

RefTr: Recurrent Refinement of Confluent Trajectories for 3D Vascular Tree Centerline Graphs

Roman Naeem

David Hagerman

Jennifer Alvén

Fredrik Kahl

Chalmers University of Technology

Abstract

Tubular trees, such as blood vessels and lung airways, are essential for material transport within the human body. Accurately detecting their centerlines with correct tree topology is critical for clinical tasks such as diagnosis, treatment planning, and surgical navigation. In these applications, maintaining high recall is crucial, as missing small branches can result in fatal mistakes caused by incomplete assessments or undetected abnormalities. We present RefTr, a 3D image-to-graph model for centerline generation of vascular trees via recurrent refinement of confluent trajectories. RefTr uses a Producer-Refiner architecture based on a Transformer decoder, where the Producer proposes a set of initial confluent trajectories that are recurrently refined by the Refiner to produce final trajectories, which forms the centerline graph. The confluent trajectory representation enables refinement of complete trajectories while explicitly enforcing a valid tree topology. The recurrent refinement scheme improves precision and reuses the same Refiner block across multiple steps, yielding a 2.4x reduction in decoder parameters compared to previous SOTA. We also introduce an efficient non-maximum suppression algorithm for spatial tree graphs to merge duplicate branches and boost precision. Across multiple public centerline datasets, RefTr achieves superior recall and comparable precision to previous SOTA, while offering faster inference and substantially fewer parameters, demonstrating its potential as a new state-of-the-art framework for vascular tree analysis in 3D medical imaging. The code is available at <https://github.com/RomStriker/RefTr>.

1. Introduction

Vascular trees, including blood vessels, airways, and lymphatic structures, are critical for transporting essential substances throughout the body, and abnormalities in these networks are linked to many diseases. Accurate and topologically correct centerlines are important for applications

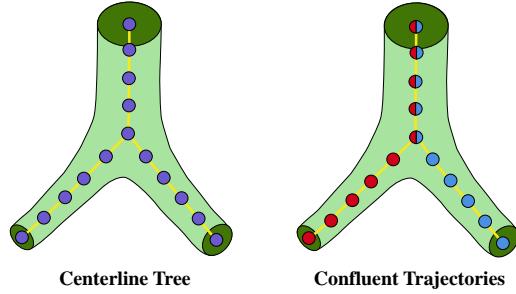


Figure 1. Illustration of confluent trajectory representation: two trajectories are confluent for the initial five points before diverging, modeling the branching structure of vessel centerlines.

such as diagnosis and treatment planning [6, 11], hemodynamic modeling [10], interventional planning [4], vascular morphometry [5], and disease assessment [3]. Centerline graphs provide a compact and interpretable representation of these trees, are easier to annotate than dense masks, and more effective enforcement of valid tree topology.

Existing methods for centerline extraction fall into three groups: (1) segmentation followed by skeletonization [2, 19], (2) point detection followed by edge prediction [15, 17], and (3) sequential breadth-first tracking [13, 14]. Segmentation methods require dense segmentation masks and often produce disconnected components or cycles. Point-and-edge detection methods similarly do not guarantee valid tree topology. Sequential tracking methods, on the other hand, suffer from low recall due to their reliance on highly imbalanced classification of bifurcation nodes.

To address these limitations, we propose RefTr, a Transformer [20] decoder based model with a Producer–Refiner architecture. Given a 3D patch, the Producer uses learned token sets to propose multiple fixed-length trajectories rooted at the patch center. The number of proposals is set to be larger than the number of true branches and a Many-to-One matching strategy assigns several proposals to each ground-truth branch. The Refiner then recurrently aligns these proposals to the target branches. Each pair of trajec-

tories remains confluent (flows together) until a divergence point, after which they separate as shown in Figure 1). The model predicts a divergence position for each pair of trajectory and an end position for each trajectory, and together they represent the final centerline tree graph.

Sequential methods classify each centerline node as intermediate, bifurcation, or end node at every step. This classification problem is highly imbalanced and even a single error can lead to missed or premature branches. RefTr avoids this issue by predicting bifurcations and terminations at the trajectory level, which produces more stable outputs. Reusing the same Refiner block across multiple refinement steps also reduces decoder parameters by $2.4\times$ compared to Trexplorer Super [14], a major advantage when working with large 3D volumes. Many-to-One matching further provides an implicit ensembling effect by allowing multiple trajectories to contribute to each target branch.

To suppress duplicates and improve precision, we introduce Tree Non-Maximum Suppression (TNMS), an efficient NMS algorithm for spatial tree graphs. Together with divergence prediction head, TNMS preserves topological consistency while removing redundant branches.

Finally, we extend Trexplorer Super’s evaluation framework by introducing radius-aware thresholds that scale with the ground-truth vessel radius. We report performance averaged across multiple thresholds, which accounts for anatomical variability and produces more reliable comparisons than fixed thresholds.

Our main contributions are:

- A parameter-efficient 3D image-to-graph model, RefTr, that uses recurrent refinement and trajectory-level predictions to achieve higher recall while maintaining state-of-the-art precision.
- Tree Non-Maximum Suppression (TNMS), an efficient algorithm for merging duplicate branches while preserving correct tree topology.
- Radius-aware evaluation thresholds for point- and branch-level metrics, enabling fair comparisons across anatomically diverse datasets.

