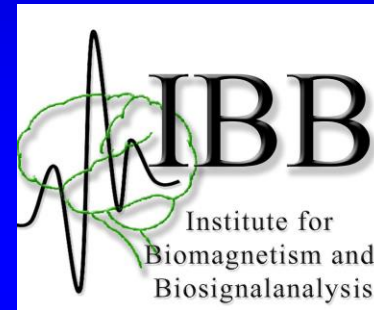


Mathematical Methods for the Segmentation of Medical Images and head model construction



Carsten Wolters

Institut für Biomagnetismus und Biosignalanalyse, Westfälische Wilhelms-Universität Münster

Lecture, April 29, 2025

Structure (segmentation)

- Fuzzy segmentation techniques
- Geometric deformable model reconstructions
- Segmentation scripts

Unsupervised clustering

- Clustering methods classify voxels by simple intensity range partitioning
- A clustering method is called unsupervised, if it automatically determines the intensity ranges for a user-given number C of clusters $\text{Clust}_j, j = 1, \dots, C$
- The ISODATA algorithm is a representative of unsupervised clustering, minimizing the following objective function:

$$J_{ISODATA}(c_j) = \sum_{j=0}^{C-1} \sum_{\mathbf{x} \in \text{Clust}_j} |b(\mathbf{x}) - c_j| \rightarrow \min!$$

Unsupervised clustering: The ISODATA algorithm

Algorithm 1 ISODATA : $(IN \in \text{IMG}(3), C \in \mathbb{N}_+) \rightarrow (OUT \in \text{IMG}(3), c_0^{\text{iso}}, \dots, c_{C-1}^{\text{iso}} \in \{i | i = 0, \dots, 255\})$

$l = 0$, Initialize $c_j^{(0)}$, $\forall j = 0, \dots, C-1$

repeat

$l = l + 1$

for all voxels \mathbf{x} of IN **do**

$min = BIGNUM$

for $j = 0, \dots, C-1$ **do**

if $|b(\mathbf{x}) - c_j^{(l-1)}| < min$ **then**

$k = j, min = |b(\mathbf{x}) - c_j^{(l-1)}|$

end if

end for

Put \mathbf{x} into Clust_k

end for

for $j = 0, \dots, C-1$ **do**

$c_j^l = \left[\frac{1}{|\text{Clust}_j|} \sum_{\mathbf{x} \in \text{Clust}_j} b(\mathbf{x}) \right]$

end for

until $\sum_{j=0}^{C-1} |c_j^{(l)} - c_j^{(l-1)}| < \epsilon$

$c_j^{\text{iso}} = c_j^{(l)}$, $\forall j = 0, \dots, C-1$

Fuzzy C-means segmentation

- **Fuzzy C-Means (FCM) segmentation algorithms do not force a voxel to belong to exclusively one class, but assign a membership value $u_j(\mathbf{x})$ with $\sum_{j=0}^{C-1} u_j(\mathbf{x}) = 1$**
- **FCM thus takes partial volume averaging effects into account.**
- **FCM minimizes the following objective function:**

$$J_{FCM}(u_j, c_j) = \sum_{\mathbf{x}} \sum_{j=0}^{C-1} u_j^2(\mathbf{x}) |b(\mathbf{x}) - c_j|^2 \rightarrow \min!$$

Adaptive Fuzzy C-means segmentation

- The following factors cause intensity inhomogeneities in MRI:
 - Radio frequency excitation field inhomogeneity (McVeigh et al., *Med.Phys.*, 1986)
 - eddy current driven field gradients (Simmons et al., *Magn.Reson.Med.*, 1994)
 - RF penetration and standing wave effects (Bottomley & Andrew, *Phys.Med.Biol.*, 1978)
- Inhomogeneities are well modeled by the product of the original image with a smoothly varying multiplier field $m(\mathbf{x})$ (Dawant et al., *IEEE Trans.Med.Imag.*, 1993)
- Functional of the adaptive fuzzy C-means (AFCM) algorithm:

$$\begin{aligned}
 J_{AFCM}(u_j, c_j, m) = & \sum_{\mathbf{x}} \sum_{j=1}^C u_j^2(\mathbf{x}) |b(\mathbf{x}) - m(\mathbf{x})c_j|^2 \\
 & + \lambda_1 \sum_{\mathbf{x}} (((D_x * m)(\mathbf{x}))^2 + ((D_y * m)(\mathbf{x}))^2 + ((D_z * m)(\mathbf{x}))^2) \\
 & + \lambda_2 \sum_{\mathbf{x}} (((D_{xx} * m)(\mathbf{x}))^2 + ((D_{yy} * m)(\mathbf{x}))^2 + ((D_{zz} * m)(\mathbf{x}))^2 \\
 & \quad + 2((D_{xy} * m)(\mathbf{x}))^2 + 2((D_{xz} * m)(\mathbf{x}))^2 \\
 & \quad + 2((D_{yz} * m)(\mathbf{x}))^2)
 \end{aligned}$$

$$\begin{aligned}
 (D_i * m)(\mathbf{x}) &:= m(\mathbf{x} + \mathbf{e}_i) - m(\mathbf{x}) \\
 (D_{ij} * m)(\mathbf{x}) &:= (D_i * (D_j * m))(\mathbf{x})
 \end{aligned}$$

AFCM: Derivation of the algorithm (centroids \mathbf{c})

$$\begin{aligned}
 J_{AFCM}(u_j, c_j, m) = & \sum_{\mathbf{x}} \sum_{j=1}^C u_j^2(\mathbf{x}) |b(\mathbf{x}) - m(\mathbf{x})c_j|^2 \\
 & + \lambda_1 \sum_{\mathbf{x}} (((D_x * m)(\mathbf{x}))^2 + ((D_y * m)(\mathbf{x}))^2 + ((D_z * m)(\mathbf{x}))^2) \\
 & + \lambda_2 \sum_{\mathbf{x}} (((D_{xx} * m)(\mathbf{x}))^2 + ((D_{yy} * m)(\mathbf{x}))^2 + ((D_{zz} * m)(\mathbf{x}))^2 \\
 & \quad + 2((D_{xy} * m)(\mathbf{x}))^2 + 2((D_{xz} * m)(\mathbf{x}))^2 \\
 & \quad + 2((D_{yz} * m)(\mathbf{x}))^2)
 \end{aligned}$$



$$\frac{\partial J_{AFCM}}{\partial c_i} = \sum_{\mathbf{x}} 2u_i^2(\mathbf{x}) (b(\mathbf{x})m(\mathbf{x}) - m^2(\mathbf{x})c_i)$$



$$c_i = \frac{\sum_{\mathbf{x}} u_i^2(\mathbf{x}) b(\mathbf{x}) m(\mathbf{x})}{\sum_{\mathbf{x}} u_i^2(\mathbf{x}) m^2(\mathbf{x})}$$

AFCM: Derivation of the algorithm (membership function u)

$$J_{\mathbf{x}}(u_j) := \sum_{j=0}^{C-1} u_j^2(\mathbf{x}) |b(\mathbf{x}) - m(\mathbf{x})c_j|^2 + \mu \left(\sum_{j=0}^{C-1} u_j(\mathbf{x}) - 1 \right)$$

$$\frac{\partial J_{\mathbf{x}}}{\partial u_i} = 2u_i(\mathbf{x}) |b(\mathbf{x}) - m(\mathbf{x})c_i|^2 + \mu$$

$$u_i(\mathbf{x}) = -\frac{1}{2}\mu |b(\mathbf{x}) - m(\mathbf{x})c_i|^{-2}$$

$$\begin{aligned} \mu &= -2 |b(\mathbf{x}) - m(\mathbf{x})c_i|^2 u_i(\mathbf{x}) \\ u_i &= 1 - \sum_{\substack{j=0 \\ j \neq i}}^{C-1} u_j \\ &= -2 |b(\mathbf{x}) - m(\mathbf{x})c_i|^2 \left(1 + \mu \sum_{\substack{j=0 \\ j \neq i}}^{C-1} \frac{1}{2} |b(\mathbf{x}) - m(\mathbf{x})c_j|^{-2} \right) \end{aligned}$$

$$\mu = -\frac{2}{\sum_{j=0}^{C-1} |b(\mathbf{x}) - m(\mathbf{x})c_j|^{-2}}$$

$$u_i(\mathbf{x}) = \frac{|b(\mathbf{x}) - m(\mathbf{x})c_i|^{-2}}{\sum_{j=0}^{C-1} |b(\mathbf{x}) - m(\mathbf{x})c_j|^{-2}}$$

AFCM: Derivation of the algorithm (multiplier m)

$$\begin{aligned}
 J_{AFCM}(u_j, c_j, m) = & \sum_{\mathbf{x}} \sum_{j=1}^C u_j^2(\mathbf{x}) |b(\mathbf{x}) - m(\mathbf{x})c_j|^2 \\
 & + \lambda_1 \sum_{\mathbf{x}} (((D_x * m)(\mathbf{x}))^2 + ((D_y * m)(\mathbf{x}))^2 + ((D_z * m)(\mathbf{x}))^2) \\
 & + \lambda_2 \sum_{\mathbf{x}} (((D_{xx} * m)(\mathbf{x}))^2 + ((D_{yy} * m)(\mathbf{x}))^2 + ((D_{zz} * m)(\mathbf{x}))^2 \\
 & + 2((D_{xy} * m)(\mathbf{x}))^2 + 2((D_{xz} * m)(\mathbf{x}))^2 \\
 & + 2((D_{yz} * m)(\mathbf{x}))^2)
 \end{aligned}$$



$$\frac{\partial J_{AFCM}}{\partial m} = \sum_{j=0}^{C-1} -2u_j^2(\mathbf{x})c_j(b(\mathbf{x}) - m(\mathbf{x})c_j) + 2\lambda_1 (H_1 * m)(\mathbf{x}) + 2\lambda_2 (H_2 * m)(\mathbf{x})$$

$$H_1 = \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & -1 & 0 \\ -1 & 6 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

$$H_2 = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 2 & -12 & 2 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 2 & -12 & 2 & 0 \\ 1 & -12 & 42 & -12 & 1 \\ 0 & 0 & -12 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 2 & -12 & 2 & 0 \\ 0 & 0 & 2 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$



$$\begin{aligned}
 Am &= f \\
 A &= w + \lambda_1 H_1 + \lambda_2 H_2 \\
 f(\mathbf{x}) &= b(\mathbf{x}) \sum_{j=0}^{C-1} u_j^2(\mathbf{x})c_j \\
 w(\mathbf{x}) &= \sum_{j=0}^{C-1} u_j^2(\mathbf{x})c_j^2
 \end{aligned}$$

AFCM: The algorithm

Algorithm 2 AFCM : $(MRI \in \text{IMG}(3), C \in \mathbb{N}, \lambda_1, \lambda_2, \epsilon, \delta \in \mathbb{R}) \rightarrow (MRI^{\text{corr}} \in \text{IMG}(3), MRI^{\text{afcm}} \in \text{IMG}(3), c_1^{\text{afcm}}, \dots, c_{C-1}^{\text{afcm}} \in \mathbb{N})$

Initialize: $l = 0, \forall \mathbf{x} : m^{(0)}(\mathbf{x}) = 1, \forall j = 0, \dots, C-1 : c_j^{(0)} = c_j^{\text{iso}}$

repeat

1. Compute new memberships, $\forall j = 0, \dots, C-1, \forall \mathbf{x}$:

$$u_j^{(l+1)}(\mathbf{x}) = \frac{|b(\mathbf{x}) - m^{(l)}(\mathbf{x})c_j^{(l)}|^{-2}}{\sum_{k=0}^{C-1} |b(\mathbf{x}) - m^{(l)}(\mathbf{x})c_k^{(l)}|^{-2}}$$

2. Compute new centroids, $\forall j = 0, \dots, C-1$:

$$c_j^{(l+1)} = \frac{\sum_{\mathbf{x}} \left(u_j^{(l+1)}(\mathbf{x}) \right)^2 m^{(l)}(\mathbf{x}) b(\mathbf{x})}{\sum_{\mathbf{x}} \left(u_j^{(l+1)}(\mathbf{x}) \right)^2 (m^{(l)}(\mathbf{x}))^2}$$

3. Comp. new mult. field (solve up to relative accuracy δ):

$$Am^{(l+1)} = f$$

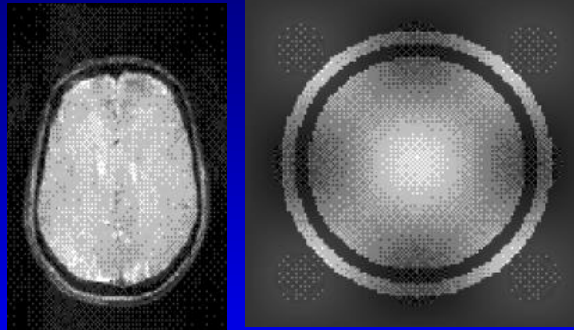
4. $l = l + 1$

until $\max_{\mathbf{x}} \left\{ \max_{j=0, \dots, C-1} |u_j^{(l)}(\mathbf{x}) - u_j^{(l-1)}(\mathbf{x})| \right\} < \epsilon$

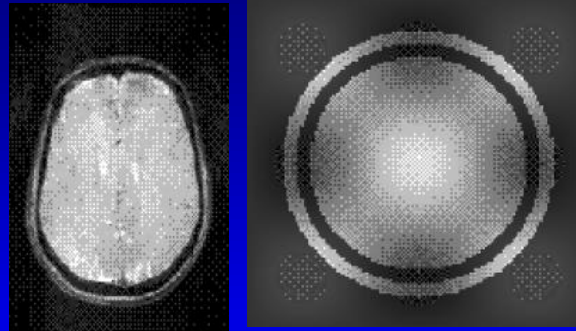
Save $\lfloor b(\mathbf{x})/m^{(l)}(\mathbf{x}) \rfloor$ to $MRI^{\text{corr}}(\mathbf{x})$

Save $\{k | u_k^{(l)}(\mathbf{x}) = \max_{j=0, \dots, C-1} u_j^{(l)}(\mathbf{x})\}$ to $MRI^{\text{afcm}}(\mathbf{x})$

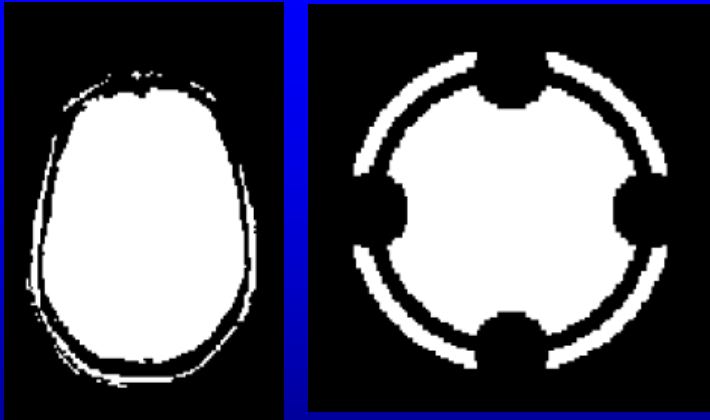
Example (PD-MRI): Results of ISODATA and AFCM [C.Wolters, Vorlesungsskriptum]



Example (PD-MRI): Results of ISODATA and AFCM

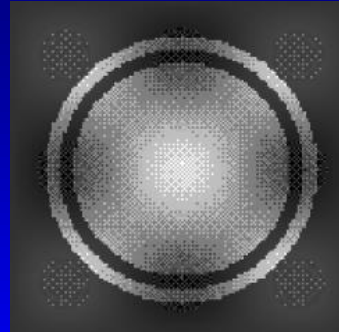
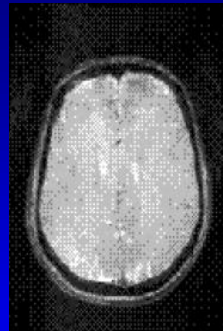


$J_{\text{ISODATA}} \rightarrow \min!$



Segmentation result
ISODATA (C=2)

