Trexplorer: Recurrent DETR for Topologically Correct Tree Centerline Tracking

Roman Naeem[0009−0001−2625−125X], David Hagerman[0000−0002−5193−3789], Lennart Svensson[0000−0003−0206−9186], and Fredrik Kahl[0000−0001−9835−3020]

Chalmers University of Technology, 412 96 Gothenburg, Sweden
{nroman,david.hagerman,lennart.svensson,fredrik.kahl}@chalmers.se

Abstract. Tubular structures with tree topology such as blood vessels, lung airways, and more are abundant in human anatomy. Tracking these structures with correct topology is crucial for many downstream tasks that help in early detection of conditions such as vascular and pulmonary diseases. Current methods for centerline tracking suffer from predicting topologically incorrect centerlines and complex model pipelines. To mitigate these issues we propose Trexplorer, a recurrent DETR based model that tracks topologically correct centerlines of tubular tree objects in 3D volumes using a simple model pipeline. We demonstrate the model's performance on a publicly available synthetic vessel centerline dataset and show that our model outperforms the state-of-the-art on centerline topology and graph-related metrics, and performs well on detection metrics. The code is available at https://github.com/RomStriker/Trexplorer.

Keywords: centerline · tracking · tree topology

1 Introduction

Tubular structures with tree topologies are ubiquitous in human anatomy and can be found in the vascular system (arteries, veins, and capillaries), lung airways, renal tubules, and more. These structures are associated with many disease groups such as cardiovascular, pulmonary, and ophthalmological diseases, and tracking them in medical images aids early diagnosis and treatment planning [13,3]. Producing an accurate topology of the tracked structure is critical for downstream tasks such as hemodynamics and blood flow modeling [12], interventional/preoperative planning [5], vascular morphometry [7] and assessment of vascular diseases such as atherosclerosis and stenosis [4]. There are many ways of representing tree structures but a centerline graph is generally preferred as it provides a concise and semantically rich representation. Tools capable of generating topologically correct centerlines are, therefore, of great interest to the medical community.

Manual extraction of these centerlines is too time-consuming, and model-based methods [23,24] suffer from poor generalizability and performance. Deep learning-based approaches have recently seen increased popularity and success. One common approach is to post-process a predicted semantic segmentation

mask using thinning and skeletonization algorithms [19,2] to produce the centerlines. Segmentation models generally aggregate image features locally to handle 3D medical images such as CT and MRI scans. Due to local aggregation, the long-range dependencies of these trees are difficult to capture, leading to disconnectivity issues in the segmentation mask. Obtaining centerlines from such segmentation masks could, therefore, result in incorrect topology, as a fully connected tree cannot be guaranteed. Some segmentation models [6,17] utilize topology information by using topology-aware losses or graph priors. This leads to better connectivity but does not ensure a tree topology. CorSegRec [15] employs a complex three-stage pipeline to join the disconnected segments to the closest largest connected component using a regularized walk algorithm. However, it can make incorrect connections, especially when dealing with multiple disconnected components or more complex trees.

Another set of models iteratively tracks the centerline by leveraging the fact that the entire tree can be reached from its root. This procedure ensures that the resulting tree has the correct topology. One deep reinforcement learning (RL) based method [21] trains an agent to find centerline points, one action at a time. However, it cannot deal with bifurcations and does not predict important information such as the radius. Subsequent RL models [22,8] mitigate these problems by utilizing an additional detector model. However, the first approach struggles to find termination points, while the second requires specific techniques to prevent backtracking and the repetitive tracing of identical branches, resulting in a more complex model pipeline.

