Supplementary Materials for "`PointScatter`: Point Set Representation for Tubular Structure Extraction"

1 Implementation details

In this section, we supplement the implementation details that are not covered in Section 4.1.

In Equation (3), the weight of regression loss is set to $\lambda = 10$. As for the focal loss, we use $\alpha = 0.6$ for the segmentation task and $\alpha = 0.7$ for the centerline extraction task. The γ in focal loss is set to 2.0 for all circumstances. Specially, we set $\alpha = 0.8$ for the centerline extraction task of MassRoads and DeepGlobe. For all datasets except DeepGlobe, we use the ADAM optimizer with the initial learning rate 1e-3 and cosine learning rate schedule to train the network end-to-end. We use the poly learning rate schedule for Deepglobe and set the initial learning rate and the power to 1e-3 and 3, respectively. The weight decay is set to be 1e-4 uniformly.

We then introduce the dataset specified hyper-parameters. Since the numbers of images are significantly different for the datasets, we set the number of iterations separately for each dataset. For DRIVE and STARE, we train the `PointScatter` for 3K iterations, and 10K for MassRoads. We use batchsize=4 for these three datasets. The DeepGlobe is much harder than the other datasets, hence we set its iteration number to 40K with batchsize=16. The sizes of input images are 384×384 for DRIVE and STARE, 1024×1024 for MassRoads and 768×768 for DeepGlobe.

2 Ablation of the Matching Methods

We compare the performance of the greedy bipartite matching method and the Hungarian algorithm on the two tasks in Table 1 and Table 2. We use the same settings to train on these two methods for each dataset to achieve a fair comparison. On both tasks, these two methods have similar volumetric scores, while the greedy method yields a slightly better topology-based score.

A succinct implementation of the greedy bipartite matching is demonstrated in Listing 1. Batching on all regions makes our implementation efficient.

```python
import torch

def batched_greedy_assignment(cost):
    """
    Args:
        cost (Tensor): shape (num_region, num_pred (N), num_gt (K))
    Returns:
        assignment (LongTensor): shape (num_region, num_gt (K))
    """
    num_region, num_pred, num_gt = cost.shape
    assignment = - torch.ones(
        size=(num_region, num_gt),
        dtype=torch.long,
        device=cost.device
    )

    for i in range(num_gt):
        cur_min_idx = torch.argmin(cost[..., i], dim=-1)
        assignment[..., i] = cur_min_idx
        index = cur_min_idx.view(num_region, 1, 1).expand(-1, -1, num_gt)
        cost.scatter_(dim=1, index=index, value=float('inf'))

    return assignment
```

Listing 1: PyTorch codes of batched greedy bipartite assignment. This function assigns each ground-truth point to a predicted point uniquely. We assume that $K \leq N$ in input cost tensor.

Table 1: Comparison of the greedy and Hungarian matching algorithm on tubular structure segmentation task.

Dataset	Method	AUC(%)	Dice(%)	clDice(%)	ACC(%)	β_0Error	β_1Error	χ_{error}
STARE	Greedy	97.86	82.73	**85.83**	97.45	**0.818**	**0.774**	**0.978**
	Hungarian	**97.88**	**82.84**	85.77	**97.48**	0.861	0.789	1.021
MassRoads	Greedy	**97.65**	77.57	86.42	96.87	0.944	1.353	1.616
	Hungarian	97.60	**77.65**	**86.60**	**96.88**	**0.902**	**1.323**	**1.558**

Table 2: Comparison of the greedy and Hungarian matching algorithm on centerline extraction task.

Dataset	Method	AUC(%)	Dice(%)	Prec(%)	Recall(%)	ACC(%)	β_0Error	β_1Error	χ_{error}
STARE	Greedy	**94.52**	**81.77**	**92.09**	**73.52**	**99.10**	**2.158**	**1.424**	**2.303**
	Hungarian	93.95	80.67	91.85	71.91	99.07	3.286	1.702	3.431
MassRoads	Greedy	93.59	69.63	70.28	**68.99**	99.01	5.955	2.244	6.135
	Hungarian	**95.51**	**69.64**	**70.86**	68.45	**99.02**	**4.315**	**2.288**	**4.766**

3 Comparison with Other Methods

In Table 3, we further compare our method with previous approaches on the STARE dataset. We adopt the data split method in RV-GAN [4] for fair comparison. Our methods achieve SOTA performance on AUC and clDice, and achieve competitive Dice and ACC scores compared with other methods.

