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

Fabian Bongratz\textsuperscript{1,}\textsuperscript{*}, Anne-Marie Rickmann\textsuperscript{1,2,}\textsuperscript{*}, Sebastian Pölsterl\textsuperscript{2}, Christian Wachinger\textsuperscript{1,2}

\textsuperscript{1}Technical University of Munich, \textsuperscript{2}Ludwig-Maximilians-University Munich

\begin{figure}[h]
\centering
\includegraphics[width=0.5\textwidth]{figure1.png}
\caption{Ground-truth points $a$ and $b$ with curvature $\kappa(a) < \kappa(b)$ and predicted points $u$ and $v$.}
\end{figure}

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., $u' = u - \lambda \frac{\partial L_C(a,u)}{\partial u}$ 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

$$\frac{\partial L_C}{\partial u} = \frac{\partial}{\partial u} \left[ \kappa(a) \left( \|a - u\|^2 + \|u - a\|^2 \right) \right] = 4\kappa(a)(u - a).$$

The calculation of $\frac{\partial L_C}{\partial u}$ works analogously. The parameter updates are given by

$$u' = u - \frac{\partial L_C}{\partial u} = u + 4\lambda\kappa(a)(a - u),$$
$$v' = v - \frac{\partial L_C}{\partial v} = v + 4\lambda\kappa(b)(b - v).$$

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

\textbf{Binary cross entropy} The cross-entropy loss between a predicted binary segmentation $B^p_i \in \{0, 1\}^{HWD}$ and a label $B^s_i \in \{0, 1\}^{HWD}$, where voxels are enumerated from 1 to $N = HWD$, is defined as

$$L_{BCE}(B^p_i, B^s_i) = -\frac{1}{N} \sum_{i=1}^{N} \left[ B^s(i) \log B^p(i) + (1 - B^s(i))(1 - \log B^p(i)) \right],$$

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

\textbf{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 \textit{intra-mesh normal consistency}, and between two meshes, which we call \textit{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 $\mathcal{P}^p_{s,c}$, $\mathcal{P}^s_{s,c}$ be predicted and ground-truth point sets with associated normals $\mathcal{N}^p_{s,c}$,
While many works [7, 8, 4] smooth the mesh with respect to vertex coordinates \( V_{p,c} \), we got inspired by [9] and apply the Laplacian operator to the displacement field \( \Delta_{p,c} \). Our ablation study confirms that this is a good choice. More precisely, \( \Delta_{p,c} \) represents the displacement vectors moving the vertices \( V_{p-1,c} \) to \( V_{p,c} \), i.e., \( V_{p,c} = V_{p-1,c} + \Delta_{p,c} \) transforms the mesh \( M_{p-1,c} \) into \( M_{p,c} \).

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) \( L_{\text{ap}} \) is considered to be a constant, i.e., the loss is not backpropagated through the creation of \( L_{\text{ap}} \).

**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

\[
\mathcal{L}_{\text{edge}}(M_{\text{p,c}}) = \frac{1}{|E_{\text{p,c}}|} \sum_{(i,j) \in E_{\text{p,c}}} \| v_i - v_j \|^2. 
\]

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:

<table>
<thead>
<tr>
<th>Surface</th>
<th>( \lambda_{1,c} )</th>
<th>( \lambda_{2,c} )</th>
<th>( \lambda_{3,c} )</th>
<th>( \lambda_{4,c} )</th>
<th>( \lambda_{5,c} )</th>
</tr>
</thead>
<tbody>
<tr>
<td>c = wm</td>
<td>1.0</td>
<td>0.01</td>
<td>0.1</td>
<td>0.001</td>
<td>5.0</td>
</tr>
<tr>
<td>c = pial</td>
<td>1.0</td>
<td>0.0125</td>
<td>0.25</td>
<td>0.00225</td>
<td>5.0</td>
</tr>
</tbody>
</table>

Mesh-loss function weights for inter-mesh normal consistency \( \mathcal{L}_{n,\text{inter}} \), intra-mesh normal consistency \( \mathcal{L}_{n,\text{intra}} \), and Laplacian smoothing \( \mathcal{L}_{\text{Lap}} \) 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.
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/CRCMAX/repos/deepcsr/browse and Vox2Mesh [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 ADNI\textsubscript{small}) and 40 epochs (ADNI\textsubscript{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 ADNI\textsubscript{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

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.

<table>
<thead>
<tr>
<th>Method</th>
<th>Pial Surfaces</th>
<th>WM Surfaces</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>CC</td>
<td>genus</td>
</tr>
<tr>
<td>Ours</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>DeepCSR</td>
<td>48.6</td>
<td>152.4</td>
</tr>
<tr>
<td>DeepCSR + top. corr.</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

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.

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.