2. Related Work

2.1. Model-based methods

Traditional model-based approaches extract hand-crafted features and apply explicit rules for tree generation [26, 27]. These methods are sensitive to parameter choices, and generalize poorly.

2.2. Segmentation and skeletonization

Segmentation-based methods [2, 19] first predict dense vessel masks and then skeletonize them. This requires costly voxel-level annotations and still does not guarantee correct topology, often producing fragmented or disconnected cen-

terlines. Skeletonization itself is non-trivial and typically involves complex post-processing. In contrast, image-to-graph models, including ours and prior baselines, require only centerline graphs for training and evaluation, which are far easier to annotate. The reliance on dense segmentation masks is the main reason these methods are not included in our comparison.

2.3. Two-stage transformer methods

Vesselformer [15] and Relationformer [17] detect centerline points and then infer edges using DETR-style transformers [1]. Although simpler than segmentation pipelines, these methods often produce disconnected graphs or cycles, and their quadratic edge prediction cost limits scalability for dense vascular trees.

2.4. Sequential generation methods

Sequential generation methods generate centerlines iteratively in a breadth-first manner. Reinforcement learning approaches [7, 24, 25] are difficult to train and limited by discretization of the 3D space. Trexplorer [13] and Trexplorer Super [14] introduce transformer-based sequential tracking with object queries. However, they suffer from low recall due to high imbalanced node-level classification of bifurcation and end nodes which represent less than 1% of training examples. The strictly sequential design also prevents correction of earlier errors.

In contrast, our proposed RefTr completely avoids node-level classification, directly predicts confluent trajectories, and refines them recurrently. By allocating multiple trajectories to each target, it creates an ensembling effect and predicts both divergence (bifurcation) and end positions at the trajectory level, resulting in higher performance, a simpler and more efficient architecture, and more reliable outputs.

3. Methodology

RefTr is a Transformer decoder-based model for generating centerline graphs of vascular trees from 3D medical images. It represents the centerline graph using a set of confluent trajectories, along with pairwise divergence positions and individual end positions, as illustrated in Figure 1. The model architecture consists of an image encoder, two customized Transformer decoder blocks, the Producer and the Refiner, and prediction heads that transform token set embeddings into output trajectories. The overall architecture is shown in Figure 2. In the following sections, we detail the formulation, architecture, and training procedure of our model.

3.1. Problem formulation

The objective is to infer a centerline tree graph $G = (V, E)$, representing a vascular tree in a 3D medical image such as a CT scan. Each node $v \in V$ is defined by a spatial location

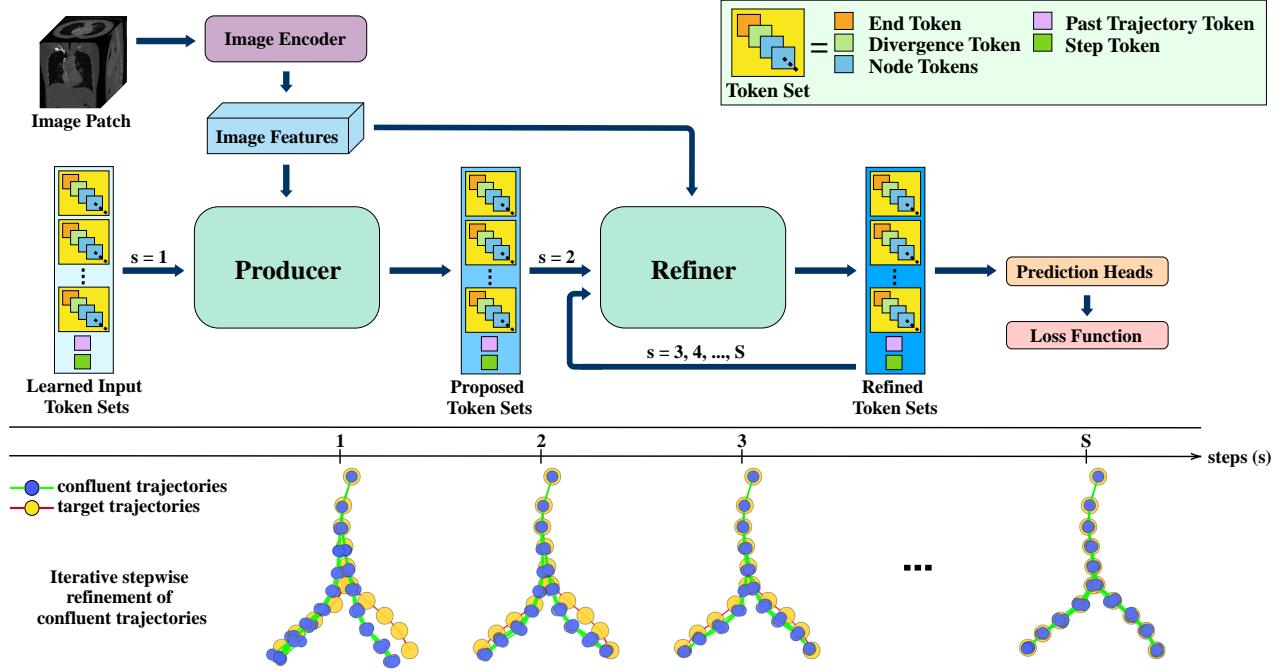


Figure 2. Overview of the RefTr architecture. The top illustrates the model, which includes an Image Encoder, the Producer, the Refiner, and prediction heads. The Producer uses learned token sets and image features to propose multiple trajectory embeddings, which are refined by the Refiner over multiple steps. Loss is computed at every step, including the Producer step (omitted for clarity). Both the Producer and Refiner also receive input from a memory bank containing refined token sets from previous image patches (also omitted for clarity). The bottom shows recurrent refinement of multiple confluent trajectories toward two target trajectories over S steps.

