PointScatter: Point Set Representation for Tubular Structure Extraction

Dong Wang[1], Zhao Zhang[2], Ziwei Zhao[2,4,5], Yuhang Liu[3], Yihong Chen[2], and Liwei Wang[1,2](✉)

[1] Key Laboratory of Machine Perception, MOE, School of Artificial Intelligence,
Peking University, Beijing, China
wangdongcis@pku.edu.cn, wanglw@cis.pku.edu.cn
[2] Center for Data Science, Peking University, Beijing, China
2201213301@stu.pku.edu.cn, zhaozw@stu.pku.edu.cn, chenyihong@pku.edu.cn
[3] Yizhun Medical AI Co., Ltd, Beijing, China
yuhang.liu@yizhun-ai.com
[4] Peng Cheng Laboratory, Shenzhen, China
[5] Pazhou Laboratory (Huangpu), Guangzhou, China

Abstract. This paper explores the point set representation for tubular structure extraction tasks. Compared with the traditional mask representation, the point set representation enjoys its flexibility and representation ability, which would not be restricted by the fixed grid as the mask. Inspired by this, we propose PointScatter, an alternative to the segmentation models for the tubular structure extraction task. PointScatter splits the image into scatter regions and parallelly predicts points for each scatter region. We further propose the greedy-based region-wise bipartite matching algorithm to train the network end-to-end and efficiently. We benchmark the PointScatter on four public tubular datasets, and the extensive experiments on tubular structure segmentation and centerline extraction task demonstrate the effectiveness of our approach. *Code is available at* https://github.com/zhangzhao2022/pointscatter.

Keywords: Tubular structure · Medical image segmentation · Centerline extraction · Point set representation

1 Introduction

Tubular structures broadly exist in computer vision tasks, especially medical image tasks, such as blood vessels [26,51], ribs [22,52], and nerves [15]. Accurate extraction of these tubular structures performs a decisive role in the downstream tasks. For instance, the diagnosis of eye-related diseases such as hypertension, diabetic retinopathy highly relies on the extraction of retinal vessels.

D. Wang and Z. Zhang—Equal contribution.

Supplementary Information The online version contains supplementary material available at https://doi.org/10.1007/978-3-031-19803-8_22.

S. Avidan et al. (Eds.): ECCV 2022, LNCS 13681, pp. 366–383, 2022.
https://doi.org/10.1007/978-3-031-19803-8_22

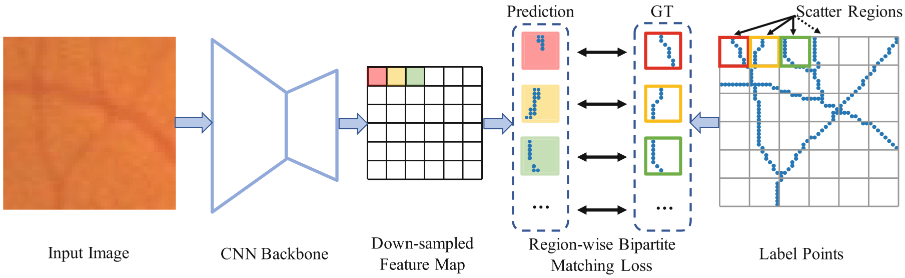


Fig. 1. `PointScatter` adopts point set representation to perform tubular structure extraction. We exhibit a small input image with size 48 × 48 to show the details clearly. `PointScatter` learns to predict points for each scatter region separately. Region-wise bipartite matching is employed in `PointScatter` to train the network end to end.

Deep learning-based methods usually model the extraction of tubular structures as a regular semantic segmentation task, which predicts segmentation masks as the representation of the structures. Therefore, previous works mostly adopt the following two routines: designing novel network components to incorporate vascular or tubular priors [37], or proposing loss functions that promote topology preservation [38].

Semantic segmentation methods apply successively upsampling on the high-level feature maps to get the predicted segmentation masks. The wide receptive field makes it more suitable to recognize large connected areas in the image. However, the paradigm of semantic segmentation has its inherent defect, which is further amplified in the task of tubular structure extraction. It is widely known that the segmentation models struggle in extracting high-frequency information accurately, such as image contours [8,9,18,19]. In tubular structures, the natural thinness makes almost all foreground regions contact with the structure boundaries. The special characteristics of the tubular structure increase the difficulty of capturing the fine-scale tubular details, which leads to false negatives of the small branches in the tubular structures.

We argue that the limitation lies in the representation of the prediction results. The semantic segmentation methods predict one score map to represent a segmentation result. The score map is arranged on a regular grid where each bin corresponds to a pixel in the input image. The fixed grid limits the flexibility of the representation and therefore restricts the ability of the network to learn fine-scale structures. Compared with the regular grids, point set representation is a more reasonable way for tubular structure extraction. Since the points can be placed at arbitrary real coordinates in the image, the point set representation enjoys more flexibility and expression ability to learn the detailed structures and is not restricted to a fixed grid.

Therefore, in this paper, we propose `PointScatter` to explore the feasibility of point set representation in tubular structure extraction. `PointScatter` (Fig. 1) is an alternative of the mask segmentation method and can apply to regular segmentation backbones (*e.g.* U-Net [35]) with minor modifications. Given a downsampled feature map output by the CNN backbone network, each localization of this feature map is responsible for predicting points in the corresponding **scatter region**. In this paper, we regard each patch as a scatter region as shown in Fig. 1, and each localization of the feature map corresponds to the image patch with the same relative position within the whole image. For each scatter region, our `PointScatter` predicts a fixed number of points with their objectness scores. When inference, a threshold is applied to filter out points with low scores. The aggregation of all scatter regions forms the final results.

Our `PointScatter` predicts points for all scatter regions parallelly at once, and the training process is also in an end-to-end and efficient manner. We apply the set matching approach separately for each scatter region to perform label assignment for training our `PointScatter`. Previous works in the object detection area (*e.g.* DETR [5]) adopt Hungarian algorithm [20] to perform one-to-one label assignment. Following this way, a straightforward way is using the Hungarian algorithm iteratively for each scatter region. However, the iteration process is inefficient for large images with thousands of scatter regions. Consequently, we propose the region-wise bipartite matching method which is based on the greedy approach. Our method reduces the computation complexity from $O(N^3)$ to $O(N^2)$ for each scatter region and is easier to be implemented on GPU using the vectorized programming by the deep learning framework (*e.g.* PyTorch [32]).

The advantages of our `PointScatter` and point sets lie in their flexibility and adaptability. 1) For the segmentation methods, each pixel of the output score maps corresponds to the pixel of the input image with the same spatial location, and has to predict the objectness score for this pixel. While in our `PointScatter`, the model can adaptively decide the assignments between the predicted and GT points within each scatter region. Since there are fewer restrictions on the assignments, the model is much easier to fit the complicated fine-scale structures in the training process. 2) During the `PointScatter` training, since we use points as GT rather than the mask, the predicted points can approach the GT points along the continuous spatial dimension. The extra dimension rather than the classification score dimension will reduce the optimization difficulty and provide more paths for the optimization algorithm to find the optimal solution during model training.