Centerline tracking can be framed as a combination of an object detection problem where we predict centerline points as objects and an edge prediction problem where we predict edges between all possible point pairs. Recently, Relationformer [16] and Vesselformer [14] have utilized this framework to perform simultaneous prediction of vertices and edges of a centerline graph in a 3D volume. These models are based on DETR [1], an end-to-end transformer [20] based object detector, and utilize object queries to detect centerline points. In contrast to iterative models, these models have a simple pipeline and utilize object queries to detect multiple branches simultaneously. However, they do not enforce the correct topology, leading to disconnected centerline components. Furthermore, for a complex tree with tortuous morphology, a large number of object queries are required to detect all the centerline points, significantly increasing model complexity. Lastly, Relationformer can only perform centerline detection on a small patch of the full volume. Vesselformer does the same but glues the output graph patches of the full volume together using custom heuristics in post-processing, which may introduce further topology errors.

Inspired by TrackFormer [10], a video multi-object tracker, we present Trexplorer, a novel recurrent DETR model with a simple pipeline that tracks the centerline graph of a tree structure in a complete volume, while also attributing important information like radius and class for each point of the centerline. In contrast to many existing methods, Trexplorer is guaranteed to generate a tree topology and does not require any post-processing steps to estimate the center-

line tree. Trexplorer combines the simplicity of transformer-based models and the dynamic programming approach of breaking down the centerline tracking into simpler sub-problems of the iterative models. Given a 3D volume and the root position of a tree in the volume, our model detects centerline points level by level, while also taking care of any new branches that might start due to a bifurcation. We train Trexplorer in an end-to-end fashion and use it to estimate the full centerline tree graph.

Our contributions are summarized as follows. **(1)** We present a novel method for centerline tracking, which is guaranteed to yield a tree topology without any post-processing. **(2)** We successfully modify a video object tracking method to generate tree-structured graphs in medical volumes. **(3)** We evaluate Trexplorer on the publicly available synthetic vessel dataset [19] and demonstrate that it yields state-of-the-art performance.

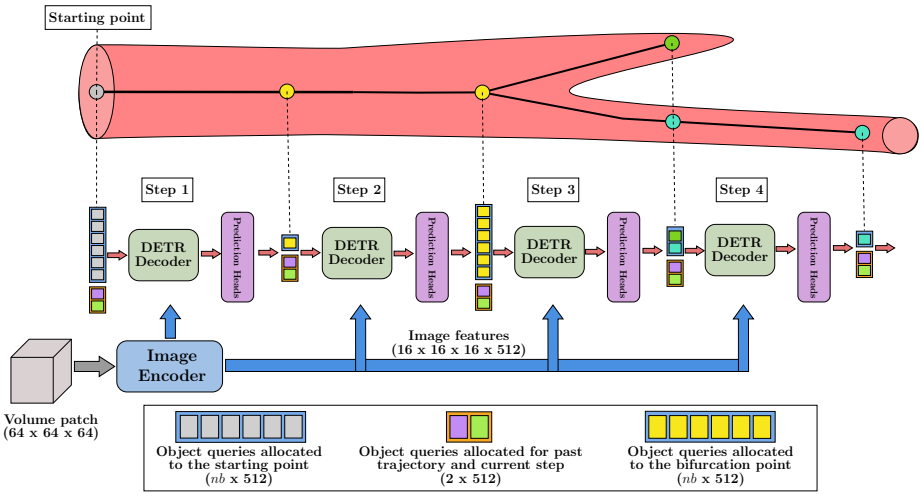

Fig. 1: Trexplorer architecture unrolled over four steps. Starting and bifurcation points use the same number of object queries nb. The prediction heads act as a filter for background and end-points while allowing intermediate points through for further tracking. For a bifurcation point, nb copies of the object query are created for tracking new branches.

2 Method

Our goal is to estimate a centerline tree. A tree is formally a graph (V, E) with nodes V and edges E. In our context, each node $\mathbf{v} \in V$ is a vector $\mathbf{v} = [\mathbf{x}, r]$ where $\mathbf{x} = [x, y, z]$ is the position of the node in our volume and r is the radius of the vessel at that point. To estimate the tree, we are given a 3D image, such as a CT or MRI scan, that contains the entire tree.

