Vox2Cortex: Fast Explicit Reconstruction of Cortical Surfaces from 3D MRI Scans with Geometric Deep Neural Networks — Supplementary Material

Fabian Bongratz[1†*], Anne-Marie Rickmann[1,2*], Sebastian Pölsterl[2], Christian Wachinger[1,2]
[1]Technical University of Munich, [2]Ludwig-Maximilians-University Munich

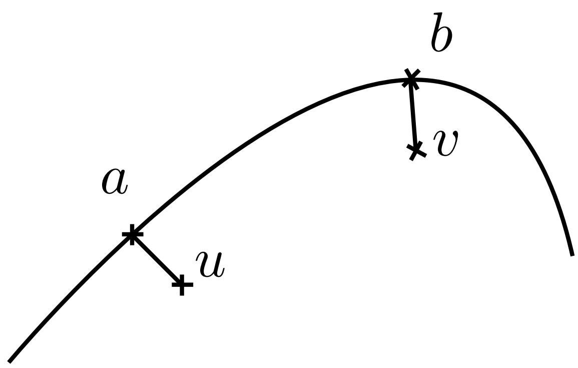

Figure 1. Ground-truth points a and b with curvature $\kappa(a) < \kappa(b)$ and predicted points u and v.

A. Proof for Cuvature-Weighted Chamfer

We want to give a brief mathematical intuition why our curvature-weighted Chamfer loss emphasizes geometric accuracy in high-curvature regions compared to low-curvature regions. Imagine therefore two ground-truth points a and b with respective curvature $\kappa(a) < \kappa(b)$ and closest predicted points u and v as shown in Figure 1. Furthermore, let the distance from the prediction to the ground truth be equal in both cases, such that $\|u - a\| = \|v - b\|$. For the sake of simplicity, we treat the predicted values u and v as the parameters that are optimized by gradient descent, i.e., $u' = u - \lambda \frac{\partial \mathcal{L}_{\mathrm{C}}(a,u)}{\partial u}$ with learning rate $\lambda > 0$. Based on Equation (7) in the main paper, the gradient of the curvature-weighted Chamfer loss with respect to u calculates as

$$\begin{aligned} \frac{\partial \mathcal{L}_{\mathrm{C}}(a,u)}{\partial u} &= \frac{\partial}{\partial u}\left[\kappa(a)\left(\|a-u\|^2 + \|u-a\|^2\right)\right] \\ &= 4\kappa(a)(u-a). \end{aligned} \tag{1}$$

The calculation of $\frac{\partial \mathcal{L}_{\mathrm{C}}(b,v)}{\partial v} = 4\kappa(b)(v-b)$ works analogously. The parameter updates are given by

$$\begin{aligned} u' &= u - \frac{\partial \mathcal{L}_{\mathrm{C}}(a,u)}{\partial u} = u + 4\lambda\kappa(a)(a-u), \\ v' &= v - \frac{\partial \mathcal{L}_{\mathrm{C}}(b,v)}{\partial v} = v + 4\lambda\kappa(b)(b-v). \end{aligned} \tag{2}$$

Further, we have $\|a-u\| = \|b-v\|$ and $\kappa(a) < \kappa(b)$, and thus we get $\|v'-b\| < \|u'-a\|$ if we assume that we don't "shoot over" the goal, i.e., $0 < 4\lambda\kappa(a) < 4\lambda\kappa(b) < 1$. That is, point v is pushed more towards b compared to u towards a within one backward pass. ∎

B. Definitions of Loss Functions

Binary cross entropy The cross-entropy loss between a predicted binary segmentation $B_l^{\mathrm{p}} \in [0,1]^{HWD}$ and a label $B^{\mathrm{gt}} \in \{0,1\}^{HWD}$, where voxels are enumerated from 1 to $N = HWD$, is defined as

$$\begin{aligned} \mathcal{L}_{\mathrm{BCE}}(B_l^{\mathrm{p}}, B^{\mathrm{gt}}) = -\frac{1}{N}\sum_{i=1}^{N} &\left[B^{\mathrm{gt}}(i)\log B_l^{\mathrm{p}}(i)\right. \\ &+ \left.(1-B^{\mathrm{gt}}(i))(1-\log B_l^{\mathrm{p}}(i))\right], \end{aligned} \tag{3}$$

where $B(i)$ is the value of voxel i.

