Topology-Preserving Image Segmentation with Spatial-Aware Persistent Feature Matching

Supplementary Material

6. Details on Datasets

6.1. CREMI

The CREMI dataset [9] (Link) has three volumes (A, B & C), volume A, B has 125 scans, volume C has 123 scans. Each scan is an 1250×1250 2D image. We use volume A,B and scan 1-14, 16-74, 76-81 in volume C for training, scan 82-92 in volume C for validation and scan 93-125 for testing. Each scan is cropped into 25 250×250 patches. In total, there are 8225 for training, 275 for validation and 825 for testing. The original CREMI dataset only provide the ground truth mask for each cell. We take the spaces between the cells to form the ground truth for the cell boundaries (extracellular matrix). The evaluation metrics are computed on the same patch size (the same for the following datasets).

6.2. DRIVE

The DRIVE dataset [30] (Link) has 20 565×584 images. We slightly remove the redundant blank margins around the circular effective imaging area and thus crop each image to 550×550. For fold 1, we use 21-32 for training, 33-34 for validation, 35-40 for testing; For fold 2, we use 29-40 for training, 27-28 for validation and 21-26 for testing; For fold 3, we use 23-28, 35-40 for training, 21-22 for validation and 29-34 for testing. Each image is then cropped into 4 275×275 image patches. In total, there are 48 for training, 8 for validation and 24 for testing.

6.3. Roads

The (Massachusetts) Roads dataset [24] (Link) has in total 1171 1500×1500 images, and is split into 1108 training, 14 validation and 49 testing by the dataset author. We follow this dataset split in our experiments. Each image is cropped into 25 300×300 image patches. In total, there are 27700 for training, 350 for validation and 1225 for testing.

6.4. CrackTree

The CrackTree dataset [37] (Link) has 260 images, most of them have a resolution of 800×600, while a few others have a resolution of 960×720. For consistency, we pad all the 800×600 images to the right to 900×600 and crop the 960×720 images from top and bottom (in equal) and right to 900×600. After sorting all the images by name: For fold 1, we used image 1-200 for training, 201-220 for validation, 221-260 for testing; For fold 2, we used image 1-116, 177-260 for training, 117-136 for validation, 137-176 for testing; For fold 3, we used image 61-260 for training, 1-20 for validation, 21-60 for testing. Each image is then cropped to 6 300×300 image patches. In total, there are 1200 for training, 120 for validation and 240 for testing.

6.5. C.Elegan-small

We follow as many details as we can in [31] to prepare the C.Elegan-small and Roads-small dataset. The C.Elegan dataset [32] (Link) has 200 696×520 images. We crop the images and only keep the center 340×340 effective imaging area and then downsample them to 96×96. For fold 1, we use A01-C22 for training, C23-D08 for validation and D09-E04 for testing; For fold 2, we use B07-E04 for training, A01-A10 for validation and A11-B06 for testing; For fold 3, we use A01-B16, C23-E04 for training, C13-C22 for validation and B17-C12 for testing. Each image is cropped into 4 48×48 patches. In total, there are 280 for training, 40 for validation and 80 for testing.

6.6. Roads-small

In total 134 images are selected from the original Roads dataset (mixing training, validation and testing, since we use 3-fold cross validation for this dataset). In [31], which images are selected is not provided so we pick image by ourselves. It is difficult to describe the selected images unless providing a long list here but the general rule is to select images with more complexed road networks and less blank regions. The selected images are then divided randomly into training, validation and testing for each fold. The images are then downsampled from 1500×1500 to 375×375. Again, details of downsampling are not provided in [31], so we use (the same for C.Elegan-small) bicubic interpolation for all images and labels and then binarized the downsampled labels by thresholding them with a value of 0.5. No further manual correction is made to the downsampled labels. Before cropping, the images are padded to the right and bottom to 384×384. Then, each image is cropped to 64 48×48 patches. In total, there are 6400 for training, 640 for validation and 1536 for testing.