Table 3: Comparison with previous works on the vessel segmentation task. We follow the data split methods in RV-GAN. Our models are based on the U-Net backbone.

Dataset	Method	AUC(%)	Dice(%)	clDice(%)	ACC(%)
STARE	R2UNet [1]	99.14	**84.75**	-	97.12
	CE-Net [2]	98.71	83.13	85.87	97.59
	DUNet [3]	98.32	81.43	-	96.41
	IterNet [5]	99.15	81.46	-	**97.82**
	RV-GAN [4]	98.87	83.23	-	97.54
	`PointScatter`	**99.24**	84.52	87.46	97.75
	softDice+`PSAUX`	98.59	84.51	87.36	97.75
	clDice+`PSAUX`	98.59	84.74	**88.29**	97.77

4 Additional Qualitative Results

In this section, we provide more qualitative analysis of our `PointScatter`. In Fig. 1, we show the points labels and points predictions of our `PointScatter`. We use the gray value to represent the objectness score of each predicted point, the darker, the higher. Since the GT points are tiled as grid shape, the predicted points are also located in the center of each pixel.

We also illustrate more tubular segmentation and centerline extraction results in Fig. 2.

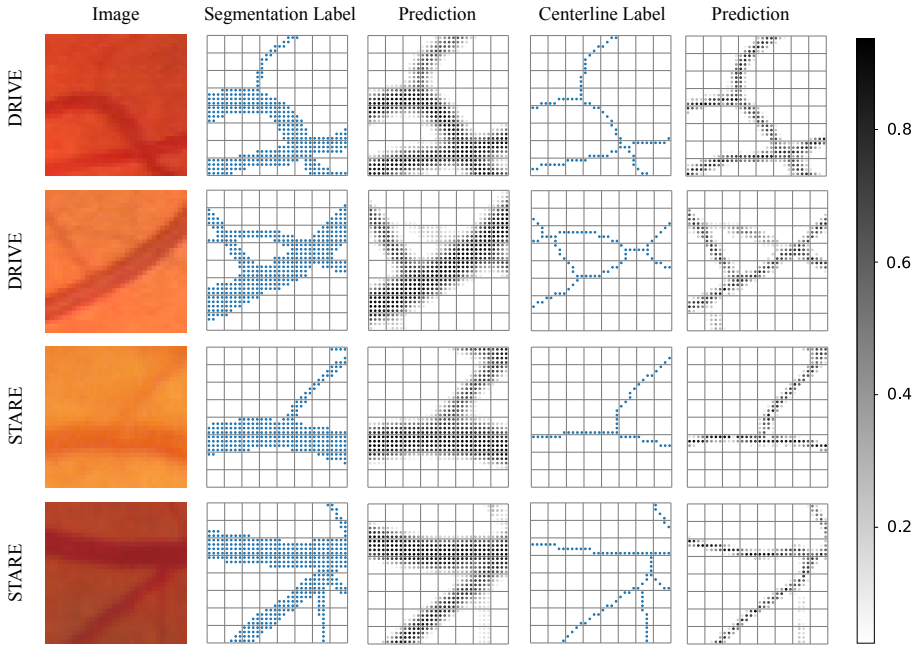

Fig. 1: Illustration of the points labels and points predictions of our `PointScatter` on the two tasks. We set $D = 4$ in our experiments and the each 4×4 bin in the images is corresponding to a scatter region.

