
Supplementary Materials for “Unsupervised Trajectory Optimization for 3D Registration in Serial Section Electron Microscopy using Neural ODEs”

Anonymous Author(s)

Affiliation

Address

email

1 Appendix

2 A1 Dataset Details and Metrics

3 A1.1 Dataset Details

4 We selected six publicly available datasets from the OpenOrganelle platform [26] for simulation
5 experiments. These datasets encompass high-resolution electron microscopy images of various
6 mouse tissues, including the heart, kidney, liver, skin, and pancreas. The availability of ground-truth
7 annotations in these datasets provides strong support for validating the applicability and robustness
8 of our method across different types of biological tissues. For real-world data, we utilized the female
9 fruit fly brain neural dataset (FemFlyBrain) [22]. These extensive datasets enabled us to validate the
10 robustness of our method against real-world data.

11 **P7 Mouse Heart.** This dataset consists of heart tissue extracted from a wild-type C57BL/6J mouse at
12 postnatal day 7. The experimental procedure followed IACUC guidelines for animal anesthesia. The
13 tissue was fixed via perfusion with glutaraldehyde and sectioned using a vibratome. Subsequently,
14 the sample underwent low-temperature reducing OTO staining, dehydration through a graded ethanol
15 series, infiltration with Durcupan resin, and polymerization in an oven at 60°C.

16 **Mouse Kidney.** This dataset consists of kidney tissue extracted from an 8-week-old wild-type
17 C57BL/6 mouse. The tissue was perfused with glutaraldehyde fixative and sectioned using a
18 vibratome. After staining with reducing OTO at room temperature, the sample underwent dehydration,
19 infiltration with a graded ethanol series, and Durcupan resin. Finally, the sample was polymerized in
20 an oven at 60°C.

21 **P7 Mouse Liver.** This dataset comprises liver tissue collected from a wild-type, postnatal day 7
22 C57BL/6J mouse. Following anesthesia in accordance with IACUC guidelines, the sample was
23 perfused with glutaraldehyde fixative and sectioned using a vibratome. After low-temperature
24 reducing OTO staining, the tissue underwent dehydration, infiltration with graded ethanol and
25 Durcupan resin, and was subsequently polymerized in a 60°C oven.

26 **Mouse Liver.** This dataset contains liver tissue from a wild-type, 8-week-old C57BL/6 mouse. The
27 sample was fixed via perfusion with glutaraldehyde and sectioned using a vibratome. Subsequently,
28 reducing OTO staining was performed at room temperature. The sample was then dehydrated,
29 infiltrated with graded ethanol and Durcupan resin, and polymerized in a 60°C oven.

30 **P7 Mouse Skin.** This dataset includes skin tissue from a wild-type, postnatal day 7 C57BL/6J mouse.
31 Mouse anesthesia was performed in accordance with IACUC guidelines, followed by perfusion
32 fixation using glutaraldehyde and vibratome sectioning. The sample underwent low-temperature

Table 1: Details of the synthetic datasets for testing phase.

Datasets	Shape	Numbers	URL
P7 Mouse Heart	1184×1184	1000	https://openorganelle.janelia.org/datasets/jrc_mus-heart-1
Mouse Kidney	1184×1184	1000	https://openorganelle.janelia.org/datasets/jrc_mus-kidney
P7 Mouse Liver	1184×1184	1127	https://openorganelle.janelia.org/datasets/jrc_mus-liver-3
Mouse Liver	1184×1184	558	https://openorganelle.janelia.org/datasets/jrc_mus-liver
P7 Mouse Pancreas	1184×1184	898	https://openorganelle.janelia.org/datasets/jrc_mus-pancreas-4
P7 Mouse Skin	1184×1184	1231	https://openorganelle.janelia.org/datasets/jrc_mus-skin-1
FemFlyBrain	1024×1024	1299	https://neurodata.io/data/takemura13/

reducing OTO staining, dehydration, infiltration with graded ethanol and Durcupan resin, and was polymerized in a 60°C oven.

P7 Mouse Pancreas. This dataset consists of pancreas tissue from a wild-type, postnatal day 7 C57BL/6 mouse. The sample was fixed via perfusion with glutaraldehyde and sectioned using a vibratome. It underwent low-temperature reducing OTO staining, followed by dehydration, infiltration with graded ethanol and Durcupan resin, and polymerization in a 60°C oven.

Female Fruit Fly Brain Neural Dataset. FemFlyBrain [22] consists of the right hemisphere of a wild-type Oregon R female fruit fly brain. The brain was continuously sectioned at a thickness of 40 nanometers, covering regions including the medulla and downstream neuropils. The sections were imaged at a magnification of 35,000. The connectome within the medulla includes 379 neurons and 8,637 chemical synaptic contacts.

A1.2 Metrics

To quantitatively evaluate the performance of our method, we calculated several metrics, including Normalized Cross-Correlation (NCC), Mutual Information (MI), and Structural Similarity Index (SSIM) between the registered image series and the ground truth (GT). These metrics provide a comprehensive evaluation of the accuracy of our alignment and registration results from different perspectives. Specifically, NCC measures the similarity between the result images and the GT, MI reflects the mutual information between the two, and SSIM evaluates the accuracy based on structural information. Dice is a set similarity metric commonly used to measure the similarity between two samples. Here, we use the Dice score to evaluate the segmentation accuracy of 3D segmentation results on the registered data.

A2 Experimental Details

A2.1 Dataset Setup

For training the trajectory tracking module, we split the original image data into 3000 image pairs $\{I_0, I_1\}$, where I_0 and I_1 represent adjacent images. These images are cropped to a size of 1184×1184 . During training, the data is normalized to the range of $[-1, 1]$. The training-validation split is set to 0.95. Additionally, we randomly select one of the images from $\{I_0, I_1\}$ and apply random elastic deformation to simulate nonlinear distortions. Specifically, we first generate a random deformation field D_{rand} , which indicates the pixel displacement matrix. This matrix is then smoothed using a Gaussian filter, and the resulting deformation is applied to create the deformed images. The deformation process is described by the following formulas:

$$\begin{cases} \phi_i = \alpha \cdot \text{size}(I_i) \cdot \text{Gauss}(D_{\text{rand}}, \sigma \cdot \text{size}(I_i)), \\ I_i = \phi_i \circ I_i, \end{cases} \quad (1)$$

where ϕ_i represents the random deformation field applied to image I_i , \circ denotes the deformation operation, D_{rand} is the generated random displacement field, $\text{Gauss}(\cdot)$ is the 2D Gaussian filter

operator, α controls the displacement magnitude, and σ determines the smoothing extent. In our experiments, we set $\sigma = 0.08$ and $\alpha = 1.0$.

During the testing phase, we cropped all six datasets into image stacks of size $1184 \times 1184 \times N$, where N denotes the number of slices. Additionally, we applied random elastic deformations to all images except the first one to simulate nonlinear distortions. The deformation parameters were set to $\sigma = 0.08$ and $\alpha = 1.0$. Detailed information regarding the datasets can be found in Table 1. Our test data consists of several hundreds to around 1000 samples, which fully demonstrates our method’s capability in handling long sequences and addressing the challenge of error accumulation.

A2.2 Network Architecture and Training Details

The trajectory tracking network adopts a 2D U-Net architecture with residual multi-kernel fusion, consisting of 1) a cascaded feature encoder with hybrid convolutional blocks, and 2) a dense feature decoder with transposed upsampling. The encoder progressively reduces spatial resolution through four strided convolutions (stride=2), doubling the number of channels from an initial 8 to 64. Specifically, the encoder is composed of four convolutional layers with output channels of 8, 16, 32, and 64, respectively. Each convolutional layer includes two convolution operations: the first convolution uses a 3×3 kernel, followed by a second convolution with a larger 7×7 kernel to expand the receptive field, similar to the approach in LKUnet [12]. The decoder consists of four layers and reconstructs dense predictions using four transposed convolutions (kernel size = 2, stride = 2), halving the number of channels from 64 to 2. Skip connections concatenate the encoder features with the corresponding decoder layers to facilitate hierarchical feature fusion. The final displacement field is generated through dual 3×3 convolutions with a Softsign activation. To enable spatial warping, a Spatial Transformer Network (STN) [10] is employed for image registration. The model is optimized using the ADAM optimizer with parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rate is set to 1×10^{-4} , and training is performed for 500 epochs with a batch size of 16. For the regularization term in the loss function, we set $\lambda = 1.5$ to encourage smoothness in displacement field.

Neural ODEs employ a 3D U-Net architecture to parameterize f_θ . The number of network layers and the number of channels per layer are kept consistent with the 2D U-Net used in the trajectory tracking module, but large convolutional kernels are avoided to reduce memory consumption. For training, we use the ADAM optimizer with parameters $\beta_1 = 0.5$ and $\beta_2 = 0.999$. The learning rate is set to 1×10^{-4} , and training is conducted for 500 epochs with a batch size of 4. For the loss function, we set $\lambda = 0.1$ and $\mu = 0.005$. For the Gauss-Seidel method used in trajectory smoothing, we set the number of iterations to 200 and $\lambda = 0.1$. For Neural ODEs, we use the simple Euler solver, with all other parameters kept at their default settings.

A2.3 Training and Testing on Real Data

A key advantage of our method lies in its unsupervised learning paradigm, which is particularly valuable for real-world datasets that lack reliable ground-truth correspondences. The training process on real data is divided into two stages: training of the trajectory tracking module and training of the trajectory smoothing module. We begin by cropping a small spatiotemporal block of size $H \times W \times T$ from the real dataset. Training samples are then constructed following the procedure described in Section A2.1, except that no additional elastic deformation is applied. The trajectory tracking network is subsequently trained in an unsupervised manner, identical to the approach used for synthetic data. For training the trajectory smoothing module, we first apply the previously trained network to obtain estimated trajectories. These are then used for the unsupervised training of the smoothing module, again following the same procedure as used for synthetic data.

For testing on real data, the networks trained on cropped data blocks can be directly applied to the full image stack. We adopt a sliding-window registration strategy to perform registration across the entire long image sequence. Specifically, given the full real image stack $\{I_i\}_{i=0}^{N-1}$, we define a receptive field of moderate size (about 100 slices). Within each windowed region, the trajectory tracking network is used to estimate the displacement trajectories. These trajectories are then refined by the trajectory smoothing module. Subsequently, inverse warping is applied based on the smoothed trajectories to perform image registration. The registered image block is then placed back into its corresponding location in the full stack. The window is shifted along the sequence, and the procedure is repeated until the entire image stack has been registered.

119 A3 Derivation of the Gauss-Seidel Iteration Update Formula

120 The Gauss-Seidel method [8] is a classical algorithm for solving linear systems through local iterative
 121 updates. Its key characteristic lies in accelerating convergence by immediately incorporating the
 122 most recently updated values. For trajectory smoothing, this method can be adapted by introducing a
 123 fidelity constraint. Below is the detailed derivation of the iteration formula. Specifically, to minimize
 124 the energy function:

$$E(u) = \lambda \sum_{i=0}^{N-1} (u_i - u_i^{(0)})^2 + \sum_{i=1}^{N-2} (u_{i-1} - 2u_i + u_{i+1})^2, \quad (2)$$

125 we take the partial derivative of E with respect to each u_i and set it to zero to obtain the optimality
 126 condition.