Experimentally, we evaluate `PointScatter` on four typical tubular datasets. For each dataset, we compare our methods with their segmentation counterparts on three strong backbone networks. We consider two tasks for tubular structure extraction: tubular structure segmentation and centerline extraction. Extensive experimental results reveal:

1. On the tubular structure segmentation task, according to the volumetric scores, our `PointScatter` achieves superior performance on most of the 12 combinations of the datasets and the backbone networks.
2. On the tubular structure segmentation task, using `PointScatter` as an auxiliary task to learn the centerline, the performance of both volumetric scores and topology-based metrics of the segmentation methods will be boosted.

3. On the centerline extraction task, our PointScatter significantly outperforms the segmentation counterparts by a large margin.
4. The qualitative analysis shows that our PointScatter is better than the segmentation methods on the small branches or bifurcation points, which verifies the expression ability of our method.

2 Related Work

2.1 Tubular Structure Segmentation

Tubular structure segmentation is a classical task due to the broad existence of tubular structures in medical images. Traditional methods [1–3,6,40] seek to exploit special geometric priors to improve the performance. Fethallah *et al.* [3] proposes an interactive method for tubular structure extraction. Once the physicians click on a small number of points, a set of minimal paths could be obtained through the marching algorithm. Amos *et al.* [40] considers centerline detection as a regression task and estimates the distance in scale space.

As for deep learning-based models, U-Net [35] and FCN [25] are the classical methods for semantic segmentation, which are also appropriate for tubular structures. To further improve the performance, approaches specially designed for tubular structures have been proposed recently. These methods can be coarsely classified into two categories: incorporating tubular priors into the network architecture [37,43,49] and designing topology-preserving loss functions [17,29,30,38]. Wang *et al.* [49] attempt to predict a segmentation mask and a distance map simultaneously for tubular structures. Then the mask could be refined through the shape prior reconstructed from the distance map. Shit *et al.* [38] introduces a new similarity measure called clDice to represent the topology architecture of tubular structures. Moreover, the differentiable version soft-clDice is proposed to train arbitrary segmentation networks. Oner *et al.* [30] proposes a connectivity-oriented loss function for training deep convolutional networks to reconstruct network-like structures. Besides these two ways, some researchers propose special approaches for their specific tasks. For instance, Li *et al.* [23] leverages a deep reinforced tree-traversal agent for efficient coronary artery centerline extraction. Different from the above methods, our PointScatter is the first to utilize points as a new representation for tubular structures, which significantly improves the segmentation performance.

2.2 Point Set Representation

Recently, points have become a popular choice to represent objects. Contributing to its flexibility and great expression capability, point representation is applied in various fields, such as image object detection [14,21,53], instance segmentation [54], pose estimation [4,31,48,58], 3D object classification and segmentation [33,34,56], *etc.* Benefiting from the advantage of points for both localization and recognition, RepPoints [53] utilizes point set as a new finer representation of objects instead of the rectangular bounding boxes. For the task of human pose estimation, detecting key points of humans is regarded as the prerequisite. Then, based on the prior knowledge of the human body, the skeletons can be obtained via the spatial connections among the detected key points. In the area of 3D object recognition, the point cloud is an important data structure. Thousands of points represented by the three coordinates (x, y, z) make up the scenes and objects. Qi *et al.* [33] provides a unified architecture for point cloud to achieve object classification and semantic segmentation. In this paper, we introduce the point set representation for tubular structures due to the expression ability of points to capture complex and fine-grained geometric structures.

2.3 Set Prediction by Deep Learning

The paradigm of set prediction has been introduced into the computer vision tasks (*e.g.* Object Detection [5,46,60]) firstly by DETR [5]. In DETR, a bipartite matching between ground truth and prediction is constructed based on the Hungarian algorithm [20], which guarantees that each target corresponds to a unique prediction. Following DETR, Wang *et al.* [50] and Sun *et al.* [42] perform one-to-one label assignment for classification to enable end-to-end object detection. More recently, researchers attempt to utilize the pattern of set prediction to improve the performance of other high-level tasks [10,11,45,50,61]. Cheng *et al.* [11] reformulates semantic segmentation as a mask set prediction problem and shows excellent empirical results. Wang *et al.* [50] predicts instance sequences directly via instance sequence set matching for video instance segmentation. In [10,61], the HOI instances can make up the triplet instance sets for both ground truth and prediction, which provides a simple and effective manner for Human Object Interaction (HOI) detection. Moreover, in the task of instance-aware human part parsing, [58] designs a specific differentiable matching method to generate the matching results for predicted limbs with different categories. In this paper, to train our `PointScatter`, we divide the image by predefined scatter regions and perform set predictions between predicted points and GT points on each of the regions in parallel.

3 Methodology

The `PointScatter` receives 2D images and produces point sets to represent the tubular structure. The training process of the set prediction task is end-to-end and efficient contributed by the region-wise bipartite matching method. We will first introduce the architecture of `PointScatter` in Sect. 3.1. Then Sect. 3.2 elaborates on the training process. An overview of our proposed `PointScatter` is shown in Fig. 2.

3.1 `PointScatter` Architecture

Our `PointScatter` formulates the tubular structure extraction task as a point set prediction task. The pipeline of the inference process of `PointScatter` is

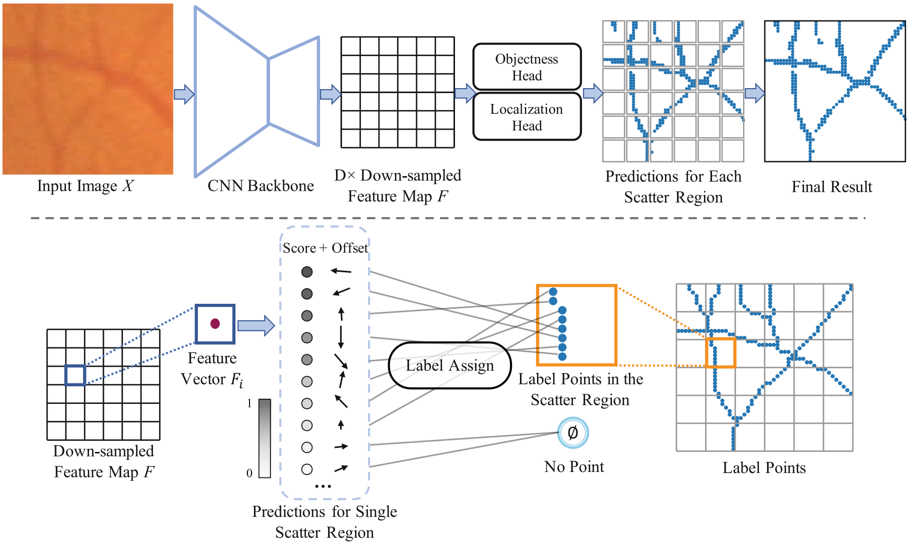


Fig. 2. The pipeline of `PointScatter`. The top part illustrates the pipeline of point set prediction of `PointScatter`. It predicts points for each scatter region separately and gathers them to form the final result. The bottom part exhibits the approach of label assignment for each scatter region. We obtain point-to-point assignments to supervise the network training precisely. The predicted points without match will be allocated a "no point" class.