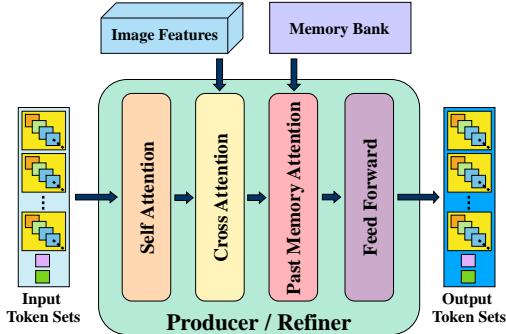


Figure 3. Detailed structure of the Producer and Refiner blocks.

and vessel radius: $\mathbf{v} = [\mathbf{x}, r]$, where $\mathbf{x} = [x, y, z]$. Edges $E \subseteq V \times V$ represent parent-child relationships, forming a rooted tree topology.

Given an input patch, RefTr predicts an intermediate representation of G as a set of confluent trajectories $\mathcal{T} = \{T_1, \dots, T_N\}$ as shown in Figure 1. Each trajectory T_i is a sequence of L ordered nodes, $T_i = \{\hat{\mathbf{v}}_i^{(1,S)}, \dots, \hat{\mathbf{v}}_i^{(L,S)}\}$, where each node $\hat{\mathbf{v}}_i^{(l,S)} = [\hat{\mathbf{x}}_i^{(l,S)}, \hat{r}_i^{(l,S)}]$ includes a predicted 3D position and radius at the final decoding step S . All trajectories originate from the center of the input patch.

Confluence is implicitly modeled by overlapping node

predictions across trajectories, where $\hat{\mathbf{D}}_{i,k}^{(S)}$ indicates the divergence position between trajectories i and k and $\hat{e}_i^{(S)}$ is the relative end position of trajectory i .

This confluent trajectory representation compactly encodes both geometry and topology of the vascular tree structure. A post-processing step converts it into the tree graph $G = (V, E)$ representation.

3.2. Model architecture

RefTr consists of four key components: an image encoder, Producer, Refiner, and prediction heads.

3.2.1. Image encoder

For a fair comparison with the previous SOTA model Tr-explorer Super [14], we use the same modified SwinUNETR [18] as our encoder. SwinUNETR is a hierarchical transformer-based UNet designed for 3D medical volumes. It uses shifted window attention to maintain computational efficiency while enabling long-range context propagation. It encodes input patches of size $64 \times 64 \times 64$ into a feature map of size $16 \times 16 \times 16 \times 512$.

3.2.2. Producer and Refiner

The Producer and Refiner use a customized Transformer decoder architecture. Each decoder block consists of a self-

attention layer, a cross-attention layer, a past-memory attention layer, and a feed-forward layer, as shown in Figure 3. The decoder input consists of learned tokens organized into n trajectory token sets, each containing L node tokens, one end token, and one divergence token. Node tokens predict position and radius, the end token predicts the end position, and paired divergence tokens predict the divergence positions.

The self-attention layer enables both intra- and inter-token set attention, allowing the model to effectively associate n proposed trajectories with m target branches. Cross-attention aggregates information from the 3D image features, while an additional past memory attention layer attends to token sets of the ancestor trajectories from the last five processed patches stored in the memory bank, improving long-range trajectory reasoning.

The Producer uses learned token sets, image features, and past memory to propose initial trajectories. The Refiner then recurrently refines these trajectories over $S - 1$ steps to align them with target branches. Reusing the same Refiner block across all steps reduces the decoder parameter count by $2.4\times$ compared to the previous SOTA Trexplorer Super, which uses six independent decoder blocks.

3.2.3. Prediction heads

At each decoding step, the model predicts structural attributes of the centerline trajectories as follows:

- Node positions $\hat{\mathbf{x}}_i^{(l,s)} \in \mathbb{R}^3$ and radii $\hat{r}_i^{(l,s)} \in \mathbb{R}_+$, for each node l along trajectory i at decoding step s .

At the final decoding step S , the model additionally predicts:

- End position $\hat{e}_i^{(S)} \in [0, 1]$, indicating the relative location along the sequence where trajectory i ends.
- Divergence position $\hat{\mathbf{D}}_{i,k}^{(S)} \in [0, 1] \forall i \neq k$, representing the divergence position between trajectory pairs (i, k) along the centerline.

We compute the divergence and end positions only after the last refinement step.

3.3. Inference and tree construction

At inference time, the model takes as input a 3D patch centered at the root and predicts a set of confluent trajectories \mathcal{T} . For each trajectory, it identifies the end position and predicts divergence position relative to other trajectories. These outputs are then converted into a centerline tree G by creating bifurcations at the divergence positions and leaf nodes at the end positions.

If any branch does not terminate within trajectory length L , a new patch is extracted centered at the last predicted node of that branch. This process is repeated recursively until all branches have been explored and the complete vascular tree structure is generated.

