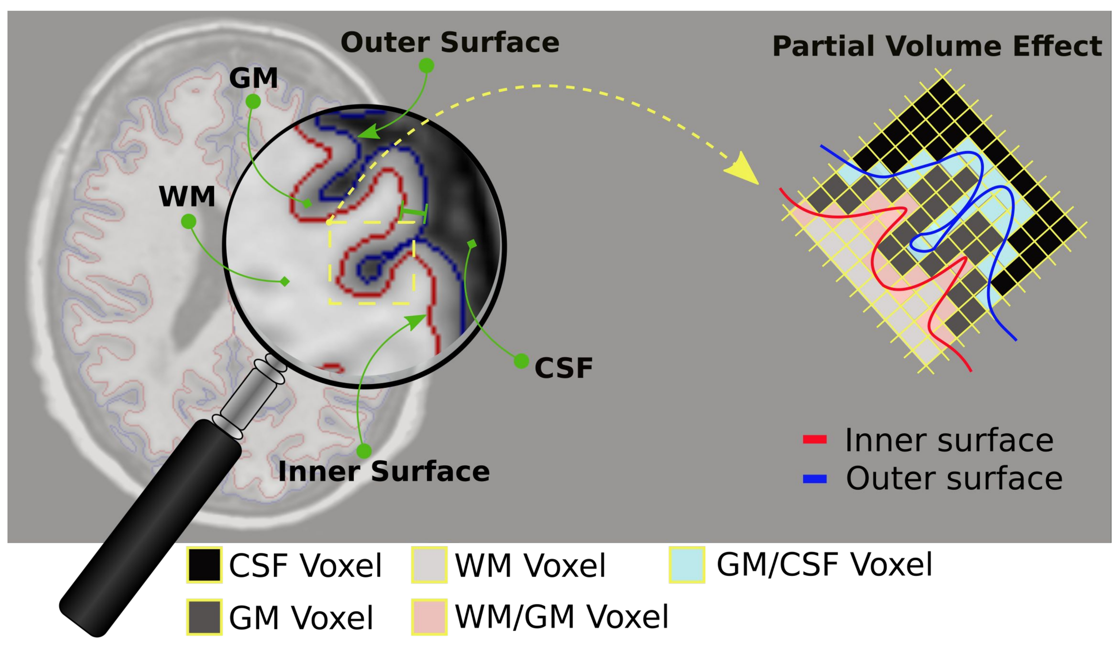
CorticalFlow++: Boosting Cortical Surface Reconstruction Accuracy, Regularity, and Interoperability

Rodrigo Santa Cruz[1,2,*], Léo Lebrat[1,2,*], Darren Fu[3], Pierrick Bourgeat[1], Jurgen Fripp[1], Clinton Fookes[2], and Olivier Salvado[1]

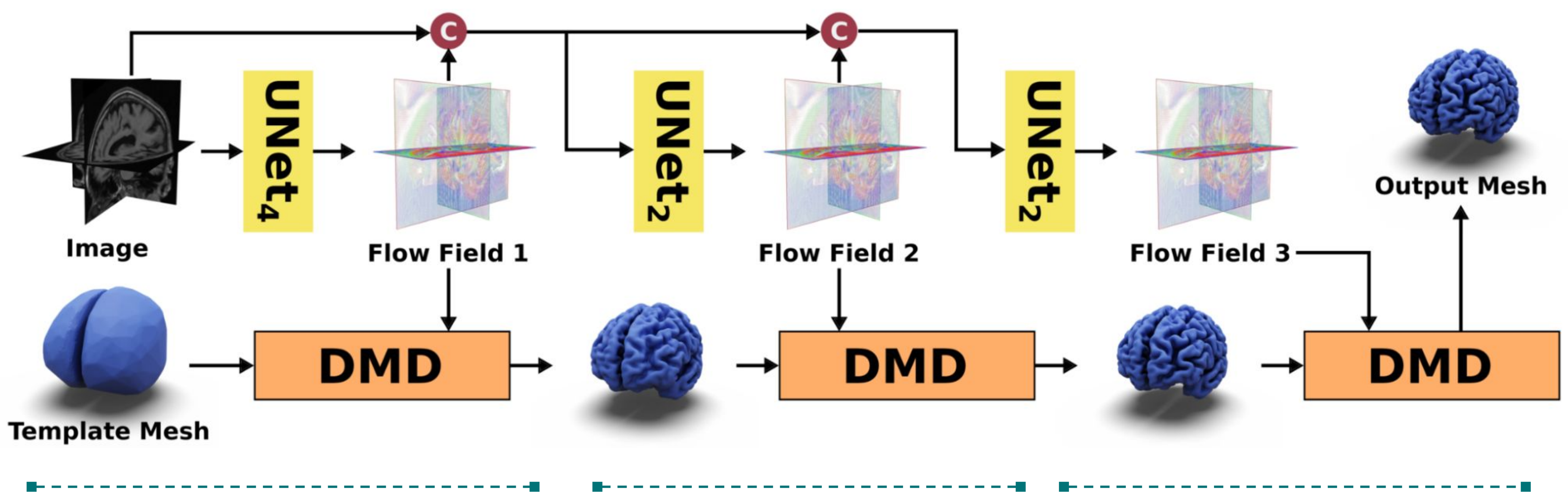
[1]The Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia. [2] Queensland University of Technology(QUT), Australia. [3] University Of Queensland (UQ), Australia. [*]Equal contribution.