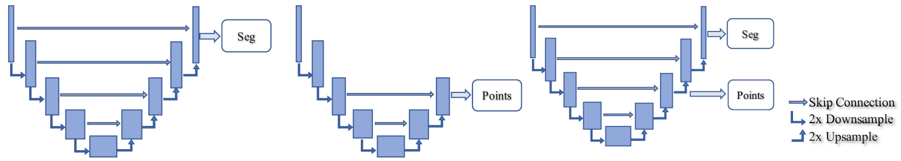
illustrated in the top part of Fig. 2. Given an input image X with shape $\mathbb{H} \times \mathbb{W}$, it is firstly fed into the CNN backbone network, and we obtain the corresponding down-sampled feature map $F \in \mathbb{R}^{C \times H \times W}$, where C is the channel size, H and W indicate the shape of the feature map. Let D denote the downsampling rate of the CNN backbone, we have

$$H = \mathbb{H}/D, \quad W = \mathbb{W}/D. \tag{1}$$

Note that we assume that $\mathbb{H}$ and $\mathbb{W}$ are divisible by D, which is the same situation as semantic segmentation.

Next, we introduce the concept of **scatter region**. In `PointScatter`, each spatial localization F_i in F is responsible for predicting the corresponding points that situate in a predefined region of the input image. i denotes the spatial index in $H \times W$. We call this predefined specific area **scatter region**. Note that the scatter region could be of arbitrary shape. In this paper, considering the natural grid shape of the feature map F, we define the scatter region as the $D \times D$ patch which has the same relative position in the input image as F_i in F. The top part of Fig. 2 provides an intuitive illustration.

We employ two head networks to perform point prediction for each scatter region. The objectness head and the localization head are responsible for producing point scores and offsets, respectively. The points of all scatter regions jointly constitute the final output.



(a) Segmentation (b) `PointScatter` (c) Segmentation + `PointScatter`

Fig. 3. Illustrations of applying segmentation method or our `PointScatter` on the U-Net backbone. We show an abstractive version of U-Net for ease of presentation. We set the downsample scale $D = 4$.

The points prediction mechanism from a high-level feature map F makes our `PointScatter` different from the mask segmentation methods (*e.g.* U-Net [35]). Instead of generating a grid of mask with the same shape as the original image, our `PointScatter` utilizes flexible points to describe the tubular structures. The ampliative representation ability enhances the power of the network to learn the complicated fine-scale structures. We will then introduce the details of the head and backbone networks in the following.

Head Networks. The head networks are responsible for predicting points for each image scatter region. They are composed of the Objectness Head (ObjHead) and the Localization Head (LocHead). For each scatter region, `PointScatter` generates N point candidates with their objectness scores and their localization, where N should be set greater than the maximum number of label points within a scatter region.

Formally, given F_i from the downsampled feature map F, where i indicates the spatial localization in $H \times W$, we denote the center localization of the corresponding image scatter region in the input image as $c_i = (X_i^c, Y_i^c)$. Then we predict the objectness score for N points and regress their coordination offsets relative to c_i by ObjHead and LocHead, respectively:

$$\text{score}_i = \text{Sigmoid}(\text{ObjHead}(F_i)) \in \mathbb{R}^N, \quad \text{offset}_i = \text{LocHead}(F_i) \in \mathbb{R}^{N\times 2}. \qquad (2)$$

Note that we apply Sigmoid operation after ObjHead to normalize the objectness score to the scale of $[0, 1]$. To acquire the coordinate of the points, we can simply apply the regressed offsets to the center point c_i, *i.e.* $p_i^j = (X_i^c + \text{offset}_i^{j,1}, Y_i^c + \text{offset}_i^{j,2})$, where p_i^j is the j_{th} point generated from F_i. During inference, the points with scores lower than a threshold T will be eliminated. Sliding these two heads on the whole feature map F and merging all predicted points, we can obtain the final results.

In practice, we instantiate ObjHead and LocHead both as a single linear layer, which can be implemented by the convolutional layer with kernel size 1×1. Although the transformer-like architecture [44] is proven to promote the interaction between object items in prior works [5,60], to maintain simplicity and focus on the point representation itself, we adopt the fully convolutional architecture in our `PointScatter`.

Backbone Networks. As introduced above, in `PointScatter`, the only requirement on the backbone network is that it should produce a $D\times$ downsampled feature map relative to the input image. The universality of `PointScatter` makes it compatible with almost all common backbone networks in semantic segmentation [25,35]. In this section, we take as an example the most famous model for medical image segmentation U-Net [35] to show how we apply `PointScatter` to a regular segmentation network.

We illustrate the utilization of the backbone network (*i.e.* U-Net) in Fig. 3. Traditional segmentation methods use the feature map with the same shape as the input image to generate the corresponding segmentation mask (Fig. 3a). Our `PointScatter` (Fig. 3b) passes the $D\times$ downsampled feature map to the head networks, while the successive upsampled feature maps are removed from the computational graph. In Fig. 3c, we show that we can simultaneously apply segmentation and `PointScatter` to the backbone network, which can be regarded as the multitask learning manner. We find that multitask learning will boost the performance of mask segmentation in the following experiments section.

3.2 Training `PointScatter`

The training of `PointScatter` also complies with the paradigm of scatter regions. Therefore, for each scatter region, we should assign the class label and the offset label for each predicted point. To achieve this goal, following prior works [5,60], we first define the cost function between predicted and ground-truth points, and then perform bipartite matching to produce one-to-one label assignment with a low global cost (bottom part of Fig. 2).

We first discuss the cost function. The matching cost should take into account both the objectness scores and the distance of the predicted and ground-truth points. Specifically, for each scatter region, we have a set of K ground-truth points $G = \{g_i\}_{i=1}^{K}$ and N predicted points $P = \{p_i\}_{i=1}^{N}$. Note that we omit the index of scatter region in this subsection for simplicity. For each predicted point p_i, its objectness score is denoted as s_i. Since the common assumption is $K \leq N$, we consider G also a set of size N, where the rest part is complemented by $\varnothing$ (no point). Therefore, for a permutation of N elements $\sigma \in \mathfrak{S}_N$, we define the cost for each point assignment as

$$\mathcal{L}_{\text{match}}(g_i, p_{\sigma(i)}) = [L_1(g_i, p_{\sigma(i)})]^{\eta} \cdot |s_{\sigma(i)} - \mathbb{1}(i \leq K)|^{1-\eta}, \tag{3}$$

where the first term in the equation describes the matching quality of point localization, and the last item indicates the classification error. L_1 is the manhattan distance in this equation, and η is a hyper-parameter determined by cross-validation. Note that we use multiplication instead of addition across the two cost terms, since the effectiveness of multiplication has been proven in [47].

Label Assignment with Region-wise Bipartite Matching. The second step is to get the optimal permutation σ. Previous works such as DETR [5] adopt the Hungarian algorithm [20] to perform set matching. Following this way, a direct generalization to our problem is to compute the bipartite matching iteratively for each scatter region. However, due to the large number of scatter regions in the images, it is inefficient to execute the iteration.