Example (PD-MRI): Results of ISODATA and AFCM

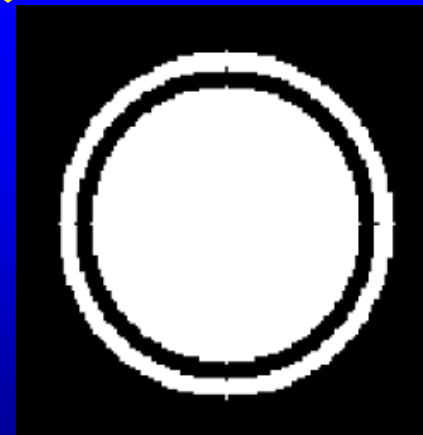


$J_{\text{ISODATA}} \rightarrow \min!$

$J_{\text{AFCM}} \rightarrow \min!$

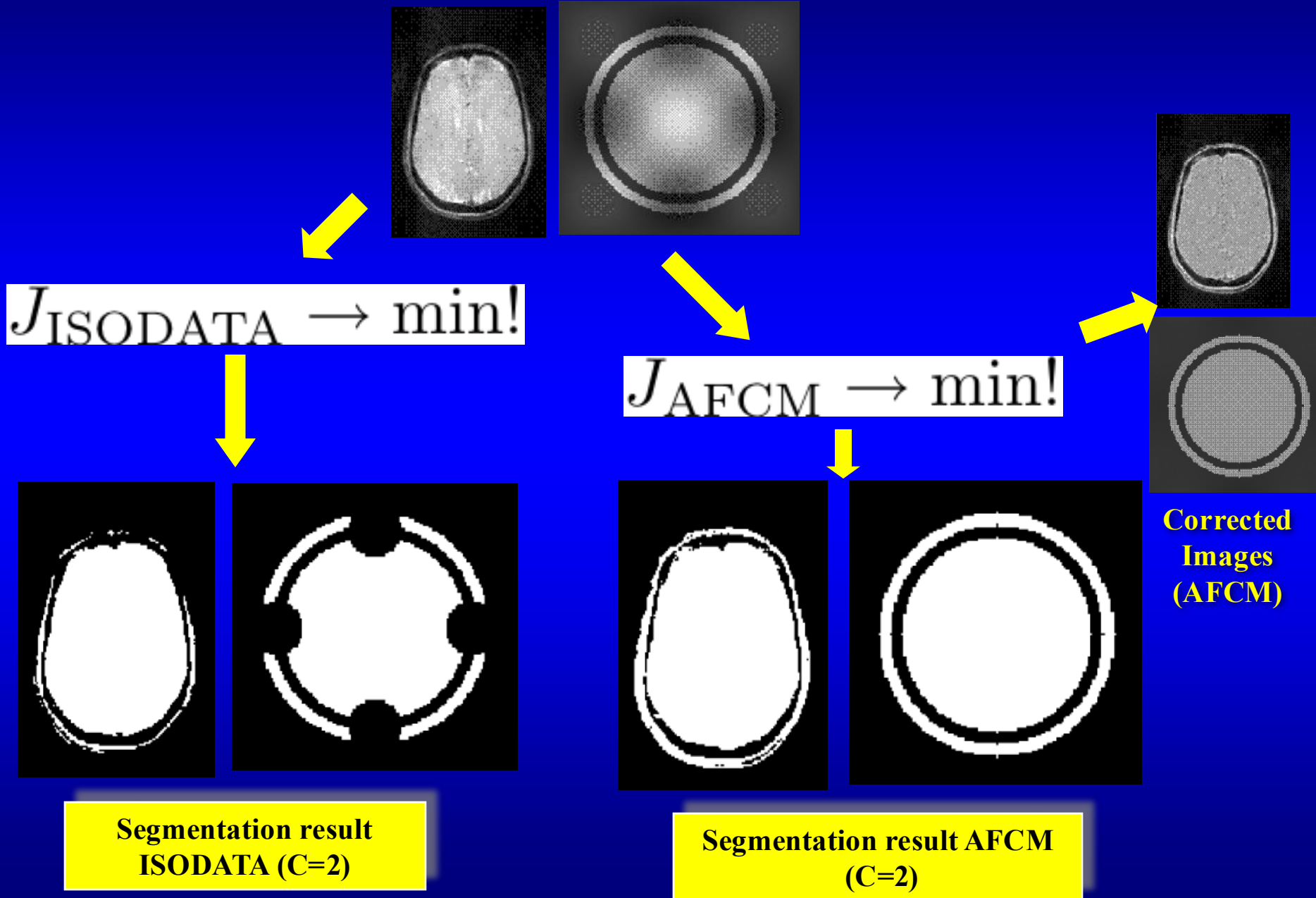


Segmentation result
ISODATA (C=2)

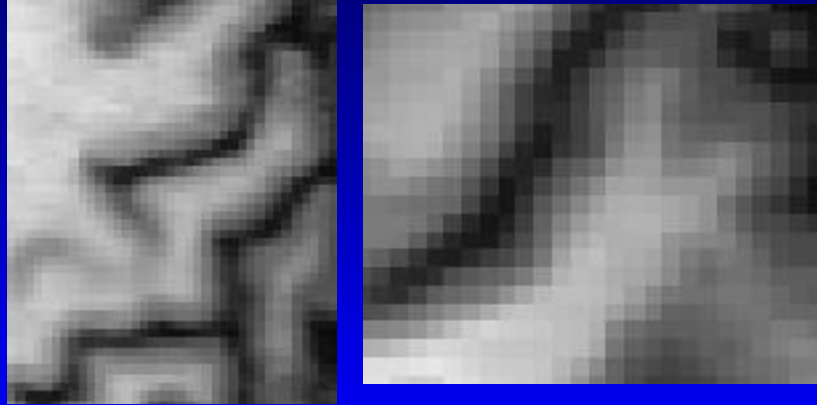


Segmentation result AFCM
(C=2)

Example (PD-MRI): Results of ISODATA and AFCM



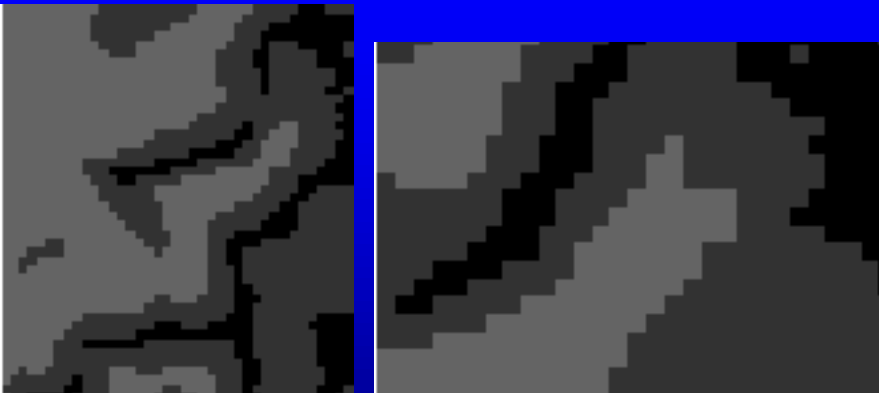
Example (T1-MRI): Results of ISODATA and AFCM



Example (T1-MRI): Results of ISODATA and AFCM

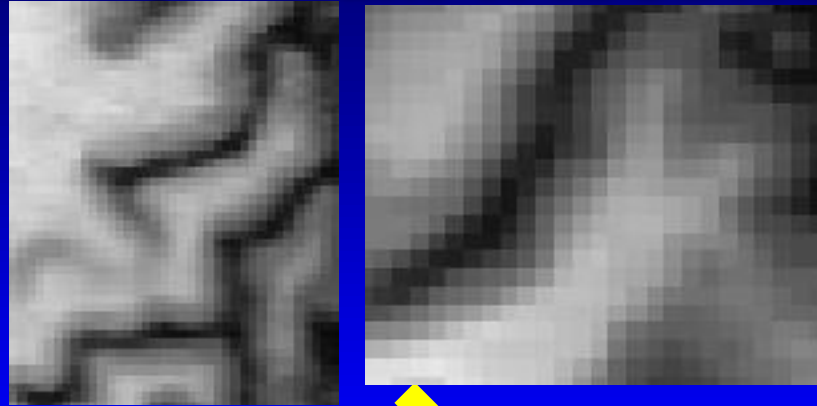


$J_{\text{ISODATA}} \rightarrow \min!$



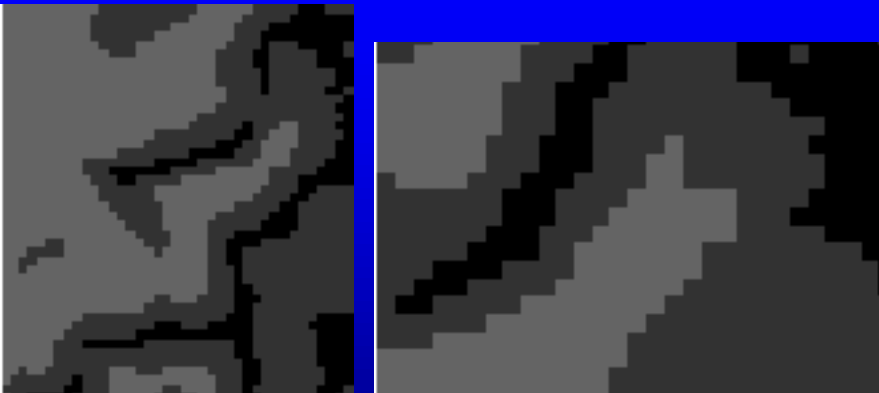
Segmentation result
ISODATA (C=3)

Example (T1-MRI): Results of ISODATA and AFCM

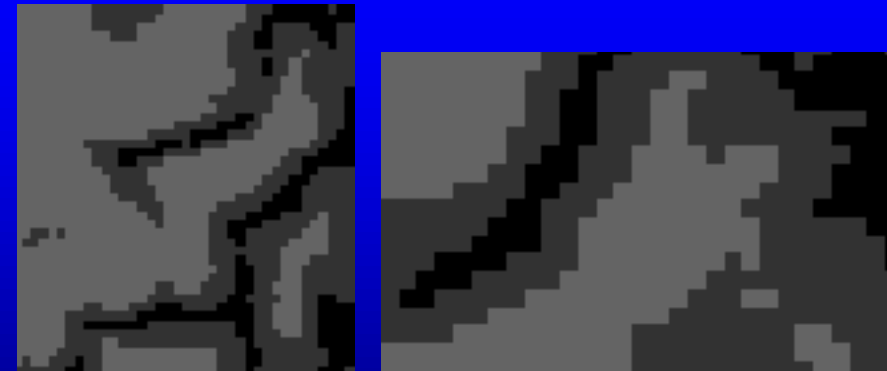


$J_{\text{ISODATA}} \rightarrow \min!$

$J_{\text{AFCM}} \rightarrow \min!$



Segmentation result
ISODATA (C=3)



Segmentation result AFCM
(C=3)

Cortical segmentation: Differences between ISODATA and AFCM

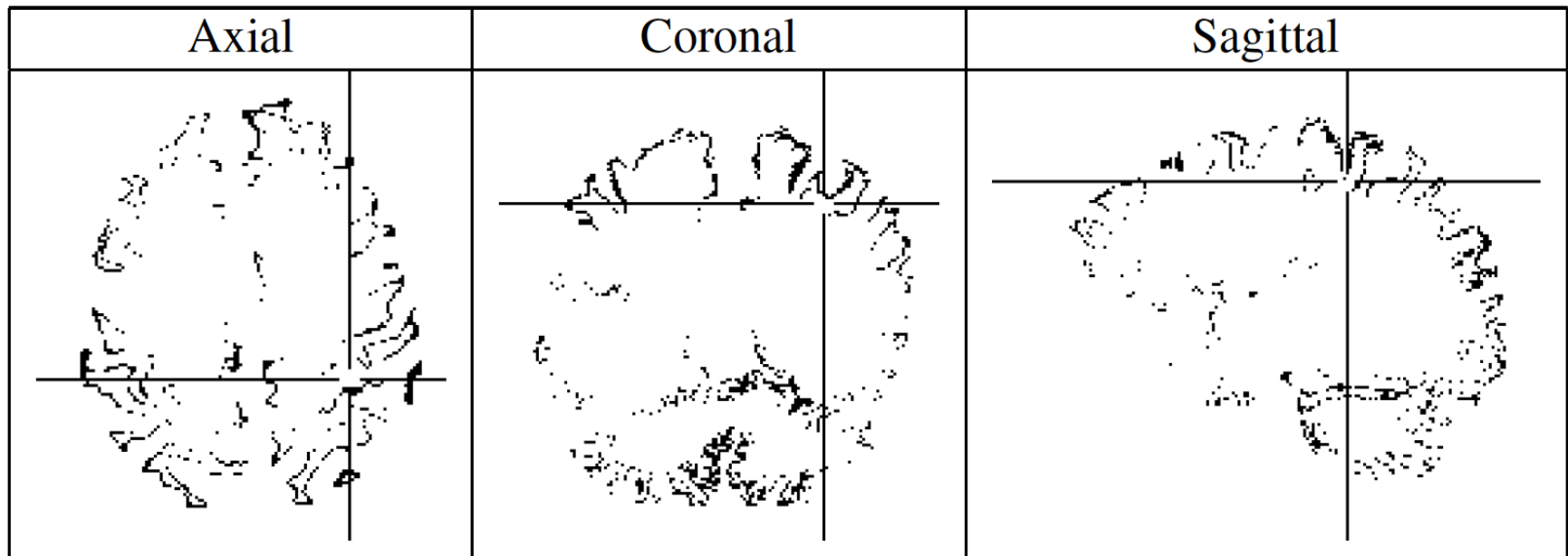


Figure 4.18: *Difference between the cortical layer thicknesses of the ISODATA and the AFCM segmentation results.*

Importance of preconditioned conjugate gradient (PCG) solver for AFCM

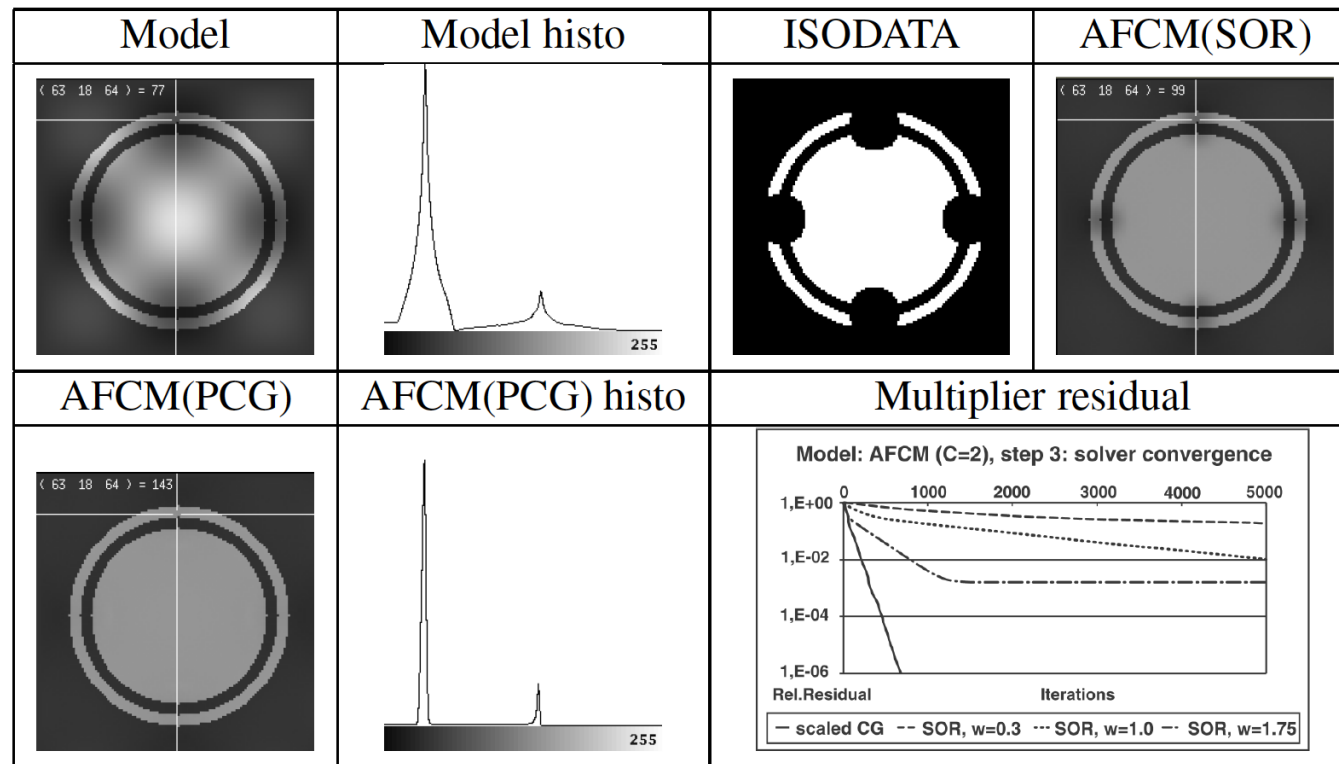


Figure 4.13: Comparison of segmentation results of Algorithms 3 and 4 for a sphere model with sinusoidal intensity inhomogeneities. Top row: The model (left), its histogram (middle, left), the classification result of ISODATA (middle, right) and the AFCM (SOR) intensity corrected model (right). Bottom row: AFCM (PCG) intensity corrected model (left), its histogram (middle) and the relative residuals $d_{(j)}^{(1)}$ (see (4.13)) for the first 5000 inner iterations of the PCG and the SOR (for various ω) solver.