This work improves CorticalFlow's accuracy, regularity, and interoperability with existing surface analysis tools without severely degrading its inference time and GPU memory consumption.

1) Cortical Surface Reconstruction from MRI (CSR):



2) CorticalFlow Model



$$\mathrm{CF}^{i+1}_{\theta_{i+1}}(\mathbf{I}, \mathcal{T}_{i+1}) = \mathrm{DMD}\Big(\mathrm{UNet}^{i+1}_{\theta_{i+1}}(\mathbf{U}_1^\frown \cdots \mathbf{U}_i^\frown \mathbf{I}), \mathrm{CF}^{i}_{\theta_i}(\mathbf{I}, \mathcal{T}_{i+1})\Big)$$

$$\mathrm{CF}^{1}_{\theta_1}(\mathbf{I}, \mathcal{T}_1) = \mathrm{DMD}\Big(\mathrm{UNet}^{1}_{\theta_1}(\mathbf{I}), \mathcal{T}_1\Big)$$

3a) Higher Order ODE Solver

DMD modules compute per vertex diffeomorphic mappings Φ from the predicted flow field **U** by solving the flow ODE,

$$\frac{d\Phi(s;\mathbf{x})}{ds} = \mathbf{U}\left(\Phi(s;\mathbf{x})\right), \text{with} \quad \Phi(0;\mathbf{x}) = x$$

CorticalFlow uses the forward **Euler** method,

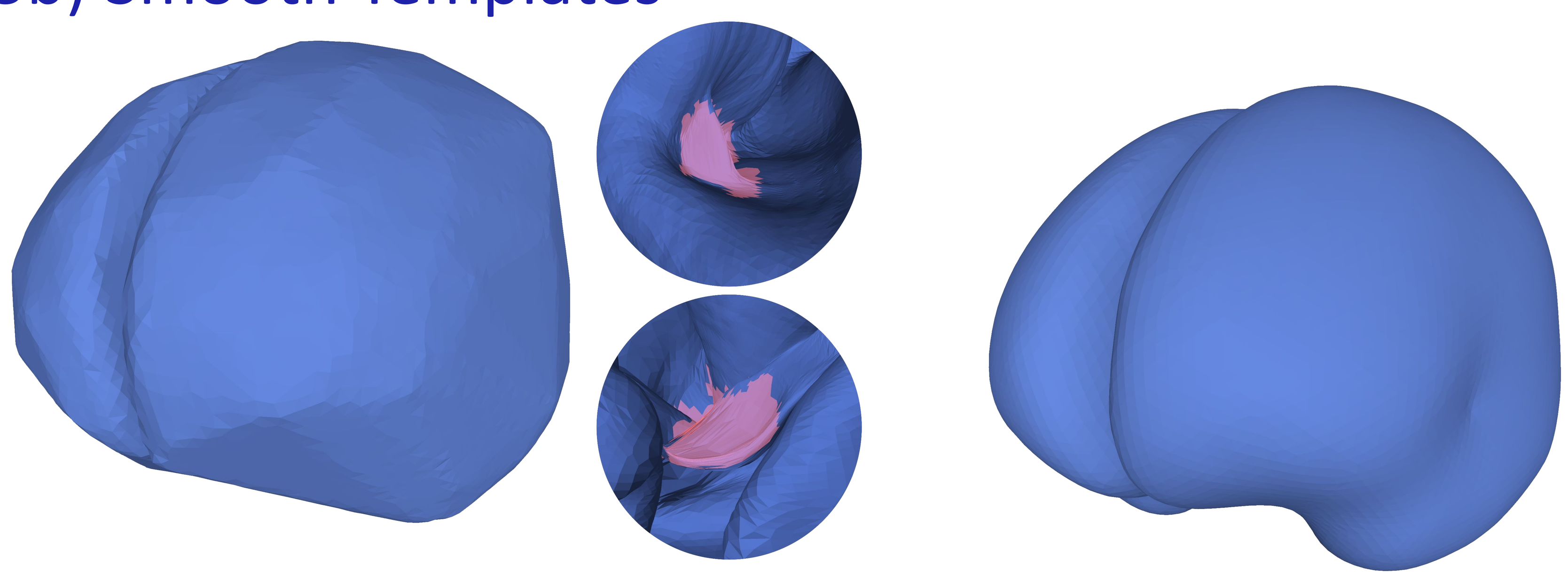
$$\hat{\Phi}(h, \mathbf{x}) = \mathbf{x} + h\mathbf{U}(\mathbf{x}),$$