To tackle this problem, we propose a greedy-based bipartite matching method. We present the matching for each image scatter region in Algorithm 1. The greedy bipartite matching iterates the ground-truth points and finds the predicted point with the minimum cost from the left predicted points. The greedy method reduces the computational complexity of the Hungarian algorithm from $O(N^3)$ to $O(N^2)$, and is easy to be implemented on GPU using the deep learning framework (*i.e.* PyTorch) for batched computation. We provide an efficient implementation of the greedy bipartite matching in the supplementary materials.

The greedy method could not generate the optimal matching results. However, the network predictions become gradually closer to the ground-truth points during training. The optimization of the network improves the quality of predicted points, which makes the matching problem easier, hence the weak greedy method is capable of allocating the point targets.

Algorithm 1 Greedy Bipartite Matching

```
Input: $G = \{g_1, g_2, ..., g_K\}$, $P = \{p_1, p_2, ..., p_N\}$, $C \in \mathbb{R}^{K \times N}$
$G$ is the list of ground truth points, $P$ is the list of predicted points, $C$ is the cost matrix, $C_{i\cdot}$ is the i-th row of $C$, $C_{\cdot j}$ is the j-th column of $C$
Output: $S = \{\sigma(1), \sigma(2), ..., \sigma(K)\}$, $\sigma(i)$ represents that the predicted point $p_{\sigma(i)}$ is assigned to the ground truth point $g_i$; the rest of predicted points $\{p_{\sigma(K+1)}, ..., p_{\sigma(N)}\}$ are assigned to "no point"
begin
    for $i = 1$ to $K$
        $n \leftarrow \text{argmin}\ C_{i\cdot}$
        $\sigma(i) \leftarrow n$
        $C_{\cdot n} \leftarrow \text{inf}$
    end
    return $S$
end
```

Loss Functions. To train `PointScatter`, the loss function is composed of objectness loss and regression loss:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{obj}} + \lambda \mathcal{L}_{\text{reg}}, \tag{4}$$

where $\mathcal{L}_{\text{obj}}$ is instantiated as Focal Loss [24] to deal with the unbalanced distribution of objectness targets and we use L_1 loss for $\mathcal{L}_{\text{reg}}$. Note that the regression loss is only applied to the positive points and the points matched "no point" will be eliminated. Practically, we normalize the total loss by dividing the number of ground-truth points to keep the optimization process stable.

It is worth mentioning that almost all current datasets for tubular structure extraction provide mask annotation, therefore we should convert the mask annotation to points to supervise the training of PointScatter. We accomplish this goal by replacing each pixel with one point located in the center of this pixel. Concretely, a mask is represented as a matrix with binary values $\mathbf{Y}^{\mathbb{H}\times\mathbb{W}} \in \{0,1\}$, and we convert it to the point sets $\{(i,j)|Y_{i,j}=1, i\in[1,\mathbb{H}], j\in[1,\mathbb{W}]\}$.

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate our PointScatter on four public tubular datasets, including two medical datasets and two satellite datasets. DRIVE [41] and STARE [16] are two retinal datasets that are commonly adopted in the medical image segmentation problem to evaluate the performance of vessel segmentation. The Massachusetts Roads (MassRoad) dataset [28] and DeepGlobe [13] are labeled with the pixel-level annotation for road segmentation. We use the official data split for DRIVE and STARE in MMSegmentation [12] and follow the data split method in previous works [38,39] for the other two datasets. We report the performance on the test set.

Tasks and Metrics. In this paper, we focus on two different tasks relevant to the understanding of tubular structures: tubular structure segmentation and centerline extraction. The above four datasets are used for the image segmentation task in previous works, and the most popular dataset for centerline extraction is the MICCAI 2008 Coronary Artery Tracking (CAT08) dataset [36]. But unfortunately, the CAT08 dataset and the evaluation server are not publicly available now. Therefore, we generate the centerline labels using the skeleton extraction method in [38] to fulfill the evaluation of centerline extraction. The labelled centerline is a set of connected pixels with a line-like structure where the width is 1 pixel. It is a more challenging task to extract the centerlines accurately by deep models. We then introduce the metrics we utilize for these two tasks.

For the tubular structure segmentation task, we consider two types of metrics: volumetric and topology-based. The volumetric scores include Dice coefficient, Accuracy, AUC, and the recently proposed clDice [38]. We also calculate the topology-based scores including the mean of absolute Betti Errors for the Betti Numbers β_0 and β_1 and the mean absolute error of Euler characteristic. We follow [38] to compute the topology-based scores.

For the centerline extraction task, we report the Dice coefficient, Accuracy, AUC, Precision, and Recall as the volumetric scores. To increase the robustness of the evaluation, we apply a three-pixel tolerance region around the centerline annotation following [15]. We adopt the same topology-based metrics as the tubular structure segmentation task.

The above metrics are designed for mask prediction, while our PointScatter generates points to describe the foreground structures. Therefore, we should convert the points to the segmentation mask in order to accommodate the evaluation

Table 1. Main results on tubular structure segmentation task. The gray lines use our `PointScatter`. We mark the best performance by bold numbers.

Dataset	Backbone	Method	Volumetric Scores (%) ↑				Topological Error ↓		
			AUC	Dice	clDice	ACC	β_0	β_1	χ_{error}
DRIVE	UNet	softDice	97.05	81.09	80.69	95.28	1.504	1.129	1.806
		clDice	96.84	81.15	81.55	95.21	1.072	0.993	1.354
		`PointScatter`	**97.69**	**81.63**	**82.89**	95.23	1.317	1.250	1.628
		softDice+`PSAUX`	97.27	81.59	81.43	**95.37**	1.004	0.980	1.269
		clDice+`PSAUX`	96.97	81.51	82.54	95.24	**0.873**	**0.944**	**1.131**
	UNet++	softDice	96.42	80.96	80.55	95.24	1.698	1.106	1.978
		clDice	96.77	81.10	81.48	95.17	1.105	0.965	1.359
		`PointScatter`	**97.45**	**81.38**	**82.34**	95.17	1.290	1.225	1.600
		softDice+`PSAUX`	96.45	81.31	81.03	**95.29**	0.936	0.956	**1.184**
		clDice+`PSAUX`	96.51	81.28	81.62	95.22	**0.924**	**0.937**	1.189
	ResUNet	softDice	97.78	82.11	82.28	95.49	1.284	1.067	1.562
		clDice	97.09	81.43	82.48	95.21	1.005	**1.006**	1.272
		`PointScatter`	97.87	81.85	82.75	95.34	1.547	1.273	1.834
		softDice+`PSAUX`	**97.97**	**82.45**	82.64	**95.59**	1.372	1.023	1.628
		clDice+`PSAUX`	97.36	82.02	**84.62**	95.31	**0.883**	1.019	**1.142**
STARE	UNet	softDice	94.86	82.27	84.87	97.45	1.093	0.667	1.260
		clDice	96.82	82.29	85.22	97.44	0.790	0.665	0.943
		`PointScatter`	**97.86**	82.73	85.83	97.45	0.818	0.774	0.978
		softDice+`PSAUX`	96.42	82.78	85.44	97.51	0.727	0.625	0.887
		clDice+`PSAUX`	97.32	**83.11**	**86.45**	**97.54**	**0.631**	**0.614**	**0.778**
	UNet++	softDice	95.05	82.22	84.60	97.45	1.005	0.667	1.163
		clDice	96.48	82.62	85.72	97.49	0.801	0.648	0.968
		`PointScatter`	**97.85**	82.80	85.98	97.43	0.844	0.745	0.997
		softDice+`PSAUX`	95.59	82.85	85.54	**97.53**	0.658	0.649	0.801
		clDice+`PSAUX`	96.36	**82.96**	**86.11**	**97.53**	**0.650**	**0.617**	**0.800**
	ResUNet	softDice	96.27	81.65	84.11	97.38	0.913	0.695	1.051
		clDice	96.65	82.51	85.33	97.47	0.731	0.650	0.884
		`PointScatter`	**97.77**	82.40	85.00	97.38	0.949	0.730	1.093
		softDice+`PSAUX`	96.59	81.80	83.55	97.41	0.796	0.670	0.944
		clDice+`PSAUX`	96.04	**82.68**	**85.60**	**97.48**	**0.601**	**0.636**	**0.748**