7. Additional Implementation Details

We train on Roads, CREMI and CrackTree datasets using a batch size of 16 and on DRIVE dataset using batch size of 2 (since bad convergence on larger batch size). During training, we follow [13] and pad the edges of the image patches with 1 (1 pixel width) to also take the 1-features enclosed by the image boundary into account. We run the main experiments on two computers, one with a i7-12700K CPU + RTX

3090 GPU, another with a i9-13900K CPU + 2 RTX 4090 GPUs. The experiments are randomly distributed into different computers. For comparison with BMLoss on Roads-small and C.Elegan-small datasets, we run on an older computer with a Xeno E5-2650 CPU + 3 GTX 1080Ti GPUs. This is because the implementation of BMLoss requires an old PyTorch and CUDA version where we find using newer GPU models encounters a random interruption issue during the training. Each experiment only takes one GPU. For implementation of the baseline, we use the official implementation for the respective methods if available (clDice [29]: Link, BMLoss [31]: Link). For methods where official implementations are unavailable (TCLoss [27], WTLoss [13], He et al. [12]), we use our own implementation similar with the SATLoss. The computations of persistent homology for these methods are exactly the same, and the difference only exists in the implementation of the loss functions, where we follow the mathematical expressions in the respective papers.

8. More Discussion on Experimental Results

One observation in the results that worth discussing is that the TCLoss [27], although a most recent method, appears at a very poor position in the baseline. On the one hand, unfortunately, in [27] the authors did not provide enough implementation details for us to reproduce their results. And they did not provide ablation studies on the topological loss weight, nor did they report the weight used for their method and the baseline methods. On the other hand, the results shown in [27] seems to have limited improvement over its baselines, including the BCELoss. And in our experiments we observe that the TCLoss has slight improvements over the BCELoss, which are similar with the results in [27]. In our opinion, the main limitation of the TCLoss is that the use of max and min functions in the Hausdorff distance makes only one pair of persistent features to have gradient from the loss function, which can make the loss less effective. In comparison, the other persistent homology-based methods optimize over all persistent features. We tried to increase the weight for TCLoss to *1e-0* until it diverges to compensate the small loss and gradient value (since they are computed from only one pair of persistent features, although it is an outlier which should usually have larger value than normal pairs). But the effort only seems to provide marginal improvement. Perhaps more investigations are needed to determine the performance and usefulness of this method.

Furthermore, we observe that the clDice loss [29] converges badly under some folds and random seeds when applied on the CrackTree dataset. It is potentially because of that the CrackTree dataset has very thin tubular labels (usually only one pixel width). And the labels, according to our examination, do contain inaccuracy and errors. This can make the overlap-based method (both clDice and the soft-Dice it used with) to act poorly. The CrackTree dataset was not used in [29]. Our experiments therefore might provide some insights into a limitation of the clDice method.

9. Additional Results

Table 5 shows the full training time comparison. The ablation results on the other datasets are provided in Fig. 7, Fig. 8, Fig. 9, Fig. 10, Fig. 11, Fig. 12. Due to limited resources and for consistency between datasets, the ablation experiments were only run on the first fold. In addition, we provide more qualitative results in Fig. 13 for our main experiments. In Fig. 14, we show more persistent feature matching results. In Fig. 15, we provide an illustrative example of the motivation of the spatial-aware matching, how it works during the matching process and how it helps with the topological accuracy of the segmentation.

