Hybrid-CSR: Supplementary Materials

Shanlin Sun[1]
shanlins@uci.edu
Tung Le[1]
thanhtul@uci.edu
Pooya Khosravi[1]
pooyak@uci.edu
Chenyu You[2]
chenyu.you@yale.edu
Kun Han[1]
khan7@uci.edu
Haoyu Ma[1]
haoyum3@uci.edu
Deying Kong[1]
deyingk@uci.edu
Xiangyi Yan[1]
xiangyy4@uci.edu
Xiaohui Xie[1]
xhx@uci.edu

[1] Department of Computer Science
University of California, Irvine
CA, USA

[2] Department of Electrical Engineering
Yale Univesity
CT, USA

1 Pipeline Review

Fig. 1 depicts the whole pipeline of Hybrid-CSR. From the template meshes $\mathcal{M}_T$, Hybrid-CSR first obtains coarsely deformed cortical meshes $\mathcal{M}_c$, given which, we estimate the positions and normals of upsampled oriented point cloud $\mathcal{O}_{up}$. Then the cortical surfaces $\hat{\mathcal{M}}$ can be reconstructed via poisson surface reconstruction from $\mathcal{O}_{up}$. To fix the topology defects in $\hat{\mathcal{M}}$, we extract non-zero level set from the signed distance grids, obtaining topologically correct meshes $\hat{\mathcal{M}}_{tc'}$, and apply optimization-based diffeomorphic registration to recover the accurate and smooth genus-0 cortical surfaces $\hat{\mathcal{M}}_{tc}$. Lastly, we refine $\hat{\mathcal{M}}_{tc}$ using a learning-based diffeomorphic transformation model, to achieve our final reconstruction results $\hat{\mathcal{M}}_f$.

2 Implementation Details

Hybrid-CSR framework is implemented using PyTorch [9] and executed on a system equipped with an NVIDIA RTX A6000 GPU and an Intel i7-7700K CPU.

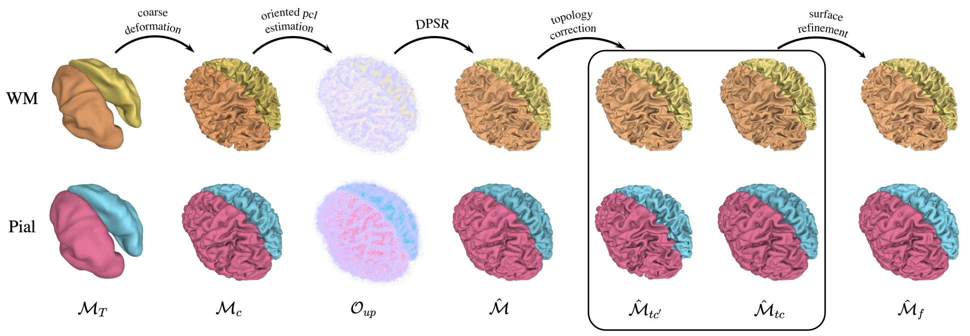

Figure 1: **The pipeline of Hybrid-CSR**.

2.0.1 Network Architecture

To model contour displacements, we encoder positions with random Fourier mapping [12] and approximate the deformation field with a SIREN [11] model. The gaussian scale and embedding length of position encoding are 5 and 128. The hidden features size and hidden layers number are 256 and 2. Our hybrid method directly optimizes the positions and normals of oriented points.

2.0.2 Optimization

To optimize neural fields for contour deformation, 1000 points with normals are separately sampled from ground truth contour and deformed coutour in each iteration. The loss function consists of geometry-consistency loss and regularization loss. For the geometry-consistency loss, we add up the chamfer distance $\mathcal{L}_{cd}$ and normal distance $\mathcal{L}_{nd}$ between two sets of 1000 sampled oriented points. The regularization loss consists of edge length $\mathcal{L}_{edge}$ [13], as well as normal consistency regularization $\mathcal{L}_{nc}$. The total mesh loss is as below:

$$\mathcal{L} = \mathcal{L}_{cd} + 0.02 * \mathcal{L}_{nd} + 0.005 * \mathcal{L}_{edge} + 0.005 * \mathcal{L}_{nc} \quad (1)$$

We apply Adam optimizer [6] with a learning rate of $1e^{-4}$ for 3000 iterations to update the parameters of neural fields.

For the hybrid method, we first uniformly sample 1000 points with normals from the deformed contour obtained above, as initializations. To optimize the positions and normals of oriented points, we minimize the L2 loss between the indicator map reconstructed from the ground truth and the optimized oriented points. We apply Adam optimizer with a learning rate of $3e^{-3}$ for 1000 iterations.