The basic idea in Trexplorer is to track every branch of the tree from the root node to the end of each branch, which means that we estimate the number of branches as well as the sequence of nodes along each branch. In each step of the algorithm, we estimate the next position of each branch (approximately one voxel away) and classify each node as either an end node, an intermediate node, or a bifurcation node. If the predicted node is an end node we stop tracking that branch, if it is an intermediate node we continue tracking it, and, if it is a bifurcation point, we start tracking all the branches leaving the bifurcation point. Trexplorer is a transformer-based model designed to solve this tracking problem in 3D volumes, see Fig. 1 for an illustration.

2.1 The Trexplorer Achitecture

The Trexplorer architecture builds on the DETR model [1] and is inspired by the TrackFormer model [10]. The DETR model contains an encoder block that extracts features from the input image and a decoder block that outputs object detections in terms of class probabilities and an estimated bounding box for each potential object. The Trexplorer architecture is a modified version of the DETR model, which outputs class probabilities for the classes *end, intermediate, bifurcation, background*, and estimates of the position of the next point and the radius of the vessel at that point.

Given the increased challenge of attention's quadratic complexity in a 3D space, both the image encoder and the transformer encoder in DETR have been replaced by a modified SwinUNETR [18] model. It utilizes windowed attention to efficiently create rich feature representations of an input volume. These features along with a set of object queries are used to compute cross-attention in the DETR decoder. To further reduce the number of tokens used in the cross-attention operation, the image features are extracted at $\frac{1}{4}$ of the initial resolution. Three MLPs are used as prediction heads to predict the position, radius, and class of each object query. See supplementary material for detailed architecture figures.

2.2 Object Queries and Bifurcations

The input object queries used in Trexplorer represent the previous state of branches that are currently tracked. Once updated through cross-attention with the image features, the updated object queries will represent the next state of those branches. As only a single object query is responsible for tracking a branch, it allows our model to predict very dense centerline trees with only a few object queries. The decision on how these updated queries are used is contingent on the classification head's output. The outputs of all object queries that are classified as intermediate points will be added as inputs to the next step. However, object queries marked as either an end-point or background are discarded, with the key distinction that the end-points are added to the global graph, marking where branches stop. If an object query is classified as a bifurcation, a fixed number nb of its copies are added to the next step's input, each with its own learned

positional embedding. The positional embedding corresponds to the nb possible directions from a voxel in a 3D grid. Intuitively, it can be seen as if the bifurcating vessel ends, up to nb new vessels could begin from that location, each with its own direction. The value of nb should be set to at least the maximum expected bifurcation degree in the data to allow the model to track all new possible branches. The object queries that do not correspond to new vessels will be classified as background in the next step and are therefore tracked for only one step and never added to the global graph. The bifurcation object queries attend to each other through self-attention and are penalized for tracking the same branch in DETR's Hungarian loss, which discourages overlapping centerlines.

2.3 Efficient Tracking Using Patches

As vessel tracking requires voxel-level image features, the number of tokens grows exponentially with the resolution of the input volume. This is an issue even considering the linear scaling of cross-attention and windowed attention. The most common strategy, also used in this work, is to train on patches taken from the complete volume. While this reduces the compute requirement significantly, it also introduces several new issues: Firstly, inference cannot simply be performed in a naive sliding-window fashion as this would create disconnected sub-trees in each patch, secondly, the vessel tracking in a patch does not have any information regarding the tree topology in the surrounding patches, and finally, the image features used as context for our object queries are now restricted to the patch.

In Trexplorer, we define a patch as a 64x64x64 cube, and vessels are tracked for ten steps starting from the central voxel. During inference, once tracking has finished in a patch, new starting points are created from the endpoints of the centerline graph of the current patch, and new patches centered around these endpoints are created. The model then tracks the centerline in each of the new patches independently. This ensures connectivity between patches and as the number of steps tracked is much smaller than the actual size of the patch, the context will always contain the previous tracked vessel. To further improve the performance when changing patches, a past trajectory token is added to the object queries. This token is obtained by embedding the past trajectory vector using an MLP. The vector is a concatenation of the position and radius from the past trajectory nodes in the previous patch. The model also uses a step token to get information about the current step which is a linear embedding of the current step number.