While **CorticalFlow++** uses the **RK4** method:

$$\hat{\hat{\Phi}}(h, \mathbf{x}) = \mathbf{x} + \frac{1}{6}[k_1 + 2k_2 + 2k_3 + k_4]$$

$$k_1 = \mathbf{U}(\mathbf{x}) \quad k_2 = \mathbf{U}\left(\mathbf{x} + h\frac{k_1}{2}\right) \quad k_3 = \mathbf{U}\left(\mathbf{x} + h\frac{k_2}{2}\right) \quad k_4 = \mathbf{U}(\mathbf{x} + hk_3)$$

3b) Smooth Templates



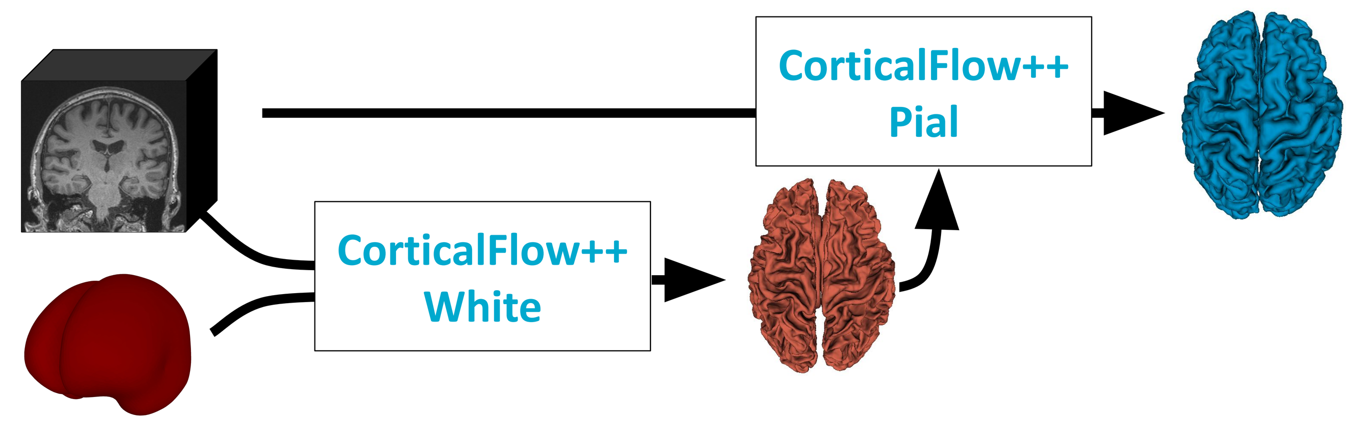
CorticalFlow's template and generated mesh artifacts — **CorticalFlow++'s template**

Procedure:

1. Compute a **signed distance grid** for every training mesh.
2. Threshold and compute the **binary union** of these grids.
3. Use **marching cubes** algorithm to obtain a coarse mesh.
4. Smooth the coarse mesh by applying **laplacian smoothing**.
5. Remesh using **delaunay triangulation**.