Importance of preconditioned conjugate gradient (PCG) solver for AFCM

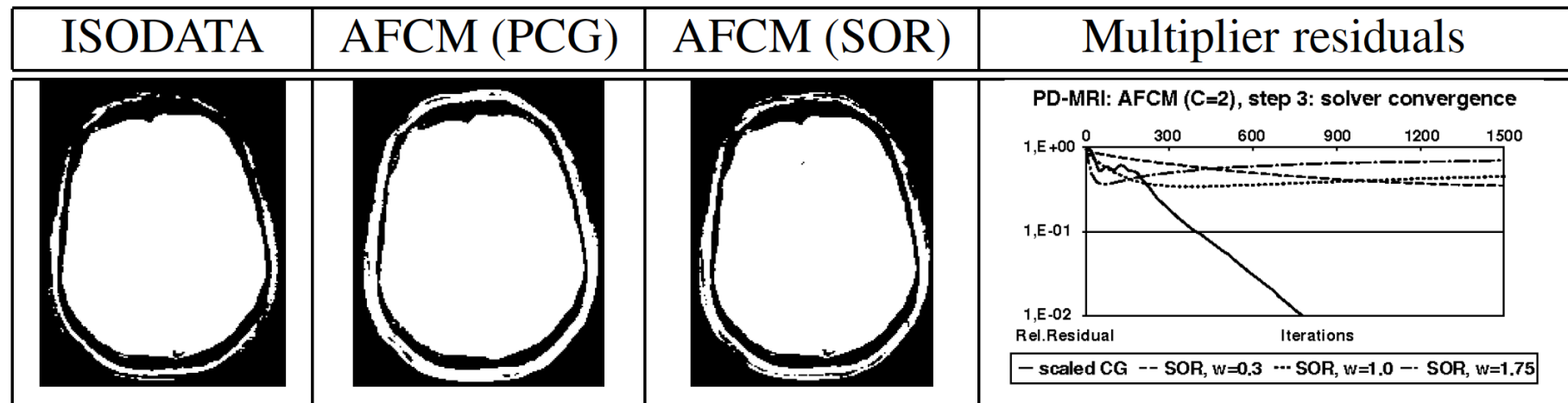


Figure 4.14: *PD-MRI: Comparison of segmentation results of Algorithms 3 (left) and 4 with scaled CG solver (left middle) and with SOR solver (right middle) and the relative residuals $d_{(j)}^{(1)}$ (see (4.13)) for the first 1500 inner iterations of PCG and SOR (for various ω) solver (right).*

Structure

- **Fuzzy segmentation techniques**
- **Geometric deformable model reconstructions**
- **Segmentation scripts**

Geometric deformable models

- Geometric deformable models **are based on the theory of front evolution and are implemented using the level set numerical method** (Osher & Sethian, 1988; Sethian, 1999)
- The level set technique represents the boundary contour** $\Gamma := \partial\Omega$ **of a domain Ω implicitly as the zero level of a level set function**

$$\Phi(x, t) : \mathbb{R}^d \times \mathbb{R}^+ \rightarrow \mathbb{R} \quad :$$
- In a velocity field V , each point evolves via the ODE**

$$\frac{dx}{dt} = V(x, t)$$

Geometric deformable models

- **For any parametric representation** $\Gamma(t) = \{x(S, t) \mid S \in \Sigma\}$
and due to the definition of the level set function $\Phi(x(S, t), t) = 0$

we find $\frac{d}{dt}\Phi(x(S, t), t) = 0$

- **By the chain rule:**

$$\begin{aligned} 0 &= \frac{d}{dt}\Phi(x(S, t), t) = \nabla\Phi(x(S, t), t) \frac{dx}{dt}(S, t) + \frac{\partial\Phi}{\partial t}(x(S, t), t) \\ &= \nabla\Phi(x(S, t), t) \cdot V(S, t) + \frac{\partial\Phi}{\partial t}(x(S, t), t) \end{aligned}$$

- **We thus obtain:**

$$\frac{\partial\Phi}{\partial t} = -\nabla\Phi \cdot V$$

Geometric deformable models

- For any parametric representation $\Gamma(t) = \{x(S, t) \mid S \in \Sigma\}$ and due to the definition of the level set function $\Phi(x(S, t), t) = 0$ we additionally find $\frac{d}{dS}\Phi(x(S, t), t) = 0$
- By the chain rule:

$$0 = \frac{d}{dS}\Phi(x(S, t), t) = \nabla\Phi(x(S, t), t) \frac{dx}{dS}(S, t)$$

- Because $\frac{\partial x}{\partial S}$ is a tangential direction, $\nabla\Phi$ has to be a radial direction:
$$n = \frac{\nabla\Phi}{|\nabla\Phi|}$$

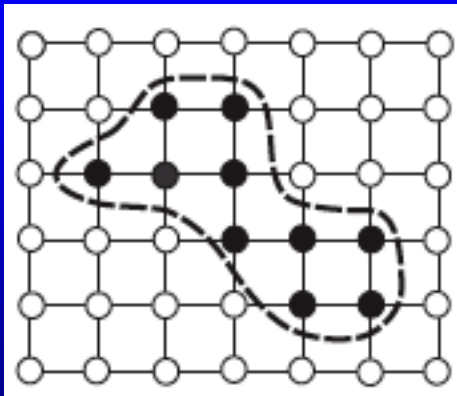
Geometric deformable models

- This finally leads to the so-called **Hamilton-Jacobi equation**

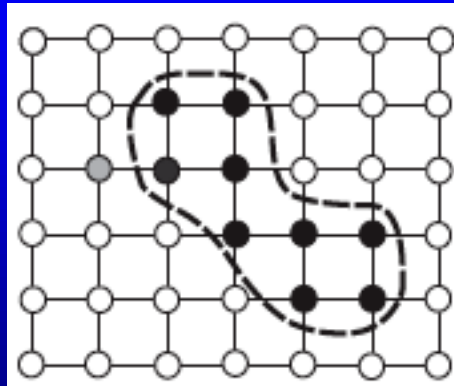
$$\begin{aligned} 0 &= \frac{\partial \Phi}{\partial t} + V \cdot \nabla \Phi = \frac{\partial \Phi}{\partial t} + v \frac{\nabla \Phi}{|\nabla \Phi|} \cdot \nabla \Phi \\ &= \frac{\partial \Phi}{\partial t} + v |\nabla \Phi| \end{aligned}$$

Example: Topology-preserving Geometric Deformable Model (TGDM)

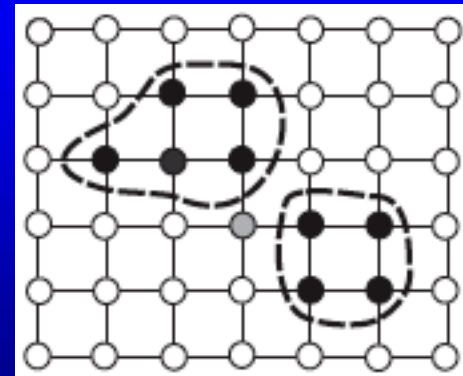
- For the reconstruction of the gray matter/white matter boundary of the human brain, a combination of an expansion/contraction speed R and the mean curvature κ was proposed for the normal velocity: $-v(x,t) = \omega_R R(x) + \omega_\kappa \kappa(x,t)$
- Choice of expansion/contraction speed: $R(x) = 2u_{wm}(x) - 1$
- TGDM: “Simple point” constraint that prevents topology-change



Original contour



Pass over simple point



Split at non-simple point

Geometric deformable models:

The TGDM algorithm

[Han, Xu and Prince, *IEEE Trans.Pattern Anal.Mach.Intell.*, 25, 2003]

[Han, Pham, Tosun, Rettmann, Xu & Prince, *NeuroImage*, 23, 2004]

Marching cubes: <http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html>

Algorithm 3 Topology-preserving Level Set Method

- **Initialize:** $m = 0$, $t_0 = 0$, $\Phi(\cdot, 0)$ signed distance function of the initial contour, $B(x_i) = \begin{cases} 1 & \text{if } \Phi(x_i, 0) \leq 0 \\ 0 & \text{otherwise} \end{cases}$

repeat

- **Built the narrow band:** Find all grid points y_i such that $|\Phi(y_i, t_m)| < W_{nb}$ with W_{nb} being the user-defined narrow band width.

for all narrow band points y_i **do**

- Compute $\Phi_{\text{temp}}(y_i) = \Phi(y_i, t_m) + \Delta t \Delta \Phi(y_i, t_m)$ with $\Delta \Phi(y_i, t_m)$ being an upwind finite difference approximation to the right-hand side of the Hamilton-Jacobi equation.
- if** $\text{sign}(\Phi_{\text{temp}}(y_i)) = \text{sign}(\Phi(y_i, t_m))$ **then**
 - $\Phi(y_i, t_{m+1}) = \Phi_{\text{temp}}(y_i)$
- else**
 - if** y_i is a simple point **then**
 - $\Phi(y_i, t_{m+1}) = \Phi_{\text{temp}}(y_i)$
 - $B(y_i) = (B(y_i) + 1) \bmod 2$
 - else**
 - To preserve the topology, do not allow the sign change: $\Phi(y_i, t_{m+1}) = \varepsilon \text{sign}(\Phi(y_i, t_m))$ with a small positive number ε .
- end if**
- end if**

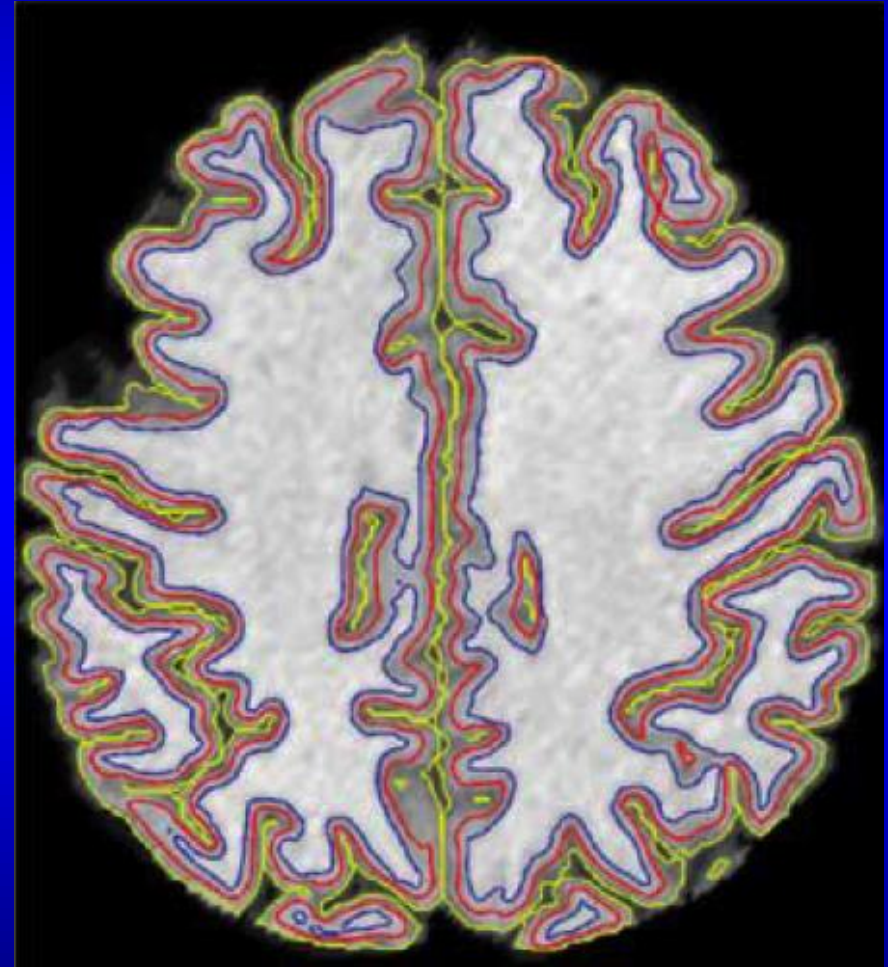
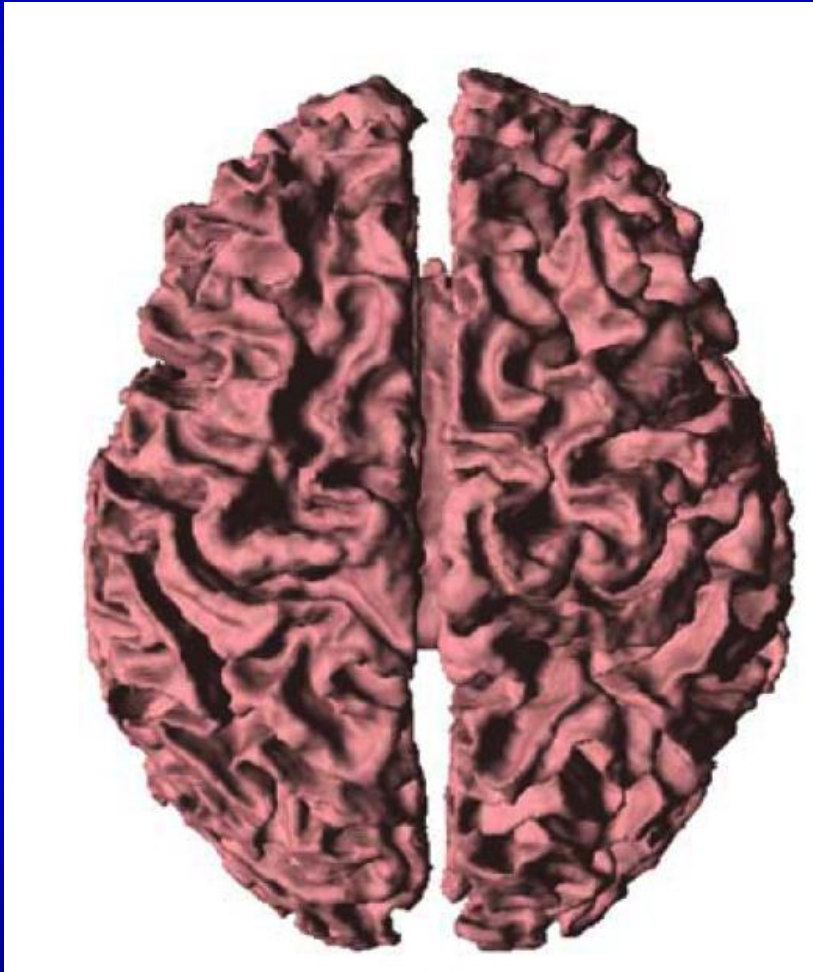
end for

- **Reinitialize:** If the zero level set of $\Phi(\cdot, t_{m+1})$ is near the boundary of the current narrow band, reinitialize $\Phi(\cdot, t_{m+1})$ to be the signed distance function of its zero level set.
- $m = m + 1$

until Zero level set is no longer moving

- Use *connectivity consistent marching cubes* algorithm to produce an explicit representation of the final contour from the zero level set.