Table 5. Training time comparison (i9-13900K+RTX 4090)

Hours	#iter	clDice	[26]	He.	Ours	BMLoss(Est.)
Roads	52k	2.9	44	43	44	>20000
CREMI	26k	1.0	16	16	16	>8000
CrackTree	3.7k	0.23	3.5	3.5	3.5	>1400
DRIVE	2.4k	0.03	0.22	0.23	0.23	>100

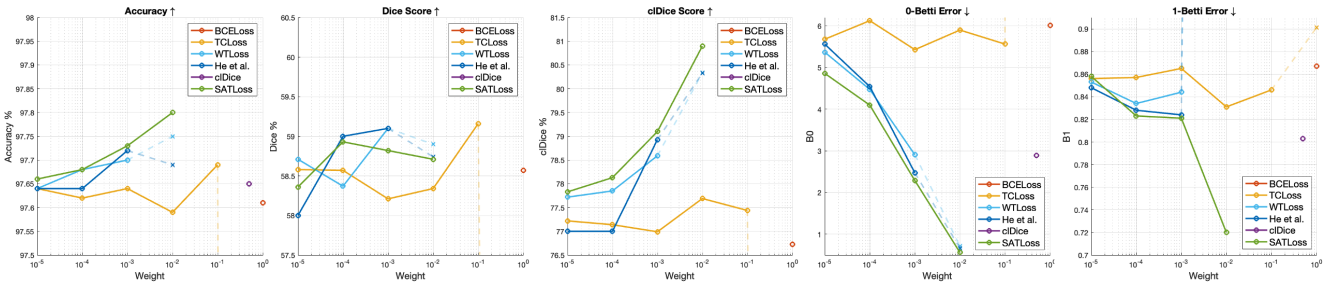

Figure 7. Ablation study results on weight λ on Roads dataset (including accuracy and Dice score).

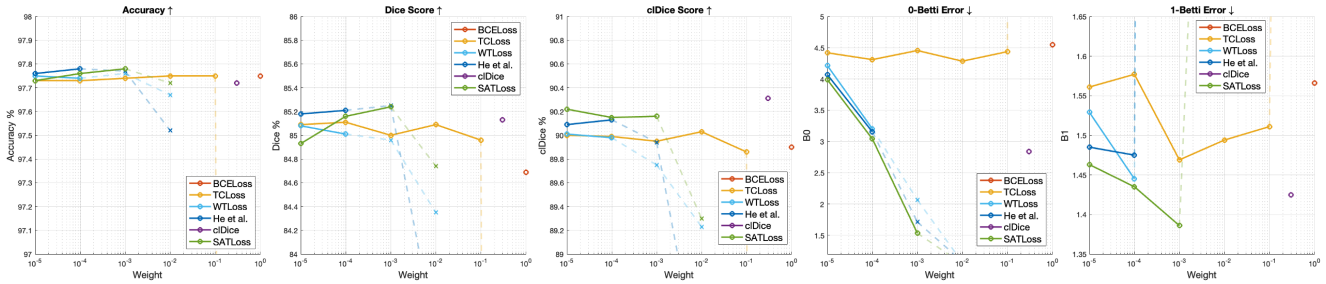

Figure 8. Ablation study results on weight λ on CREMI dataset.

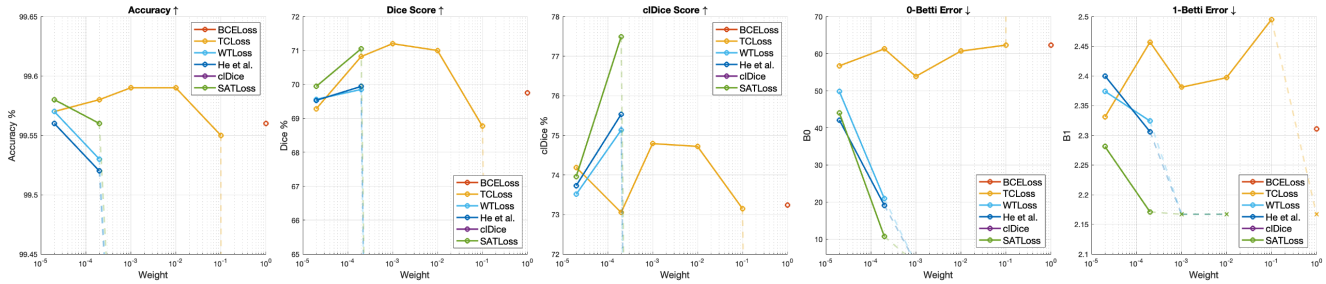

Figure 9. Ablation study results on weight λ on CrackTree dataset.