2.4 Loss Functions

Let $(\mathbf{p}, \hat{\mathbf{x}}, \hat{r})$ denote the predicted class probabilities, position, and radius for a single node, whereas c is the node class and $\mathbf{x}$ and r denote the position of and radius at the node if the class is not *background*. For the class probabilities, we use a class-balanced multi-class focal loss [9],

$$\mathrm{FL}(\mathbf{p}, c) = -(1 - w_c)(1 - \mathbf{p}_c)^{\gamma} \log(\mathbf{p}_c), \tag{1}$$

where $\mathbf{p}_c$ is the c^{th} element in $\mathbf{p}$, $(1 - w_c)$ is a class balancing parameter and w_c represents the frequency of class c in our training data. The total loss for predicting $(\mathbf{p}, \hat{\mathbf{x}}, \hat{r})$ is

$$L = \frac{w_{cls}}{Q}\text{FL}(\mathbf{p}, c) + \frac{I(c)}{N}\left(w_{\text{pos}}\|x - \hat{\mathbf{x}}\|_1 + w_{rad}|r - \hat{r}|\right), \tag{2}$$

where Q is number of allocated queries in the step, N is number of target points in the step, and w_{cls}, w_{pos}, w_{rad} are the weighting coefficients of the different losses and $I(c)$ is an indicator function that takes the value 1 unless $c =$background for which $I(c) = 0$.

After a bifurcation point, we will make multiple (Q) predictions of multiple (N) ground truth nodes. The association between predictions and ground truth nodes is generally unknown. To compute the loss function we follow the DETR model's Hungarian loss and solve an assignment problem where we minimize the total loss of all assignments. A prediction not assigned to a ground truth node is assumed to be background.

3 Experiments and Results

3.1 Dataset

We evaluate Trexplorer on the synthetic vessel dataset [19], which is the only publicly available vessel centerline dataset to the best of our knowledge. The synthetic vessel dataset contains 136 3D volumes of size (325 x 204 x 600). Each volume contains $\{min : 11, max : 21, mean : 16.1\}$ vessel trees. Each tree has a width of $\{min : 1, max : 97, mean : 19.6\}$ and the depth is $\{min : 34, max : 1319, mean : 438.7\}$. The bifurcation degree is always 2. The max radius of each vessel tree is $\{min : 11, max : 21, mean : 16.1\}$ voxels. The same training and test splits as reported by Vesselformer [14] are used, i.e. the first 40 volumes for training and validation, and the next 10 volumes for testing.

3.2 Experiments

Trexplorer is trained on the synthetic dataset from scratch. We train the model for 240,000 iterations with a batch size of 8. The number of tokens allocated to a bifurcation point nb is set to 2, while the max number of concurrently tracking tokens mq is set to 10. The models have been implemented using the open-source libraries Pytorch and MONAI, and are trained using 4 A100 GPUs on a single node with mixed precision enabled.

3.3 Results

We report the same metrics as reported by Vesselformer for a fair comparison. The reported metrics include the Street Mover Distance (SMD) which is the graph Wasserstein distance, the relative error in % of Betti-0 (the number of

connected components) and Betti-1 (the number of cycles), mean average precision (mAP) and mean average recall (mAR) for both nodes and edges, and the mean absolute error for the predicted radius. We also include the results obtained by Voreen [11], an open-source framework for the analysis of multi-modal volumetric data. The results for Voreen and Vesselformer are taken from results reported by Vesselformer's authors.