3.4. Tree Non-Max Suppression

Tree Non-Max Suppression (TNMS) is an efficient algorithm that merges overlapping branches in predicted trees to ensure topological consistency. While the model’s divergence head resolves many duplicates, TNMS handles any remaining overlaps.

Given a directed tree $G = (V, E)$ with 3D positions and radii at each node, TNMS performs a top-down (pre-order) traversal from the root. It maintains a list of visited nodes and uses a KD-tree for fast spatial queries. The spatial threshold for matching is adaptive, scaled by each node’s radius using a factor $\tau_{\text{pos}} = 0.3$, with a minimum threshold $\tau_{\text{min}} = 2.0$. Nodes within this threshold of visited nodes are flagged as duplicates. Hyperparameters were selected via simple grid search, and the radius-aware threshold allows TNMS to handle vascular trees of varying sizes.

Branches with a significant fraction of flagged nodes (above a threshold $\rho = 0.2$) are marked for merging. Duplicate nodes are grouped and replaced by a single node, and cycles are removed by keeping only the edge closest to the root. The procedure is summarized in Algorithm 1.

TNMS is fast and scalable. It requires a single traversal, uses sublinear KD-tree lookups, and compares each node only to earlier nodes, yielding $\mathcal{O}(N \log N)$ complexity for N nodes. Merging and cycle resolution are lightweight, allowing TNMS to process graphs with thousands of nodes in under a second on CPU.

3.5. Matching and loss function

3.5.1. Matching cost

Given n predicted trajectories $\hat{\mathcal{T}} = \{\hat{T}_i\}_{i=1}^n$ and m target paths $\mathcal{T} = \{T_j\}_{j=1}^m$, we compute a matching from predictions to targets based on a cost matrix $\mathbf{C} \in \mathbb{R}^{n \times m}$. Each entry $\mathbf{C}_{i,j}$ reflects the cost of matching predicted trajectory \hat{T}_i to target trajectory T_j , and is defined as a weighted sum of position and radius cost:

$$\mathbf{C}_{i,j} = \lambda_{\text{pos}} \cdot \mathcal{C}_{\text{pos}}(\hat{T}_i, T_j) + \lambda_{\text{rad}} \cdot \mathcal{C}_{\text{rad}}(\hat{T}_i, T_j), \quad (1)$$

where:

$$\mathcal{C}_{\text{pos}}(\hat{T}_i, T_j) = \frac{1}{L} \sum_{l=1}^L \left\| \hat{\mathbf{x}}_i^{(l)} - \mathbf{x}_j^{(l)} \right\|_1, \quad (2)$$

$$\mathcal{C}_{\text{rad}}(\hat{T}_i, T_j) = \frac{1}{L} \sum_{l=1}^L \left| \hat{r}_i^{(l)} - r_j^{(l)} \right|, \quad (3)$$

and λ are the weight coefficients to balance the contributions of each cost component.

To ensure consistent supervision, the matching is computed only once using the predictions from the first decoding step. This fixed assignment is reused for all remaining $S - 1$ steps during loss computation.

Algorithm 1 Tree Non-Max Suppression

Require: Graph $G = (V, E)$, radius scaling threshold τ_{pos} , minimum distance threshold τ_{\min} , duplicate ratio ρ

Ensure: Deduplicated graph G

- 1: Initialize visited nodes $\mathcal{V} \leftarrow []$, positions $\mathcal{P} \leftarrow []$
- 2: Initialize branch nodes $\mathcal{B}_v \leftarrow []$, positions $\mathcal{B}_p \leftarrow []$
- 3: Set root node v_{root}
- 4: **for** each node v in pre-order from v_{root} **do**
- 5: **if** v starts a new branch **then**
- 6: Append $\mathcal{B}_v, \mathcal{B}_p$ to \mathcal{V}, \mathcal{P} ; rebuild KD-tree
- 7: Reset $\mathcal{B}_v \leftarrow []$, $\mathcal{B}_p \leftarrow []$
- 8: **end if**
- 9: Append v and its position to $\mathcal{B}_v, \mathcal{B}_p$
- 10: Query KD-tree for nearest neighbor u within $\tau = \max(\tau_{\min}, \tau_{\text{pos}} \cdot r_v)$
- 11: **if** match found **then**
- 12: Mark v as duplicate of u
- 13: **end if**
- 14: **end for**
- 15: Identify branches with duplicate ratio $\geq \rho$
- 16: Group duplicate pairs into sets and merge each set into a single node
- 17: **for** each node v with in-degree > 1 **do**
- 18: Keep only the edge from the parent closest to the root
- 19: **end for**
- 20: **return** G

3.5.2. Many-to-one matching

We choose $n > m$, and enable many-to-one matching by replicating the target set to form a square cost matrix $\tilde{\mathbf{C}} \in \mathbb{R}^{n \times n}$. This allows us to use the Hungarian algorithm to perform bipartite matching between the n predicted and n (replicated) target trajectories:

$$\sigma^* = \arg \min_{\sigma \in \mathfrak{S}_n} \sum_{i=1}^n \tilde{\mathbf{C}}_{\sigma(i), i}, \quad (4)$$

where \mathfrak{S}_n denotes the set of all permutations over n elements, and σ^* is the permutation that minimizes the total cost.

Since the target trajectories are replicated, we recover the index of the original (non-replicated) target trajectory via modulo indexing:

$$\hat{j}_i = \sigma^*(i) \bmod m. \quad (5)$$

where \hat{j}_i denotes the assigned target trajectory index for the i -th predicted trajectory.