Inter-mesh normal consistency loss While the Chamfer distance takes into account the spatial position of two meshes, i.e., enforcing surface points to lie "at the right location", the *cosine distance* considers the orientation of meshes. In general, one can compute the cosine distance within one mesh, which we refer to as *intra-mesh normal consistency*, and between two meshes, which we call *inter-mesh normal consistency*.

The inter-mesh normal consistency loss is defined based on the normal vectors of adjacent points in the predicted and the ground-truth mesh. Let $\mathcal{P}_{s,c}^{\mathrm{p}}, \mathcal{P}_c^{\mathrm{gt}}$ be predicted and ground-truth point sets with associated normals ${\mathcal{N}_{s,c}}^{\mathrm{p}} =$

$\{\mathbf{n}(\mathbf{p})|\mathbf{p} \in \mathcal{P}^{\mathrm{p}}_{s,c}\}$ and $\mathcal{N}_c{}^{\mathrm{gt}} = \{\mathbf{n}(\mathbf{p})|\mathbf{p} \in \mathcal{P}^{\mathrm{gt}}_c\}$, respectively. Then, the inter-mesh normal consistency loss is given by

$$\begin{aligned}\mathcal{L}_{\mathrm{n},\,inter}(\mathcal{M}^{\mathrm{p}}_{s,c}, \mathcal{M}^{\mathrm{gt}}_c) = \frac{1}{|\mathcal{P}^{\mathrm{gt}}_c|} \sum_{\mathbf{u}\in\mathcal{P}^{\mathrm{gt}}_c} 1 - \cos(\mathbf{n}(\mathbf{u}), \mathbf{n}(\tilde{\mathbf{v}})) \\ + \frac{1}{|\mathcal{P}^{\mathrm{p}}_{s,c}|} \sum_{\mathbf{v}\in\mathcal{P}^{\mathrm{p}}_c} 1 - \cos(\mathbf{n}(\mathbf{v}), \mathbf{n}(\tilde{\mathbf{u}})),\end{aligned} \tag{4}$$

where $\tilde{\mathbf{v}} = \underset{\mathbf{r}\in\mathcal{P}^{\mathrm{p}}_{s,c}}{\arg\min}\|\mathbf{u} - \mathbf{r}\|^2$ and $\tilde{\mathbf{u}} = \underset{\mathbf{r}\in\mathcal{P}^{\mathrm{gt}}_c}{\arg\min}\|\mathbf{v} - \mathbf{r}\|^2$.

In other words, each normal vector at a certain point $\mathbf{p}$ is compared to the normal belonging to the nearest neighbor of $\mathbf{p}$ in the respective other point set. Since nearest-neighbor correspondences are also required for the computation of the Chamfer loss, we use the same point sets $\mathcal{P}^{\mathrm{p}}$, $\mathcal{P}^{\mathrm{gt}}$ for the computation of the Chamfer and inter-mesh normal consistency loss in practice (see paragraph "Curvature-weighted Chamfer loss" in the main paper for details about the point sets).