As shown in Table 1, Trexplorer has the lowest SMD score which is a graph similarity score, showcasing our model's great performance on whole graph tracking. Both Betti-0 and Betti-1 topology errors are zero for Trexplorer, due to its constrained topology output. Our model has lower node and edge mAP compared to Vesselformer. However, Vesselformer is trained and tested on a pruned version of the centerlines, where nodes of degree 2 with neighboring edges forming angles larger than 160 degrees are removed, and the neighboring nodes are connected. In contrast, Trexplorer is trained on dense centerlines with node spacing of around 1 voxel. This results in a predicted centerline graph with many nodes for a single volume. To be able to compute matching-based mAP and mAR scores in a reasonable time, we prune our predicted and ground truth vessel centerlines in the same manner as described above. This may result in the pruning of true positives, leading to lower scores. Another reason for a lower mAP is that in the synthetic dataset, some end-points of a vessel tree can be connected or be very close to a different vessel tree, and upon reaching those end-points, the model starts tracking this connected tree, resulting in extra false positives.

Table 1: Comparison of Trexplorer with Vesselformer and Voreen. Some of the results are missing for Voreen due to instability in metric computation.

Model	SMD ↓	%-Betti Error ↓		Node ↑		Edge ↑		MAE ↓
		Betti-0	Betti-1	mAP	mAR	mAP	mAR	(radius)
Voreen	0.03071	0.2955	0.2766	36.17	43.35	*	*	1.79
Vesselformer	0.01381	0.2188	0.2054	**72.32**	**80.11**	**72.19**	76.24	**0.52**
Trexplorer	**0.0075**	**0.0000**	**0.0000**	60.88	77.88	59.74	**82.45**	0.74

We compare the outputs of Trexplorer and Vesselformer with the ground truth centerline graph in Figure 2. The examples show that Trexplorer can predict complicated vessel trees accurately with correct topology while Vesselformer struggles to get the topology right and even misses some branches altogether. Vesselformer predicts sparse centerline graphs and for the sake of this comparison, we add extra nodes between the connected edges to make it dense.

3.4 Ablations

In this section, we ablate two components that are added to the DETR model to help it with the recurrent flow of information, namely the past trajectory token

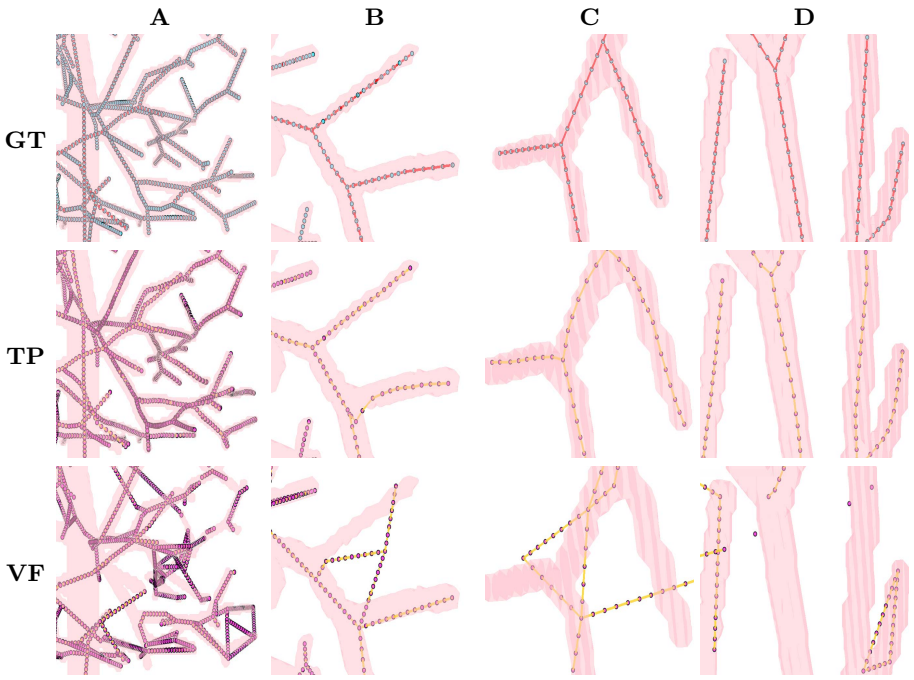

Fig. 2: Visual comparison between ground truth (GT), Trexplorer (TP), and Vesselformer (VF) centerlines using four examples patches (A, B, C, D) from the synthetic vessel dataset.