Dataset	Backbone	Method	Volumetric Scores (%) ↑				Topological Error ↓		
			AUC	Dice	clDice	ACC	β_0	β_1	χ_{error}
MassRoads	UNet	softDice	97.02	76.96	86.33	96.86	0.686	1.361	1.356
		clDice	95.76	76.11	85.68	96.68	0.679	1.380	1.334
		`PointScatter`	**97.65**	77.57	86.42	96.87	0.944	1.353	1.616
		softDice+`PSAUX`	97.59	**78.14**	**87.38**	**96.98**	0.526	**1.257**	1.190
		clDice+`PSAUX`	96.60	77.68	87.34	96.88	**0.498**	1.316	**1.187**
	UNet++	softDice	97.10	76.88	86.08	96.82	0.690	1.373	1.351
		clDice	95.80	76.39	86.15	96.72	0.685	1.455	1.373
		`PointScatter`	**97.62**	77.65	86.40	96.90	0.836	1.315	1.503
		softDice+`PSAUX`	97.60	**78.10**	87.24	**96.99**	0.559	**1.306**	1.252
		clDice+`PSAUX`	96.41	77.81	**87.34**	96.91	**0.498**	1.316	**1.181**
	ResUNet	softDice	96.93	76.04	85.57	96.73	0.992	1.478	1.658
		clDice	96.12	75.97	85.69	96.68	0.887	1.521	1.571
		`PointScatter`	**97.40**	76.34	85.07	96.74	1.448	1.423	2.039
		softDice+`PSAUX`	**97.40**	**77.08**	86.31	**96.85**	**0.745**	**1.416**	**1.435**
		clDice+`PSAUX`	96.69	76.67	**86.36**	96.76	0.803	1.456	1.472
DeepGlobe	UNet	softDice	97.65	74.71	80.08	97.89	1.154	0.605	1.166
		clDice	96.03	74.96	81.16	97.90	0.691	0.556	0.751
		`PointScatter`	**98.64**	78.07	82.38	98.12	0.855	0.541	0.907
		softDice+`PSAUX`	98.27	**78.09**	**83.96**	**98.15**	0.492	**0.449**	0.530
		clDice+`PSAUX`	96.87	77.20	83.31	98.07	**0.435**	0.472	**0.485**
	LinkNet34	softDice	97.51	75.64	81.58	97.95	0.704	0.549	0.763
		clDice	97.03	75.63	82.03	97.94	0.590	0.646	0.706
		`PointScatter`	**98.59**	**79.21**	84.04	98.20	0.710	0.543	0.802
		softDice+`PSAUX`	98.00	78.88	**85.43**	**98.23**	0.491	**0.446**	0.549
		clDice+`PSAUX`	97.55	78.58	85.08	98.18	**0.451**	0.448	**0.516**
	DinkNet34	softDice	97.45	75.60	81.59	97.95	0.604	0.524	0.655
		clDice	97.07	75.23	82.18	97.92	0.938	0.562	1.009
		`PointScatter`	**98.66**	**79.39**	84.36	98.20	0.749	0.558	0.833
		softDice+`PSAUX`	98.30	78.95	**85.33**	**98.21**	0.513	**0.440**	**0.571**
		clDice+`PSAUX`	97.78	78.29	85.01	98.16	**0.511**	0.549	0.613

protocol. Specifically, an image can be regarded as a grid with the size of each bin 1×1, and each point is expected to be located in one bin. We first initialize an empty score map with the same size as the input image. For each bin in the output score map, we directly set the objectness score of the point located in this bin as its score. The bins without any point will be endowed with zero scores. To get the segmentation mask, we can threshold the score map by 0.5.

Implementation Details. As discussed in Sect. 3.1, our `PointScatter` is compatible with various segmentation backbones with an encoder-decoder shape, the adjustable parameters are the downsample rate D and the number of points in each scatter region N. Experimentally, we set $D = 4$ and $N = 16$ by default. The threshold of objectness score during inference is $T = 0.1$. During training, we set $\eta = 0.8$ in Eq. 3 to balance the localization cost and classification cost. Additional implementation details are depicted in the supplementary materials. We implement our model based on PyTorch and MMSegmentation [12].

4.2 Main Results

Our `PointScatter` can be regarded as an alternative for the segmentation approach. We compare our `PointScatter` with two very competitive segmentation methods (*i.e.* softDice [27] and clDice [38]) on various mainstream backbone networks [7,35,55,57,59]. We use the same training settings for the segmentation methods with our `PointScatter`, including the optimizer, training schedule, *etc.*

Table 2. Main results on centerline extraction task.