127 The fidelity term contributes a derivative of:

$$\frac{\partial}{\partial u_i} (\lambda(u_i - u_i^{(0)})^2) = 2\lambda(u_i - u_i^{(0)}). \quad (3)$$

128 The smoothing term consists of a sum of squared second-order differences, where each point u_i
 129 appears in multiple overlapping terms, together with u_{i-1} and u_{i+1} . For clarity, we focus on a
 130 representative term where u_i is the center, to illustrate how the smoothness constraint acts locally on
 131 the trajectory:

$$\frac{\partial}{\partial u_i} (u_{i-1} - 2u_i + u_{i+1})^2 = -4(u_{i-1} - 2u_i + u_{i+1}). \quad (4)$$

132 Combining the derivatives of both terms, we obtain the total gradient:

$$\frac{\partial E}{\partial u_i} = 2\lambda(u_i - u_i^{(0)}) - 4(u_{i-1} - 2u_i + u_{i+1}) = 0. \quad (5)$$

133 Expanding and rearranging terms gives:

$$2\lambda(u_i - u_i^{(0)}) - 4u_{i-1} + 8u_i - 4u_{i+1} = 0. \quad (6)$$

134 Solving for u_i , we derive the following linear equation:

$$(2\lambda + 8)u_i = 2\lambda u_i^{(0)} + 4u_{i-1} + 4u_{i+1}. \quad (7)$$

135 This leads to the Gauss-Seidel iteration update formula:

$$u_i^{(k+1)} = \frac{1}{2\lambda + 8} \left(2\lambda u_i^{(0)} + 4u_{i-1}^{(k+1)} + 4u_{i+1}^{(k)} \right), \quad (8)$$

136 Simplifying further, we obtain

$$u_i^{(k+1)} = \frac{1}{\lambda + 2} \left(\lambda u_i^{(0)} + u_{i-1}^{(k+1)} + u_{i+1}^{(k)} \right), \quad (9)$$

137 where λ is halved from its original value. We use the updated $u_{i-1}^{(k+1)}$ from the current iteration and
 138 the old $u_{i+1}^{(k)}$ from the previous iteration to update $u_i^{(k+1)}$.

139 A4 Algorithmic Details of Bilinear Splatting

140 The goal of bilinear splatting is to map discrete feature points onto a regular grid using bilinear
 141 interpolation. Given a set of normalized coordinates and their associated feature vectors, we need to
 142 compute the grid values using the surrounding grid points. The coordinates are normalized to the
 143 range $[0, H)$ and $[0, W)$, where H and W are the height and width of the target grid. The feature
 144 vectors are associated with these coordinates and are projected to the grid using bilinear interpolation.

145 Let the coordinates of the discrete points be represented as $\text{coords} = \{(y_1, x_1), \dots, (y_N, x_N)\}$,
 146 where each (y_i, x_i) is a 2D coordinate within the normalized range. The corresponding feature

147 vectors for each point are denoted by values = $\{v_1, \dots, v_N\}$, where $v_i \in \mathbb{R}^C$ is the feature vector
 148 for point i , and C is the feature dimension.

149 To perform bilinear splatting, we begin by identifying the four nearest grid points that surround each
 150 continuous coordinate (y_i, x_i) . These grid points form the corners of the smallest axis-aligned square
 151 in the discrete grid that encloses the given coordinate. Specifically, we obtain them by applying the
 152 floor and ceiling operations to both the y - and x -coordinates:

$$\begin{cases} y_{\text{floor}} = \lfloor y_i \rfloor, & x_{\text{floor}} = \lfloor x_i \rfloor, \\ y_{\text{ceil}} = \lfloor y_i \rfloor + 1, & x_{\text{ceil}} = \lfloor x_i \rfloor + 1. \end{cases} \quad (10)$$

153 These points form the basis for distributing the value at (y_i, x_i) across the surrounding pixels based
 154 on their bilinear interpolation weights.

155 Next, the weights for bilinear splatting are calculated based on the fractional parts of the coordinates.
 156 Let $\Delta y = y_i - \lfloor y_i \rfloor$ and $\Delta x = x_i - \lfloor x_i \rfloor$ represent the fractional parts of the coordinates. The four
 157 interpolation weights are given by:

$$\begin{cases} w_1 = (1 - \Delta x)(1 - \Delta y), & w_2 = (1 - \Delta x)\Delta y, \\ w_3 = \Delta x(1 - \Delta y), & w_4 = \Delta x\Delta y. \end{cases} \quad (11)$$

158 These weights represent the contribution of each neighboring grid point to the interpolated value
 159 at the target coordinate (y_i, x_i) . Now, we distribute the feature values to the corresponding grid
 160 locations based on these weights. For each coordinate (y_i, x_i) , we update the feature grid by adding
 161 the weighted value of the point to the grid locations corresponding to the four neighboring points.
 162 This is done using the following scatter operation:

$$\begin{cases} \text{Grid}[y_{\text{floor}}, x_{\text{floor}}] = \text{Grid}[y_{\text{floor}}, x_{\text{floor}}] + v_i \cdot w_1, \\ \text{Grid}[y_{\text{ceil}}, x_{\text{floor}}] = \text{Grid}[y_{\text{ceil}}, x_{\text{floor}}] + v_i \cdot w_2, \\ \text{Grid}[y_{\text{floor}}, x_{\text{ceil}}] = \text{Grid}[y_{\text{floor}}, x_{\text{ceil}}] + v_i \cdot w_3, \\ \text{Grid}[y_{\text{ceil}}, x_{\text{ceil}}] = \text{Grid}[y_{\text{ceil}}, x_{\text{ceil}}] + v_i \cdot w_4. \end{cases} \quad (12)$$

163 The grid values are accumulated at the corresponding grid positions, and the sum of the weights at
 164 each grid location is also accumulated for normalization purposes.

165 Finally, to ensure proper averaging when multiple points contribute to the same grid position, we
 166 normalize the resulting grid by the sum of the weights:

$$\text{Grid}[y, x] = \frac{\sum v_i \cdot w_i}{\sum w_i} \quad (13)$$

167 To avoid division by zero in case of no contribution to a grid location, a small epsilon $\epsilon = 1e - 8$ is
 168 added to the denominator during the normalization step. Thus, the bilinear splatting operation effi-
 169 ciently interpolates the feature values from discrete points to the target grid, with smooth interpolation
 170 properties ensured by the bilinear weights.

171 A5 Limitations and Future Work

172 Our approach can be divided into three main modules: 1) pixel trajectory tracking, 2) dynamics-based
 173 trajectory optimization, and 3) trajectory inversion and image registration. For each module, we
 174 have not explored the impact of more complex frameworks (e.g., using more sophisticated network
 175 architectures) on the registration performance. Additionally, we have not addressed the interference
 176 caused by complex scenarios, such as high-noise artifacts and wrinkles, which may potentially affect
 177 the accuracy of trajectory tracking. Therefore, future work should investigate 3D registration methods
 178 that are robust to artifacts and wrinkles, as well as explore the use of more advanced architectures
 179 architectures for improving registration accuracy and generalizability. Furthermore, Table 3 presents
 180 the registration accuracy and execution time on the Mus Liver-3 dataset, where the test slice resolution
 181 is set to 1024×1024 . Our method achieves the highest SSIM, demonstrating superior accuracy
 182 compared to other approaches. However, due to the computational demands of the ODE solver, it
 183 does not achieve the fastest runtime.

Table 2: Execution time and performance.

	TrakEM2 [19]	EFSR [25]	EMReg [15]	SEMLeSS [18]	GaussReg [29]	Ours
Time(s)	3.92	3.44	0.28	27.11	0.92	2.46
SSIM	0.66	0.52	0.65	0.61	0.82	0.87

A6 Broader Impacts and Discussion

Our work focuses on 3D registration for Series Section Electron Microscopy (ssEM). One of the key advantages of our approach is its fully unsupervised training, which eliminates the reliance on ground truth and makes it applicable to real-world ssEM scenarios. For ssEM applications [28, 5, 6], this is particularly important for the increasingly complex and diverse electron microscopy image datasets [6, 26], as acquiring comprehensive ground truth across various tissue and cell types is challenging. Our method can be applied to the modeling and analysis of biological cell tissues, assisting researchers in exploring the structure and function of cellular tissues.

For other downstream tasks in ssEM, such as 3D segmentation [13, 14, 17], our method provides well-structured image data that aligns with natural biological morphology. By improving the axial continuity of the raw image data while preserving the structural integrity of superstructural details, our approach can enhance the performance of downstream tasks.

For the design of 3D registration algorithms, we propose a novel paradigm from the perspective of trajectory optimization. Our approach can be divided into three main components: 1) pixel trajectory tracking, 2) dynamics-based trajectory optimization, and 3) trajectory inversion and image registration. These three modules are decoupled and interchangeable. In theory, each module can be replaced with a more efficient algorithm. For example, more efficient feature point tracking algorithms [3, 2] or optical flow estimation algorithms [21, 27] can be used for trajectory tracking. Low-pass filtering [11] and convex quadratic programming (QP) [4, 7] can be used for trajectory smoothing. Radial basis function interpolation [24] and other splatting techniques [1] can be used for trajectory inversion. This helps researchers explore more efficient 3D registration methods based on this paradigm.

Due to time and resource constraints, we were unable to validate the performance of our method on larger-scale scenarios, particularly on ultra-high-resolution image data. The challenges posed by computational complexity in larger environments and the variability of data noise remain. In future work, we will refine our approach and adapt it into a microscopy image processing tool suitable for large-scale real-world datasets.

Table 3: Performance of different registration methods on downstream segmentation tasks across six datasets. We evaluate segmentation accuracy using the Dice score (Avg %).

	EMReg [15]	EFSR [25]	SEMLeSS [18]	TrakEM2 [19]	GaussReg [29]	Ours
Mus Heart	0.35 \pm 0.06	0.82 \pm 0.02	0.65 \pm 0.05	0.68 \pm 0.03	0.89 \pm 0.01	0.88 \pm 0.01
Mus Kidney	0.41 \pm 0.07	0.75 \pm 0.02	0.67 \pm 0.02	0.75 \pm 0.02	0.82 \pm 0.01	0.84 \pm 0.01
Mus Liver-3	0.41 \pm 0.08	0.90 \pm 0.02	0.86 \pm 0.02	0.90 \pm 0.02	0.95 \pm 0.01	0.94 \pm 0.01
Mus Liver	0.57 \pm 0.09	0.89 \pm 0.04	0.86 \pm 0.05	0.92 \pm 0.04	0.94 \pm 0.03	0.94 \pm 0.03
Mus Pancreas	0.46 \pm 0.04	0.87 \pm 0.02	0.79 \pm 0.02	0.90 \pm 0.01	0.94 \pm 0.01	0.93 \pm 0.01
Mus Skin	0.50 \pm 0.07	0.87 \pm 0.03	0.76 \pm 0.04	0.90 \pm 0.02	0.94 \pm 0.01	0.94 \pm 0.03

B Additional Quantitative Results and Visualization

B.1 More Experimental Results

Robustness to Error Accumulation. In practical applications, serial section electron microscopy (ssEM) datasets often contain hundreds or even thousands of images. This large number of slices poses a challenge for long-sequence registration, as cumulative errors can easily arise, eventually leading to substantial sequence drift and compromising the accurate reconstruction of the biological specimen’s true 3D structure. To systematically evaluate the ability of our method to suppress such cumulative errors, we conducted comparative experiments on six long-sequence datasets listed in Table 1, focusing on the robustness of our method versus GaussReg [29] and SEMLeSS [18].