and the step token. For the past trajectory, we change the number of previous positions and the radius. For this ablation study, we use a patch-wise mean average precision (mAP) and bipartite matched F1-score (BP-F1), evaluated on 640 validation patches. The results are shown in Table 2. Using the past trajectory token results in a significant performance boost as it provides the decoder with past context and helps it determine which way to go next. Adding more past positions, corresponding radii, and the step token leads to small improvements.

Table 2: Ablation experiments for examining the past trajectory and step token.

Index	Past trajectory token		Step Token	BP-F1	mAP
	Num. Prev. Pos.	Radius			
1	0	✗	✓	0.900	0.851
2	5	✗	✓	0.944	0.900
3	5	✗	✗	0.941	0.900
4	10	✓	✓	**0.946**	**0.901**

4 Conclusion

We propose a novel recurrent DETR model and demonstrate that our model can effectively track centerline graphs with correct topology using a simple pipeline. Trexplorer does not require post-processing and can produce a vessel tree for huge volumes by processing only relevant patches. The results show that it performs significantly better than the state-of-the-art model on the graph and topology-related metrics while performing comparably on other detection-based metrics. Although the results are impressive, our model has limitations such as possible premature tracking termination and duplicate tracking. Future works can potentially address these limitations by utilizing DETR-variants with stronger priors and advanced tree-matching algorithms.

Acknowledgments. The compute resources were provided by the National Academic Infrastructure for Supercomputing in Sweden (NAISS), partially funded by the Swedish Research Council through grant agreement no. 2022-06725.

Disclosure of Interests. The authors declare that they have no competing interests relevant to the content of this article.

References

1. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision. pp. 213–229. Springer (2020)
2. Chen, L., Liu, W., Balu, N., Mossa-Basha, M., Hatsukami, T.S., Hwang, J.N., Yuan, C.: Deep open snake tracker for vessel tracing. In: Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part VI 24. pp. 579–589. Springer (2021)
3. Cheng, G., Wu, X., Xiang, W., Guo, C., Ji, H., He, L.: Segmentation of the airway tree from chest ct using tiny atrous convolutional network. IEEE Access **9**, 33583–33594 (2021)
4. Choi, A.D., Marques, H., Kumar, V., Griffin, W.F., Rahban, H., Karlsberg, R.P., Zeman, R.K., Katz, R.J., Earls, J.P.: Ct evaluation by artificial intelligence for atherosclerosis, stenosis and vascular morphology(clarify): A multi-center, international study. Journal of Cardiovascular Computed Tomography **15**(6), 470–476 (2021)
5. Huang, D., Tang, W., Ding, Y., Wan, T., Chen, Y.: An interactive 3d preoperative planning and training system for minimally invasive vascular surgery. In: 2011 12th International Conference on Computer-Aided Design and Computer Graphics. pp. 443–449. IEEE (2011)
6. Keshwani, D., Kitamura, Y., Ihara, S., Iizuka, S., Simo-Serra, E.: Topnet: Topology preserving metric learning for vessel tree reconstruction and labelling. In: Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part VI 23. pp. 14–23. Springer (2020)