3.5.3. Loss components

Let \hat{j}_i denote the target trajectory assigned to predicted trajectory i . For each decoder step $s = 1, \dots, S$, we define the

position and radius losses as:

$$\mathcal{L}_{\text{pos}}^{(s)} = \frac{1}{nL} \sum_{i=1}^n \sum_{l=1}^L \left\| \hat{\mathbf{x}}_i^{(l,s)} - \mathbf{x}_{\hat{j}_i}^{(l)} \right\|_1, \quad (6)$$

$$\mathcal{L}_{\text{rad}}^{(s)} = \frac{1}{nL} \sum_{i=1}^n \sum_{l=1}^L \left| \hat{r}_i^{(l,s)} - r_{\hat{j}_i}^{(l)} \right|. \quad (7)$$

The end loss and divergence loss are computed only at the final decoding step $s = S$, where the predicted trajectories are most refined. The end loss is defined as

$$\mathcal{L}_{\text{end}} = \frac{1}{n} \sum_{i=1}^n \left| \hat{e}_i^{(S)} - e_{\hat{j}_i} \right|. \quad (8)$$

To compute the divergence loss, we reorder the target divergence matrix to align with the assigned targets:

$$\mathbf{D}'_{i,k} = \mathbf{D}_{j_i, \hat{j}_k}, \quad \forall i \neq k. \quad (9)$$

The divergence loss is then:

$$\mathcal{L}_{\text{div}} = \frac{1}{n(n-1)} \sum_{\substack{i,k=1 \\ i \neq k}}^n \left| \hat{\mathbf{D}}_{i,k}^{(S)} - \mathbf{D}'_{i,k} \right|. \quad (10)$$

3.5.4. Total loss

The overall loss aggregates the position and radius losses across all steps, and adds the end and divergence losses from the final step:

$$\mathcal{L} = \sum_{s=1}^S \left(\alpha_{\text{pos}} \mathcal{L}_{\text{pos}}^{(s)} + \alpha_{\text{rad}} \mathcal{L}_{\text{rad}}^{(s)} \right) + \alpha_{\text{end}} \mathcal{L}_{\text{end}} + \alpha_{\text{div}} \mathcal{L}_{\text{div}}, \quad (11)$$

where α_{\cdot} are scalar weights used to balance the contributions of each loss component.

3.6. Training strategy

We adopt the training strategy proposed in Trexplorer Super [14] to efficiently train on long trajectory sequences and enhance contextual reasoning from past information. Each training sample consists of a long trajectory with 55 nodes, which is divided into a sequence of six overlapping trajectories of 10 nodes each (using a sliding window of size 10 and stride 9). Each trajectory is paired with a 3D volume patch centered at its starting node. The model is trained sequentially over these trajectories. For each trajectory, it receives:

- the corresponding 3D image patch,
- a memory bank containing up to five token sets, each representing an ancestor trajectory that encodes the direct parent trajectory of the current starting point within previous image patches,

- a past trajectory token, a linear projection of up to S previous node positions and radii, and
- a step token, indicating the current refinement step.

Using these inputs, the model predicts the confluent trajectories over S refinement steps, along with divergence and endpoint positions for the final step. The total training loss is accumulated as defined in Equation 11.

For the first trajectory of the sequence, the memory bank is empty, and past context is provided by the past trajectory token. As training progresses, the model updates the memory bank with embeddings from previously predicted ancestor trajectories. To reduce memory consumption, gradient flow is stopped between consecutive trajectories. This sequential training process is repeated for all six trajectories in each sample.

4. Experiments and results

4.1. Datasets

We evaluate our method using the benchmark centerline graph datasets introduced in Trexplorer Super [14]. These include centerline graph annotations for: (1) a synthetic dataset, (2) the Airway Tree Modelling Challenge 2022 (ATM’22) dataset [16, 21–23, 28], and (3) the Pulmonary Artery Segmentation Challenge 2022 (Parse 2022) dataset [9]. These datasets provide a diverse evaluation setting across synthetic, airway, and vascular anatomies. All datasets are publicly available on Zenodo [12].

4.2. Evaluation metrics

For consistency with past work [14], we report both node-level and branch-level metrics. However, accurate centerline evaluation requires metrics that account for anatomical variability and localization uncertainty. The fixed 1.5-voxel threshold used by Trexplorer Super to match predicted and ground-truth points can disproportionately penalize predictions near large vessels and may encourage overfitting. To address this, we propose radius-aware thresholds τ^{rad} scaled by the ground-truth vessel radius (r), resulting in final threshold of $\max(1.5, \tau^{\text{rad}} \cdot r)$. A predicted node is a true positive if it falls within this threshold of an unmatched target node. The fixed lower bound of 1.5 voxels is chosen to match the minimum vessel radius of 1 with a small buffer of 0.5, to avoid false positives on thin vessels.

We evaluate node-level metrics, precision, recall and F1, using $\tau^{\text{rad}} \in [0.25, 0.75]$ with 0.05 increments and report their average indicated by radius-aware Average Precision (rAP), Average Recall (rAR), and Average F1-score (rF1) @ $\tau^{\text{rad}} = [0.25:0.05:0.75]$. This strategy, inspired by object detection metrics like mean Average Precision [8], reduces sensitivity to threshold choice, improves robustness across vessel sizes, and provides a fairer and clinically meaningful performance comparison.