2.0.3 Results of diffeomorphic transformation

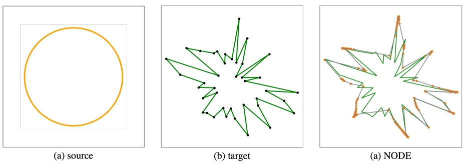

Figure 2: Explicit contour representation

In NMF [4], they present that diffeomorphic transformation can avoid the "regularizer's dilemma", but in our toy experiment, we found neural ode (NODE) [2] is not suitable to model large and sharp deformations, as is shown in Fig. 2. We model the dynamic function of NODE, i.e., neural velocity fields, using the same neural fields as that for contour deformation and optimize with only chamfer distance and normal distance without any regularization loss, optimized via Adam optimizer with a learning rate of $1e^{-4}$ for 3000 iterations.

2.1 Coarse Mesh Deformation

2.1.1 Network Architecture

We apply Vox2Cortex [1] to deform template meshes. Same as Vox2Cortex,

- we train the volumetric segmentation branch and mesh deformation branch end-to-end;
- volumetric segmentation branch is based on the Res-Unet;
- we take four cortical surfaces template as one and use residual GCN-based modules to encode features on graphs;
- we use the same feature extraction strategy. That is, in the first step of mesh deformation, the volumetric feature associated with meshes vertices are from the 4th, 5th, 6th and 7th layers of Res-Unet. And in the second step of mesh deformation, the meshes vertices are extracted from the 3rd, 4th, 7th and 8th layers of Res-Unet;

Different from Vox2Cortex, in the coarse mesh deformation module of Hybrid-CSR

- we deform template meshes in two steps, instead of four steps. In other words, our superiority in performance doesn't come from more steps of surface reconstruction;
- In both training and inference, we use smaller templates (≈ 42000 vertices per surface).

2.1.2 Training

Same as Vox2Cortex, we apply the loss function composed of voxel loss $\mathcal{L}_vox$, curvature-weighted chamfer loss $\mathcal{L}$, normal distance, laplacian smoothing, normal consistency as well as edge length regularizations. The coarse mesh deformation module as well as segmentation branch are first optimized for 50 epochs and then they will be optimized together with the oriented point cloud estimation module for another 100 epochs. The other implementation details can be found in the supplementary material of Vox2Cortex.

2.2 Oriented Point Cloud Estimation

2.2.1 Gated Linear Unit (GLU) for Point Estimation

Let S denote the upsample ratio, $p_i^{up} \in \mathbb{R}^{S\times 3}$ denote the position of upsampled oriented point clouds, $v_i^{up} \in \mathbb{R}^{S\times 3}$ denote the vertex of deformed meshes repeated by S times and p_i^{up} denote the upsampled displacements of vertex v_i. Let's also define the intermediate position of upsampled oriented point clouds as $p'^{up}_i \in \mathbb{R}^{S\times 3}$, such that

$$p_i^{up} = (1 - m_i) \odot v_i^{up} + m_i \odot p'^{up}_i \tag{2}$$

where each dimension of m_i is between 0 and 1, controlling the "confidence" of the intermediate results. We can represent p'^{up}_i as $v^{up}_i + d'_i$, where d'^{up}_i is intermediate displacement associated with upsampled vertex v^{up}_i. Therefore, the Eq. 2 can be rewritten as:

$$p^{up}_i = v^{up}_i + m_i \odot d'^{up}_i \tag{3}$$

As we have demonstrated in the main paper, using GLU, the displacements can be written as $d_i = (\mathbf{W}_0\mathbf{f}_i + \mathbf{b}_0) \odot \sigma(\mathbf{W}_1\mathbf{f}_i + \mathbf{b}_1)$. Then, we have

$$(\mathbf{W}_0\mathbf{f}_i + \mathbf{b}_0) \odot \sigma(\mathbf{W}_1\mathbf{f}_i + \mathbf{b}_1) = m_i \odot d'^{up}_i \tag{4}$$

Thus, we suppose $\mathbf{W}_0 \in \mathbb{R}^{(S\times3)\times(d_{out}+S\times3)}$ and $\mathbf{b}_0 \in \mathbb{R}^{(S\times3)}$ modulate the displacement vector, while $\mathbf{W}_1 \in \mathbb{R}^{(S\times3)\times(d_{out}+S\times3)}$ and $\mathbf{b}_1 \in \mathbb{R}^{(S\times3)}$ modulate the "confidence " of the predicted displacement vector. In our experiments, $S = 7$ and $d_{out} = 64$.