7. Khan, Z., Ngo, J.P., Le, B., Evans, R.G., Pearson, J.T., Gardiner, B.S., Smith, D.W.: Three-dimensional morphometric analysis of the renal vasculature. American Journal of Physiology-Renal Physiology **314**(5), F715–F725 (2018)
8. Li, Z., Xia, Q., Hu, Z., Wang, W., Xu, L., Zhang, S.: A deep reinforced tree-traversal agent for coronary artery centerline extraction. In: Medical Image Computing and Computer Assisted Intervention–MICCAI 2021: 24th International Conference, Strasbourg, France, September 27–October 1, 2021, Proceedings, Part V 24. pp. 418–428. Springer (2021)
9. 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)
10. Meinhardt, T., Kirillov, A., Leal-Taixe, L., Feichtenhofer, C.: Trackformer: Multi-object tracking with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 8844–8854 (2022)
11. Meyer-Spradow, J., Ropinski, T., Mensmann, J., Hinrichs, K.: Voreen: A rapid-prototyping environment for ray-casting-based volume visualizations. IEEE Computer Graphics and Applications **29**(6), 6–13 (2009)
12. Miraucourt, O., Salmon, S., Szopos, M., Thiriet, M.: Blood flow in the cerebral venous system: modeling and simulation. Computer methods in biomechanics and biomedical engineering **20**(5), 471–482 (2017)
13. Moccia, S., De Momi, E., El Hadji, S., Mattos, L.S.: Blood vessel segmentation algorithms—review of methods, datasets and evaluation metrics. Computer methods and programs in biomedicine **158**, 71–91 (2018)
14. Prabhakar, C., Shit, S., Paetzold, J.C., Ezhov, I., Koner, R., Li, H., Kofler, F.S., Menze, B.: Vesselformer: Towards complete 3d vessel graph generation from images. In: Medical Imaging with Deep Learning. pp. 320–331. PMLR (2024)
15. Qiu, Y., Li, Z., Wang, Y., Dong, P., Wu, D., Yang, X., Hong, Q., Shen, D.: Corsegrec: a topology-preserving scheme for extracting fully-connected coronary arteries from ct angiography. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 670–680. Springer (2023)
16. Shit, S., Koner, R., Wittmann, B., Paetzold, J., Ezhov, I., Li, H., Pan, J., Sharifzadeh, S., Kaissis, G., Tresp, V., et al.: Relationformer: A unified framework for image-to-graph generation. In: European Conference on Computer Vision. pp. 422–439. Springer (2022)
17. Tan, Z., Feng, J., Zhou, J.: Sgnet: Structure-aware graph-based network for airway semantic segmentation. In: International Conference on Medical Image Computing and Computer-Assisted Intervention. pp. 153–163. Springer (2021)
18. Tang, Y., Yang, D., Li, W., Roth, H.R., Landman, B., Xu, D., Nath, V., Hatamizadeh, A.: Self-supervised pre-training of swin transformers for 3d medical image analysis. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 20730–20740 (2022)
19. Tetteh, G., Efremov, V., Forkert, N.D., Schneider, M., Kirschke, J., Weber, B., Zimmer, C., Piraud, M., Menze, B.H.: Deepvesselnet: Vessel segmentation, centerline prediction, and bifurcation detection in 3-d angiographic volumes. Frontiers in Neuroscience **14**, 1285 (2020)
20. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)
21. Zhang, P., Wang, F., Zheng, Y.: Deep reinforcement learning for vessel centerline tracing in multi-modality 3d volumes. In: Medical Image Computing and

Computer Assisted Intervention–MICCAI 2018: 21st International Conference, Granada, Spain, September 16-20, 2018, Proceedings, Part IV 11. pp. 755–763. Springer (2018)
22. Zhang, Y., Luo, G., Wang, W., Wang, K.: Branch-aware double dqn for centerline extraction in coronary ct angiography. In: Medical Image Computing and Computer Assisted Intervention–MICCAI 2020: 23rd International Conference, Lima, Peru, October 4–8, 2020, Proceedings, Part VI 23. pp. 35–44. Springer (2020)
23. Zhang, Z., Marin, D., Chesakov, E., Maza, M.M., Drangova, M., Boykov, Y.: Divergence prior and vessel-tree reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10216–10224 (2019)
24. Zhang, Z., Marin, D., Drangova, M., Boykov, Y.: Confluent vessel trees with accurate bifurcations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 9573–9582 (2021)