Intra-mesh normal consistency loss Instead of computing the cosine distances among the normals of two different meshes as described above, it is also possible to compare normal vectors of two adjacent faces of the same mesh. Two faces f_1 and f_2 are adjacent if they share a common edge $e = f_1 \cap f_2$. This intra-mesh normal-consistency, which we denote as $\mathcal{L}_{\mathrm{n},\,intra}$, intuitively measures the smoothness of a mesh as it is lowest for meshes with no curvature. Formally, it is defined as

$$\mathcal{L}_{\mathrm{n},\,intra}(\mathcal{M}^{\mathrm{p}}_{s,c}) = \sum_{\substack{e=f_1\cap f_2\neq\emptyset \\ f_1,f_2\in\mathcal{F}^{\mathrm{p}}_{s,c}}} 1 - \cos(\mathbf{n}(f_1), \mathbf{n}(f_2)), \tag{5}$$

where $\mathbf{n}(f)$ assigns a normal to each face of the mesh. As this loss is only computed based on a predicted mesh not taking into account any ground truth, it belongs to the group of mesh-regularization losses.

Laplacian loss Another measure for the smoothness of a mesh is computed based on the uniform Laplacian operator $\mathbf{L} = \mathbf{D}^{-1}\mathbf{A} - \mathbf{I}$, where $\mathbf{D}$ is the degree and $\mathbf{A}$ is the adjacency matrix of the mesh. More precisely, *Laplacian smoothing* is defined as

$$\mathcal{L}_{\mathrm{Lap},\,rel}(\mathcal{M}^{\mathrm{p}}_{s,c}) = \frac{1}{|\mathcal{V}^{\mathrm{p}}_{s,c}|} \sum_{i=1}^{|\mathcal{V}^{\mathrm{p}}_{s,c}|} \|(\mathbf{L}^{\mathrm{p}}_{s,c} \cdot \mathbf{\Delta}^{\mathrm{p}}_{s,c})_i\| \tag{6}$$

This is a well-known objective for smooth meshes [6]. While many works [7, 8, 4] smooth the mesh with respect to vertex coordinates $\mathcal{V}^{\mathrm{p}}_{s,c}$, we got inspired by [9] and apply the Laplacian operator to the displacement field $\mathbf{\Delta}^{\mathrm{p}}_{s,c}$. Our ablation study confirms that this is a good choice. More precisely, $\mathbf{\Delta}^{\mathrm{p}}_{s,c}$ represents the displacement vectors moving the vertices $\mathbf{V}^{\mathrm{p}}_{s-1,c}$ to $\mathbf{V}^{\mathrm{p}}_{s,c}$, i.e., $\mathbf{V}^{\mathrm{p}}_{s,c} = \mathbf{V}^{\mathrm{p}}_{s-1,c} + \mathbf{\Delta}^{\mathrm{p}}_{s,c}$ transforms the mesh $\mathcal{M}^{\mathrm{p}}_{s-1,c}$ into $\mathcal{M}^{\mathrm{p}}_{s,c}$.

Even though a Laplacian loss does not guarantee that the predicted meshes are free of self-intersections, it generally enforces the predicted meshes to have a smooth surface, i.e., few self-intersections. Also note that in Eq. (6) $\mathbf{L}^{\mathrm{p}}_{s,c}$ is considered to be a constant, i.e., the loss is not backpropagated through the creation of $\mathbf{L}^{\mathrm{p}}_{s,c}$.

Edge loss Yet another mesh loss function with regularizing purposes is given by the *edge loss*. The edge loss with respect to a predicted mesh is defined as

$$\mathcal{L}_{\mathrm{edge}}(\mathcal{M}^{\mathrm{p}}_{s,c}) = \frac{1}{|\mathcal{E}^{\mathrm{p}}_{s,c}|} \sum_{(i,j)\in\mathcal{E}^{\mathrm{p}}_{s,c}} \|\mathbf{v}_i - \mathbf{v}_j\|^2. \tag{7}$$

Intuitively, this loss function enforces meshes with homogeneous edge-lengths, leading to a homogeneous distribution of vertices on the surface. In general, this is desirable in the context of cortical surfaces since the folds of the cortex are also distributed homogeneously.