2.2.2 Network Architecture for Normal Estimation

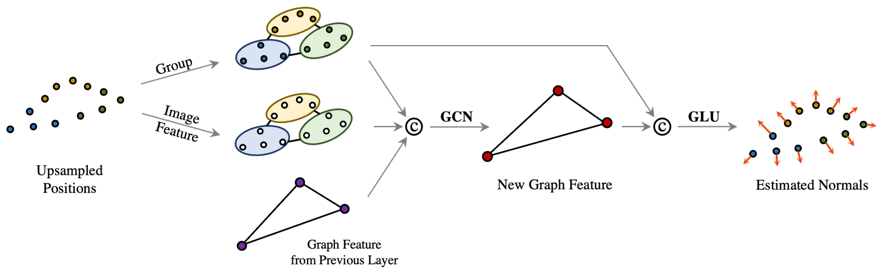

Figure 3: **Network Architecture of Normal Estimation Module.** We apply GCN to encode features on graph and GLU to estimate normals of upsampled positions.

As is shown in Fig. 3, GCN is used to encode features on graph and GLU is used to estimate normals of upsampled positions. The input of GCN is the concatenation of grouped positions, grouped image features associated with points and graph feature generated by the previous GCN layer. The positions and associated features will be grouped into the same node if they are displaced from the same vertices. We sample the image features from the 2nd, 3rd, 8th and 9th layers of Res-Unet given the point positions via linear interpolation, and the channel number of these volumetric features are 32, 64, 16 and 8. The graph feature from the previous layer is in length of 64. Thus, the input feature channel number of GCN is $S*(3+(32+64+16+8))+64 = 925$ and the output feature channel number is 64.

The output of GCN concatenated with the grouped positions is taken as the input of GLU, whose input feature channel number is therefore $64+S*3 = 85$. The output of GLU is the normals of upsampled point clouds, thus the output channel number is $S*3 = 21$.

2.2.3 Training

We apply weighted mean square error $\mathcal{L}_{DPSR}$ to measure the difference between predicted and ground truth indicator grids. The weight map is the smoothed edge map of the ground truth indicator grid. Together with mesh-based loss proposed in Vox2Cortex, $\mathcal{L}_{DPSR}$ is used to optimize Hybrid-CSR in an end-to-end manner for additional 100 epochs. The parameters of oriented point cloud estimation module are optimized via an Adam optimizer of a learning rate of $5e^{-5}$.

2.3 Topology Correction

We model the neural velocity fields $\mathcal{F}_\theta$, using the same network architecture as described in Sec. 2.0.1 and Sec. 2.0.3. In the forward pass, the destination position $\hat{p}_{tc} \in \hat{\mathcal{V}}_{tc}$ starting from $\hat{p}_{tc'} \in \hat{\mathcal{V}}_{tc'}$ is estimated by integrating $\mathcal{F}_\theta(p)$ from $t = 0$ to $t = 1$ via 4th-order Runge–Kutta methods with step size being 0.2. For backpropagation, NODE adopts the adjoint sensitivity method [10], which retrieves the gradient by solving the adjoint ODE backwards in time and allows solving with O(1) memory usage no matter how many steps the ODE solver takes. The network parameters θ are optimized with chamfer distance using the Adam optimizer for 75 iterations with a step size of 3e-4. To compute the chamfer distance, we sample 150,000 points with normals from both $\hat{\mathcal{M}}_{tc}$ and $\hat{\mathcal{M}}_{tc'}$ per iteration. Other implementation details are included in the main paper.

Different from the toy example, in the procedure of topology correction, the source surface $\hat{\mathcal{M}}_{tc'}$ and target surface $\hat{\mathcal{M}}_{tc}$ have already been well-aligned, so that diffeomorphic transformation optimized by chamfer distance is able to provide accurate surface registration performance.

2.4 Surface Refinement

The network architecture of surface refinement is the same as CortexODE [7]. But there are some differences in training pial surface refinement model. In the original CortexODE, they learn to map the ground truth WM surface to the ground truth pial surface. Since the ground truth WM and pial surface share the same topology, they can train with the L2 distance. However, during inference, the source surface is the predicted WM surface obtained from Marching Cubes so there exists a discrepancy between inference and training. Instead, we learn to map the topological correct pial surface to the ground truth pial surface, supervised by chamfer distance. And during inference, the initial pial surface is also generated by the topology correction procedure. In terms of other implementation details, we follow the original CortexODE.

3 Dataset

ADNI We use a subset of the ADNI dataset [5] containing a total of 419 T1-weighted (T1w) brain MRI from subjects aged from 55 to 90 years old. We stratify the dataset into 299 scans for training ($\approx 70\%$), 40 scans for validation($\approx 10\%$), and 80 scans for testing ($\approx 20\%$). We report all of our experiment results on the test set.