Specifically, Figures 6 to 11 illustrate how the registration accuracy changes as the number of slices increases. The results show that our method consistently achieves higher average registration accuracy across the entire sequence, with smaller fluctuations in the accuracy curve. This indicates superior stability and robustness when handling long sequences. Moreover, Figures 1 and 2 present the 3D reconstruction results on the remaining four datasets. It is evident that our method accurately recovers the spatial structures of various biological tissues, further demonstrating its generalizability and robustness across different types of data.

Performance on 3D Segmentation Tasks. 3D segmentation [13, 14] is a important task in serial section electron microscopy (ssEM) and has been widely applied in various biological image analysis domains [9, 16]. High-quality 3D registration plays a crucial role in segmentation tasks, as it can significantly mitigate artifacts caused by structural deformation, scale variations, and differences in imaging modalities, thereby improving segmentation accuracy. To comprehensively evaluate the applicability of our method in 3D segmentation, we conducted experiments on six datasets involving various organelles, including nuclei and mitochondria. Specifically, we employed a segmentation network [23] to perform segmentation on the images registered by each method. The segmentation performance was quantified by computing the Dice similarity coefficient between the predicted results and the ground-truth (GT) label stacks.

Table 3 summarizes the Dice scores for 3D segmentation results across the six datasets. As shown, our method achieves the best performance on three datasets and the second-best on the remaining three, with overall results slightly below those of the supervised method GaussReg (with differences no greater than 0.01). This discrepancy may be attributed to the segmentation model’s sensitivity to subtle misalignments, while our method focuses on global trajectory optimization and may have limitations in handling local details. For instance, as illustrated in Fig. 2 for the Mus Liver-3 dataset, minor local misalignments can still be observed in our registration results. Nevertheless, our method attains accuracy comparable to that of the supervised GaussReg. Furthermore, Figs. 3, 4, and 5 present 3D visualizations of the segmentation results on multiple datasets. These visual results demonstrate that our method can reliably reconstruct the correct 3D structure of biological tissues, leading to clearer and more consistent segmentation boundaries, further validating its practicality in downstream tasks.

Results on Real-World Data. To further evaluate the performance of our method on real-world datasets, we selected four volume samples from the FemFlyBrain dataset [22], with the size of each image block listed in Table 1. These volumes enable us to assess the robustness and generalization

Table 4: Ablation study on the number of iterations in the Gauss-Seidel method.

Iterations	50	100	150	200	250	300
SSIM	0.591	0.656	0.695	0.721	0.724	0.724

Table 5: Ablation study of the components in our method.

Case	Gauss-Seidel	Neural ODEs	SSIM
(a)	✓	✗	0.721
(b)	✗	✓	0.712
(c)	✓	✓	0.746

Table 6: Ablation study of different solvers for Neural ODEs.

Case	Solver	SSIM
(a)	Euler	0.746
(b)	Dopri5	0.749
(c)	RK4	0.748

ability of our method in practical scenarios. We then followed the procedure described in Section A2.3 to train and test our model, and compared it with GaussReg [29].

The registration results are visualized in Figures 12 and 13, where we used the interpolation and visualization tools provided by FIJI [20] to display the side views of the image stacks. As shown in Volume 1 of Figure 12, although GaussReg can reduce nonlinear distortions, it also incorrectly removes some structural textures, resulting in noticeable deviations from the original data. In contrast, our method not only effectively corrects nonlinear distortions in the image stack but also better preserves local structural details.

B.2 More Ablation Experiments

To further evaluate the effectiveness of each component in our method, we conducted a more detailed ablation study on the EFPL dataset. As shown in Table 5, we analyze the contributions of key components in the trajectory smoothing scheme. Specifically, our method can be decomposed into two main modules: the Gauss-Seidel method and Neural ODEs. The Gauss-Seidel method achieves fast convergence through iterative updates, while Neural ODEs effectively capture complex nonlinear deviations by modeling implicit dynamics. The results demonstrate the necessity and effectiveness of both components. In addition, Table 4 explores the impact of different iteration counts in the Gauss-Seidel method. From 0 to 200 iterations, the registration performance improves significantly and then gradually converges. Considering the trade-off between accuracy and efficiency, we select 200 iterations in practice to achieve a good balance. Table 6 further investigates the effect of different solvers for Neural ODEs, including Euler, Dopri5, and RK4. The results show that more accurate solvers lead to slight improvements in registration quality. To avoid additional computational overhead, we adopt the Euler solver, balancing performance and computational cost.

B.3 Explainable Visualization Study

To better understand the performance and explainability of our new paradigm, we conducted a visualization analysis of the intermediate trajectories. Specifically, we performed trajectory tracking on data with nonlinear distortions, ground truth data, and registration results, ensuring consistency between the trajectories across different datas. The visualization results are shown in Figure 14. It is evident from the figure that the trajectories of the nonlinear distorted data exhibit irregular noise jitter, leading to abrupt changes in local curvature, which reflect the impact of distortion on the motion trajectory. In contrast, the trajectory of the ground truth data maintains the natural evolutionary pattern of the organism’s movement, exhibiting smooth and continuous changes. Our method successfully overcame abnormal physical deformations in the registration results, faithfully and precisely restoring the natural motion trajectory.

B.4 Failure Cases

We observe that our method is to some extent dependent on the accuracy of the trajectory tracking module. When the tracking is suboptimal, the subsequent image registration performance may be affected. This issue becomes particularly prominent when handling real-world anisotropic data, where

288 the axial resolution is often significantly lower than the lateral (XY) resolution. Substantial structural
289 and textural differences arise between adjacent slices (as shown in Fig. 15). Moreover, real datasets
290 often suffer from high noise levels, imaging artifacts, and missing slices, further complicating reliable
291 trajectory estimation. In Fig. 16, we illustrate several representative failure cases, in which the axial
292 resolution of the image stacks is approximately one-tenth of the lateral resolution, with noticeable
293 noise and missing slices. Although our method is still capable of performing registration under such
294 challenging conditions, the visual quality may degrade due to the aforementioned interfering factors.
295 We hope future research will explore more robust and efficient trajectory tracking modules, as well as
296 3D registration methods that are better suited for anisotropic and noisy real-world data.

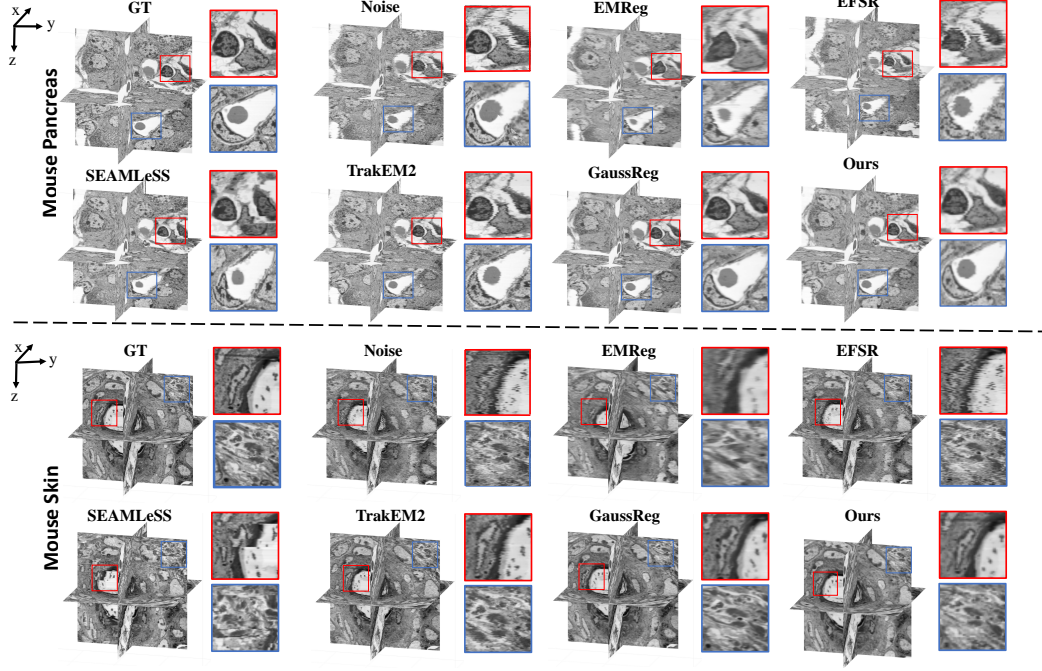


Figure 1: More 3D visualization of registration results on Mus Pancreas and Mus Skin datasets.

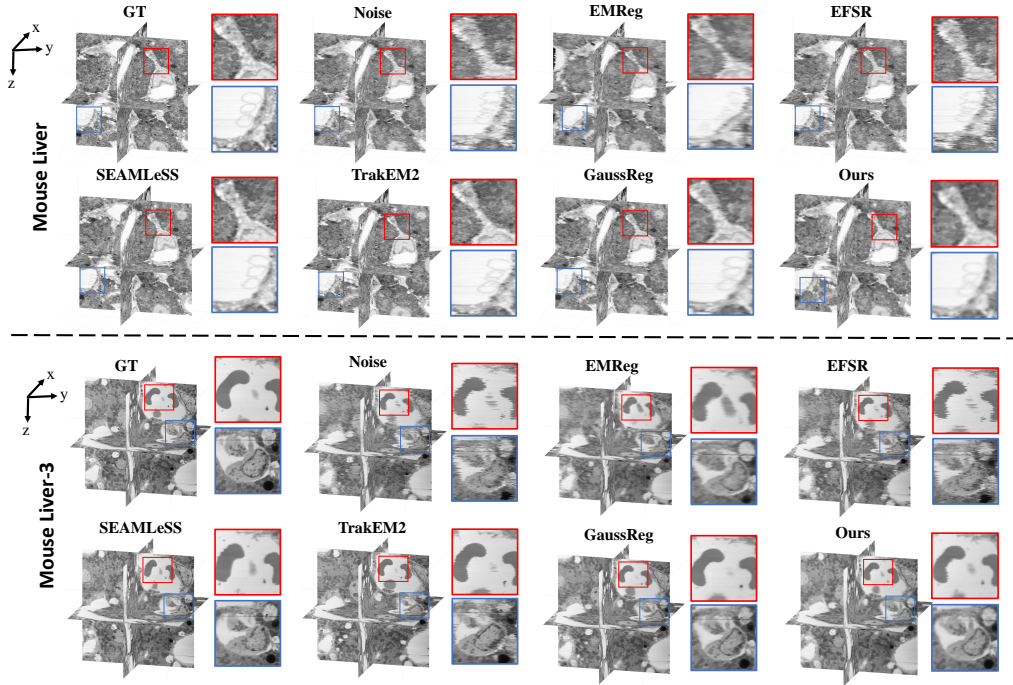


Figure 2: More 3D visualization of registration results on Mus Liver and Mus Liver-3 datasets.