Table 1. Training hyperparameters

Hyperparameter	Value
Training Iterations	1.92M
Sequence length L	10
Refining steps S	8
Predicted branches n	20
Position loss weight α_{pos}	4.2
Position cost weight λ_{pos}	3
Radius loss weight α_{rad}	1.15
Radius cost weight λ_{rad}	1
End loss weight α_{end}	0.94
Divergence loss weight α_{div}	0.3

For branch-level metrics, we fix $\tau^{\text{rad}} = 0.5$ to reduce computational cost. According to Trexplorer Super, a predicted branch is considered a true positive if it overlaps with at least 80% of an unmatched target branch’s nodes within the matching threshold. However, this criterion penalizes cases where multiple predictions collectively cover a single target branch, overcounting false positives and underestimating performance. To address this, we redefine the criterion: a predicted branch is counted as a true positive if it overlaps with at least τ^{match} fraction of nodes from any unmatched target centerline segment within the threshold. This relaxation allows partial matches to be recognized while maintaining meaningful correspondence. Because a fixed $\tau^{\text{match}} = 0.8$ does not reflect performance across varying match levels, we evaluate over a range of thresholds $\tau^{\text{match}} \in [0.5, 0.9]$ with increments of 0.05, and report the averaged branch-level metrics: rBAP, rBAR, and rBF1@ $\tau^{\text{match}} = [0.5:0.05:0.9]$.

We omit graph-level metrics because RefTr, like Trexplorer and Trexplorer Super, always produces a valid tree structure, meaning a single connected component without cycles. In contrast, Vesselformer does not guarantee a tree topology, as noted in the Trexplorer Super evaluation.

4.3. Experiments

All experiments were conducted on a single compute node equipped with four NVIDIA A100 GPUs, utilizing mixed-precision training to improve efficiency. The key hyperparameters used for training are provided in Table 1.

For fair comparison with Trexplorer and Trexplorer Super [13, 14], we computed the updated evaluation metrics on the test results provided by the authors. For Vesselformer [15], the best result was obtained for hyperparameters published by the authors with 80 object tokens and 12 million training iterations.

4.4. Results

Table 2 presents point- and branch-level metrics for Vesselformer, Trexplorer, Trexplorer Super, and our method RefTr on three benchmark datasets: Synthetic, ATM’22,

Table 2. Comparison of different models using point-level and branch-level metrics on the Synthetic, ATM’22, and Parse 2022 datasets.

Model	Point-level@ $\tau^{\text{rad}} = [0.25:0.05:0.75]$				Branch-level@ $\tau^{\text{match}} = [0.5:0.05:0.9]$		
	rAP(%) \uparrow	rAR(%) \uparrow	rF1(%) \uparrow	Radius(MAE) \downarrow	rBAP(%) \uparrow	rBAR(%) \uparrow	rBF1(%) \uparrow
Synthetic Dataset							
Vesselformer	48.53 \pm 10.35	70.05 \pm 2.76	53.15 \pm 8.19	0.43 \pm 0.011	15.84 \pm 1.54	36.25 \pm 4.32	20.54 \pm 1.74
Trexplorer	31.10 \pm 9.52	78.64 \pm 4.10	39.64 \pm 8.68	0.23 \pm 0.032	29.16 \pm 8.52	72.68 \pm 4.25	37.37 \pm 7.66
Trexplorer Super	92.10 \pm 3.29	70.58 \pm 3.03	77.98 \pm 1.90	0.10 \pm 0.006	90.05 \pm 3.45	89.51 \pm 0.78	88.94 \pm 0.0213
RefTr (ours)	93.35 \pm 1.63	79.38 \pm 1.19	84.64 \pm 1.52	0.53 \pm 0.124	88.95 \pm 2.98	84.69 \pm 1.50	86.34 \pm 2.17
ATM’22 Dataset							
Vesselformer	24.85 \pm 1.47	38.48 \pm 1.00	29.84 \pm 1.37	0.79 \pm 0.020	1.86 \pm 0.25	5.15 \pm 0.57	2.71 \pm 0.34
Trexplorer	4.54 \pm 1.44	5.90 \pm 0.70	4.49 \pm 0.29	0.98 \pm 0.086	3.42 \pm 1.59	1.63 \pm 0.43	1.83 \pm 0.28
Trexplorer Super	71.32 \pm 1.22	60.84 \pm 3.38	61.45 \pm 1.23	0.40 \pm 0.021	57.96 \pm 1.35	54.38 \pm 3.33	51.99 \pm 1.21
RefTr (ours)	73.34 \pm 1.38	66.37 \pm 0.75	66.78 \pm 0.56	0.64 \pm 0.047	54.24 \pm 1.45	53.40 \pm 0.35	50.88 \pm 0.82
Parse 2022 Dataset							
Vesselformer	21.93 \pm 2.06	18.21 \pm 1.00	19.54 \pm 2.69	1.11 \pm 0.027	3.45 \pm 0.18	3.09 \pm 0.32	3.12 \pm 0.21
Trexplorer	11.37 \pm 3.73	13.42 \pm 7.80	11.36 \pm 5.17	1.22 \pm 0.304	8.96 \pm 3.17	32.26 \pm 5.24	13.54 \pm 4.47
Trexplorer Super	53.65 \pm 5.91	35.91 \pm 3.22	40.01 \pm 3.86	0.58 \pm 0.025	41.97 \pm 5.63	56.68 \pm 3.69	46.57 \pm 4.70
RefTr (ours)	53.03 \pm 0.71	36.69 \pm 0.28	42.15 \pm 0.17	0.73 \pm 0.032	32.80 \pm 0.49	30.91 \pm 0.72	31.48 \pm 0.10