Mesh-loss weights We condition the mesh-loss weights on the surface class, even though this increases the number of hyperparameters, as we have found that different weights are necessary for white matter and pial surfaces in order to achieve optimal reconstruction quality. In practice, we tuned the mesh-loss weights for white matter and pial surfaces independently of each other (ignoring the respective other surfaces in those runs and considering only one hemisphere) on the small MALC dataset [5]. It contains 15 training scans, 7 validation scans, and 8 test scans (which we ignore since testing our model on such a few scans is probably not meaningful). From the tuning, we got the following loss weights:

Surface	$\lambda_{1,c}$	$\lambda_{2,c}$	$\lambda_{3,c}$	$\lambda_{4,c}$	$\lambda_{5,c}$
$c =$ wm	1.0	0.01	0.1	0.001	5.0
$c =$ pial	1.0	0.0125	0.25	0.00225	5.0

Mesh-loss function weights for inter-mesh normal consistency $\mathcal{L}_{\mathrm{n},\,inter}$, intra-mesh normal consistency $\mathcal{L}_{\mathrm{n},\,intra}$, and Laplacian smoothing $\mathcal{L}_{\mathrm{Lap}}$ were first tuned with a grid search containing 0.1, 0.01, 0.001 and then fine-tuned with the values $x + 0.5x$, x, $x - 0.5x$, where x was the respective best value of the first tuning. Weights for Chamfer and edge losses were set to 1 in this procedure and the edge-loss weight was later tuned separately trying the values 1, 5, and 10.

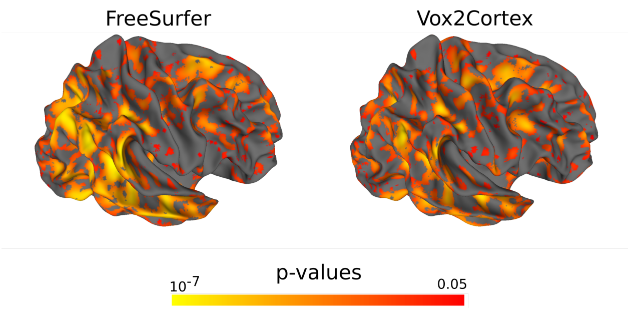

Figure 2. Group comparison of cortical atrophy in the right hemisphere between patients diagnosed with Alzheimer's disease and healthy controls on the $\text{ADNI}_{\text{large}}$ test-split.

C. Implementation Details

We implemented our method based on pytorch v1.7.1 https://pytorch.org/ and pytorch3d v0.4.0 https://pytorch3d.readthedocs.io. We ran experiments on NVIDIA Quadro and Titan RTX GPUs with 24GB memory each (one GPU per training). In addition, we used CUDA v10.2.89, CUDNN v7.6.5, python v3.8.8, and the repositories from DeepCSR [1] https://bitbucket.csiro.au/projects/CRCPMAX/repos/deepcsr/browse and Voxel2Mesh [8] https://github.com/cvlab-epfl/voxel2mesh/blob/master/README.md.

D. Hyperparameters

A list of hyperparameters is in Table 1. We trained our models for 100 epochs (OASIS and $\text{ADNI}_{\text{small}}$) and 40 epochs ($\text{ADNI}_{\text{large}}$) and chose the best model with respect to the respective validation set in terms of voxel IoU and Hausdorff distance.

E. Additional Analysis of Experiments

Cortical atrophy We show the study of cortical atrophy (Figure 5 in the main paper, left hemisphere) for the right hemisphere in Figure 2.

Visual analysis of Freesurfer fails In our $\text{ADNI}_{\text{large}}$ dataset, we removed samples in which FreeSurfer failed. As it is quite difficult to perform automated quality control of the FreeSurfer surface pipeline, we removed all scans that failed in the segmentation of one or more regions as identified by UCSF quality control guidelines [2]. We then applied the trained model to the previously removed cases where FreeSurfer failed and visualize results in Figure 3, where we focus on pial surfaces due to better visibility. The first case is a mild case where FreeSurfer was able to generate 4 surfaces, but we can observe that the left pial surface extends into the dura. Our model does not produce those artifacts. We further display a more extreme case, where FreeSurfer was not able to generate surfaces for the right hemisphere and also failed to segment parts of the left temporal lobe correctly.