Example: Topology-preserving Geometric Deformable Model (TGDM)



Inner cortical surface for $\omega_R=1$ and $\omega_K=-0.02$

Structure

- **Fuzzy segmentation techniques**
- **Geometric deformable model reconstructions**
- **Segmentation scripts**

Example of a segmentation script:
We thus now furthermore need to learn about basic components such as, e.g., morphological operations

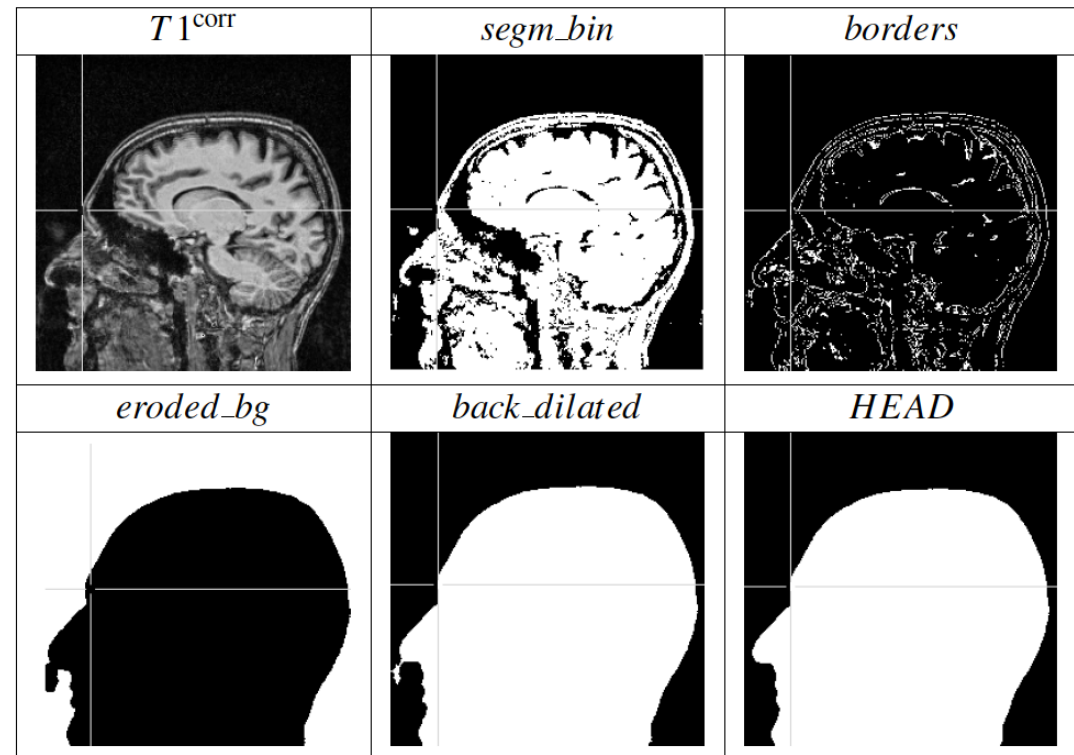
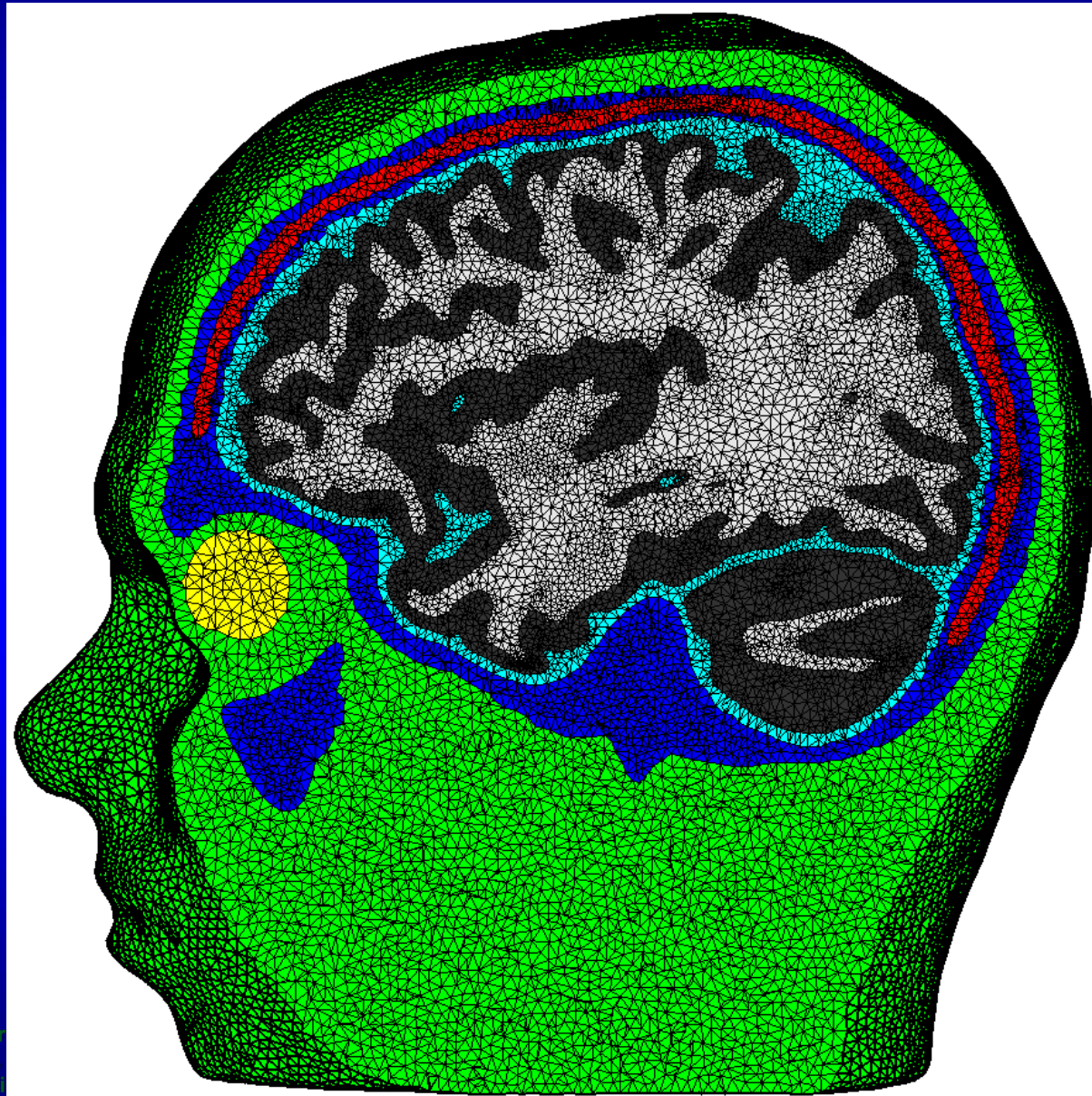


Figure 4.16: Intermediate results for the generation of the head mask.

Script 4.7.1. HEAD : $(MRI \in \mathcal{MR}I, C \in \mathbb{N}, \lambda_1, \lambda_2, \varepsilon, \delta \in \mathbb{R}, d_1, d_2, d_3 \in \mathbb{N}_0, OSS \in \mathcal{BI}) \rightarrow (HEAD, SCALP \in \mathcal{BI})$

1. $(MRI^{corr}, MRI^{afcm}, c_j^{afcm}) = \mathbf{AFCM}(MRI, C, \lambda_1, \lambda_2, \varepsilon, \delta)$ /* class. */
2. $segm_bin = \mathbf{BINARIZE}(MRI^{afcm}, g_1(C), g_2(C))$ /* fusion */
3. $borders = \mathbf{BORDER}(\mathbf{BIGGESTCOMP}(segm_bin))$ /* closing holes */
4. $eroded_bg = \mathbf{BIGGESTCOMP}(\mathbf{BINARIZE}(\mathbf{DIST}(borders), d_1, \infty))$
5. $back_dilated = \mathbf{BINARIZE}(\mathbf{DIST}(eroded_bg), d_1, \infty)$
6. $HEAD = \mathbf{SMOOTHING}(back_dilated, d_2)$ /* smoothing */
7. $SCALP = \mathbf{AND}(HEAD, \mathbf{DILATE}(OSS, d_3))$ /* for multi-tissue model */

6 compartment FE head model





Thank you for your attention!

Structure

- **Fuzzy segmentation techniques**
- **Geometric deformable model reconstructions**
- **Basic definitions and operations on images and meshes**
- **Segmentation scripts**

Example of a segmentation script:
We thus now furthermore need to learn about basic components such as, e.g., morphological operations

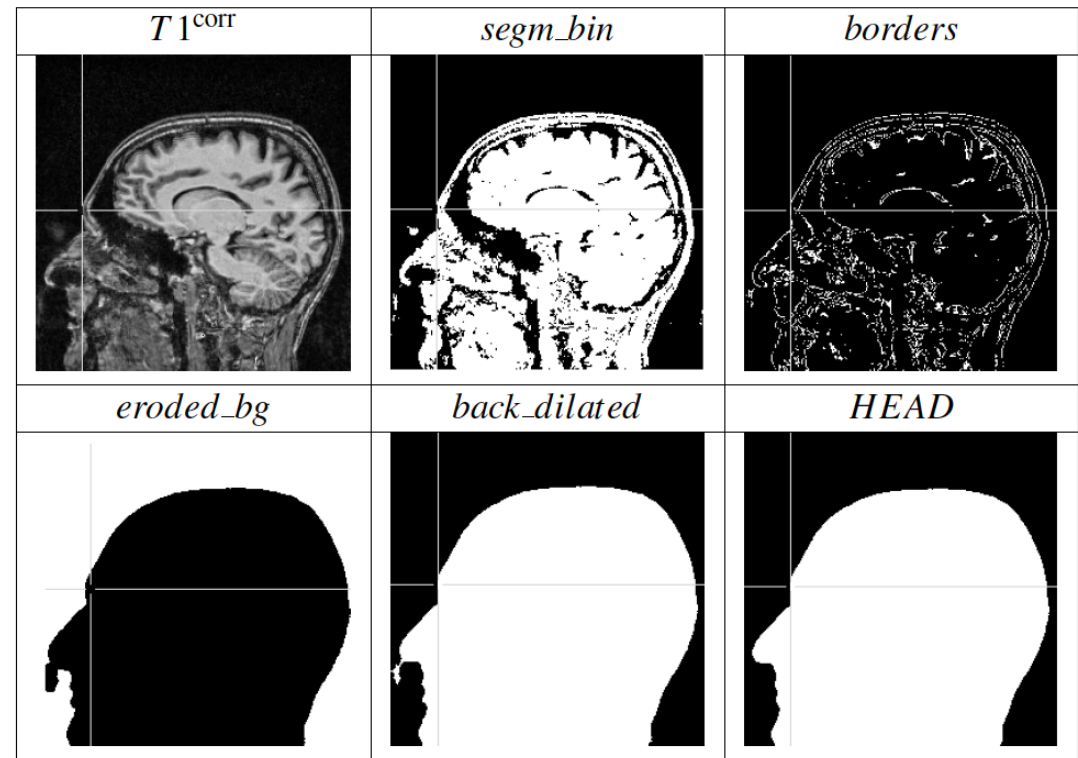


Figure 4.16: Intermediate results for the generation of the head mask.

Script 4.7.1. HEAD : $(MRI \in \mathcal{MR}I, C \in \mathbb{N}, \lambda_1, \lambda_2, \varepsilon, \delta \in \mathbb{R}, d_1, d_2, d_3 \in \mathbb{N}_0, OSS \in \mathcal{BI}) \rightarrow (HEAD, SCALP \in \mathcal{BI})$

1. $(MRI^{corr}, MRI^{afcm}, c_j^{afcm}) = \mathbf{AFCM}(MRI, C, \lambda_1, \lambda_2, \varepsilon, \delta)$ /* class. */
2. $segm_bin = \mathbf{BINARIZE}(MRI^{afcm}, g_1(C), g_2(C))$ /* fusion */
3. $borders = \mathbf{BORDER}(\mathbf{BIGGESTCOMP}(segm_bin))$ /* closing holes */
4. $eroded_bg = \mathbf{BIGGESTCOMP}(\mathbf{BINARIZE}(\mathbf{DIST}(borders), d_1, \infty))$
5. $back_dilated = \mathbf{BINARIZE}(\mathbf{DIST}(eroded_bg), d_1, \infty)$
6. $HEAD = \mathbf{SMOOTHING}(back_dilated, d_2)$ /* smoothing */
7. $SCALP = \mathbf{AND}(HEAD, \mathbf{DILATE}(OSS, d_3))$ /* for multi-tissue model */

Literature

[C.Wolters, *Vorlesungsskriptum*]

[180] G. Lohmann. *Volumetric Image Analysis*. John Wiley & Sons, Chicester, 1998.

Basic definitions

4.4.1 Definitions of images and meshes

A three dimensional image is composed of a stack of two-dimensional *slices*, which are indexed from top (slice 0) to bottom (the orientations are already given with respect to the measured object, i.e., the human head). Each two dimensional slice is discretized into *rows* (front to back) and *columns* (left to right), resulting in the *image lattice*

$$L = \{(s, r, c) \in \mathbb{N}_0^3 | (0 \leq s < nslices), (0 \leq r < nrows), (0 \leq c < ncolumns)\} .$$

A *feature space* G represents the set of values of an *image intensity function* $I : L \rightarrow G$. The feature space is restricted here to the one-dimensional case, so that each *image lattice address* (s, r, c) is assigned a one-dimensional intensity $I(s, r, c)$. Cartesian coordinates $\mathbf{x}_{(s,r,c)} \in \mathbb{R}^3$ of the lattice address (s, r, c) are calculated by multiplying each component with its discretization size, denoted by $ssize \in \mathbb{R}$, $rsize \in \mathbb{R}$ and $csize \in \mathbb{R}$. A cubic volume element with barycentre $\mathbf{x}_{(s,r,c)}$, cubic edge lengths $ssize$, $rsize$ and $csize$ and the constant intensity $I(s, r, c)$ throughout its volume will be called a *voxel*. We denote the number of voxels with N_x , i.e., $N_x = nslices \cdot nrows \cdot ncolumns$. In order to simplify indices, a voxel is identified with its image lattice address, so that denotations like $\mathbf{x} \in L$, $\mathbf{x} = (s, r, c)$ and $I(\mathbf{x}) := I(s, r, c)$ are used.

Basic definitions

Definition 4.4.1. A “general 3D image” $GI = (L, G, I)$ consists of the image lattice $L \subseteq \mathbb{N}_0^3$, a feature space $G \subseteq \mathbb{R}$, also called “gray code” and an intensity function $I : L \rightarrow G$. The set of all 3D images GI will be denoted by $\mathcal{G}I$.

An image, resulting from a restriction to the set of integer gray codes $G = G_{MRI} = \{i | i = 0, \dots, 255\}$ will be denoted by MRI with the corresponding class $\mathcal{M}RI$. “Black” and “white” are used for the gray codes 0 and 255, respectively, while the intermediate values represent various shades of gray. An image, resulting from a restriction to the triplet $(L, G_{0,1}, I)$ with the set of boolean gray codes $G_{0,1} = \{i | i = 0, 1\}$ will be called a “binary image” or a “binary mask” and will be denoted by BI with the corresponding class $\mathcal{B}I$. A voxel \mathbf{x} with intensity $I(\mathbf{x}) = 0$ will be called a “background” or “black” voxel and with intensity $I(\mathbf{x}) = 1$ a “foreground” or “white” voxel. An image, resulting from a restriction to the triplet $(L, G_{\mathbb{N}_0}, I)$ with $G_{\mathbb{N}_0} = \{i | i \in \mathbb{N}_0\}$ will be called a “positive integer image” PI with the corresponding class $\mathcal{P}I$. An image, resulting from a restriction to the triplet (L, G_C, I) with $G_C = \{i | i = 0, \dots, C - 1\}$ will be called a “C-class positive integer image” CPI with the corresponding class $\mathcal{C}PI$.