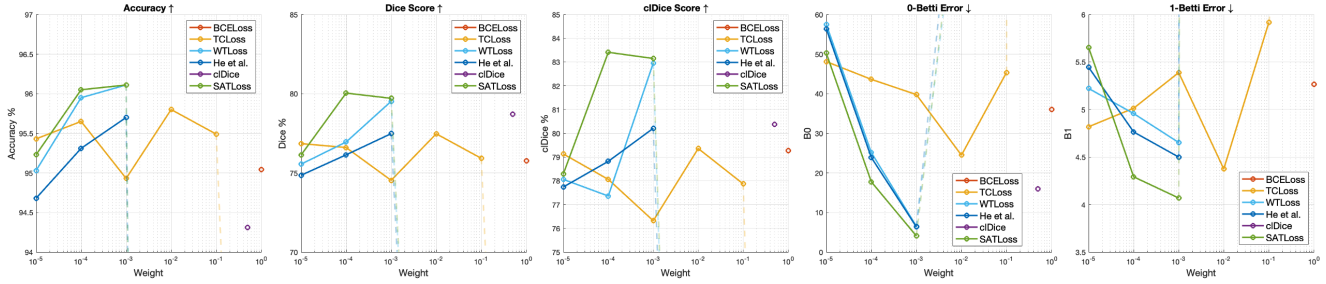

Figure 10. Ablation study results on weight λ on DRIVE dataset.

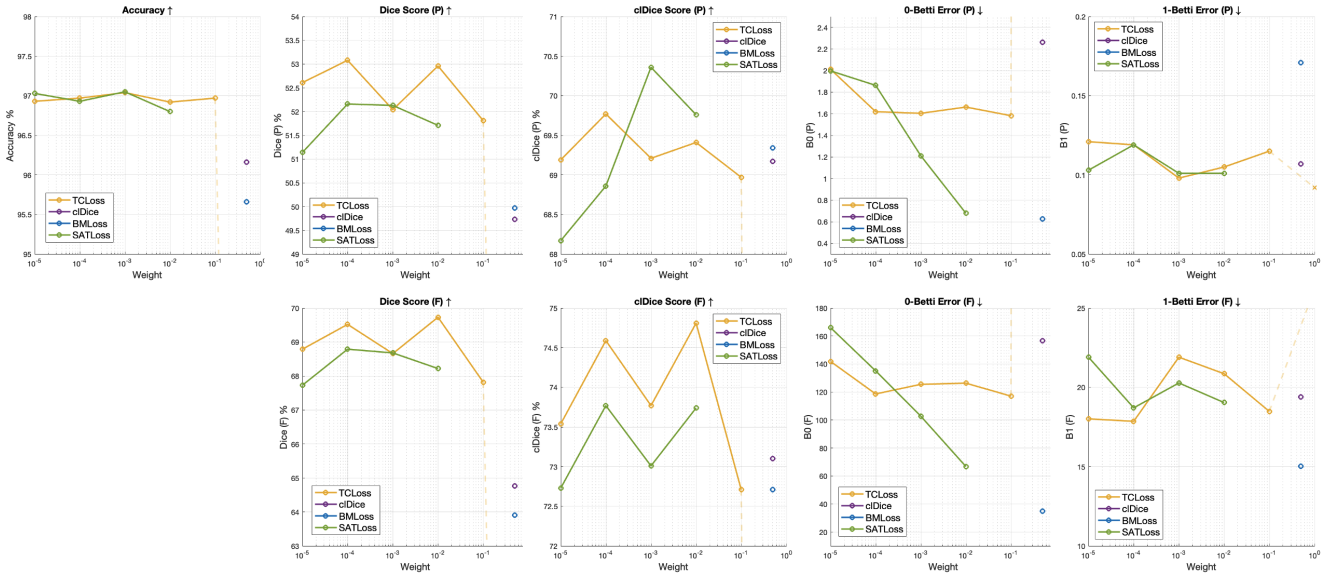

Figure 11. Ablation study results on weight λ on Roads-small dataset.