Cortical thickness on OASIS We visualize thickness measurements on an exemplary subject from the OASIS dataset in Figure 5. It can be well observed that measurements on Vox2Cortex meshes largely coincide with measurements on FreeSurfer pseudo-ground-truth meshes.

References

[1] Rodrigo Santa Cruz, Léo Lebrat, P. Bourgeat, C. Fookes, J. Fripp, and O. Salvado. Deepcsr: A 3d deep learning approach for cortical surface reconstruction. *2021 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 806–815, 2021. 3

[2] Miriam Hartig, Diana Truran-Sacrey, Sky Raptentsetsang, Alix Simonson, Adam Mezher, Norbert Schuff, and Michael Weiner. Ucsf freesurfer methods. Technical report, Alzheimer's Disease Neuroimaging Initiative, 2014. 3

[3] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In Yoshua Bengio and Yann LeCun, editors, *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015. 4

[4] Fanwei Kong, Nathan Wilson, and Shawn C. Shadden. A deep-learning approach for direct whole-heart mesh reconstruction, 2021. 2

[5] B. Landman and S. Warfield. Miccai 2012 workshop on multi-atlas labeling. In *MICCAI Grand Challenge and Workshop on Multi-Atlas Labeling, CreateSpace Independent Publishing Platform, Nice, France*, 2012. 2

[6] Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. Laplacian mesh optimization. In Y. T. Lee, Siti Mariyam Hj. Shamsuddin, Diego Gutierrez, and Norhaida Mohd. Suaib, editors, *Proceedings of the 4th International Conference on Computer Graphics and Interactive Techniques in Australasia and Southeast Asia 2006, Kuala Lumpur, Malaysia, November 29 - December 2, 2006*, pages 381–389. ACM, 2006. 2

[7] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Computer Vision - ECCV 2018 - 15th European Conference, Munich, Germany, September 8-14, 2018, Proceedings, Part XI*, volume 11215 of *Lecture Notes in Computer Science*, pages 55–71. Springer, 2018. 2

[8] Udaranga Wickramasinghe, Edoardo Remelli, Graham Knott, and Pascal Fua. Voxel2mesh: 3d mesh model generation from volumetric data. In Anne L. Martel, Purang Abolmaesumi, Danail Stoyanov, Diana Mateus, Maria A. Zuluaga, S. Kevin Zhou, Daniel Racoceanu, and Leo Joskowicz, editors, *Medical Image Computing and Computer Assisted Intervention – MICCAI 2020*, pages 299–308, Cham, 2020. Springer International Publishing. 2, 3

Optimizer	CNN learning rate	GNN learning rate	Batch size	Mixed precision	CNN channels	GNN channels	Gradient clipping
Adam [3] $\beta_1 = 0.9$, $\beta_2 = 0.999$	$1e^{-4}$	$5e^{-5}$	2 (1 for OASIS)	yes	16, 32, 64, 128 , 256, 64, 32, 16, 8	255, 64, 64, 64, 64	$2e^{5}$

Table 1. Hyperparameters used in our experiments.

	Pial Surfaces				WM Surfaces			
Method	CC	genus	# faces	# vertices	CC	genus	# faces	# vertices
Ours	1	0	336112	168058	1	0	336112	168058
DeepCSR	48.6	152.4	1341838.3	670711.5	18.3	15.8	1209313.5	604661.7
DeepCSR + top. corr.	1	0	1291385.5	645694.8	1	0	1160980.8	580492.4

Table 2. Comparison of topological measures (number of connected components (CC) and genus) and quantification of mesh complexity in number of faces and vertices. We compare predictions of our method to DeepCSR with and without topology correction on the OASIS test-set. For our method, the number of faces and vertices is defined by the initial template and does not change. Presented values represent the average over white and pial surfaces.