Basic definitions

Definition 4.4.2. *The image histogram $h_{MRI} : G_{MRI} \rightarrow G_{N_x^0}$ with $G_{N_x^0} = \{i | i = 0, \dots, N_x\}$ counts the occurrence of each gray code in $MRI \in \mathcal{M} \mathcal{R} I$, i.e., $h_{MRI}(g) = |\{\mathbf{x} | \mathbf{x} \in \mathcal{L} \wedge I(\mathbf{x}) = g\}|$. A 2D histogram $h_{MRI_1, MRI_2} : G_{MRI} \times G_{MRI} \rightarrow G_{N_x^0}$ for $MRI_1, MRI_2 \in \mathcal{M} \mathcal{R} I$ with intensity functions I_1 and I_2 , resp., is defined through*

$$h_{MRI_1, MRI_2}(g_1, g_2) = |\{\mathbf{x} | \mathbf{x} \in \mathcal{L} \wedge I_1(\mathbf{x}) = g_1 \wedge I_2(\mathbf{x}) = g_2\}|.$$

Basic definitions

In the following, cuts through a 3D image with fixed value for s will be called *axial*, cuts with fixed value for r *coronal* and with fixed value for c *sagittal*. Some basic definitions about neighborhood, adjacency and connectivity in an image lattice L have to be made, formulated in

Basic definitions

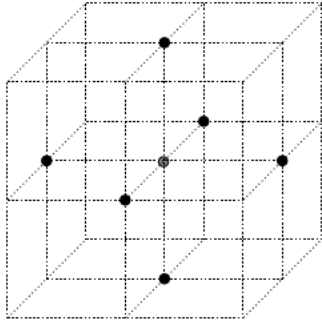
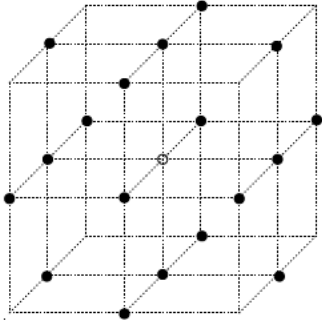
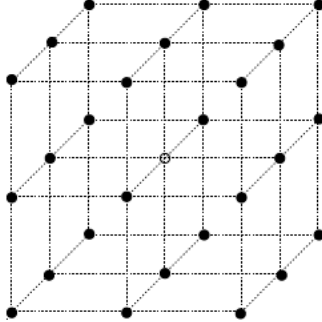
6-neighborhood	18-neighborhood	26-neighborhood
		

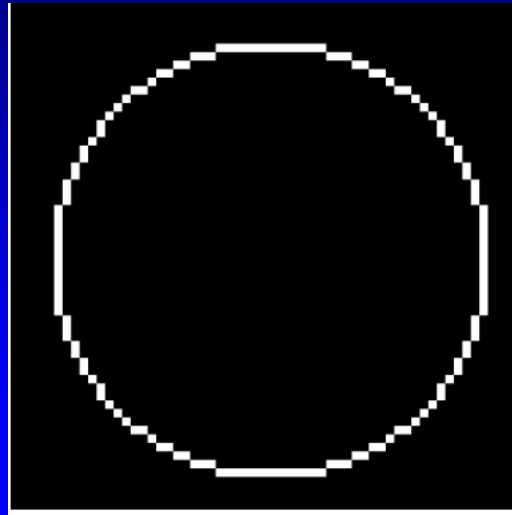
Figure 4.5: *Neighborhoods in 3D from Lohmann [180].*

Definition 4.4.3. *The set of the 6 neighbors of a voxel \mathbf{x} in the 3D image lattice L , differing (by one) in at most one coordinate is called the 6-neighborhood, the set of those, differing in at most two coordinates is called the 18-neighborhood and the set of voxels with at most three different coordinates is called the 26-neighborhood.*

Two voxels are said to be n -adjacent, iff both are n -neighbors of one another.

A set of voxels is called n -connected if for any two of them, a sequence of voxels $(\mathbf{x}_{i_1}, \dots, \mathbf{x}_{i_k})$ can be found that are pairwise (i.e., $(\mathbf{x}_{i_j}, \mathbf{x}_{i_{j+1}}), \forall j = \{1, \dots, k\})$ n -adjacent.

Basic definitions



The gray code of a voxel did not play a role in the above definition. In order to make sure for $BI \in \mathcal{B} I$, that a closed m -connected surface of foreground voxels completely encloses a k -connected background component and thus divides the background into two k -connected components (the one interior to the closed foreground surface and the one exterior to it, see Figure above as an illustration in 2D), the restriction to the (m, k) pairs $(26, 6)$, $(6, 26)$, $(18, 6)$ and $(6, 18)$ has to be made [180].

In the following, $(m, k) = (26, 6)$ will be used for all $BI \in \mathcal{B} I$:

Basic definitions

Definition 4.4.4. For $BI \in \mathcal{B} I$, two foreground voxels are called adjacent if they are 26-adjacent and two background voxels are called adjacent if they are 6-adjacent. A set of foreground voxels is called connected if it is 26-connected and a set of background voxels is called connected if it is 6-connected.

Definition 4.4.5. For $BI \in \mathcal{B} I$, a foreground voxel is called border voxel, iff at least one of its 6-neighbors is a background voxel.

Basic definitions

The Euler number is an important measure for several topological characteristics [180], so that it will be defined in

Definition 4.4.6. *The Euler number of a binary object $IN \in \mathcal{B} I$ is defined as the number of connected components plus the number of cavities, i.e., totally enclosed components of the background, minus the number of handles (imagine a handle of a tea-cub).*

A binary object is *topologically equivalent to a sphere*, if it has the Euler number 1

Basic definitions

Definition 4.4.7. *Two markers, STOP and PASS, for an $MRI \in \mathcal{M} \mathcal{R} I$ are introduced, offering a possibility to influence image segmentation independently of the image intensity. Each marker can be seen as a binary image, defined on the image lattice of its corresponding MRI, so that each lattice point \mathbf{x} of the "extended" image MRI_{ext} has three attributes, a gray value $I(\mathbf{x})$ and a binary value (0 for "not set" or deleted and 1 for "set") for each of its two markers $STOP(\mathbf{x})$ and $PASS(\mathbf{x})$. For each \mathbf{x} , the setting of one of the markers is automatically followed by the deletion of the other, i.e., the combination $STOP(\mathbf{x}) = PASS(\mathbf{x}) = 1$ is impossible. The class of extended images MRI_{ext} will be denoted by $\mathcal{M} \mathcal{R} I_{ext}$*

Basic definitions

The rest of this subsection is concerned with structures for geometrical models on irregular grids. A geometrical model is composed of a tuple (\mathbf{v}, Δ) of *vertices* \mathbf{v} and *primitives* Δ . To make an example, such a model could be a triangulated surface in 3D, i.e. a 2D manifold with \mathbf{v}_i the 3D Cartesian coordinates of the triangle node i and Δ_i the description of the i th triangle by means of its three node points \mathbf{v}_{i_1} , \mathbf{v}_{i_2} and \mathbf{v}_{i_3} .

Basic definitions

Definition 4.4.8. A mesh $= \{(\mathbf{v}_i, \Delta_j), i = 1, \dots, N_v, j = 1, \dots, N_p\}$ consists of a list of N_v vertices $\mathbf{v}_i \in \mathbb{R}^3$ and a list of N_p primitives $\Delta_j \in \Delta$, where each primitive consists of a subset of vertices of the mesh, i.e., $\Delta = \{\mathbf{v}_{i_k} | k = 1, \dots, \Delta_v\}$. The number of vertices Δ_v , a primitive is consisting of, depends on its type. A triangle comprises $\Delta_v = 3$, a tetrahedron $\Delta_v = 4$ and a cube $\Delta_v = 8$ vertices. The class of all meshes will be denoted by \mathcal{MESH} .

Basic definitions

Definition 4.4.9. *The Euler characteristic of a triang. mesh is defined as the number of vertices N_v plus the number of faces N_p minus the number of edges N_e .*

A triangulated mesh is *topologically equivalent to a sphere*, if it has the Euler number 2 (Agoston, 1976 [1]).

Basic operations on images

4.4.2 Operations on images and meshes

Operations on 3D images

Within this subsection, the basic operations on 3D images will be presented by definitions and short explanations which are frequently used later in this chapter. Refer to [180] for a more detailed description. The first simple operation is

Basic operations on images

Operation 4.4.10. *By means of*

$$OUT = AND(IN1, IN2),$$

each voxel of $IN1 \in \mathcal{G} I$ will be taken over to $OUT \in \mathcal{G} I$, if it is a foreground voxel in $IN2 \in \mathcal{B} I$. By means of

$$OUT = OR(IN1, IN2),$$

each voxel of $IN1 \in \mathcal{G} I$ will be taken over to $OUT \in \mathcal{G} I$, if its intensity is not equal to 0 or if it is a foreground voxel in $IN2 \in \mathcal{B} I$.

If $IN1$ is restricted to a binary mask, the operation AND (OR) leads to a binary mask OUT which is the minimum (maximum) of both input masks.

Basic operations on images

Definition 4.4.5. For $BI \in \mathcal{B} I$, a foreground voxel is called border voxel, iff at least one of its 6-neighbors is a background voxel.

Remember Definition 4.4.5 for the following operation:

Operation 4.4.11. By means of

$$OUT = BORDER(IN),$$

border voxels of $IN \in \mathcal{B} I$ are detected and written out as the new foreground of $OUT \in \mathcal{B} I$.

Within the algorithm, which realizes the above operation, background voxels will be searched in the 6-neighborhood of each foreground voxel of IN . As soon as at least one is found, the examined foreground voxel is successfully detected as a border voxel.

Basic operations on images

Operation 4.4.12. *By means of*

$$OUT = LABEL(IN),$$

connected foreground components of $IN \in \mathcal{B} I$ are detected and labeled in order. The result is written to $OUT \in \mathcal{P} I$. The operation

$$OUT = SELBIG(IN)$$

selects the connected component with the most frequent label value of $IN \in \mathcal{P} I$. All voxels of that component become foreground voxels in $OUT \in \mathcal{B} I$. The combination

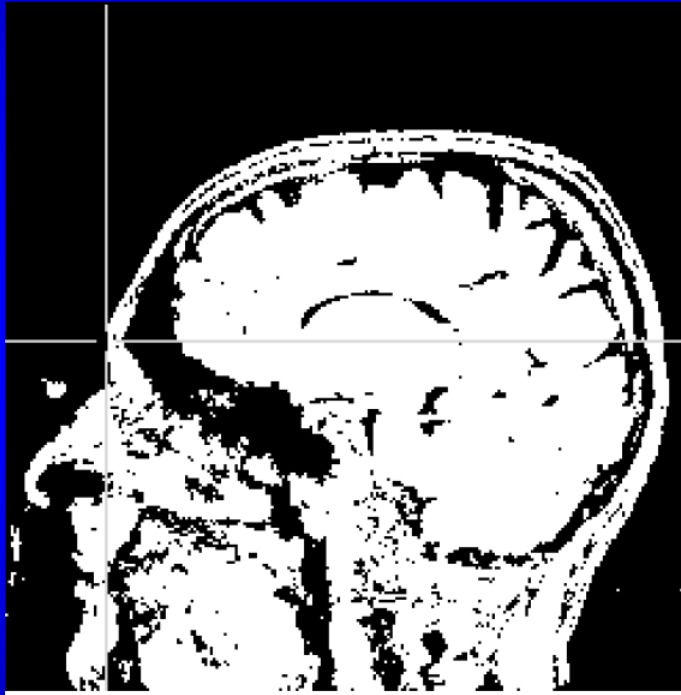
$$OUT = BIGGESTCOMP(IN) := SELBIG(LABEL(IN))$$

finds a biggest connected component in $IN \in \mathcal{B} I$.

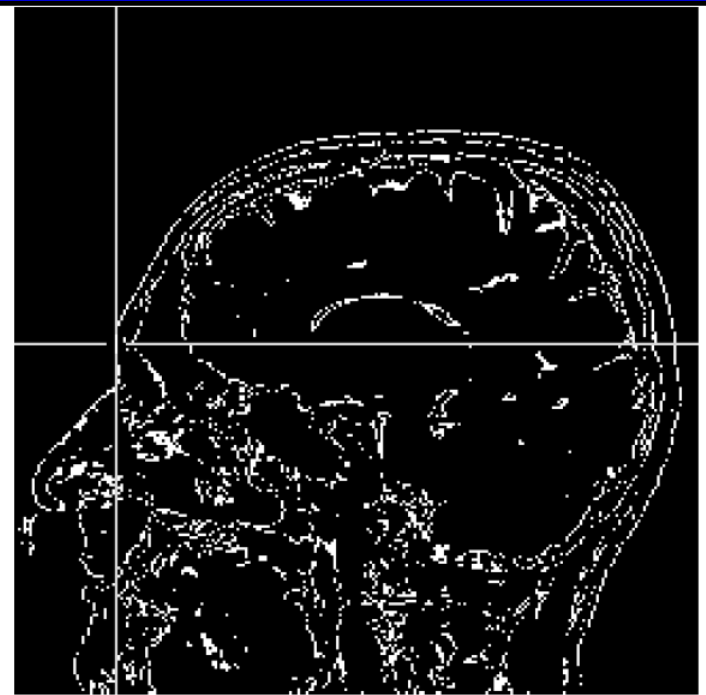
The labeling algorithm begins by selecting an arbitrary foreground voxel and assigns the label "1" to it. It then propagates this label recursively to all adjacent foreground voxels, until no more foreground voxel connected to any already labeled one can be found. The algorithm then tries to find a further foreground voxel, which has still not been labeled and starts a second round with label "2". The procedure is stopped when all foreground voxels have been labeled [180].

Basic operations on images

OUT=BORDER(BIGGESTCOMP (IN))



IN



OUT

Basic operations on images

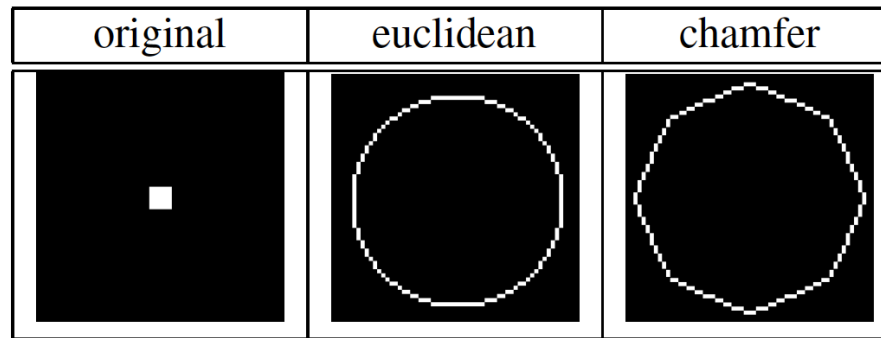


Figure 4.6: *Original binary mask (left), iso-euclidian distance points (middle) and iso-chamfer distance points (right).*

Operation 4.4.13. *The distance transform $OUT = DIST(IN)$ on $IN \in \mathcal{B} I$ with $OUT \in \mathcal{F} I$ attaches a label to each background voxel which encodes the shortest distance towards the closest foreground voxel. The distance within $DIST$ is measured by an euclidian metric, whereas the distance transform $DIST_CHAM$ uses a chamfer metric.*

The operation $DIST_CHAM$ can be seen as a fast method to approximate $DIST$. To illustrate the difference: If a binary mask with only one foreground voxel in the middle of the mask will be transformed, iso-euclidian distance points are arranged as a circle and iso-chamfer distance points as a hexagon around this foreground voxel, see Figure 4.6 [180].