Dataset	Backbone	Method	Volumetric Scores (%) ↑					Topological Error ↓		
			AUC	Dice	Prec	Recall	ACC	β_0	β_1	χ_{error}
DRIVE	UNet	softDice	89.53	73.41	90.97	61.52	97.63	3.177	1.843	3.555
		PointScatter	**94.46**	**81.92**	**92.52**	**73.51**	**98.05**	5.203	2.612	5.509
		softDice+PS	93.13	75.92	91.08	65.08	97.76	**2.677**	**1.753**	**3.051**
	UNet++	softDice	83.83	72.06	90.48	59.87	97.54	5.651	2.371	6.006
		PointScatter	**93.29**	**82.23**	91.14	**74.91**	**98.04**	**1.959**	**1.657**	**2.282**
		softDice+PS	87.88	72.93	**91.29**	60.72	97.61	6.360	2.600	6.671
	ResUNet	softDice	90.83	74.86	91.36	63.40	97.70	2.774	1.723	3.147
		PointScatter	94.79	**83.91**	91.73	**77.31**	**98.18**	3.169	1.983	3.495
		softDice+PS	**95.40**	81.52	**92.95**	72.60	98.12	**2.479**	**1.673**	**2.853**
STARE	UNet	softDice	86.99	72.14	**93.01**	58.91	98.91	3.053	1.562	3.253
		PointScatter	**94.52**	**81.77**	92.09	**73.52**	**99.10**	2.158	1.424	2.303
		softDice+PS	88.34	75.10	92.51	63.20	98.98	**1.870**	**1.219**	**2.061**
	UNet++	softDice	84.94	73.08	92.87	60.24	98.93	3.388	1.556	3.588
		PointScatter	**93.07**	**80.03**	92.74	**70.38**	**99.07**	2.582	1.613	2.743
		softDice+PS	84.91	74.20	**93.20**	61.64	98.96	**2.300**	**1.368**	**2.487**
	ResUNet	softDice	86.56	73.93	92.66	61.49	98.95	2.568	1.341	2.760
		PointScatter	**95.93**	**82.44**	92.15	**74.58**	**99.12**	2.495	1.451	2.641
		softDice+PS	93.08	77.55	**93.24**	66.39	99.04	**1.964**	**1.199**	**2.152**
MassRoads	UNet	softDice	95.09	66.24	72.36	61.08	99.06	2.225	2.589	2.919
		PointScatter	93.59	**69.63**	70.28	**68.99**	99.01	5.955	2.244	6.135
		softDice+PS	**96.23**	67.60	**73.74**	62.40	**99.09**	**1.990**	**2.438**	**2.683**
	UNet++	softDice	95.07	66.01	71.94	60.99	99.05	**2.315**	2.699	**3.011**
		PointScatter	**96.54**	**69.93**	71.18	**68.73**	99.03	2.353	**2.407**	3.033
		softDice+PS	96.05	67.76	**73.06**	63.17	**99.08**	2.329	2.582	3.030
	ResUNet	softDice	94.84	65.84	71.52	61.01	99.04	2.816	2.784	3.512
		PointScatter	95.42	67.45	70.83	**64.38**	99.02	5.666	2.651	5.409
		softDice+PS	**96.43**	**67.52**	**73.00**	62.82	**99.08**	**2.295**	**2.628**	**2.996**
DeepGlobe	UNet	softDice	96.59	56.58	62.56	51.64	99.60	1.823	1.157	1.845
		PointScatter	**97.84**	**62.78**	66.38	**59.55**	99.62	2.127	1.059	2.145
		softDice+PS	97.59	61.17	**67.08**	56.21	**99.63**	**1.358**	**1.001**	**1.380**
	LinkNet34	softDice	96.62	57.42	62.82	52.87	99.60	1.481	1.108	1.503
		PointScatter	95.04	59.36	63.85	55.46	99.60	9.880	1.933	8.516
		softDice+PS	**97.44**	**61.55**	**66.97**	**56.95**	**99.63**	**1.395**	**1.022**	**1.417**
	DinkNet34	softDice	96.08	56.42	62.37	51.51	99.60	1.528	1.121	1.549
		PointScatter	95.54	60.89	64.20	**57.91**	99.60	7.341	1.375	6.520
		softDice+PS	**97.71**	**61.71**	**66.89**	57.27	**99.63**	**1.436**	**0.997**	**1.458**

These methods are also implemented by MMSegmentation for a fair comparison. Except for using `PointScatter` directly, we also study the effect of using our `PointScatter` as an auxiliary task for the segmentation method. We combine these two methods as shown in Fig. 3c and use the sum of the loss function of these two methods as the objective to train the network. We denote this method as `PSAUX` (abbreviation for `PointScatter` AUXiliary). We use the centerline labels to train the `PointScatter` branch in `PSAUX`.

Tubular Structure Segmentation. We exhibit the results in Table 1. According to the volumetric metrics, we can conclude that our `PointScatter` achieves superior performance compared to the segmentation methods on most of the combinations of the datasets and the backbone networks, which confirms the effectiveness of our `PointScatter`. When applying `PSAUX` to the segmentation method, we also observe improvements for most of the cases. The performance of `PSAUX` certifies that the point set representation leads to better feature learning for the backbone network. Our `PointScatter` obtains inferior performance than clDice according to the topology-based scores. We argue that it is because our `PointScatter` can capture more fine-scale structures which cannot be discovered by the segmentation models. These detailed predictions are beneficial to the volumetric scores but harmful to the topology. We will qualitatively analyze this phenomenon later. In addition, it is worth mentioning that `PSAUX` can improve the topology scores as shown in Table 1.

Centerline Extraction. We also conduct extensive experiments to validate the advantage of `PointScatter` for the centerline extraction task. As shown in Table 2, our `PointScatter` consistently surpasses the performance of the segmentation methods by a large margin according to the volumetric scores. Our `PointScatter` achieves similar precision to softDice, while complies with significantly higher recall values. It confirms again that our `PointScatter` can capture fine-scale details which cannot be detected by the segmentation model. The effect of `PSAUX` is similar to the tubular structure segmentation task.

Table 3. Ablation on N. ($D = 4$)

		Segmentation			Centerline	
Dataset	N	Dice (%)	clDice (%)	ACC (%)	Dice (%)	ACC (%)
DRIVE	8	64.80	66.71	92.27	78.31	97.79
	16	**81.63**	**82.89**	**95.23**	**81.92**	**98.05**
	32	78.73	80.73	94.60	79.07	97.85
	64	78.33	80.57	94.54	80.08	97.91
MassRoads	8	57.61	58.23	95.20	64.73	98.94
	16	**77.57**	86.42	96.87	69.63	99.01
	32	77.52	**86.55**	96.87	70.05	99.03
	64	77.54	86.40	**96.89**	**70.38**	**99.04**

Table 4. Ablation on D.

		Segmentation			Centerline	
Dataset	D	Dice (%)	clDice (%)	ACC (%)	Dice (%)	ACC (%)
DRIVE	2	81.26	82.16	95.20	**82.70**	**98.07**
	4	**81.63**	**82.89**	**95.23**	81.92	98.05
	8	79.80	80.48	94.77	78.59	97.81
MassRoads	2	**77.90**	**86.92**	**96.93**	**69.87**	99.02
	4	77.57	86.42	96.87	69.63	99.01
	8	77.54	86.31	96.86	68.79	**99.03**

4.3 Ablation Study

Number of Points (N). We ablate the number of predicted points (N) within each scatter region in Table 3. With $D = 4$, the maximum number of ground-truth points in each scatter region is 16. Therefore, the performance is not satisfactory when $N = 8$. Increasing N has marginal improvement on the performance when $N \geq 16$.

Downsample Rate (D). We compare the effect of different downsample rates D in Table 4. For the DRIVE dataset, $D = 4$ shows the best performance on the segmentation task while $D = 2$ is slightly better on the centerline extraction task. For the MassRoads dataset, different D yield similar performances on both tasks.

Greedy Bipartite Matching. Our greedy bipartite matching is theoretically faster than the Hungarian method and can be easily implemented on GPU. We compare the running time in each training iteration of these two methods in Table 5. We execute the greedy method on GPU TITAN RTX and the Hungarian algorithm on Intel(R) Xeon(R) CPU E5-2680 v4. The results show that our greedy method is at least three orders of magnitude faster than the Hungarian algorithm. The latency of our greedy method is negligible compared to the computation time of neural networks, whereas the latency of the Hungarian algorithm is unaffordable for large images.

Table 5. Running time (seconds) of Greedy and Hungarian bipartite matching. We set $D = 4$ and batchsize = 4.