[9] Fenqiang Zhao, Zhengwang Wu, Li Wang, Weili Lin, Shunren Xia, Dinggang Shen, and Gang Li. Unsupervised learning for spherical surface registration. In Mingxia Liu, Pingkun Yan, Chunfeng Lian, and Xiaohuan Cao, editors, *Machine Learning in Medical Imaging*, pages 373–383, Cham, 2020. Springer International Publishing. 2

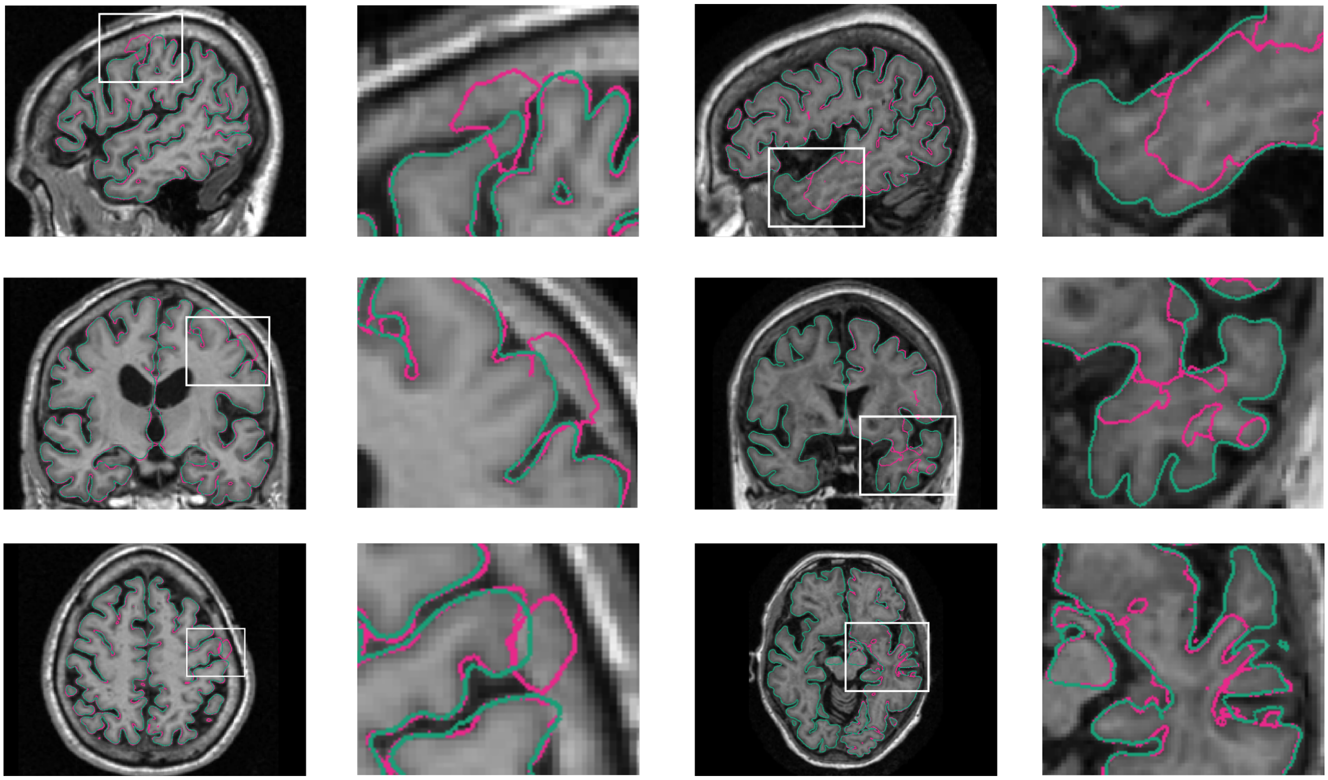

Figure 3. MRI scans with overlaying pial surfaces generated by FreeSurfer (pink) and Vox2Cortex (green). From top to bottom we show sagittal, coronal, and axial slices of two subjects with zoomed in parts where FreeSurfer failed.