Basic operations on images

Operation 4.4.14. *The operation $OUT = \text{BINARIZE}(IN, \min, \max)$ thresholds $IN \in \mathcal{G} I$ to produce $OUT \in \mathcal{B} I$. The parameter \min specifies the minimal and the parameter \max the maximal foreground value.*

In Figure 4.6, all foreground voxels in the middle and the right figure have a distance of 25 to the original foreground voxel, realized by choosing minimal and maximal foreground value as 25.

$$OUT = \text{BINARIZE}(\text{DIST}(IN), 25, 25)$$



IN

OUT

Basic operations on images

Operation 4.4.15. *The operation $OUT = INVERT(IN)$ inverts $IN \in \mathcal{B} I$ so that foreground voxels in IN become background voxels in $OUT \in \mathcal{B} I$ and vice versa.*

Basic operations on images

Based on the last three basic operations, the following morphological basis operations can be derived:

Operation 4.4.16. *The operation $OUT = DILATION(IN, rad)$ with $IN, OUT \in \mathcal{B} I$ will be defined by thresholding a distance transform, i.e.*

$$DILATION(IN, rad) := BINARIZE(DIST(IN), 0, rad).$$

The operation $OUT = EROSION(IN, rad)$ with $IN, OUT \in \mathcal{B} I$ will be defined by

$$EROSION(IN, rad) := BINARIZE(DIST(INVERT(IN)), rad, \infty).$$

It should be mentioned that the operations *DILATION* and *EROSION* are basically defined in mathematical morphology and that, in fact, they are only simulated by the above operations. Nevertheless, if a distance transform with an euclidian metric is used, the result of the above procedures will be the same as a morphological filtering with a sphere-shaped structuring element of radius *rad* [180].

Basic operations on images

By means of these two morphological basis operations, we can further define

Operation 4.4.17. *The operation $OUT = OPENING(IN, rad)$ with $IN, OUT \in \mathcal{B} I$ will be defined by*

$$OPENING(IN, rad) := DILATION(EROSION(IN, rad), rad).$$

The operation $OUT = CLOSING(IN, rad)$ with $IN, OUT \in \mathcal{B} I$ will be defined by

$$CLOSING(IN, rad) := EROSION(DILATION(IN, rad), rad).$$

Basic operations on images

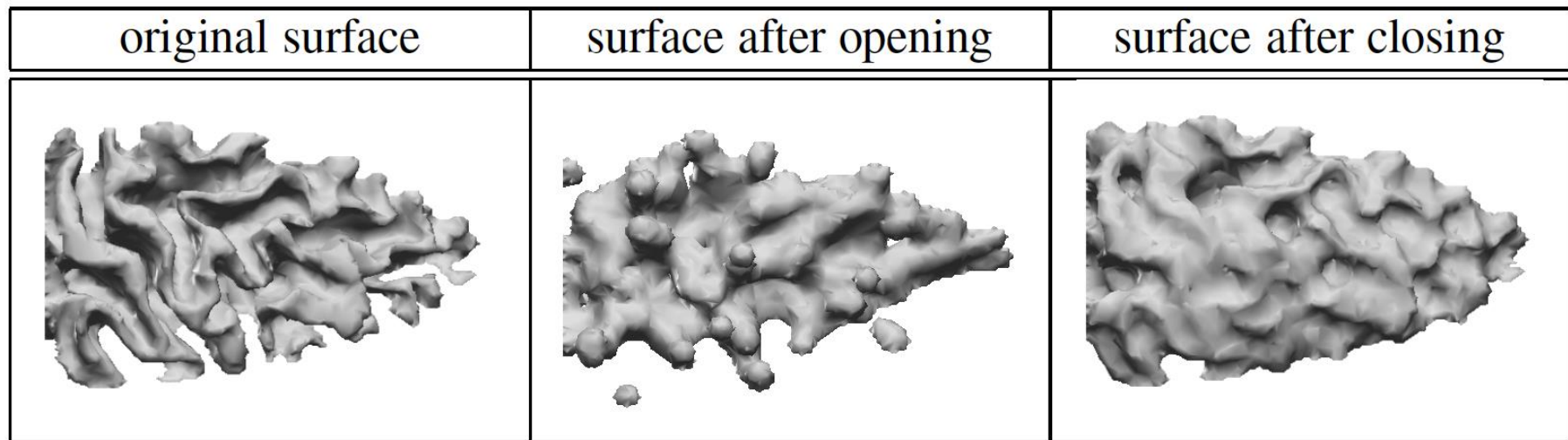


Figure 4.7: Visualization of two morphological operations from Lohmann [180].

The operation $OUT = SMOOTHING(IN, rad)$ with $IN, OUT \in \mathcal{B} I$ will be defined by

$$SMOOTHING(IN, rad) := OPENING(CLOSING(IN, rad), rad).$$

Figure 4.7 illustrates the effect of opening and closing on a brain WM surface. Thin gyri disappear during opening. The closing operator fills the sulci but does not cause the gyri to rise.

Basic operations on images

Two image filters will now be introduced. The first is the well-known operation, which convolves an image with a Gauss filter with standard deviation σ [147]:

Operation 4.4.18. *The operation $OUT = GAUSS(IN, \sigma)$ applies a Gauss-filter with standard deviation σ to the image $IN \in \mathcal{M} \mathcal{R} I$ in order to produce a smoothed and noise-reduced $OUT \in \mathcal{M} \mathcal{R} I$.*

The Gauss-filter also smoothes the edges of an image, which is not wanted for certain applications. Lee filters, also called sigma filters [176], replace each voxel with the mean value of its surrounding window, where the mean is taken only from the neighboring voxels whose intensities do not differ by more than a σ threshold from the value of the current voxel.

Operation 4.4.19. *The operation $OUT = LEE(IN, \sigma)$ applies a sigma filter with a σ threshold to the image $IN \in \mathcal{M} \mathcal{R} I$ and produces an edge-preserved smoothed and noise-reduced $OUT \in \mathcal{M} \mathcal{R} I$.*

Basic operations on images

In Chapter 5, the following operation is needed for modifying the intensity values of an MRI:

Operation 4.4.20. *The operation*

$$MRI_{mod} = MODINT(MRI, BI, I^{new})$$

sets the intensity values of all those lattice points of the image $MRI \in \mathcal{M} \mathcal{R} I$ to the new intensity value $I^{new} \in G_{MRI}$, which are foreground in the binary mask $BI \in \mathcal{B} I$.

The last two operations are concerned with the image markers.

Operation 4.4.21. *The operation*

$$MRI_{ext} = INTRODUCE_MARKERS(MRI)$$

extends MRI , i.e., copies the intensity for each lattice point and defines its markers as "not set".

Operation 4.4.22. *By means of the operation*

$$MRI_{ext} = MARK(MRI_{ext}, BI, marker),$$

the marker = STOP, PASS of all those lattice points of the extended image $MRI_{ext} \in \mathcal{M} \mathcal{R} I_{ext}$ will be set, which are foreground in the binary mask $BI \in \mathcal{B} I$.

Basic operations on meshes

Mesh operations

The later introduced deformable models use mesh representations of surfaces which are defined as iso-surfaces in digitized volumetric data. It will be focussed here on the extraction of a border of a binary mask in form of a mesh consisting of triangle elements (see Def.4.4.8). The *marching tetrahedra algorithm* (Payne and Toga, 1990 [238]) is now presented as such an extraction method resulting in a closed and oriented triangular surface mesh, i.e., the vertices of each triangle are ordered so that, viewed from the outside, the vertex-cycle is traversed in counterclockwise ordering.

Basic operations on meshes

Mesh operations

The later introduced deformable models use mesh representations of surfaces which are defined as iso-surfaces in digitized volumetric data. It will be focussed here on the extraction of a border of a binary mask in form of a mesh consisting of triangle elements (see Def.4.4.8). The *marching tetrahedra algorithm* (Payne and Toga, 1990 [238]) is now presented as such an extraction method resulting in a closed and oriented triangular surface mesh, i.e., the vertices of each triangle are ordered so that, viewed from the outside, the vertex-cycle is traversed in counterclockwise ordering.

Basic operations on meshes

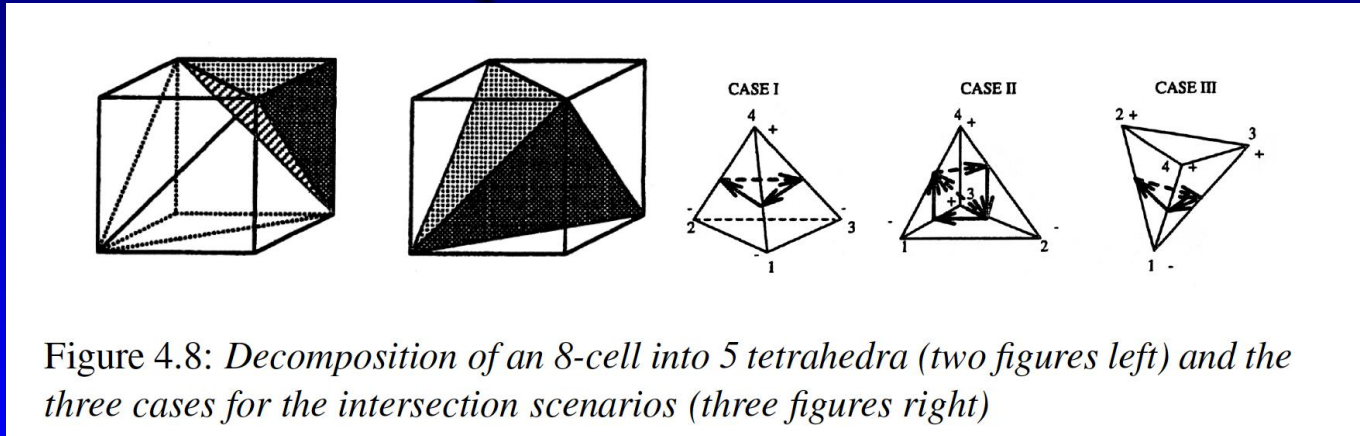


Figure 4.8: *Decomposition of an 8-cell into 5 tetrahedra (two figures left) and the three cases for the intersection scenarios (three figures right)*

The starting point within the algorithm is a decomposition of each so-called 8-cell (or cube) into 5 tetrahedra, where the vertices of the cube are the addresses of 8 neighbored voxels in the image lattice (see Figure 4.8). Two such decompositions are possible and it must be alternated in a 3D checkerboard fashion between both of them so that faces and edges of 8-cell tetrahedra match those of neighboring ones. The second step in the algorithm is an identification of those tetrahedra with vertices $\mathbf{v}_{i_1}, \mathbf{v}_{i_2}, \mathbf{v}_{i_3}, \mathbf{v}_{i_4}$, intersecting with the iso-surface of the binary mask. This is controlled by means of a sign-change of $I(\mathbf{v}_{i_k}) - I_0$ with $I_0 \in]0, 1[$. The intersection points along the edges of a tetrahedron and thus the vertices of the resulting triangle mesh, are dependent on the used interpolation basis function. Gueziec and Hummel [111] proposed a bilinear basis function on an 8-cell, since a linear led to an excessive spikeness of the resulting triangle surface mesh. The bilinear basis function reduces to a linear along the edges of an

Basic operations on meshes

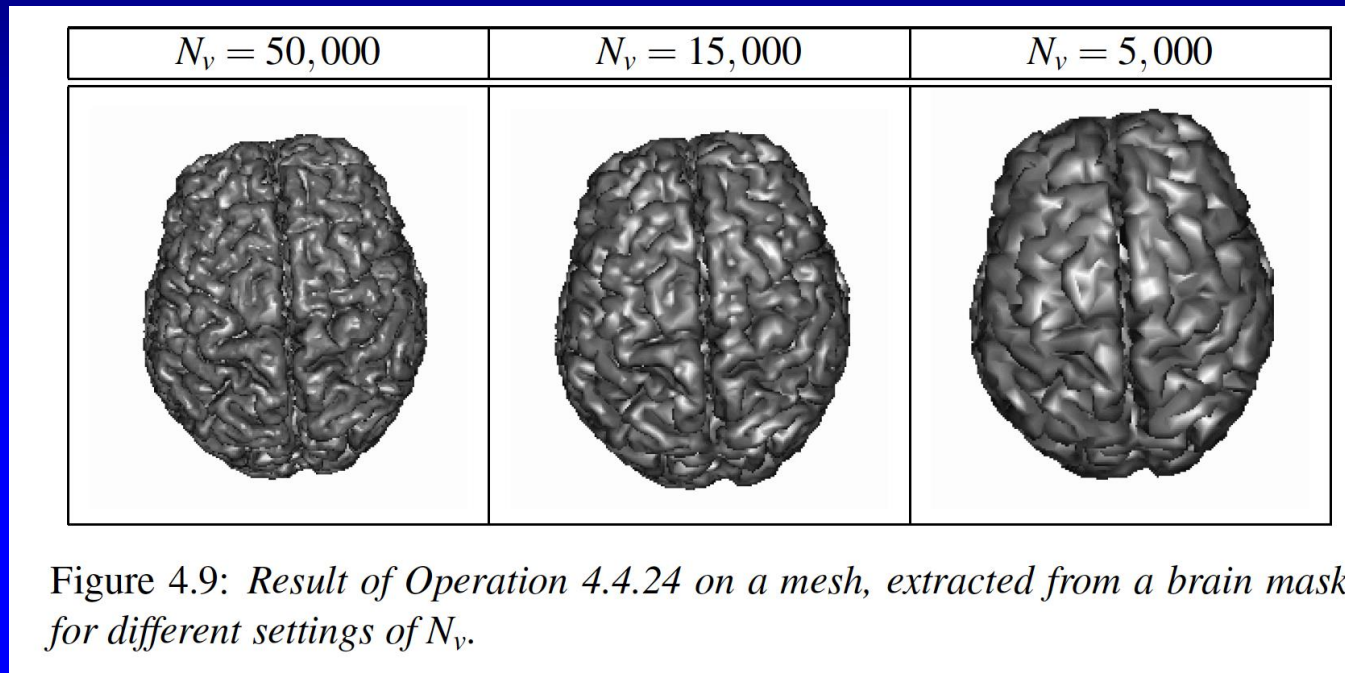
8-cell and to a quadratic along 8-cell diagonals. Triangle vertices are calculated as the zero-crossings of the interpolation function within a tetrahedron. Three different intersection scenarios have to be distinguished, one of them results in two triangles for the surface mesh, the other two in one (see Figure 4.8). During the above process, the first surface triangle will be assigned a correct orientation by appropriately ordering its vertices. Since any given edge of the mesh has to be traversed in opposite directions by the vertex-cycles from the two neighboring triangles, the correct orientation of the surface mesh can then be ensured by a recursive procedure. The above explanations led to

Operation 4.4.23. *The operation $OUT = EXTRACT(IN)$ extracts the border of $IN \in \mathcal{BI}$ as a closed and well-oriented triangulated surface mesh $OUT \in \mathcal{MESH}$ by means of the marching tetrahedra method.*

Basic operations on meshes

The extracted mesh generally consists of a large number of triangles. In order to remove badly shaped ones and to coarsen the mesh in regions of small curvature, a mesh simplification follows the extraction. In Gueziec and Hummel [111], each edge $(\mathbf{v}_{i_1}, \mathbf{v}_{i_2})$ goes through a deletion test and if the tests are positive, it will be replaced by $\mathbf{v} = (\mathbf{v}_{i_1} + \mathbf{v}_{i_2})/2$, i.e., former edges to \mathbf{v}_{i_1} and \mathbf{v}_{i_2} will be replaced by edges to \mathbf{v} and two triangles are removed. If \mathbf{v}_{i_3} and \mathbf{v}_{i_4} denote the remaining vertices of the two triangles which contain the edge $(\mathbf{v}_{i_1}, \mathbf{v}_{i_2})$, the first demand is, that the distance between \mathbf{v}_{i_1} and \mathbf{v}_{i_2} is smaller than the distance between \mathbf{v}_{i_3} and \mathbf{v}_{i_4} . A second prerequisite is a limit for the projection distance of \mathbf{v}_{i_1} and \mathbf{v}_{i_2} on the resulting surface after the deletion.

Basic operations on meshes



Operation 4.4.24. The operation $OUT = SIMPLIFY(IN, N_v)$ with $IN, OUT \in \mathcal{MESH}$ simplifies an extracted mesh to N_v vertices by a coarsening in regions of small curvature and by removing badly shaped triangles.

Figure 4.9 shows the result of Operation 4.4.24 on a mesh with 750,000 vertices, extracted from a brain mask for different settings of N_v . It can be seen, that 5.000 vertices are no longer sufficient to correctly represent the neocortical surface.