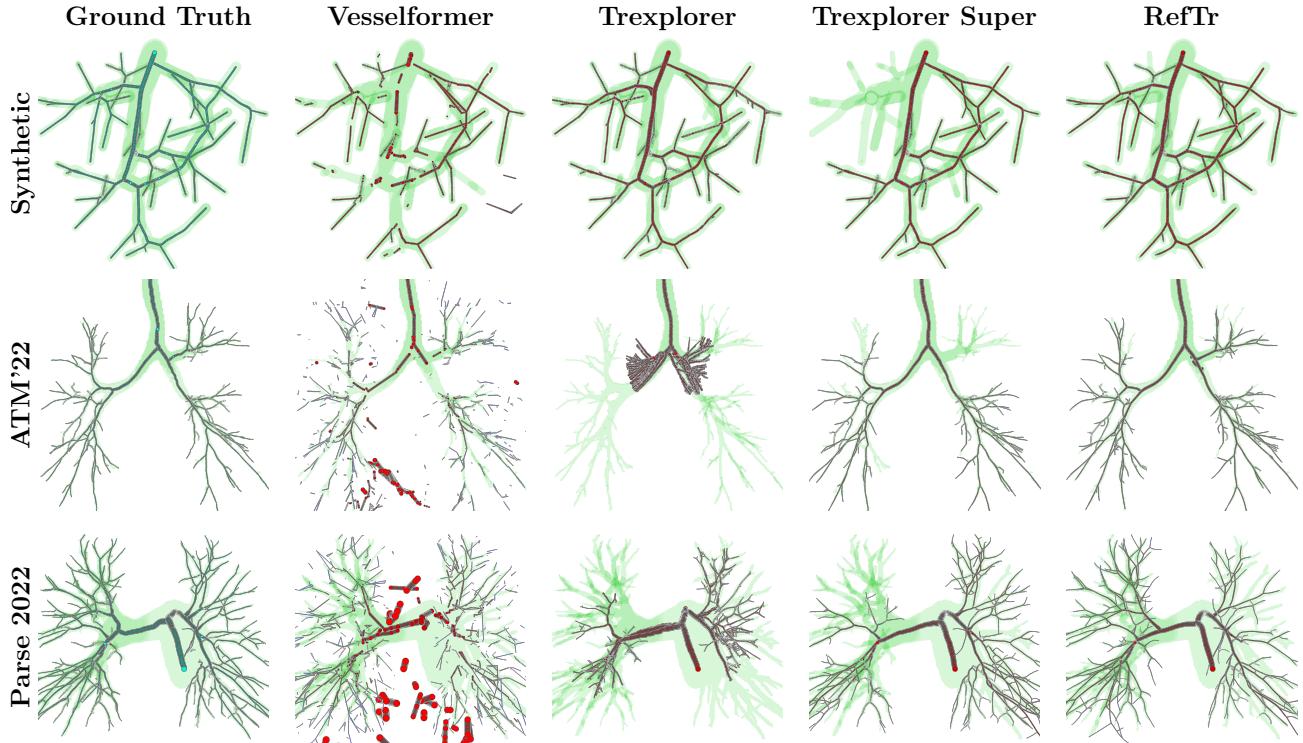


Figure 4. Visual comparison of ground truth and predictions from Vesselformer, Trexplorer, Trexplorer Super, and RefTr on one sample from each dataset. Centerline marker sizes are scaled according to the radius at each node. The vessel mask (green) is shown only for visualization and is not used during training or evaluation.

and Parse 2022. We report mean and standard deviation over five runs. Qualitative examples are shown in Figure 4. Table 3 lists parameters and per-patch runtimes.

4.4.1. Point-level metrics

RefTr achieves the highest recall across all datasets while maintaining comparable precision, resulting in the best

Table 3. Number of parameters and runtime for compared models.
Note: The effective runtime for Vesselformer is much higher due to a very time-consuming (multiple hours) post-processing step that glues graphs of adjacent patches.

Model	Number of Parameters (M)			Runtime per patch (ms)
	Encoder	Decoder	Total	
Vesselformer	82.83	13.59	98.73	23*
Trexplorer	23.31	25.22	55.46	222
Trexplorer Super	23.12	25.22	55.27	189
RefTr (ours)	23.12	10.50	42.32	156

Table 4. Ablation study evaluating the impact of the key components Recurrent Refinement (RR), Many-to-One Matching (MOM), and Tree Non-Max Suppression (TNMS) in the RefTr model on the ATM’22 dataset.