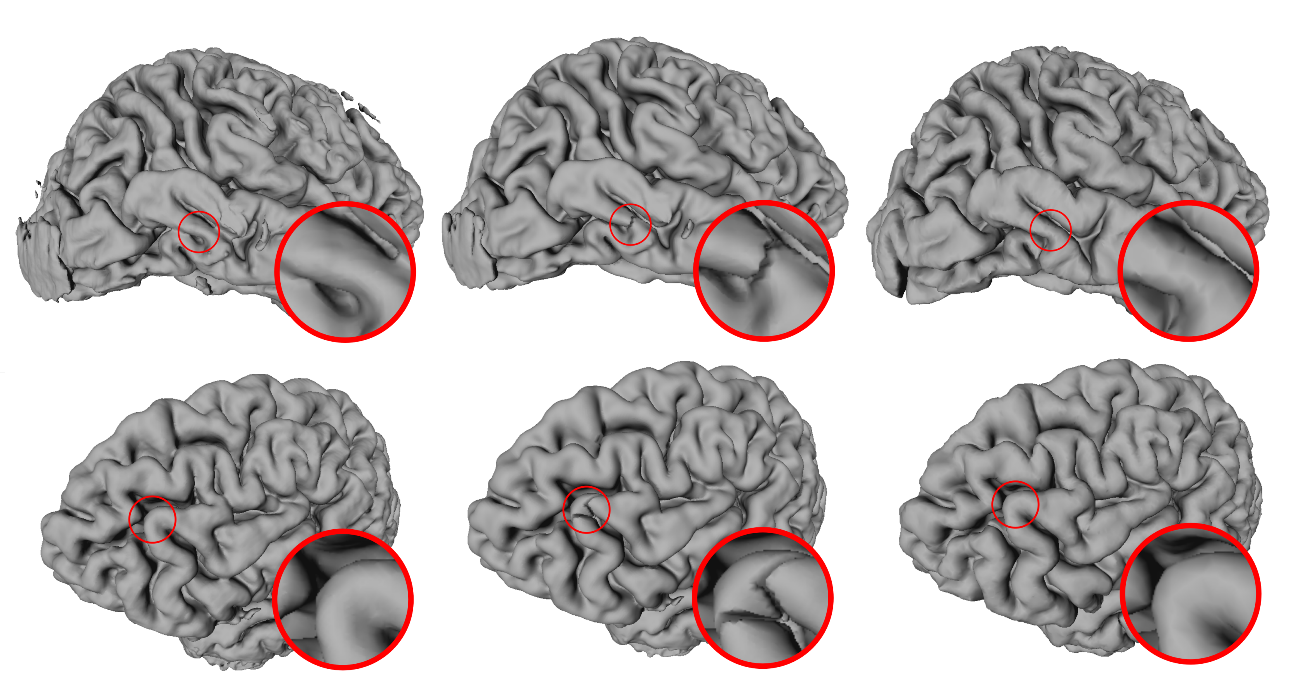

Figure 4. Visualization of incorrect anatomy due to topology correction. We show pial surfaces from 2 different patients from the OASIS dataset. Left: prediction by DeepCSR before topology correction, middle: after topology correction, right: FreeSurfer pseudo ground truth.