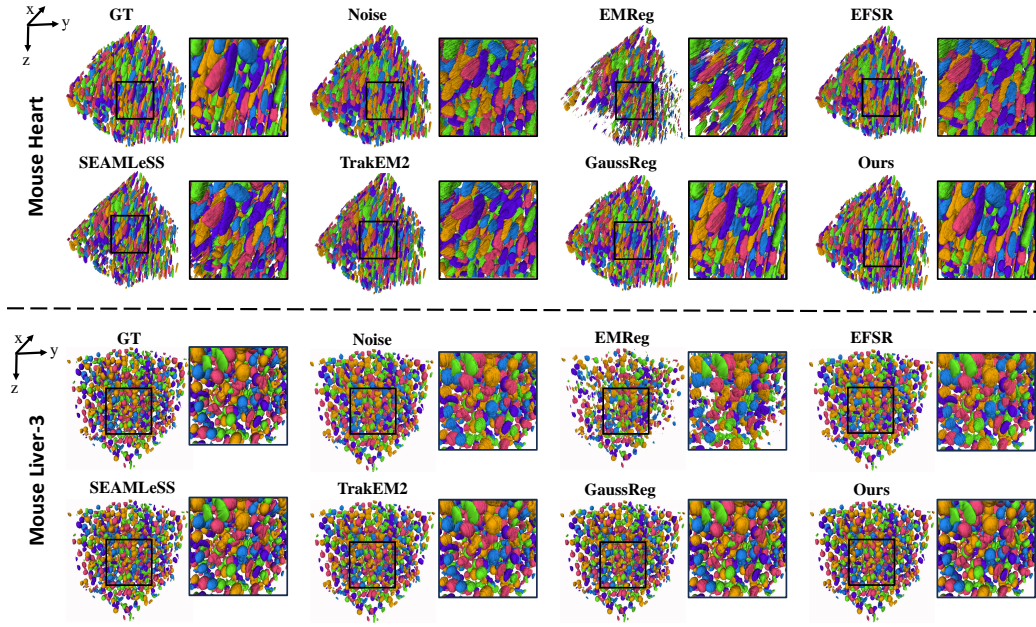


Figure 3: 3D segmentation visualizations of registration results using various methods on the Mus Heart and Mus Liver-3 datasets.

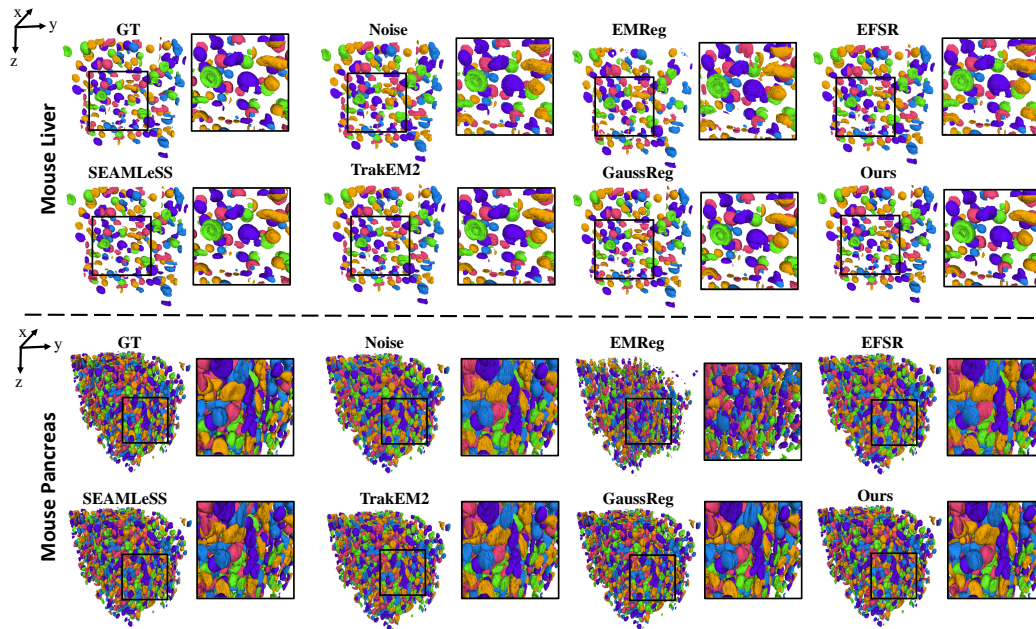


Figure 4: 3D segmentation visualizations of registration results using various methods on the Mus Liver and Mus Pancreas datasets.

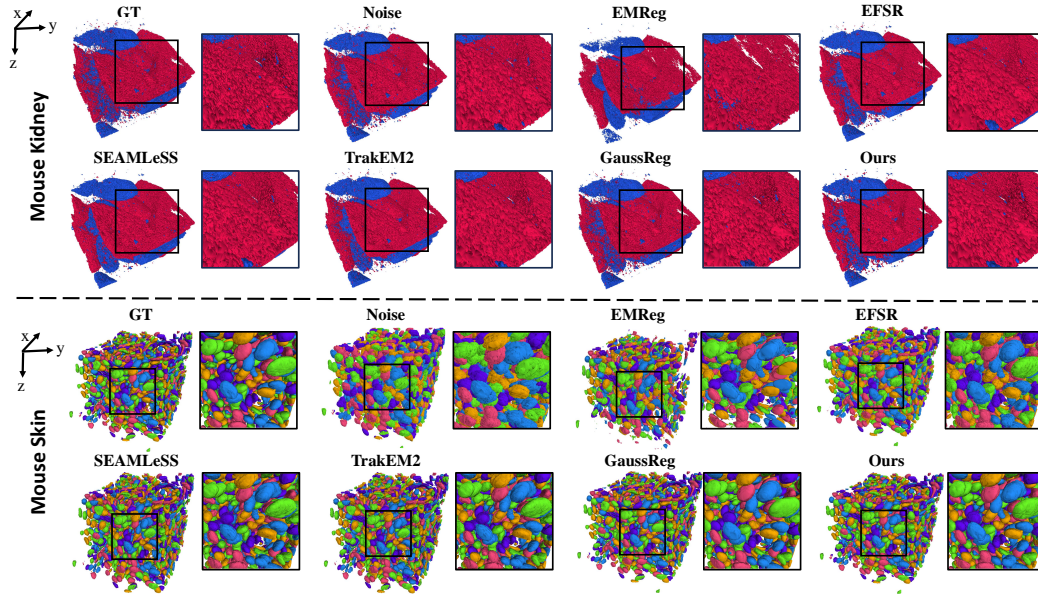


Figure 5: 3D segmentation visualizations of registration results using various methods on the Mus Kidney and Mus Skin datasets.

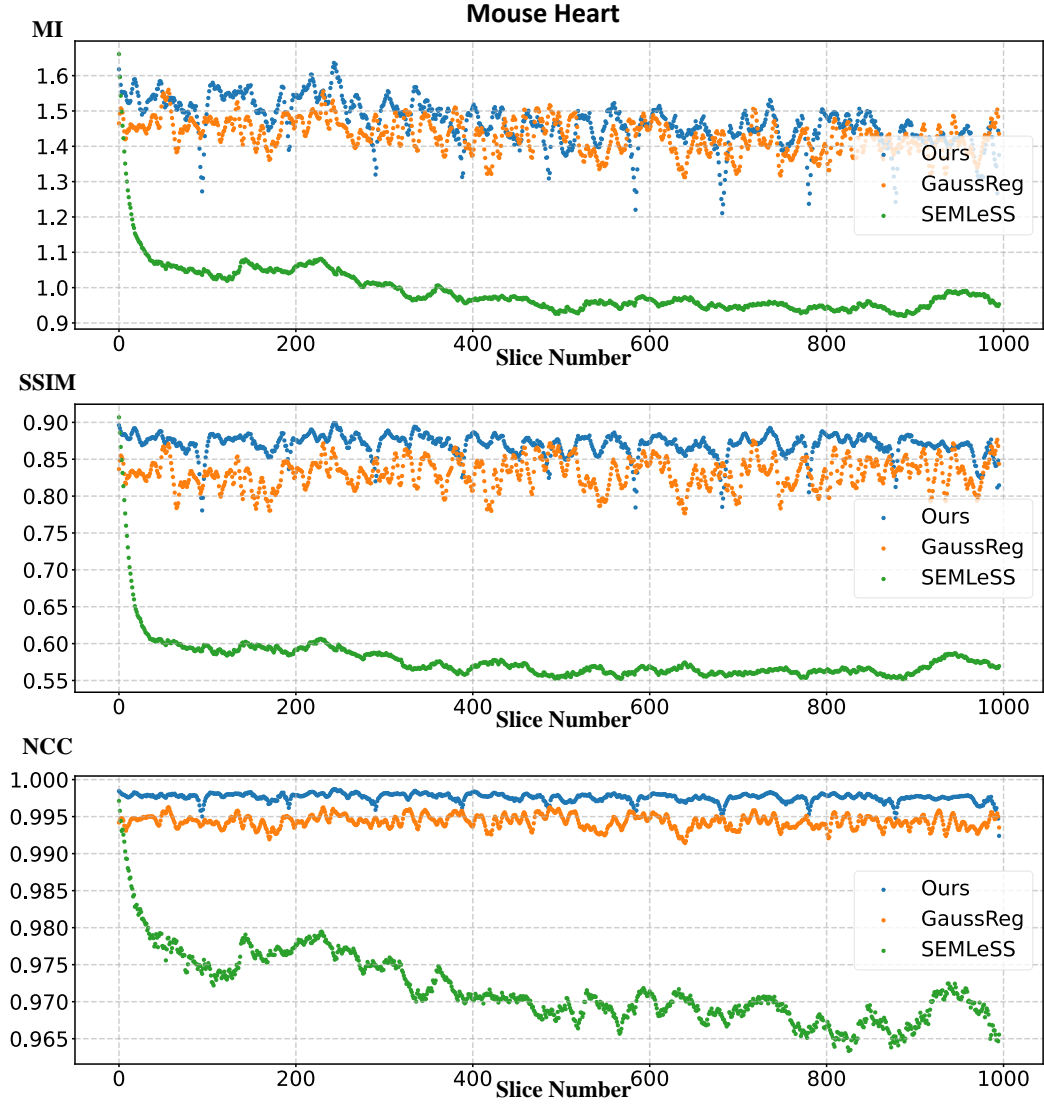


Figure 6: Error accumulation comparison between GaussReg [29] and SEMLeSS [18] and ours on Mus Heart dataset.