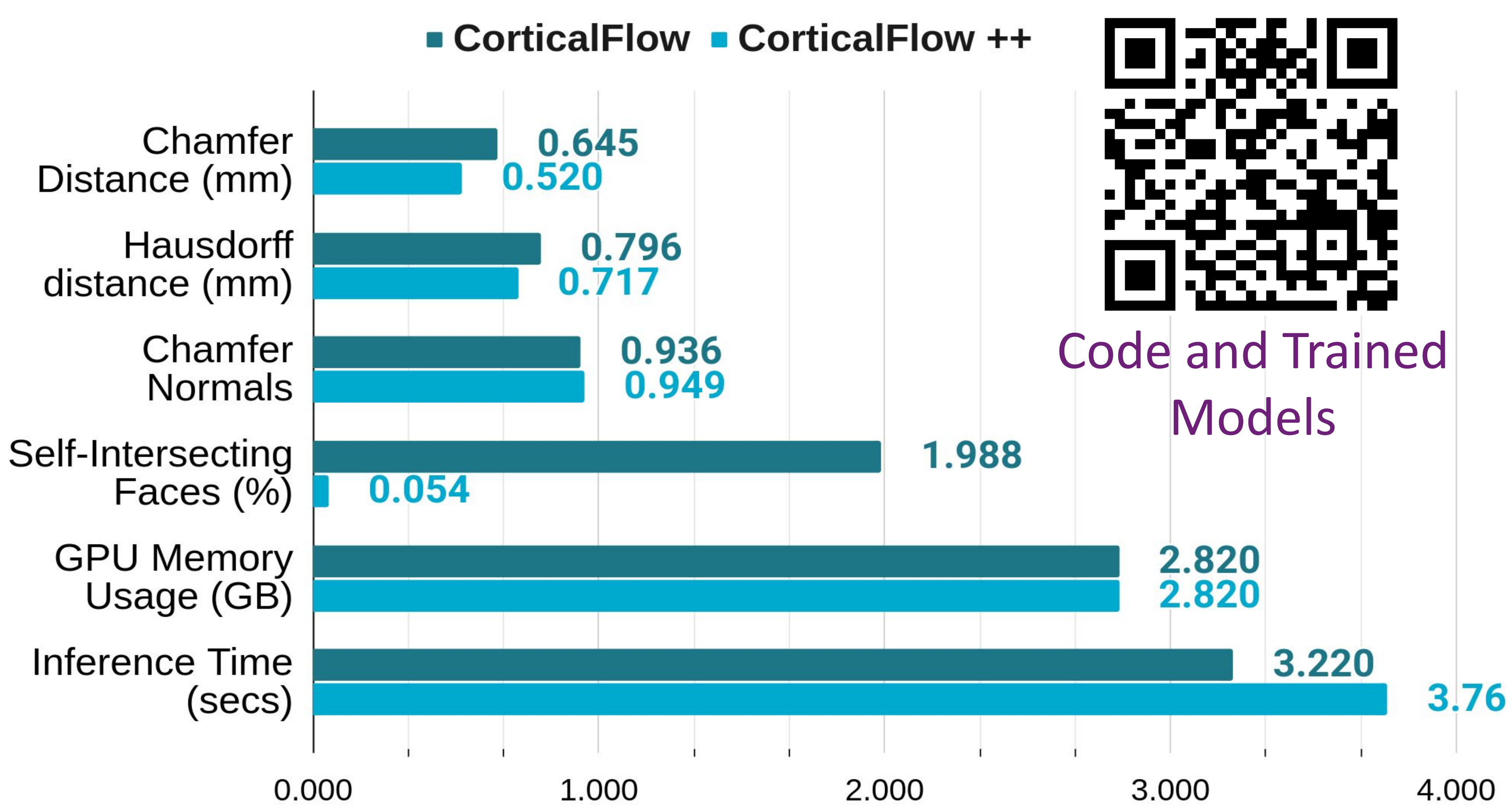
3c) White To Pial Surface Morphing

CorticalFlow deforms separate template meshes leading to reconstructed meshes **without a one-to-one mapping** between the vertices in the white and pial surfaces. Instead, **CorticalFlow++** predicts pial surfaces by deforming its corresponding **predicted white surface**.