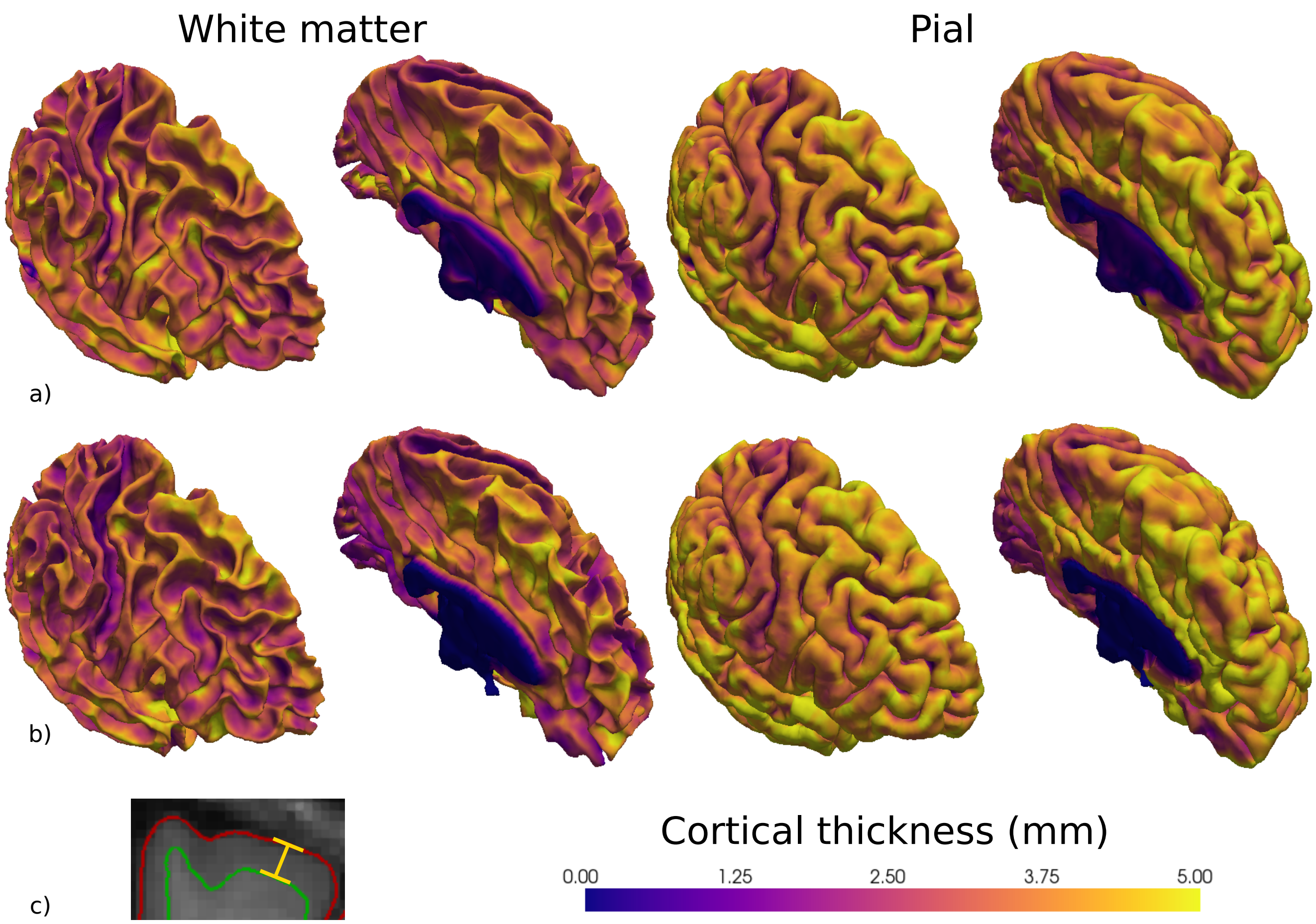

Figure 5. OASIS meshes color-coded with cortical thickness per vertex in mm. a) Vox2Cortex meshes, b) FreeSurfer meshes, c) cortical thickness between white matter (green) and pial (red) surface.