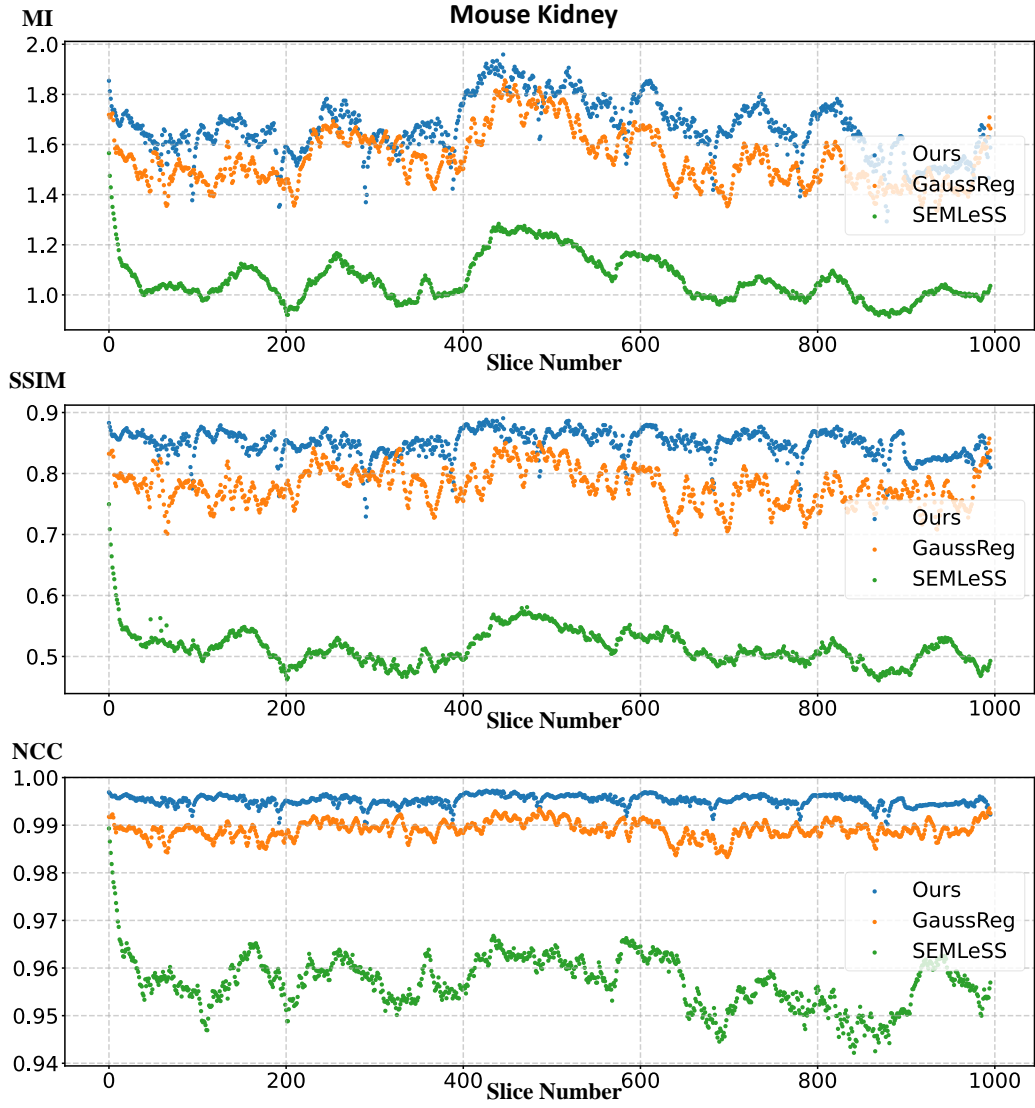


Figure 7: Error accumulation comparison between GaussReg [29] and SEMLeSS [18] and ours on Mus Kidney dataset.

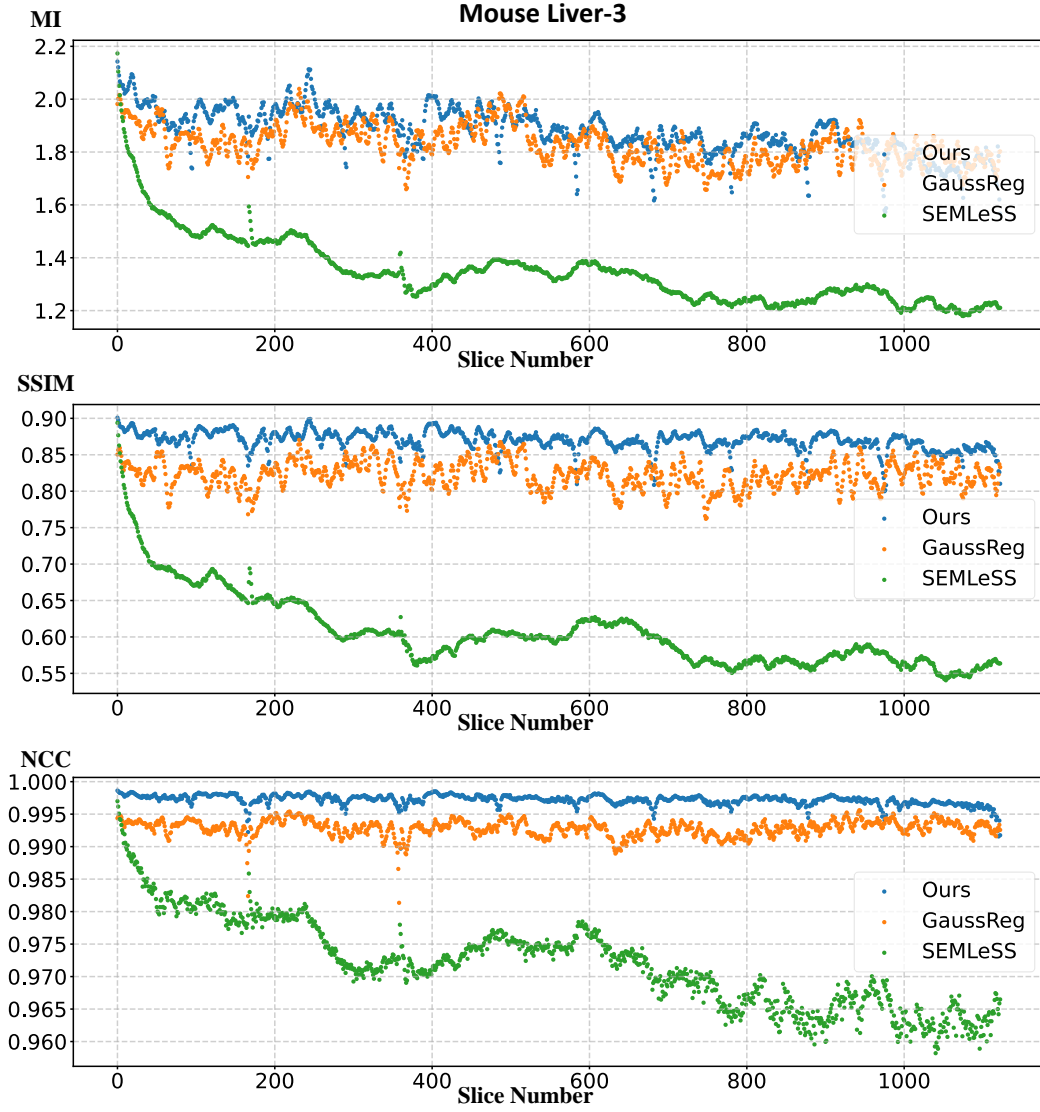


Figure 8: Error accumulation comparison between GaussReg [29] and SEMLeSS [18] and ours on Mus Liver-3 dataset.

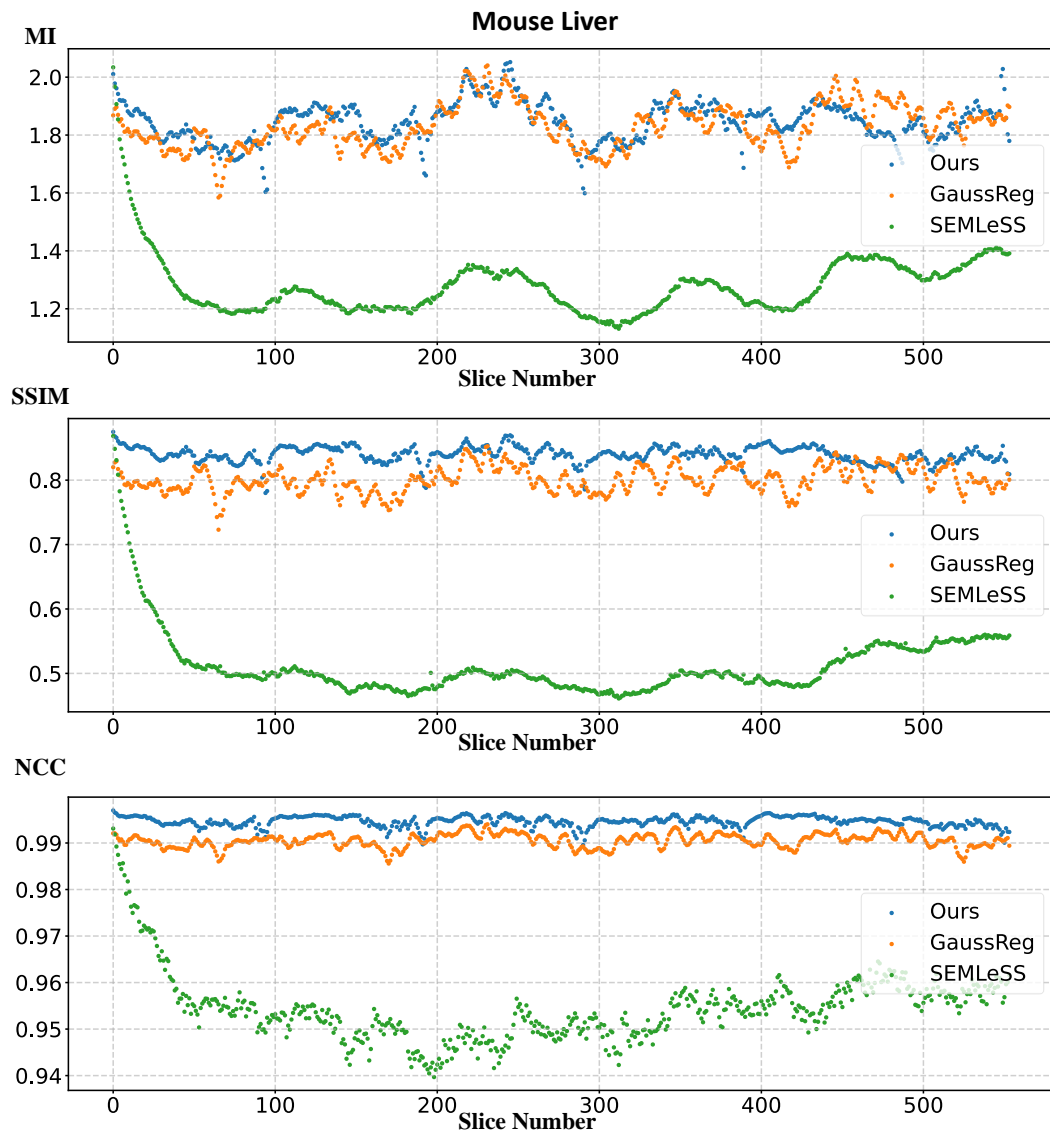


Figure 9: Error accumulation comparison between GaussReg [29] and SEMLeSS [18] and ours on Mus Liver dataset.