Basic operations on meshes

Later in this chapter, the transformation of a triangle mesh into a binary image is needed. The following operation deals with this problem:

Operation 4.4.25. *By means of $mask = VOXELIZE(mesh)$ with $mesh \in \mathcal{M} \subseteq \mathcal{SH}$ consisting of triangle primitives ($\Delta_v = 3$), a triangle mesh is transformed into a binary image $mask \in \mathcal{BI}$. Therefore, each voxel of $mask$ becomes a foreground voxel, if the distance of its lattice point projection onto the closest triangle primitive is smaller than $\sqrt{3}/2$.*

Operation 4.4.25 yields a mask, which has to be filled in certain situations. The filling process is described in Script 4.4.26 and illustrated in Figure 4.10. In the second step of the script, the contour of the transformed mesh is dilated as the background component with the parameter d , so that the biggest connected component of the resulting mask is the d eroded image background. The dilation is then canceled by a d erosion and the filled *mask* is achieved.

Segmentation scripts for head modeling

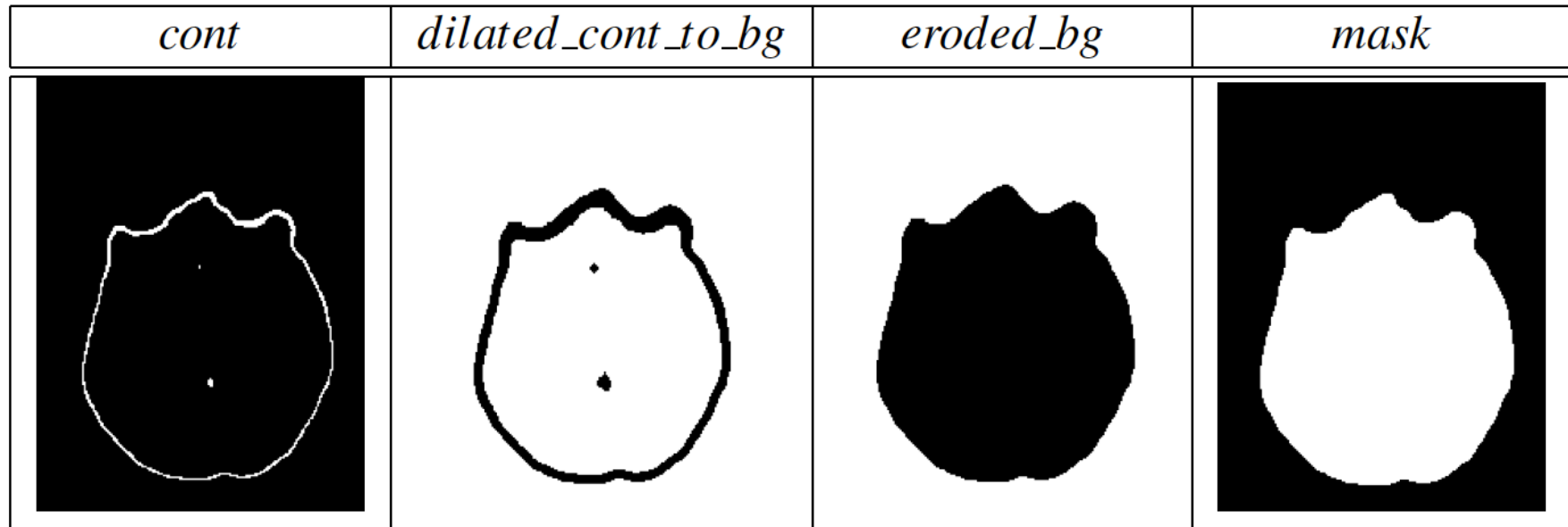


Figure 4.10: *Example for Script 4.4.26: Filling the ISS mesh.*

Script 4.4.26. FILL : $(mesh \in \mathcal{MESH}, d \in \mathbb{R}) \rightarrow (mask \in \mathcal{BI})$:

1. $cont = VOXELIZE(mesh)$
2. $dilated_cont_to_bg = BINARIZE(DIST(cont), d, \infty)$
3. $eroded_bg = BIGGESTCOMP(dilated_cont_to_bg)$
4. $mask = INVERT(BINARIZE(DIST(eroded_bg), 0, d))$

Structure

- **Fuzzy segmentation techniques**
- **Geometric deformable model reconstructions**
- **Basic definitions and operations on images and meshes**
- **Segmentation scripts**

Head mask segmentation script

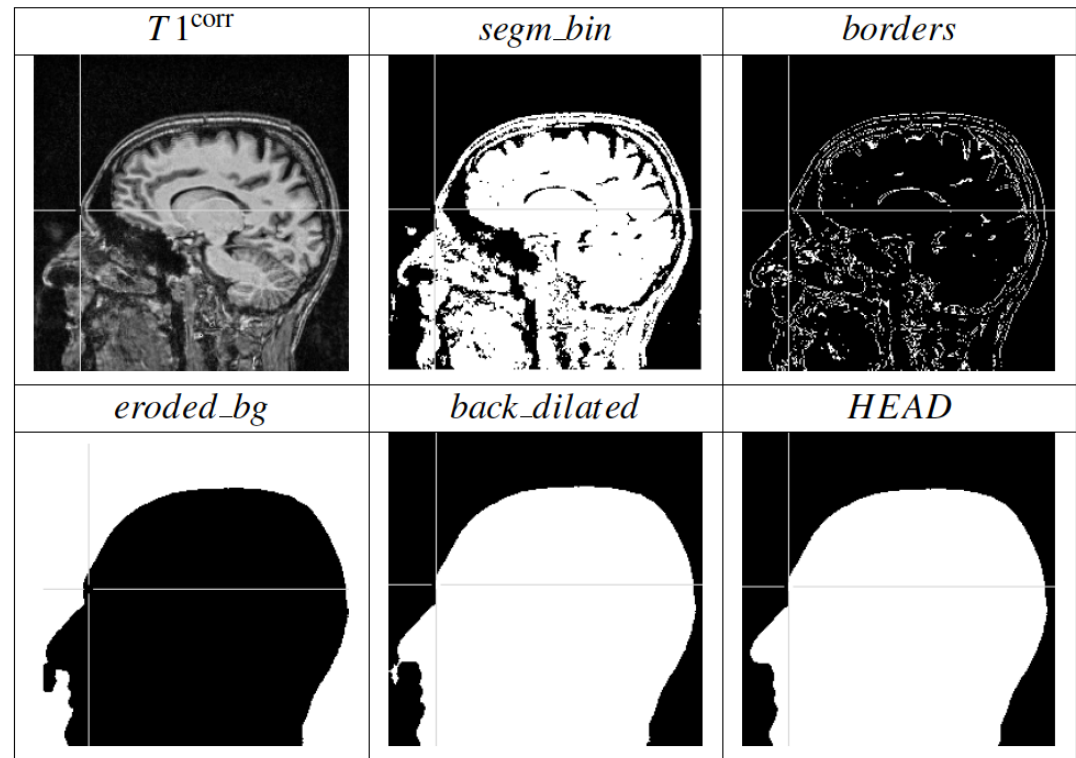


Figure 4.16: *Intermediate results for the generation of the head mask.*

Script 4.7.1. HEAD : $(MRI \in \mathcal{MR}I, C \in \mathbb{N}, \lambda_1, \lambda_2, \varepsilon, \delta \in \mathbb{R}, d_1, d_2, d_3 \in \mathbb{N}_0, OSS \in \mathcal{BI}) \rightarrow (HEAD, SCALP \in \mathcal{BI})$

1. $(MRI^{corr}, MRI^{afcm}, c_j^{afcm}) = \mathbf{AFCM}(MRI, C, \lambda_1, \lambda_2, \varepsilon, \delta)$ /* class. */
2. $segm_bin = \mathbf{BINARIZE}(MRI^{afcm}, g_1(C), g_2(C))$ /* fusion */
3. $borders = \mathbf{BORDER}(\mathbf{BIGGESTCOMP}(segm_bin))$ /* closing holes */
4. $eroded_bg = \mathbf{BIGGESTCOMP}(\mathbf{BINARIZE}(\mathbf{DIST}(borders), d_1, \infty))$
5. $back_dilated = \mathbf{BINARIZE}(\mathbf{DIST}(eroded_bg), d_1, \infty)$
6. $HEAD = \mathbf{SMOOTHING}(back_dilated, d_2)$ /* smoothing */
7. $SCALP = \mathbf{AND}(HEAD, \mathbf{DILATE}(OSS, d_3))$ /* for multi-tissue model */

Brain peeling segmentation script

Script 4.7.2. PEELING : $(T1 \in \mathcal{M} \mathcal{R} I, d_1, d_2 \in \mathbb{R}) \rightarrow (PEEL \in \mathcal{B} I)$

1. $t1_filt = LEE(t1)$ */* binarize filtered T1 */*
2. $bin = BINARIZE(ISODATA(t1_filt, 2), 1, 1)$
3. $and1 = AND(bin, EROSION_SPEC(bin, d_1))$ */* erode */*
4. $big = BIGGESTCOMP(OPENING(and1, 3))$ */* extract */*
5. $and2 = AND(bin, DILATION_SPEC(big, d_2))$ */* inflate */*
6. $PEEL = CLOSING(and2, 12)$

Script for creating brain mask

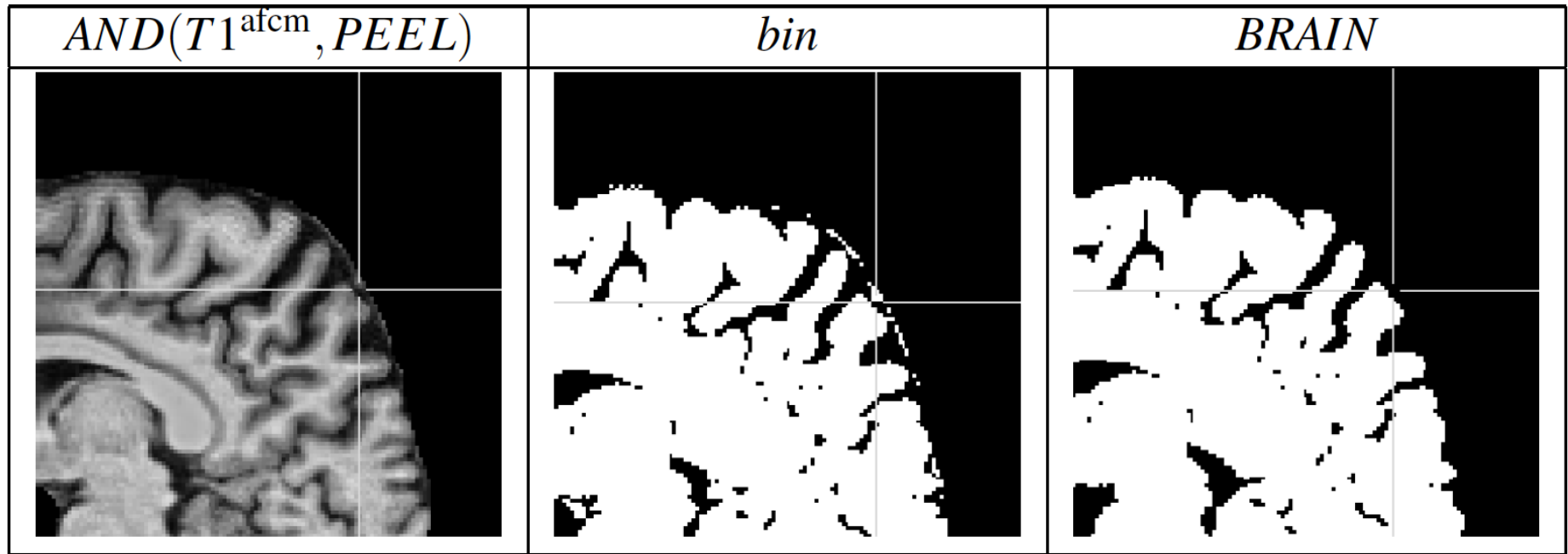


Figure 4.17: *Intermediate results of Script 4.7.3.*

Script 4.7.3. GEN_BRAIN : $(T1^{afcm} \in \mathcal{CPI}, PEEL \in \mathcal{BI}, d \in \mathbb{N}) \rightarrow (BRAIN \in \mathcal{BI})$:

1. $bin = BINARIZE(AND(T1^{afcm}, PEEL), 1, 2)$
2. $BRAIN = DILATION(BIGGESTCOMP(EROSION(bin, d)), d)$

Inner skull surface (ISS) segmentation script

Script 4.7.4. ISS : $(T1, PD^{reg} \in \mathcal{M} \mathcal{R} I, \lambda_1, \lambda_2, N_v \in \mathbb{N}, \varepsilon, \delta, \omega_{int}, \omega_{ext}, \tau, d \in \mathbb{R}) \rightarrow (ISS \in \mathcal{B} I)$

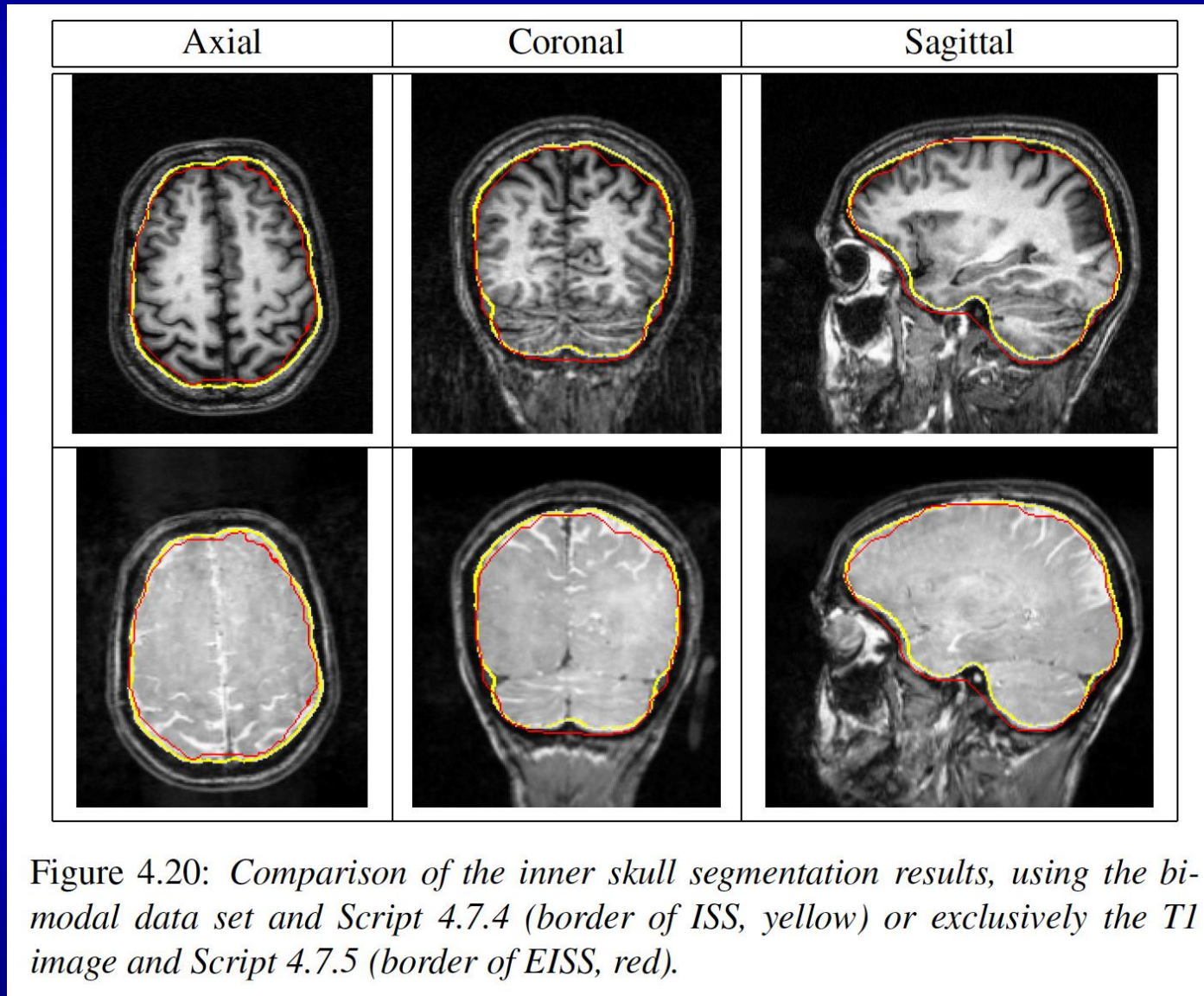
1. $(PD^{corr}, PD^{afcm}, c_0^{PD}, c_1^{PD}) = \mathbf{AFCM}(PD^{reg}, 2, \lambda_1, \lambda_2, \varepsilon, \delta)$ */* class. */*
2. $segm_bin = \mathbf{BIGGESTCOMP}(\mathbf{BINARIZE}(PD^{afcm}, 1, 1))$ */* peeling */*
3. $small_inner = \mathbf{BIGGESTCOMP}(\mathbf{EROSION}(segm_bin, 6))$
4. $too_large_inner = \mathbf{DILATION}(small_inner, 8)$
5. $peeled_inner = \mathbf{AND}(segm_bin, too_large_inner)$
6. $del_iso_points = \mathbf{BIGGESTCOMP}(peeled_inner)$ */* cleaning */*
7. $IN = \mathbf{INVERT}(\mathbf{BIGGESTCOMP}(\mathbf{INVERT}(del_iso_points)))$
8. $IS = (\mathbf{DEFORM}(PD^{corr}, IN, N_v, \frac{c_0^{PD} + c_1^{PD}}{2}, \omega_{int}, \omega_{ext}, \tau))$ */* Improve */*
9. $ISS = \mathbf{FILL}(IS, d)$ */* fill */*

