voxels wide (Fig. 6). In general, the boundaries are accurate. The biggest limitation of the method is the disappearance of the initial small regions due to the construction of the pyramid. This is a topic for future research.

In order to demonstrate the quality of the algorithm in terms of CPU time, we implemented a very simple alternative algorithm that consists of three steps: (1) Smoothing the data in the volume with an algorithm that preserves boundaries by applying 10 iterations of the Adaptive Gaussian Weighting Filter (AGWF) of size $3\times3\times3$. (2) Finding the classes by a local centroid clustering and the regions by a minimum-distance classification. (3) Removing insignificant regions which contain up to 27 connected voxels. The AGWF was chosen because it is well documented and tested in the literature, both in terms of CPU time and characteristics [e.g. 3]. The second and third steps are similar to those explained in section 3 and 4. The results obtained with this alternative method show completely misclassified boundaries. Furthermore, the application of only one single iteration of the AGWF is a factor of 2.7 slower than applying our entire segmentation algorithm.

7. Conclusions

We presented an algorithm for a fast and unsupervised 3D segmentation which avoids *a priori* knowledge of the classes. However, it is not completely unsupervised because it is necessary to select the level $L_{\max}$ in the octree at which the clustering is done. This level depends on the characteristics of the data volume at hand and can only be determined by means of experiments. Experiments have shown that the strategy of eliminating insignificant regions (≤ 27 voxels) only at the pyramid level $L_{\max}$ proved to be optimal, because the voxels misclassified at that level disappear as well as their propagation through the pyramid. Finally, we have shown that an extension of the Schroeter and Bigün [1] 2D method to 3D is relatively straightforward and not expensive in CPU time even in the case of large volumetric data sets. In other words, a more sophisticated processing of the boundary voxels would improve the quality while still keeping the CPU time acceptable (in the order of a few minutes on an SGI Origin 200, using a single CPU, for the data sets shown here).

8. Acknowledgements

The NMR data visualised in this study are part of the Chapel Hill Volume Rendering Test Data Set (ftp://cs.unc.edu/pub/projects/image) of SoftLab Software Systems Laboratory, Department of Computer Science, University of North Carolina, Chapel Hill, NC, USA.

References

[1] Schroeter, P. and Bigün, J. (1995) Hierarchical image segmentation by multi-dimensional clustering and orientation-adaptive boundary refinement. Pattern Recognition, Vol 28, No 5, pp. 695-709.
[2] Spann, M. and Wilson, R. (1985) A quad-tree approach to image segmentation which combines statistical and spatial information. Pattern Recognition, Vol 18, No 3/4, pp. 257-269.
[3] du Buf, J.M.H. and Campbell, T.G. (1990) A quantitative comparison of edge-preserving smoothing techniques. Signal Processing, Vol 21, pp. 289-301.