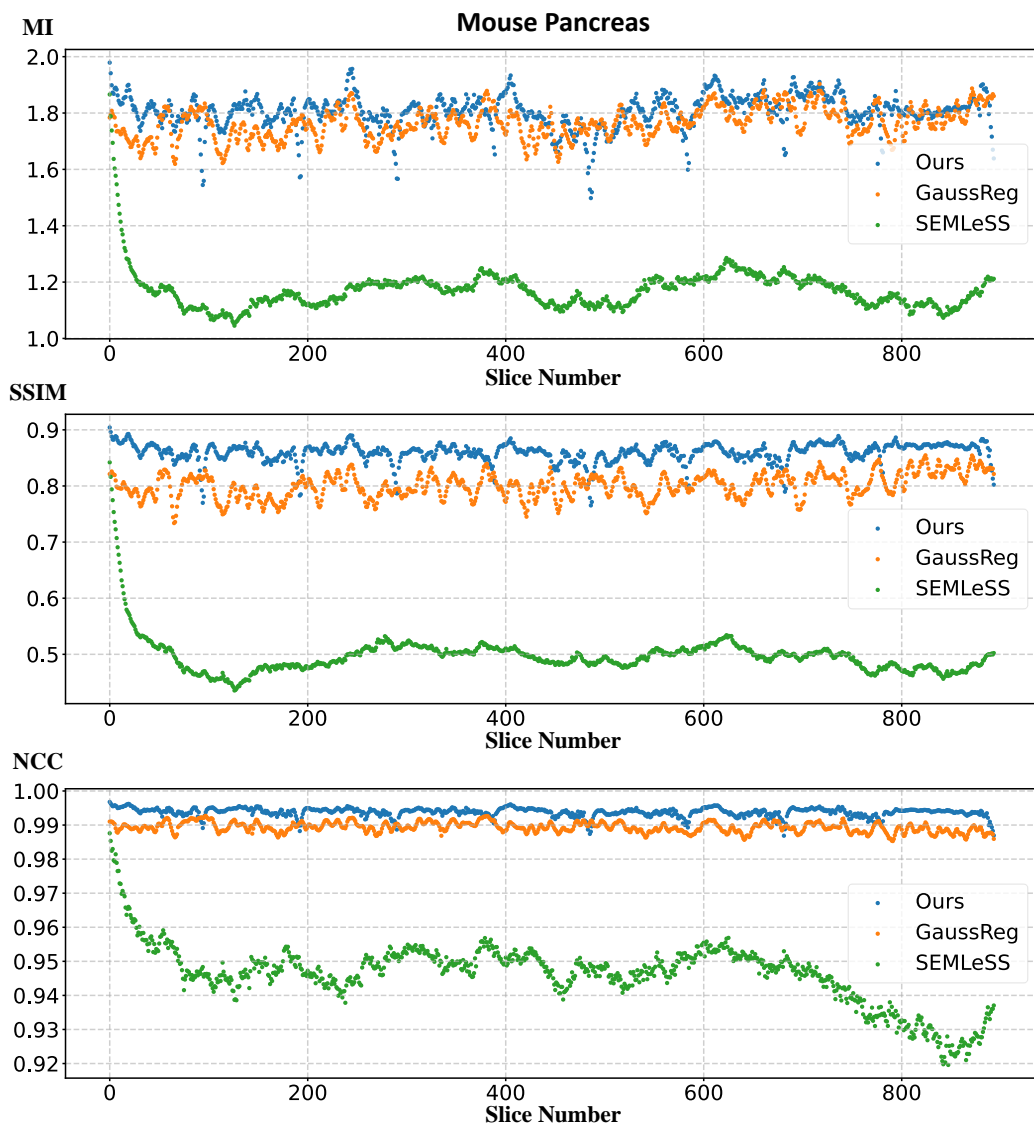


Figure 10: Error accumulation comparison between GaussReg [29] and SEMLeSS [18] and ours on Mus Pancreas dataset.

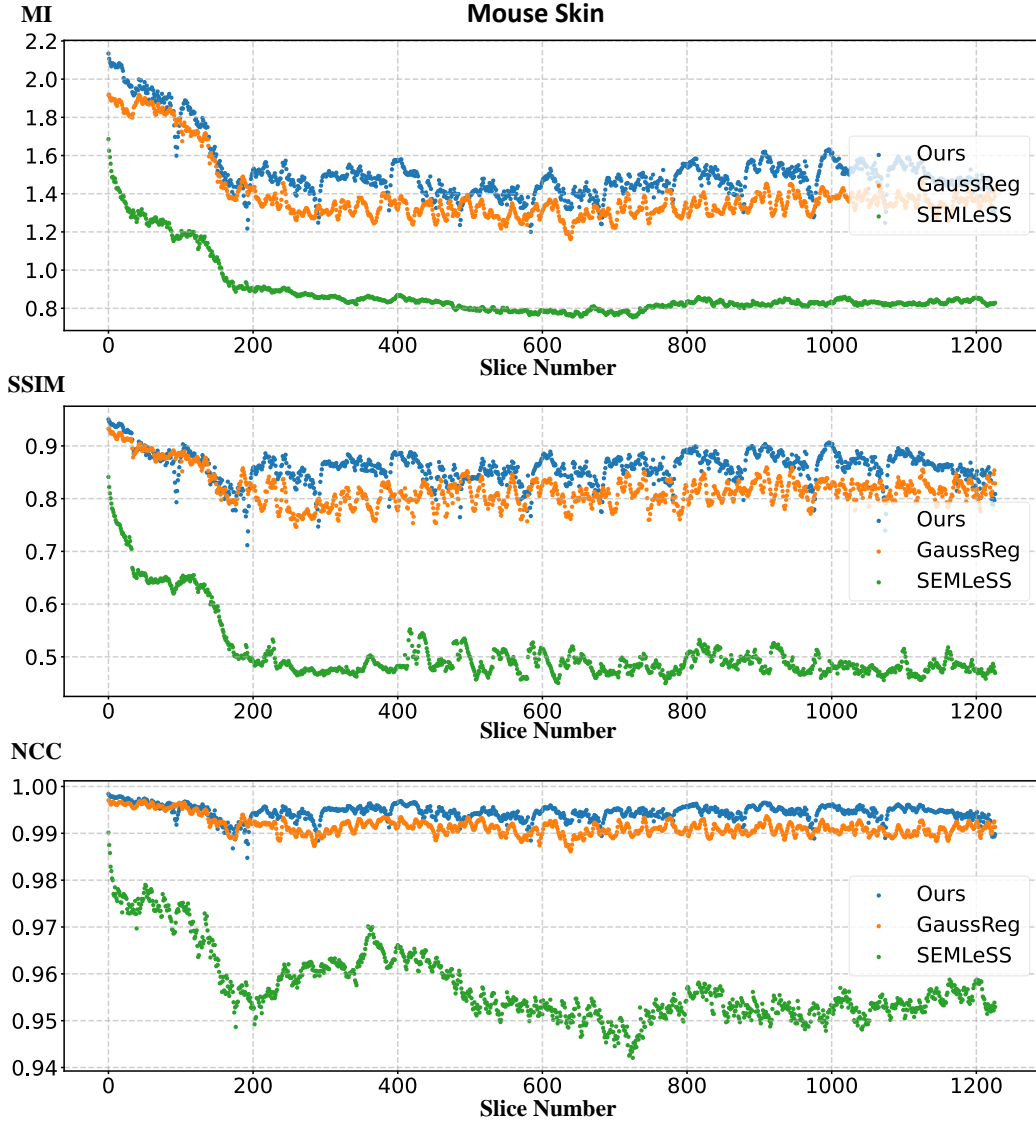


Figure 11: Error accumulation comparison between GaussReg [29] and SEMLeSS [18] and ours on Mus Skin dataset.

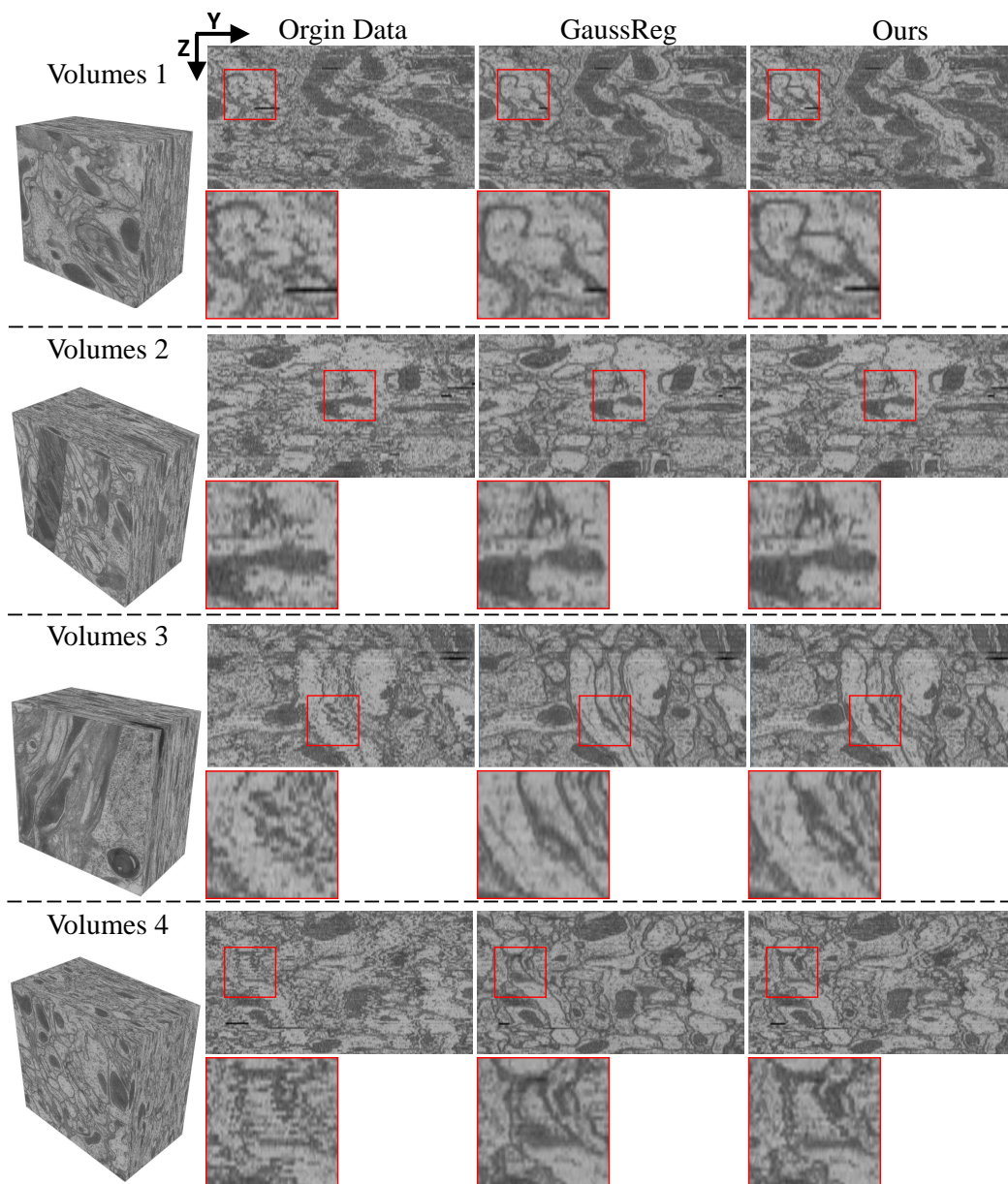


Figure 12: Side views of the original data, the registration results of GaussReg [29], and our method on four sampled volumes from the FemFlyBrain dataset [22].