Estimated (from T1-MRI) inner skull surface (EISS) segmentation scripts

Script 4.7.5. EISS : $(T1 \in \mathcal{M} \mathcal{R} I, BRAIN \in \mathcal{B} I, \lambda_1, \lambda_2 \in \mathbb{N}, \varepsilon, \delta, d_1, d_2 \in \mathbb{R}, I_{min}, I_{max} \in G_{MRI}) \rightarrow (EISS \in \mathcal{B} I)$

1. $(T1^{corr}, T1^{afcm}, c_0^{T1}, c_1^{T1}, c_2^{T1}) = AFCM(T1, 3, \lambda_1, \lambda_2, \varepsilon, \delta)$ */* class. */*
2. $close = \mathbf{CLOSING}(BRAIN, d_1)$ */* Generate minimal inner mask */*
3. $min_inner = \mathbf{DILATION}(close, d_2)$
4. $max_inner = \mathbf{DILATION}(min_inner, 2)$ */* Gen. maximal inner mask */*
5. $T1_{ext}^{corr} = \mathbf{INTRODUCE_MARKERS}(T1^{corr})$ */* extend MRI */*
6. $T1_{ext}^{corr} = \mathbf{MARK}(T1_{ext}^{corr}, min_inner, PASS)$ */* Meninges/skull segm. */*
7. $T1_{ext}^{corr} = \mathbf{MARK}(T1_{ext}^{corr}, \mathbf{INVERT}(max_inner), STOP)$
8. $meninges = \mathbf{ERG}(T1_{ext}^{corr}, 0, I_{min}, I_{max})$
9. $EISS = \mathbf{OR}(\mathbf{SMOOTHING}(meninges, 20), min_inner)$ */* Smoothing */*

Differences between ISS and EISS segmentation



Differences between ISS and EISS segmentation

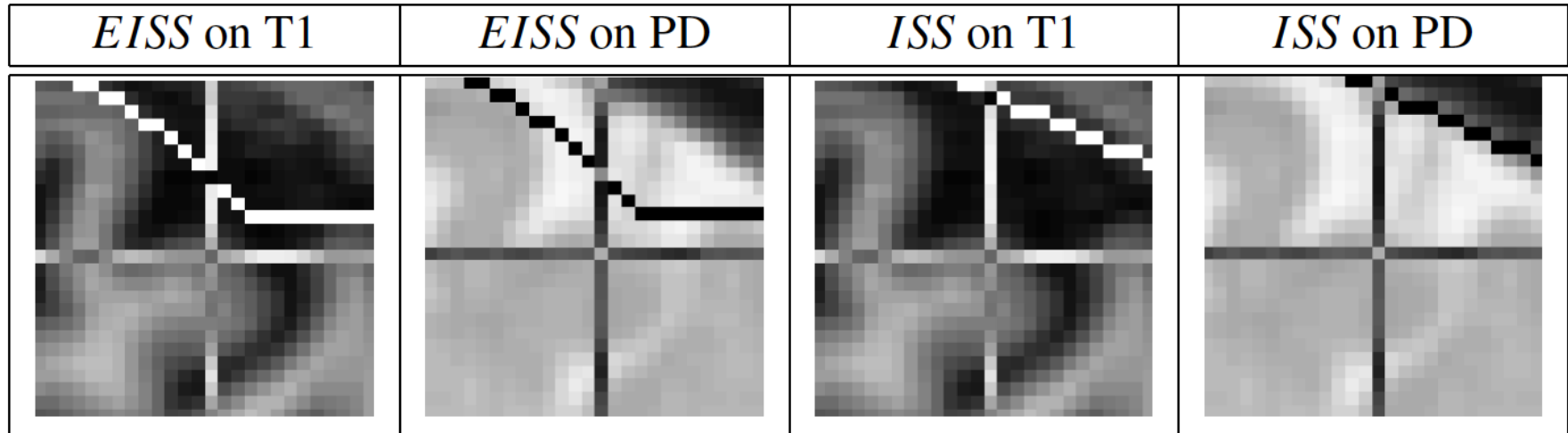


Figure 4.21: *Magnification of the parietal area of the neurocranial roof where the CSF layer is thicker than being estimated by means of the T1-MRI based EISS, as the PD image and the border of the ISS show.*

Outer skull surface (OSS) segmentation script

Script 4.7.6. OSS : $(T1^{corr} \in \mathcal{M} \mathcal{R} I, ISS, HEAD \in \mathcal{B} I, elastic \in G_{0,1}, I_{min}, I_{max}, N_v, c_0^{afcm}, c_1^{afcm} \in \mathbb{N}, \omega_{int}, \omega_{ext}, \tau \in \mathbb{R}) \rightarrow (OSS \in \mathcal{B} I)$

1. $T1_{ext}^{corr} = INTRODUCE_MARKERS(T1^{corr})$ /* extend MRI */
2. $min = DILATION(ISS, 3)$ /* minimal outer mask */
3. $max = AND(DILATION(min, 6), HEAD)$ /* maximal outer mask */
4. $T1_{ext}^{corr} = MARK(T1_{ext}^{corr}, min, PASS)$ /* segm. initial mask */
5. $T1_{ext}^{corr} = MARK(T1_{ext}^{corr}, INVERT(max), STOP)$
6. $OSS_{in} = OR(SMOOTHING(\mathbf{ERG}(T1_{ext}^{corr}, elastic, I_{min}, I_{max}), 6), min)$
7. $OS = \mathbf{DEFORM}(T1^{corr}, OSS_{in}, N_v, \frac{c_0^{afcm} + c_1^{afcm}}{2}, \omega_{int}, \omega_{ext}, \tau)$ /* Improve */
8. $OSS = FILL(OS, 1)$

Outer skull surface (OSS) segmentation script

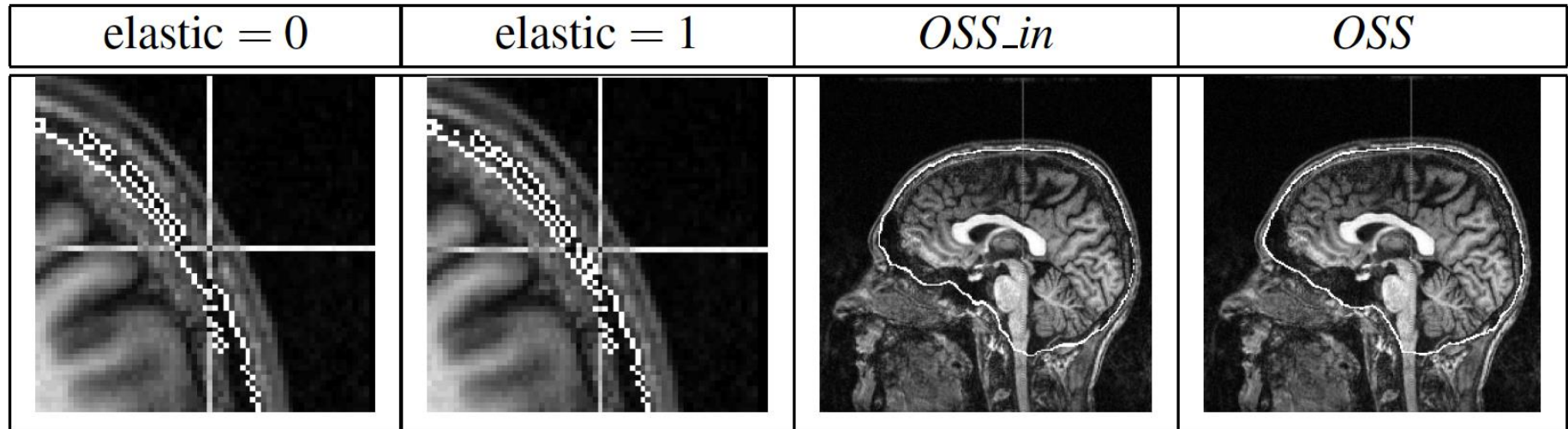


Figure 4.22: Results of script 4.7.6: ERG surface results for low (left) and higher (left middle) elasticity, border of OSS_{in} (right middle) and of OSS (right) on underlying T1 image for high elasticity.

3D rendering of segmented surfaces

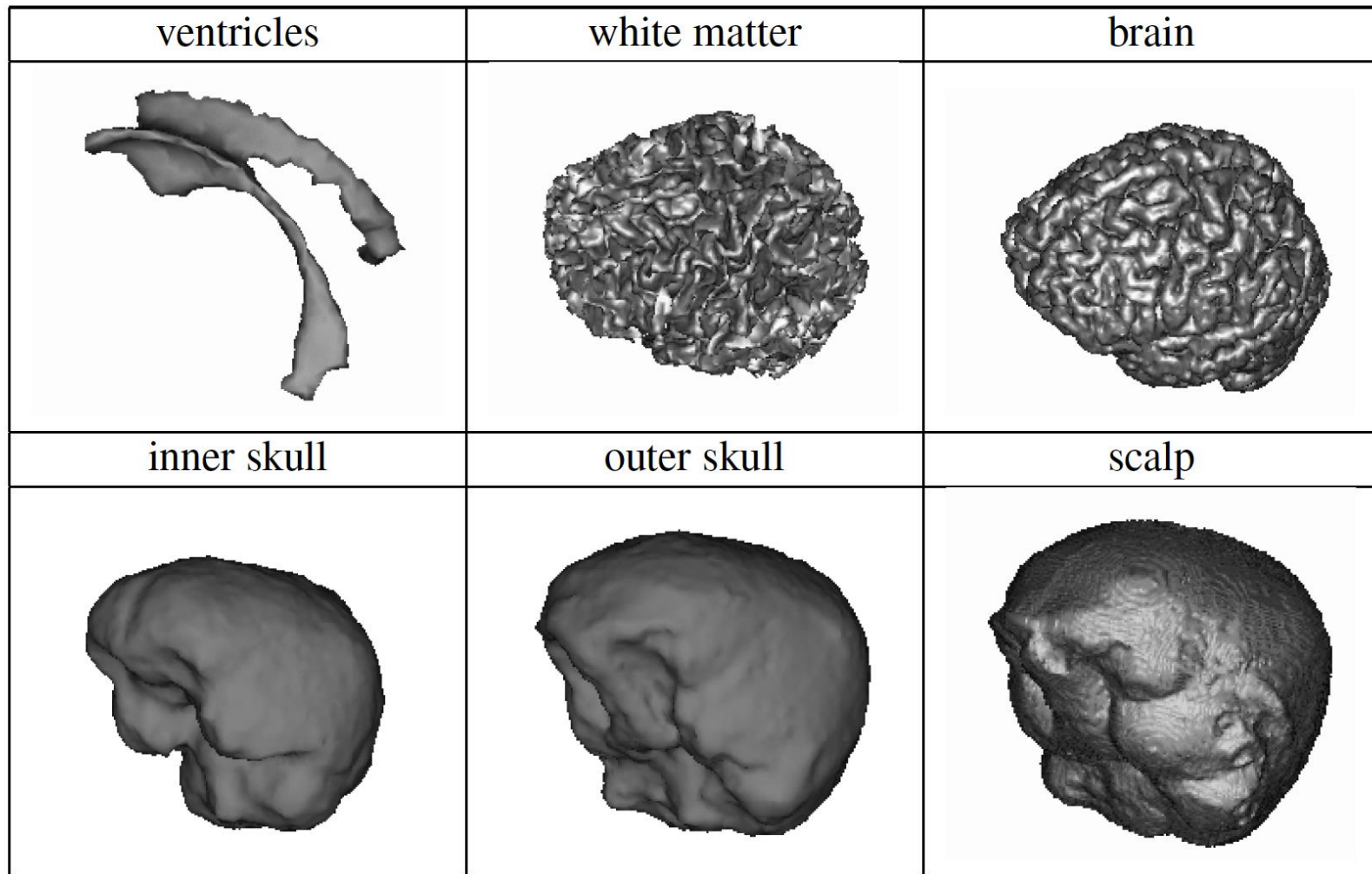


Figure 4.23: 3D rendering of tissue boundaries in multicompartiment head model.

5 compartment head model

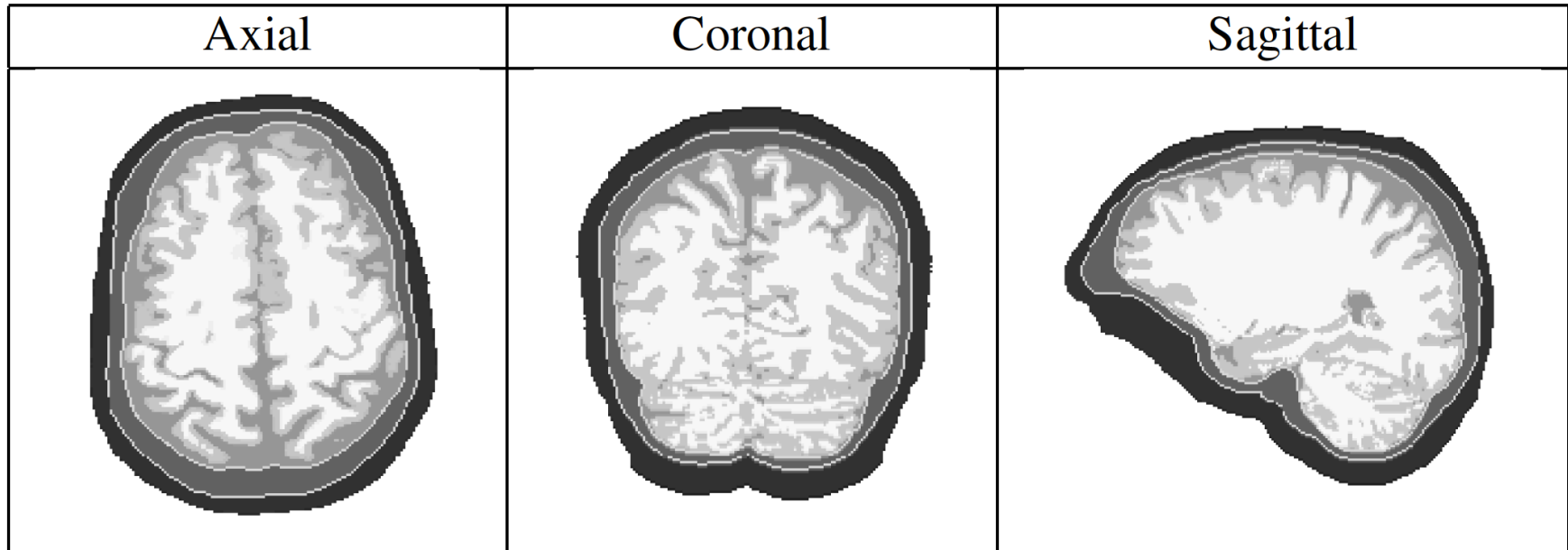


Figure 4.24: *The 5 tissue head model.*