Method	Complexity	Image size	Running time (seconds)
Greedy	$O(M^2)$	384 × 384	0.0076
		768 × 768	0.0100
		1024 × 1024	0.0123
Hungarian	$O(M^3)$	384 × 384	3.0043
		768 × 768	12.7027
		1024 × 1024	20.9922

4.4 Qualitative Analysis

We qualitatively compare our method and the mask segmentation methods in Fig. 4. Our PointScatter performs better on small branches or bifurcation points. It shows a better ability for our PointScatter to learn the complicated fine-scale information, which is contributed by the flexibility of the point set representation. Note that sometimes the small branches detected by PointScatter are not densely connected (*e.g.* the top left image), which decreases the performance on the topology-based metrics. However, it is better to extract tubular segments than miss the whole branch. We will leave future work to improve the topology performance of our PointScatter.

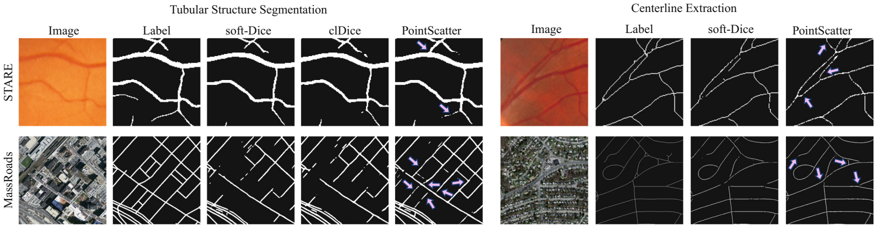


Fig. 4. Visual comparison for our PointScatter with other methods (zoom for details). The areas pointed by the arrows are missed by other models, while extracted by our PointScatter. More qualitative results can be found in the supplementary materials.

5 Conclusion

This paper proposes PointScatter, a novel architecture that introduces the point set representation for tubular structure extraction. This network can be trained end-to-end and efficiently with our proposed greedy bipartite matching algorithm. The extensive experiments reveal that our PointScatter achieves superior performance to the segmentation counterparts on the tubular structure segmentation task in most of the experiments, and significantly surpasses other methods on the centerline extraction task.

This novel design presents the potential of point set representation for tubular structures, and future work may include:

- Exploring the performance of PointScatter on the more challenging 3D tubular extraction tasks such as coronary vessel extraction.
- Improving the topology of predicted points of PointScatter to enhance the performance of the topology-based metrics.
- Promoting the point set representation for the general segmentation task.

Acknowledgement. This work is supported by Exploratory Research Project of Zhejiang Lab (No. 2022RC0AN02), Project 2020BD006 supported by PKUBaidu Fund.

References

1. Alvarez, L., et al.: Tracking the aortic lumen geometry by optimizing the 3D orientation of its cross-sections. In: Descoteaux, M., Maier-Hein, L., Franz, A., Jannin, P., Collins, D.L., Duchesne, S. (eds.) MICCAI 2017. LNCS, vol. 10434, pp. 174–181. Springer, Cham (2017). https://doi.org/10.1007/978-3-319-66185-8_20
2. Bauer, C., Pock, T., Sorantin, E., Bischof, H., Beichel, R.: Segmentation of interwoven 3d tubular tree structures utilizing shape priors and graph cuts. Med. Image Anal. **14**(2), 172–184 (2010)
3. Benmansour, F., Cohen, L.D.: Tubular structure segmentation based on minimal path method and anisotropic enhancement. Int. J. Comput. Vision **92**(2), 192–210 (2011)
4. Cao, Z., Simon, T., Wei, S.E., Sheikh, Y.: Realtime multi-person 2d pose estimation using part affinity fields. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 7291–7299 (2017)
5. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12346, pp. 213–229. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58452-8_13
6. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. Int. J. Comput. Vision **22**(1), 61–79 (1997)
7. Chaurasia, A., Culurciello, E.: LinkNet: exploiting encoder representations for efficient semantic segmentation. In: 2017 IEEE Visual Communications and Image Processing (VCIP), pp. 1–4. IEEE (2017)
8. Chen, L.C., Papandreou, G., Kokkinos, I., Murphy, K., Yuille, A.L.: DeepLab: semantic image segmentation with deep convolutional nets, Atrous convolution, and fully connected CRFs. IEEE Trans. Pattern Anal. Mach. Intell. **40**(4), 834–848 (2017)
9. Chen, L.-C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with Atrous separable convolution for semantic image segmentation. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) ECCV 2018. LNCS, vol. 11211, pp. 833–851. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01234-2_49
10. Chen, M., Liao, Y., Liu, S., Chen, Z., Wang, F., Qian, C.: Reformulating HOI detection as adaptive set prediction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9004–9013 (2021)
11. Cheng, B., Schwing, A., Kirillov, A.: Per-pixel classification is not all you need for semantic segmentation. Advances in Neural Information Processing Systems 34 (2021)
12. MMS Contributors: MMSegmentation: OpenMMLab semantic segmentation toolbox and benchmark (2020). www.github.com/open-mmlab/mmsegmentation
13. Demir, I., et al.: DeepGlobe 2018: a challenge to parse the earth through satellite images. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 172–181 (2018)
14. Duan, K., Bai, S., Xie, L., Qi, H., Huang, Q., Tian, Q.: CenterNet: keypoint triplets for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 6569–6578 (2019)
15. Guimaraes, P., Wigdahl, J., Ruggeri, A.: A fast and efficient technique for the automatic tracing of corneal nerves in confocal microscopy. Transl. Vision Sci. Technol. **5**(5), 7 (2016)

16. Hoover, A., Kouznetsova, V., Goldbaum, M.: Locating blood vessels in retinal images by piecewise threshold probing of a matched filter response. IEEE Trans. Med. Imaging **19**(3), 203–210 (2000)
17. Hu, X., Li, F., Samaras, D., Chen, C.: Topology-preserving deep image segmentation. In: Advances in Neural Information Processing Systems 32 (2019)
18. Kirillov, A., Levinkov, E., Andres, B., Savchynskyy, B., Rother, C.: InstanceCut: from edges to instances with MultiCut. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 5008–5017 (2017)
19. Kirillov, A., Wu, Y., He, K., Girshick, R.: PointRend: image segmentation as rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9799–9808 (2020)
20. Kuhn, H.W.: The Hungarian method for the assignment problem. Nav. Res. Logist. Q. **2**(1–2), 83–97 (1955)
21. Law, H., Deng, J.: CornerNet: detecting objects as paired keypoints. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. LNCS, vol. 11218, pp. 765–781. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01264-9_45
22. Lenga, M., Klinder, T., Bürger, C., von Berg, J., Franz, A., Lorenz, C.: Deep learning based rib centerline extraction and labeling. In: Vrtovec, T., Yao, J., Zheng, G., Pozo, J.M. (eds.) MSKI 2018. LNCS, vol. 11404, pp. 99–113. Springer, Cham (2019). https://doi.org/10.1007/978-3-030-11166-3_9
23. Li, Z., Xia, Q., Hu, Z., Wang, W., Xu, L., Zhang, S.: A deep reinforced tree-traversal agent for coronary artery centerline extraction. In: de Bruijne, M., et al. (eds.) MICCAI 2021. LNCS, vol. 12905, pp. 418–428. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-87240-3_40
24. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE International Conference on Computer Vision, pp. 2980–2988 (2017)
25. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3431–3440 (2015)
26. Ma, Y., et al.: ROSE: a retinal OCT-angiography vessel segmentation dataset and new model. IEEE Trans. Med. Imaging **40**(3), 928–939 (2020)
27. Milletari, F., Navab, N., Ahmadi, S.A.: V-Net: fully convolutional neural networks for volumetric medical image segmentation. In: 2016 Fourth International Conference on 3D Vision (3DV), pp. 565–571. IEEE (2016)
28. Mnih, V.: Machine learning for aerial image labeling. University of Toronto (Canada) (2013)
29. Mosinska, A., Marquez-Neila, P., Koziński, M., Fua, P.: Beyond the pixel-wise loss for topology-aware delineation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3136–3145 (2018)
30. Oner, D., Koziński, M., Citraro, L., Dadap, N.C., Konings, A.G., Fua, P.: Promoting connectivity of network-like structures by enforcing region separation. arXiv preprint arXiv:2009.07011 (2020)
31. Papandreou, G., Zhu, T., Chen, L.-C., Gidaris, S., Tompson, J., Murphy, K.: PersonLab: person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In: Ferrari, V., Hebert, M., Sminchisescu, C., Weiss, Y. (eds.) Computer Vision – ECCV 2018. LNCS, vol. 11218, pp. 282–299. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-01264-9_17