OASIS For the OASIS dataset [8], we use all of 416 T1-weighted (T1w) brain MRI images. We stratify the dataset into 292 scans for training ($\approx 70\%$), 44 scans for validation ($\approx 10\%$), and 80 scans for testing ($\approx 20\%$). We report all of our experiment results on the test set.

Test-retest To analyze the consistency of our approach, we evaluate all 120 scans from three different subjects, where each subject is scanned twice in 20 days.

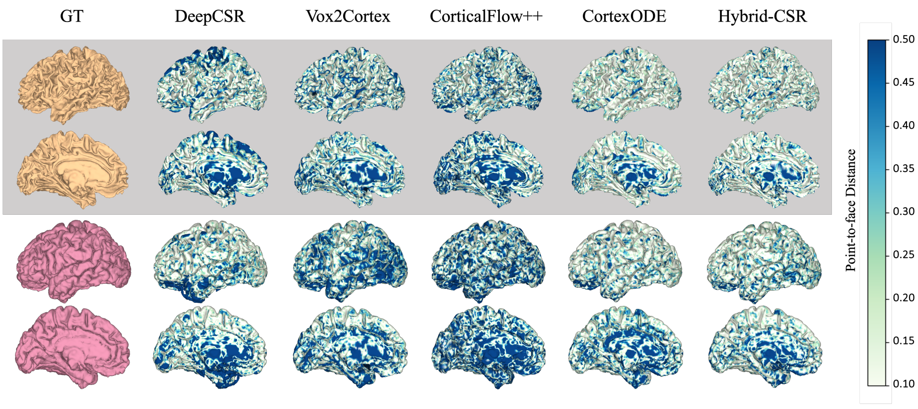

Figure 4: Visual Comparison on OASIS dataset

3.1 Visual Comparisons on OASIS dataset

Fig. 4 present comparisons on OASIS dataset, between our Hybrid-CSR and other competing methods, including Vox2Cortex, CorticalFlow++, CortexODE as well as DeepCSR [3].

References

[1] Fabian Bongratz, Anne-Marie Rickmann, Sebastian Pölsterl, and Christian Wachinger. Vox2cortex: fast explicit reconstruction of cortical surfaces from 3d mri scans with geometric deep neural networks. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20773–20783, 2022.

[2] Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

[3] Rodrigo Santa Cruz, Leo Lebrat, Pierrick Bourgeat, Clinton Fookes, Jurgen Fripp, and Olivier Salvado. Deepcsr: A 3d deep learning approach for cortical surface reconstruction. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 806–815, 2021.

[4] Kunal Gupta. *Neural mesh flow: 3d manifold mesh generation via diffeomorphic flows*. University of California, San Diego, 2020.

[5] Clifford R Jack Jr, Matt A Bernstein, Nick C Fox, Paul Thompson, Gene Alexander, Danielle Harvey, Bret Borowski, Paula J Britson, Jennifer L. Whitwell, Chadwick Ward, et al. The alzheimer's disease neuroimaging initiative (adni): Mri methods. *Journal of Magnetic Resonance Imaging: An Official Journal of the International Society for Magnetic Resonance in Medicine*, 27(4):685–691, 2008.

[6] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[7] Qiang Ma, Liu Li, Emma C Robinson, Bernhard Kainz, Daniel Rueckert, and Amir Alansary. Cortexode: Learning cortical surface reconstruction by neural odes. *IEEE Transactions on Medical Imaging*, 2022.

[8] Daniel S Marcus, Tracy H Wang, Jamie Parker, John G Csernansky, John C Morris, and Randy L Buckner. Open access series of imaging studies (oasis): cross-sectional mri data in young, middle aged, nondemented, and demented older adults. *Journal of cognitive neuroscience*, 19(9):1498–1507, 2007.

[9] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019.

[10] Lev Semenovich Pontryagin. *Mathematical theory of optimal processes*. CRC press, 1987.

[11] Vincent Sitzmann, Julien Martel, Alexander Bergman, David Lindell, and Gordon Wetzstein. Implicit neural representations with periodic activation functions. *Advances in Neural Information Processing Systems*, 33:7462–7473, 2020.

[12] Matthew Tancik, Pratul Srinivasan, Ben Mildenhall, Sara Fridovich-Keil, Nithin Raghavan, Utkarsh Singhal, Ravi Ramamoorthi, Jonathan Barron, and Ren Ng. Fourier features let networks learn high frequency functions in low dimensional domains. *Advances in Neural Information Processing Systems*, 33:7537–7547, 2020.

[13] Nanyang Wang, Yinda Zhang, Zhuwen Li, Yanwei Fu, Wei Liu, and Yu-Gang Jiang. Pixel2mesh: Generating 3d mesh models from single rgb images. In *Proceedings of the European conference on computer vision (ECCV)*, pages 52–67, 2018.