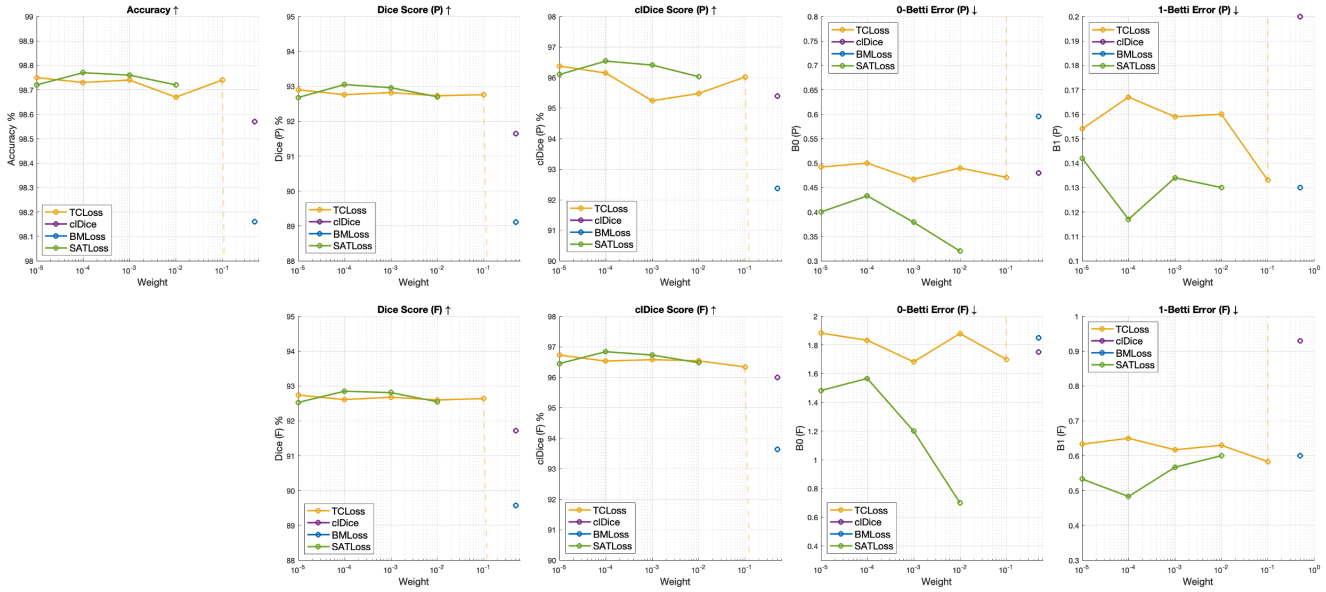

Figure 12. Ablation study results on weight λ on C.Elegan-small dataset.