32. Paszke, A., et al.: PyTorch: an imperative style, high-performance deep learning library. In: Wallach, H.M., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E.B., Garnett, R. (eds.) NeurIPS (2019)
33. Qi, C.R., Su, H., Mo, K., Guibas, L.J.: PointNet: deep learning on point sets for 3d classification and segmentation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 652–660 (2017)
34. Qi, C.R., Yi, L., Su, H., Guibas, L.J.: Pointnet++: deep hierarchical feature learning on point sets in a metric space. In: Advances in Neural Information Processing Systems 30 (2017)
35. Ronneberger, O., Fischer, P., Brox, T.: U-Net: convolutional networks for biomedical image segmentation. In: Navab, N., Hornegger, J., Wells, W.M., Frangi, A.F. (eds.) MICCAI 2015. LNCS, vol. 9351, pp. 234–241. Springer, Cham (2015). https://doi.org/10.1007/978-3-319-24574-4_28
36. Schaap, M., et al.: Standardized evaluation methodology and reference database for evaluating coronary artery centerline extraction algorithms. Med. Image Anal. **13**(5), 701–714 (2009)
37. Shin, S.Y., Lee, S., Yun, I.D., Lee, K.M.: Deep vessel segmentation by learning graphical connectivity. Med. Image Anal. **58**, 101556 (2019)
38. Shit, S., et al.: clDice-a novel topology-preserving loss function for tubular structure segmentation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 16560–16569 (2021)
39. Singh, S., et al.: Self-supervised feature learning for semantic segmentation of overhead imagery. In: BMVC, vol. 1, p. 4 (2018)
40. Sironi, A., Lepetit, V., Fua, P.: Multiscale centerline detection by learning a scale-space distance transform. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 2697–2704 (2014)
41. Staal, J., Abràmoff, M.D., Niemeijer, M., Viergever, M.A., Van Ginneken, B.: Ridge-based vessel segmentation in color images of the retina. IEEE Trans. Med. Imaging **23**(4), 501–509 (2004)
42. Sun, P., et al.: What makes for end-to-end object detection? In: International Conference on Machine Learning, pp. 9934–9944. PMLR (2021)
43. Tetteh, G., et al.: DeepVesselNet: vessel segmentation, centerline prediction, and bifurcation detection in 3-d angiographic volumes. Front. Neurosci., 1285 (2020)
44. Vaswani, A., et al.: Attention is all you need. In: Advances in Neural Information Processing Systems 30 (2017)
45. Wang, H., Zhu, Y., Adam, H., Yuille, A., Chen, L.C.: MaX-DeepLab: end-to-end panoptic segmentation with mask transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5463–5474 (2021)
46. Wang, J., Song, L., Li, Z., Sun, H., Sun, J., Zheng, N.: End-to-end object detection with fully convolutional network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 15849–15858 (2021)
47. Wang, J., Song, L., Li, Z., Sun, H., Sun, J., Zheng, N.: End-to-end object detection with fully convolutional network. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 15849–15858, June 2021
48. Wang, J., et al.: Deep high-resolution representation learning for visual recognition. IEEE Trans. Pattern Anal. Mach. Intell. **43**(10), 3349–3364 (2020)
49. Wang, Y., et al.: Deep distance transform for tubular structure segmentation in CT scans. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 3833–3842 (2020)

50. Wang, Y., et al.: End-to-end video instance segmentation with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8741–8750 (2021)
51. Wu, H., Wang, W., Zhong, J., Lei, B., Wen, Z., Qin, J.: SCS-Net: a scale and context sensitive network for retinal vessel segmentation. Med. Image Anal. **70**, 102025 (2021)
52. Yang, J., Gu, S., Wei, D., Pfister, H., Ni, B.: RibSeg dataset and strong point cloud baselines for rib segmentation from CT scans. In: de Bruijne, M., et al. (eds.) MICCAI 2021. LNCS, vol. 12901, pp. 611–621. Springer, Cham (2021). https://doi.org/10.1007/978-3-030-87193-2_58
53. Yang, Z., Liu, S., Hu, H., Wang, L., Lin, S.: RepPoints: point set representation for object detection. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9657–9666 (2019)
54. Yang, Z., et al.: Dense RepPoints: representing visual objects with dense point sets. In: Vedaldi, A., Bischof, H., Brox, T., Frahm, J.-M. (eds.) ECCV 2020. LNCS, vol. 12366, pp. 227–244. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-58589-1_14
55. Zhang, Z., Liu, Q., Wang, Y.: Road extraction by deep residual U-Net. IEEE Geosci. Remote Sens. Lett. **15**(5), 749–753 (2018)
56. Zhao, H., Jiang, L., Jia, J., Torr, P.H., Koltun, V.: Point transformer. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 16259–16268 (2021)
57. Zhou, L., Zhang, C., Wu, M.: D-LinkNet: LinkNet with pretrained encoder and dilated convolution for high resolution satellite imagery road extraction. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 182–186 (2018)
58. Zhou, T., Wang, W., Liu, S., Yang, Y., Van Gool, L.: Differentiable multi-granularity human representation learning for instance-aware human semantic parsing. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 1622–1631 (2021)
59. Zhou, Z., Rahman Siddiquee, M.M., Tajbakhsh, N., Liang, J.: UNet++: a nested U-Net architecture for medical image segmentation. In: Stoyanov, D., et al. (eds.) DLMIA/ML-CDS -2018. LNCS, vol. 11045, pp. 3–11. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00889-5_1
60. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable DETR: deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020)
61. Zou, C., et al.: End-to-end human object interaction detection with HOI transformer. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 11825–11834 (2021)