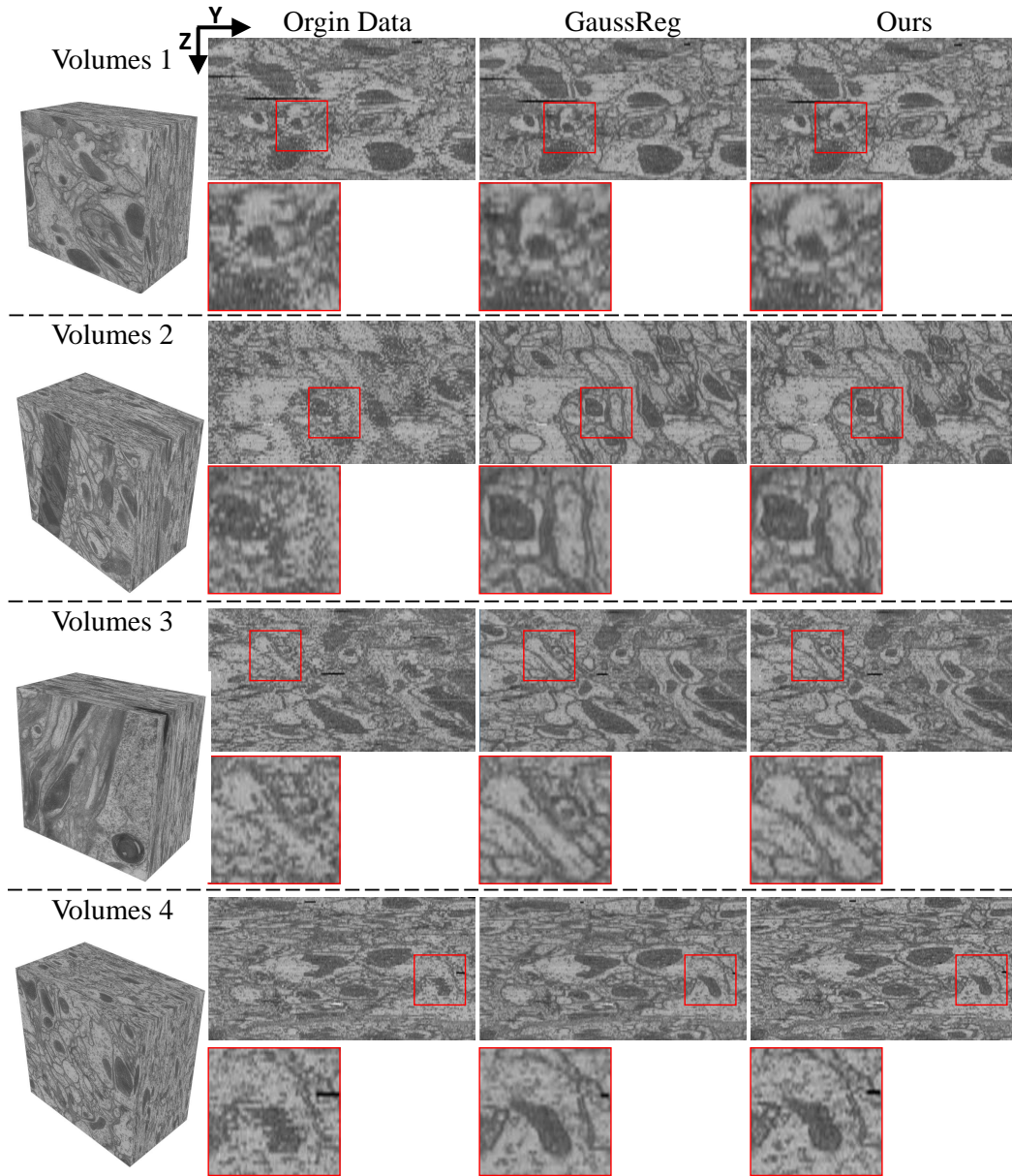


Figure 13: Side views of the original data, the registration results of GaussReg [29], and our method on four sampled volumes from the FemFlyBrain dataset [22].

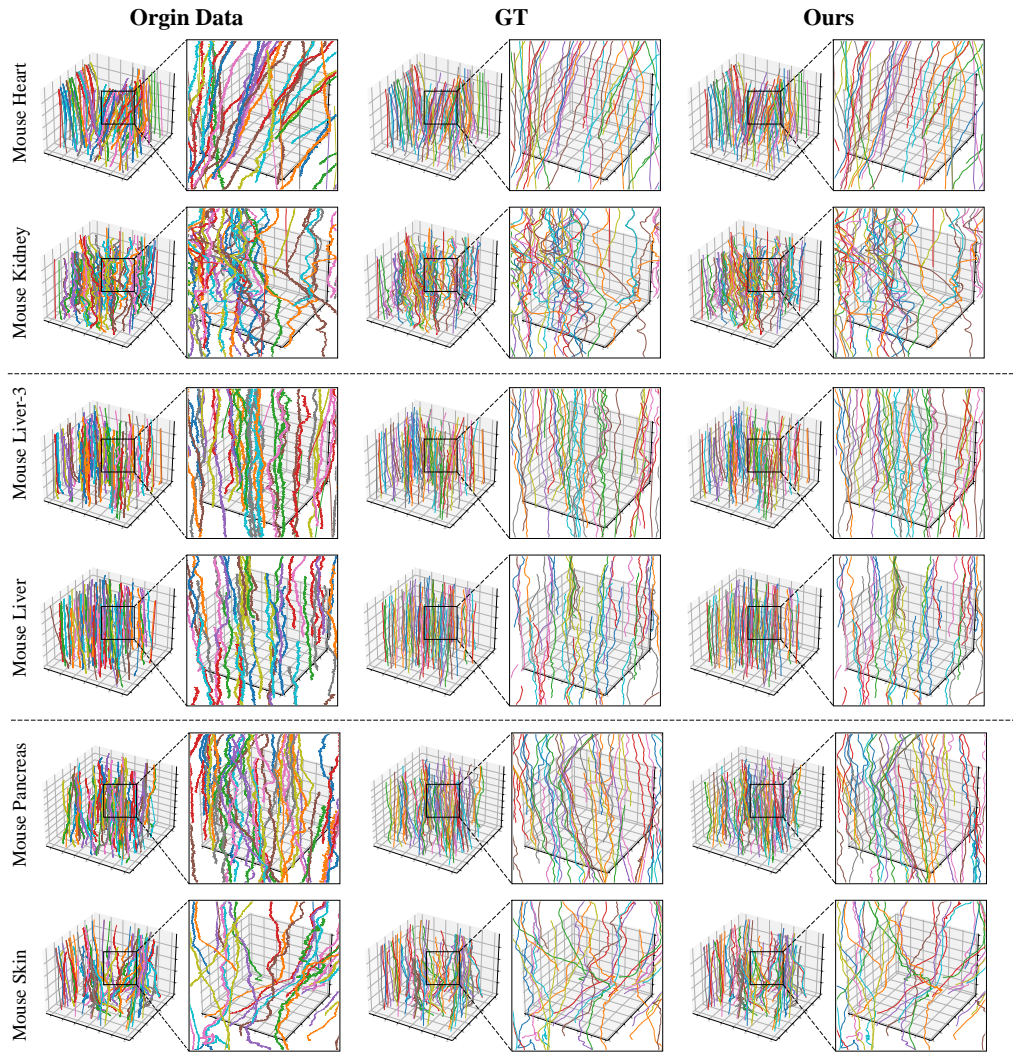


Figure 14: Trajectory visualization of the original data, ground truth, and registration results.

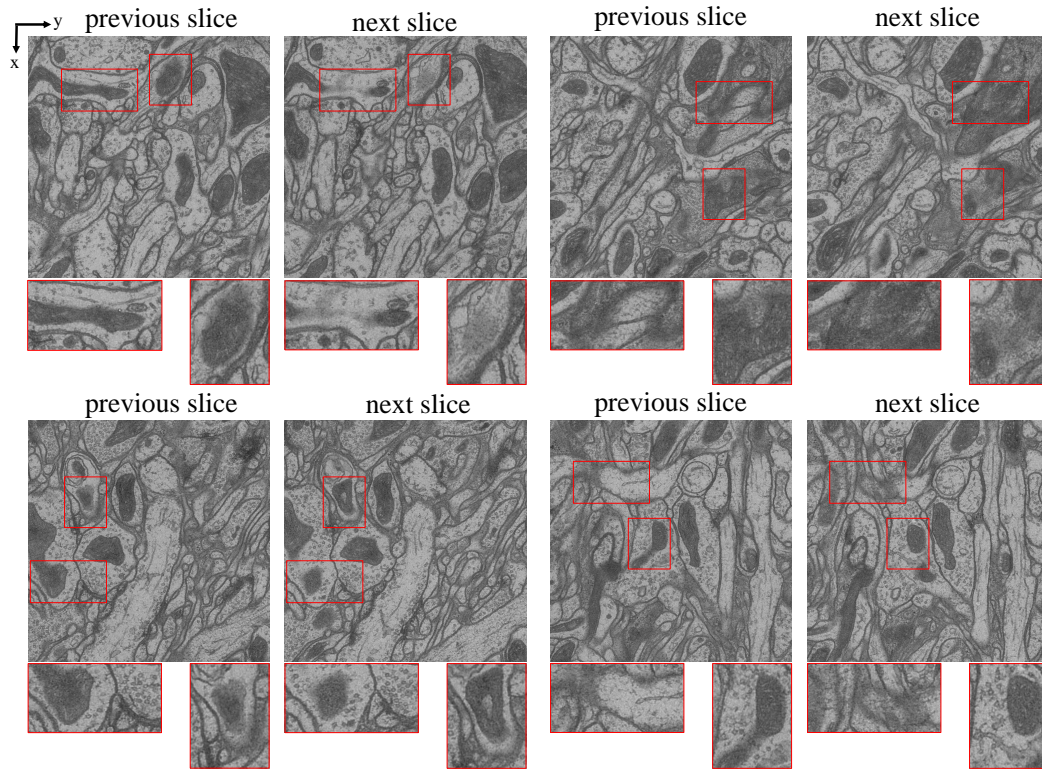


Figure 15: Four examples showing texture structure differences between adjacent slices of anisotropic data.