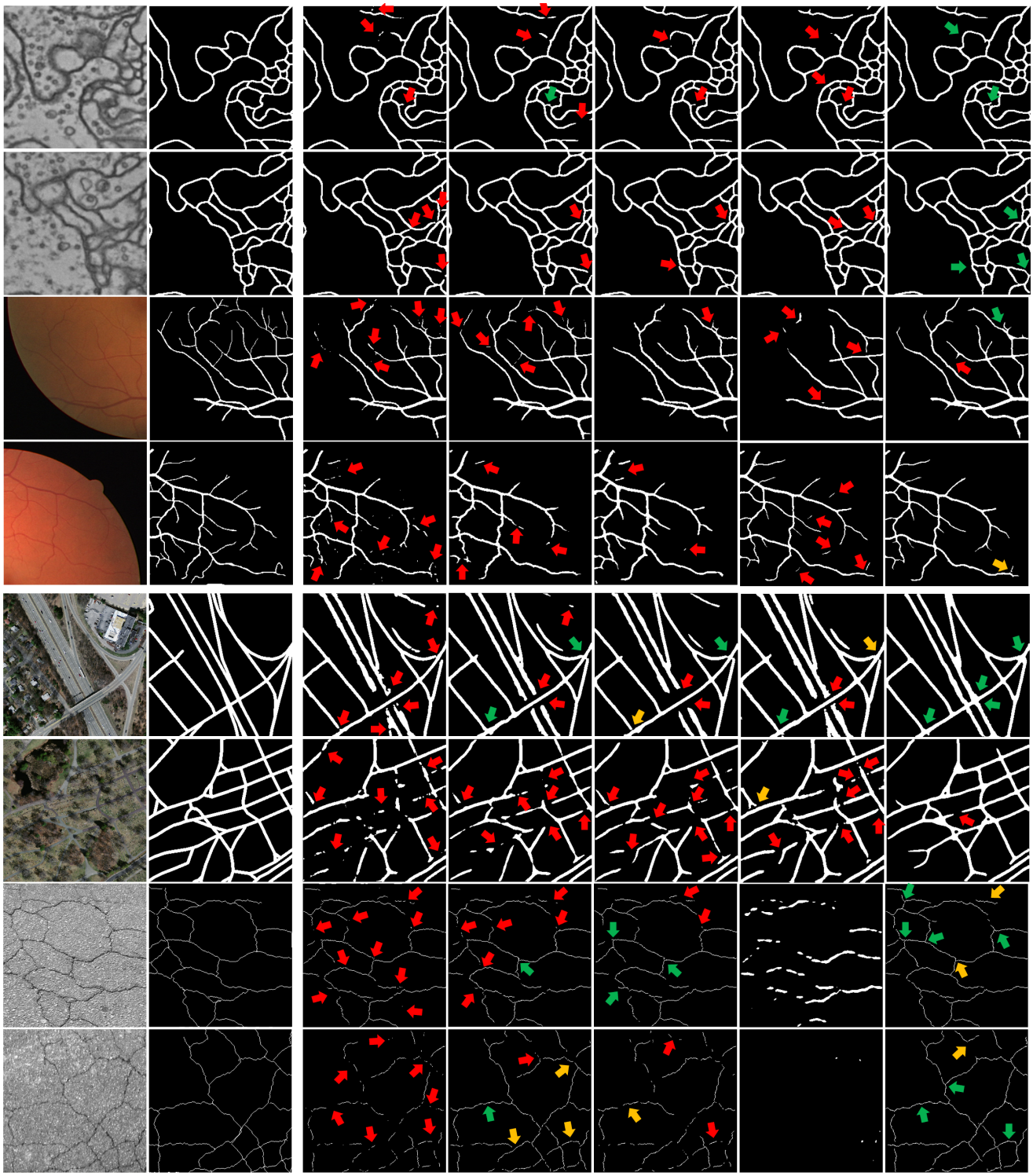

Figure 13. Extra qualitative results on main comparison with SOTA methods. From left to right: image, ground truth, BCELoss, WTLoss, He et al., clDice, SATLoss.