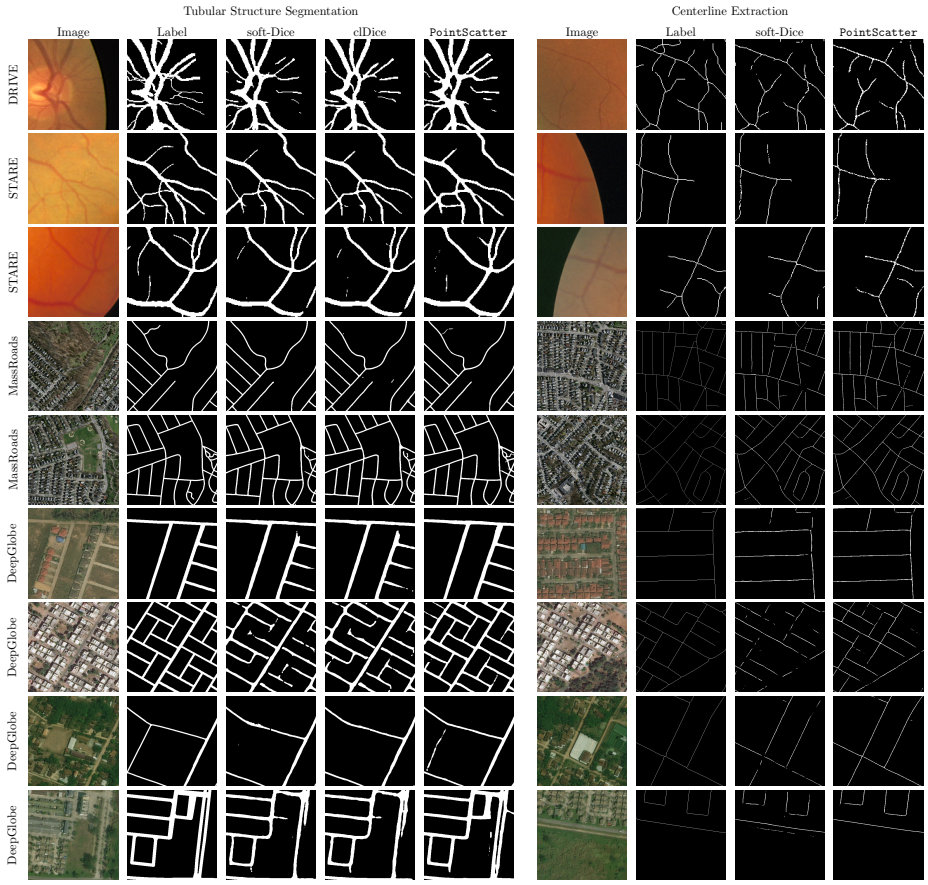

Fig. 2: Additional visual comparison for our `PointScatter` with other methods.

References

1. Alom, M.Z., Hasan, M., Yakopcic, C., Taha, T.M., Asari, V.K.: Recurrent residual convolutional neural network based on u-net (r2u-net) for medical image segmentation. arXiv preprint arXiv:1802.06955 (2018) 3
2. Gu, Z., Cheng, J., Fu, H., Zhou, K., Hao, H., Zhao, Y., Zhang, T., Gao, S., Liu, J.: Ce-net: Context encoder network for 2d medical image segmentation. IEEE transactions on medical imaging **38**(10), 2281–2292 (2019) 3
3. Jin, Q., Meng, Z., Pham, T.D., Chen, Q., Wei, L., Su, R.: Dunet: A deformable network for retinal vessel segmentation. Knowledge-Based Systems **178**, 149–162 (2019) 3
4. Kamran, S.A., Hossain, K.F., Tavakkoli, A., Zuckerbrod, S.L., Sanders, K.M., Baker, S.A.: Rv-gan: segmenting retinal vascular structure in fundus photographs using a novel multi-scale generative adversarial network. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 34–44. Springer (2021) 3
5. Li, L., Verma, M., Nakashima, Y., Nagahara, H., Kawasaki, R.: Iternet: Retinal image segmentation utilizing structural redundancy in vessel networks. In: Proceedings of the IEEE/CVF winter conference on applications of computer vision. pp. 3656–3665 (2020) 3