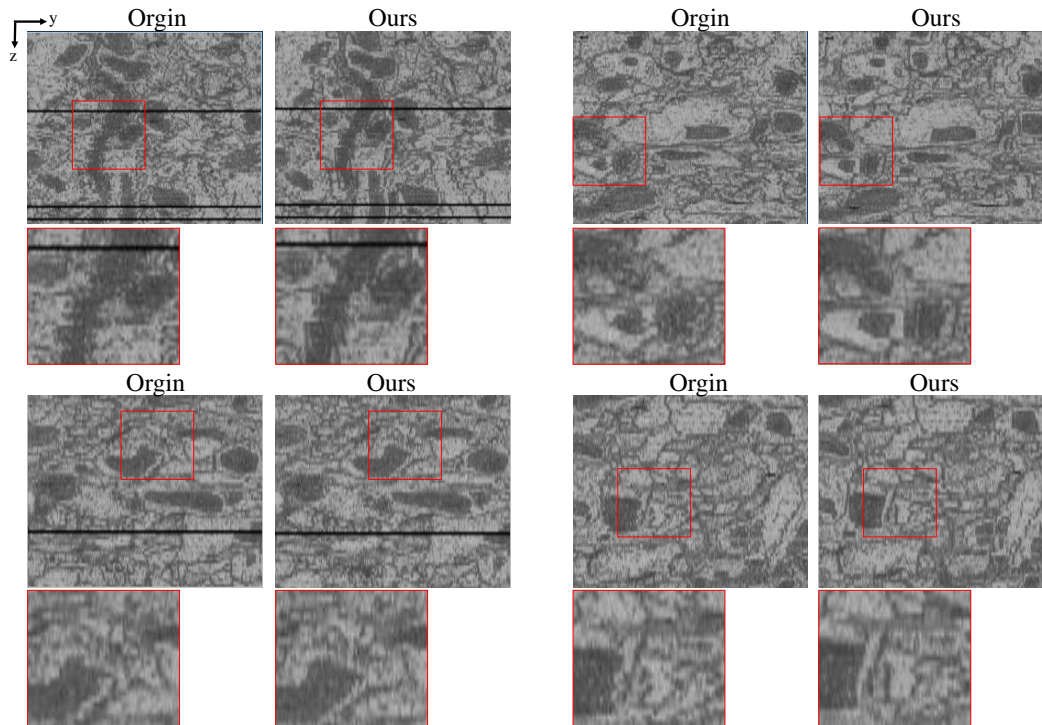


Figure 16: Four failure cases from the FemFlyBrain dataset. These examples are affected by low axial resolution, missing slices, and high noise.

References

- [1] Mario Botsch, Alexander Hornung, Matthias Zwicker, and Leif Kobbelt. High-quality surface splatting on today's gpus. In *Proceedings Eurographics/IEEE VGTC Symposium Point-Based Graphics, 2005.*, pages 17–141. IEEE, 2005.
- [2] Carl Doersch, Ankush Gupta, Larisa Markeeva, Adria Recasens, Lucas Smaira, Yusuf Aytar, Joao Carreira, Andrew Zisserman, and Yi Yang. Tap-vid: A benchmark for tracking any point in a video. *Advances in Neural Information Processing Systems*, 35:13610–13626, 2022.
- [3] Carl Doersch, Yi Yang, Mel Vecerik, Dilara Gokay, Ankush Gupta, Yusuf Aytar, Joao Carreira, and Andrew Zisserman. Tapir: Tracking any point with per-frame initialization and temporal refinement. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10061–10072, 2023.
- [4] Dmitri Dolgov, Sebastian Thrun, Michael Montemerlo, and James Diebel. Path planning for autonomous vehicles in unknown semi-structured environments. *The international journal of robotics research*, 29(5):485–501, 2010.
- [5] Sven Dorkenwald, Casey M Schneider-Mizell, Derrick Brittain, Akhilesh Halageri, Chris Jordan, Nico Kemnitz, Manuel A Castro, William Silversmith, Jeremy Maitin-Shephard, Jakob Troidl, et al. Cave: Connectome annotation versioning engine. *Nature Methods*, pages 1–9, 2025.
- [6] Clare R Gamlin, Casey M Schneider-Mizell, Matthew Mallory, Leila Elabbady, Nathan Gouwens, Grace Williams, Alice Mukora, Rachel Dalley, Agnes L Bodor, Derrick Brittain, et al. Connectomics of predicted sst transcriptomic types in mouse visual cortex. *Nature*, 640(8058):497–505, 2025.
- [7] Philip E Gill, Walter Murray, and Michael A Saunders. Snpot: An sqp algorithm for large-scale constrained optimization. *SIAM review*, 47(1):99–131, 2005.
- [8] Gene H Golub et al. Cf vanloan, matrix computations. *The Johns Hopkins*, 113(10):23–36, 1996.
- [9] Anjali Gour, Kevin M Boergens, Natalie Heike, Yunfeng Hua, Philip Laserstein, Kun Song, and Moritz Helmstaedter. Postnatal connectomic development of inhibition in mouse barrel cortex. *Science*, 371(6528):eabb4534, 2021.
- [10] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. *Advances in neural information processing systems*, 28, 2015.
- [11] Bernd Jähne. *Digital image processing*. Springer Science & Business Media, 2005.
- [12] Xi Jia, Joseph Bartlett, Tianyang Zhang, Wenqi Lu, Zhaowen Qiu, and Jinming Duan. U-net vs transformer: Is u-net outdated in medical image registration? In *International Workshop on Machine Learning in Medical Imaging*, pages 151–160. Springer, 2022.
- [13] Xiaoyu Liu, Bo Hu, Mingxing Li, Wei Huang, Yueyi Zhang, and Zhiwei Xiong. A soma segmentation benchmark in full adult fly brain. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7402–7411, 2023.
- [14] Xiaoyu Liu, Wei Huang, Zhiwei Xiong, Shenglong Zhou, Yueyi Zhang, Xuejin Chen, Zheng-Jun Zha, and Feng Wu. Learning cross-representation affinity consistency for sparsely supervised biomedical instance segmentation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 21107–21117, 2023.
- [15] Xinzhaio Liu, Yueyi Zhang, Shenglong Zhou, Zhiwei Xiong, and Xiaoyan Sun. Electron microscopy image registration using correlation volume. In *2023 IEEE 20th International Symposium on Biomedical Imaging (ISBI)*, pages 1–5. IEEE, 2023.
- [16] Sahil Loomba, Jakob Straehle, Vijayan Gangadharan, Natalie Heike, Abdelrahman Khalifa, Alessandro Motta, Niansheng Ju, Meike Sievers, Jens Gempt, Hanno S Meyer, et al. Connectomic comparison of mouse and human cortex. *Science*, 377(6602):eabo0924, 2022.
- [17] Naisong Luo, Rui Sun, Yuwen Pan, Tianzhu Zhang, and Feng Wu. Electron microscopy images as set of fragments for mitochondrial segmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 3981–3989, 2024.
- [18] Sergiy Popovych, Thomas Macrina, Nico Kemnitz, Manuel Castro, Barak Nehoran, Zhen Jia, J Alexander Bae, Eric Mitchell, Shang Mu, Eric T Trautman, et al. Petascale pipeline for precise alignment of images from serial section electron microscopy. *Nature Communications*, 15(1):289, 2024.

- 347 [19] Stephan Saalfeld, Richard Fetter, Albert Cardona, and Pavel Tomancak. Elastic volume reconstruction
348 from series of ultra-thin microscopy sections. *Nature methods*, 9(7):717–720, 2012.
- 349 [20] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias
350 Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, et al. Fiji: an open-source
351 platform for biological-image analysis. *Nature methods*, 9(7):676–682, 2012.
- 352 [21] Xiaoyu Shi, Zhaoyang Huang, Dasong Li, Manyuan Zhang, Ka Chun Cheung, Simon See, Hongwei Qin,
353 Jifeng Dai, and Hongsheng Li. Flowformer++: Masked cost volume autoencoding for pretraining optical
354 flow estimation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*,
355 pages 1599–1610, 2023.
- 356 [22] Shin-ya Takemura, Arjun Bharioke, Zhiyuan Lu, Aljoscha Nern, Shiv Vitaladevuni, Patricia K Rivlin,
357 William T Katz, Donald J Olbris, Stephen M Plaza, Philip Winston, et al. A visual motion detection circuit
358 suggested by drosophila connectomics. *Nature*, 500(7461):175–181, 2013.
- 359 [23] Jeya Maria Jose Valanarasu and Vishal M Patel. Unext: Mlp-based rapid medical image segmentation
360 network. In *International conference on medical image computing and computer-assisted intervention*,
361 pages 23–33. Springer, 2022.
- 362 [24] Grady Barrett Wright. *Radial basis function interpolation: numerical and analytical developments*.
363 University of Colorado at Boulder, 2003.
- 364 [25] Tong Xin, Yanan Lv, Haoran Chen, Linlin Li, Lijun Shen, Guangcun Shan, Xi Chen, and Hua Han. A novel
365 registration method for long-serial section images of em with a serial split technique based on unsupervised
366 optical flow network. *Bioinformatics*, 39(8):btad436, 2023.
- 367 [26] C Shan Xu, Song Pang, Gleb Shtengel, Andreas Müller, Alex T Ritter, Huxley K Hoffman, Shin-ya
368 Takemura, Zhiyuan Lu, H Amalia Pasolli, Nirmala Iyer, et al. An open-access volume electron microscopy
369 atlas of whole cells and tissues. *Nature*, 599(7883):147–151, 2021.
- 370 [27] Haoifei Xu, Jing Zhang, Jianfei Cai, Hamid RezaTofighi, and Dacheng Tao. Gmflow: Learning optical
371 flow via global matching. In *Proceedings of the IEEE/CVF conference on computer vision and pattern
372 recognition*, pages 8121–8130, 2022.
- 373 [28] Wenjing Yin, Derrick Brittain, Jay Borseth, Marie E Scott, Derric Williams, Jedediah Perkins, Christopher S
374 Own, Matthew Murfitt, Russel M Torres, Daniel Kapner, et al. A petascale automated imaging pipeline for
375 mapping neuronal circuits with high-throughput transmission electron microscopy. *Nature communications*,
376 11(1):4949, 2020.
- 377 [29] Zhenbang Zhang, Hongjia Li, Zhiqiang Xu, Wenjia Meng, and Renmin Han. A gaussian filter-based 3d
378 registration method for series section electron microscopy. In *Proceedings of the AAAI Conference on
379 Artificial Intelligence*, volume 39, pages 1156–1164, 2025.