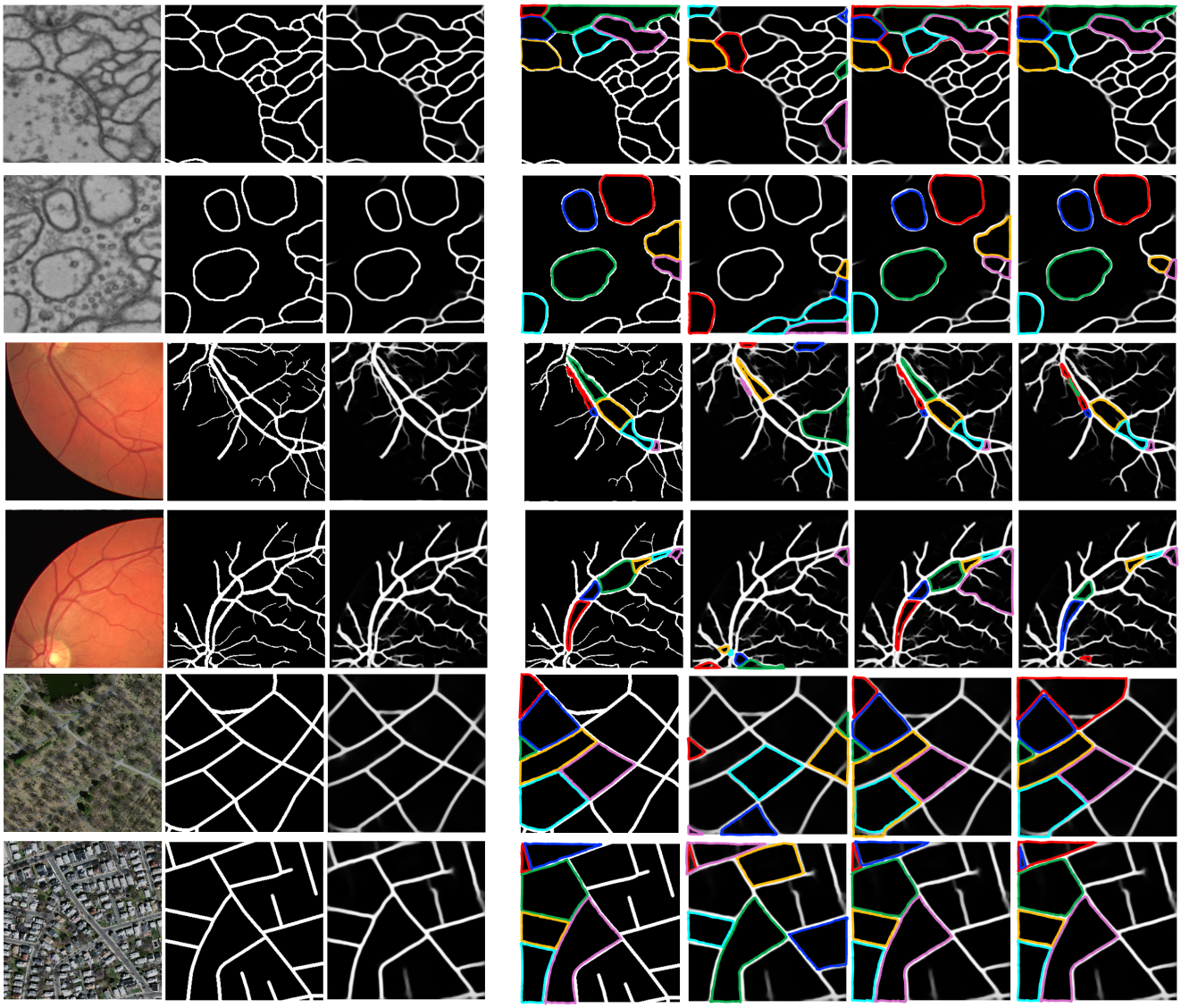

Figure 14. Matching of persistent features from likelihoods predicted by a converged model. Our method usually makes better matching on smaller features, and can make more mistakes on larger and longer features. In general, our method considerably improves the Wasserstein matching (which is usually almost totally messy) at a much lower computational cost than Betti-matching.