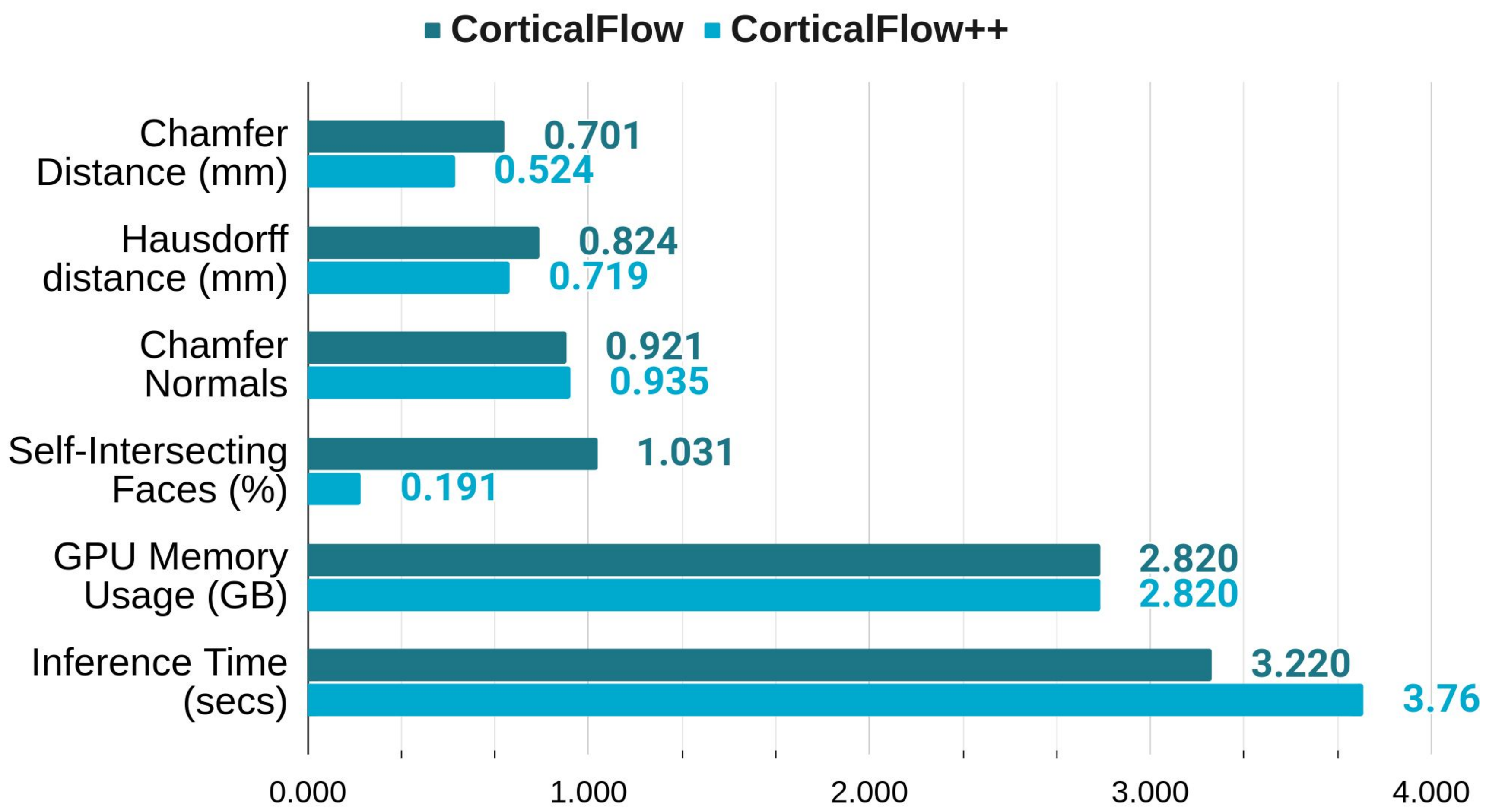
4) Results:

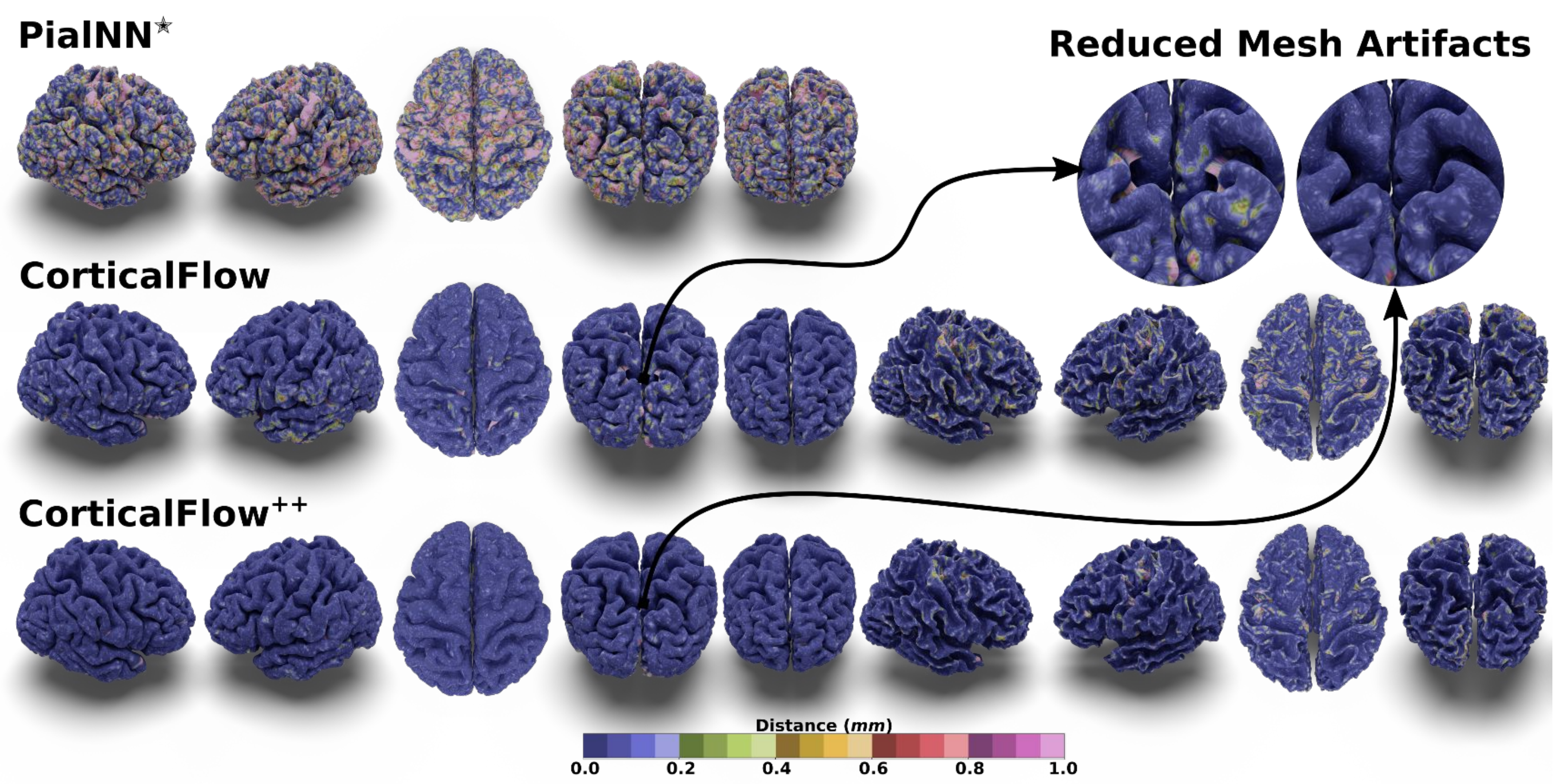
ADNI Benchmark



Code and Trained Models

OASIS3 (Out-Of-Train Distribution)





Contact us:

rodrigo.santacruz@csiro.au

leo.lebrat@csiro.au