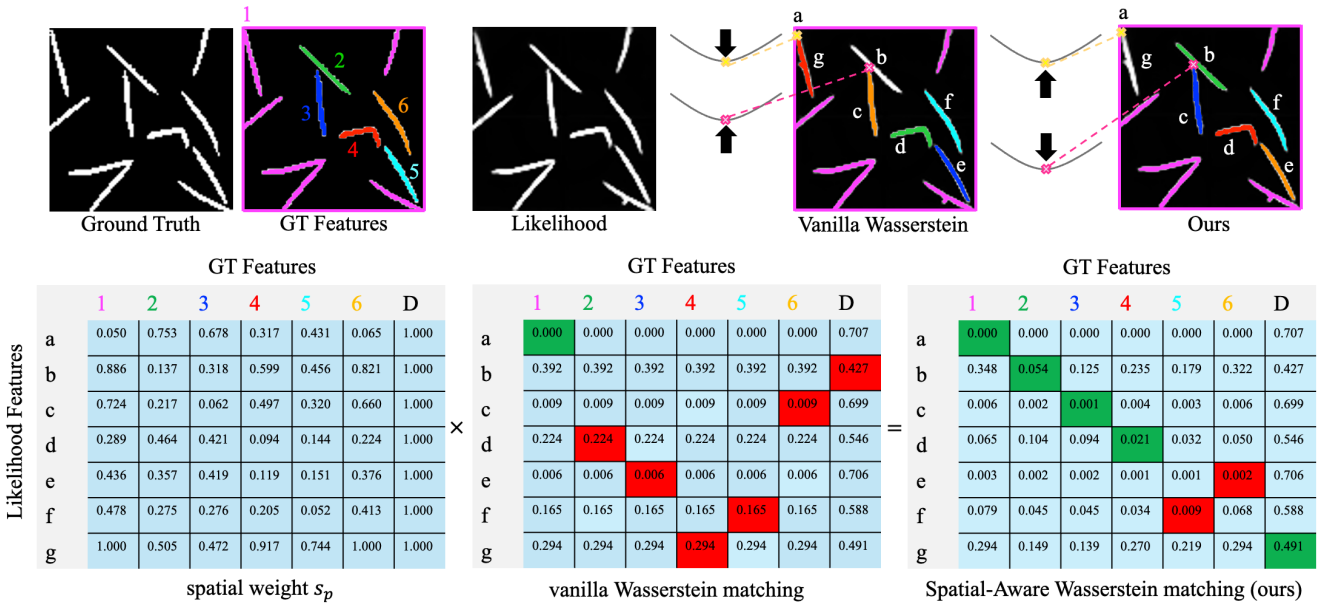

Figure 15. An illustrative example of how the matching works and how spatial-aware matching helps improve topological accuracy in the segmentation. The lower three tables show the spatial weight between each pair of persistent features, the cost matrix used by vanilla Wasserstein matching in previous methods, and the cost matrix after weighted by the spatial weight used by our proposed SATLoss, respectively. The green boxes refer to correct matchings and the red boxes refer to incorrect matchings. D refers to the diagonal of the persistent diagram. Note that by Wasserstein distance (optimal transport) the matching plan giving the least overall cost is selected, to each feature there is no guarantee that the minimal-cost matching is selected. The upper figures shows the matching results and the gradient behavior (see Sec. 3.2.2 for explanation). At the destroyer of the feature g (yellow cross), by vanilla Wasserstein matching the pixel value is pushed down (since matched with GT features), making feature a and g more separated (which should be connected). Similarly, at the destroyer of the feature b (pink cross), the gradient pushes the value up (since matched with diagonal), making feature b and c connected (which should be separated). In comparison, using the proposed spatial-aware matching, at the destroyer of feature g the pixel value is pushed up, making it more connected to feature a; at the destroyer of feature b the pixel value is pushed down, making it more separated from feature c. Therefore, allowing the segmentation model to make topologically more correct predictions. Similar process occurs repeatedly at different locations on the image in the optimization iterations.