Key Components		Point-level@ $\tau^{\text{rad}} = [0.25:0.05:0.75]$		
RR MOM TNMS		rAP(%) \uparrow	rAR(%) \uparrow	rF1(%) \uparrow
\times	\checkmark	\checkmark	52.89 ± 2.79	6.02 ± 0.71
\checkmark	\times	\checkmark	66.70 ± 1.27	63.25 ± 3.49
\checkmark	\checkmark	\times	23.44 ± 2.42	74.00 ± 1.12
\checkmark	\checkmark	\checkmark	73.21 ± 1.57	25.22 ± 3.87
				66.37 ± 0.88
				66.81 ± 0.46

overall F1-scores. The slightly lower precision on Parse 2022 stems from RefTr’s recall-focused behavior. It continues tracking faint or thin vessels that are often missing from the ground truth, increasing false positives. This was verified through a radiologist review. Qualitative examples of this behavior are shown in the supplementary material. RefTr has a higher radius MAE than Trexplorer Super, but it remains well below one voxel, making it acceptable.

4.4.2. Branch-level metrics

RefTr shows lower branch-level scores than Trexplorer Super, due to its recall bias. The model tends to bifurcate earlier than the annotated centerlines, which often branch later, to not miss any new branches. While this leads to structural differences, both topologies are anatomically valid, especially in thicker vessels where bifurcation locations are ambiguous. However, this results in lower branch-level metrics at higher thresholds. Supplementary material provides qualitative examples of this behavior.

4.5. Ablations

We perform an ablation study to assess the contribution of three key components of our model: Recurrent Refinement (RR), Many-to-One Matching (MOM), and Tree Non-Max Suppression (TNMS). Table 4 reports the mean and standard deviation over three runs for each configuration.

Removing RR results in a substantial drop in recall. The Refiner plays a critical role in aligning predicted trajectories to newly emerging branches that the Producer may miss,

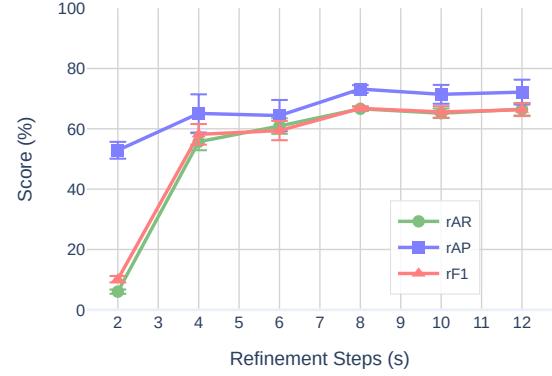


Figure 5. Point-level performance metrics versus the number of refinement steps for ATM’22 Dataset.

thereby improving recall and overall performance. Figure 5 shows performance with increasing refinement steps.

To evaluate the role of MOM, we build an alternative model that replaces MOM with standard one-to-one bipartite matching. This variant classifies each token set as either vessel or background and trains only vessel-labeled sets. Although it performs reasonably well, it exhibits lower recall and precision than the full model. The benefit of MOM arises from an implicit ensembling effect: multiple token sets are assigned to a single ground-truth branch, allowing several predictions to contribute to the final representation, which are later merged using divergence positions.

Finally, TNMS is essential for merging duplicate predictions, yielding a clear improvement in precision. This merging can introduce a slight reduction in recall because the fused branch may deviate more from the ground truth than the individual unmerged predictions. This effect is more pronounced in larger vessels where the centerline location is inherently ambiguous.

5. Conclusion

We introduce RefTr, a method for robust and efficient generation of topologically correct vascular centerline graphs from 3D medical images. RefTr utilizes a Producer-Refiner architecture that proposes and recurrently refines confluent trajectories to match the target vascular tree. A many-to-one matching strategy enables implicit ensembling, while a novel and efficient Tree-NMS algorithm merges duplicate branches to improve precision. We also propose radius-aware evaluation thresholds that account for anatomical variability and produce more reliable metric comparisons. Experiments across multiple datasets show that RefTr achieves state-of-the performance on all datasets. Performance on the Parse 2022 dataset may improve further with a stronger pretrained image encoder, given the high anatomical complexity of pulmonary arteries and the limited number of training samples.

6. Acknowledgements

Compute and storage resources were provided by NAISS (grant 2022-06725, Swedish Research Council) and the Berzelius system at the National Supercomputer Centre, funded by the Knut and Alice Wallenberg Foundation.

References

- [1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European conference on computer vision*, pages 213–229. Springer, 2020. [2](#)
- [2] Li Chen, Wenjin Liu, Niranjan Balu, Mahmud Mossa-Basha, Thomas S Hatsuhami, Jenq-Neng Hwang, and Chun Yuan. 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*, pages 579–589. Springer, 2021. [1, 2](#)
- [3] Andrew D Choi, Hugo Marques, Vishak Kumar, William F Griffin, Habib Rahban, Ronald P Karlsberg, Robert K Zeman, Richard J Katz, and James P Earls. 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. [1](#)
- [4] Dongjin Huang, Wen Tang, Youdong Ding, Taoruan Wan, and Yimin Chen. 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*, pages 443–449. IEEE, 2011. [1](#)
- [5] Zohaib Khan, Jennifer P Ngo, Bianca Le, Roger G Evans, James T Pearson, Bruce S Gardiner, and David W Smith. Three-dimensional morphometric analysis of the renal vasculature. *American Journal of Physiology-Renal Physiology*, 314(5):F715–F725, 2018. [1](#)
- [6] Hao Li, Zeyu Tang, Yang Nan, and Guang Yang. Human treelike tubular structure segmentation: A comprehensive review and future perspectives. *Computers in biology and medicine*, 151:106241, 2022. [1](#)
- [7] Zhuowei Li, Qing Xia, Zhiqiang Hu, Wenji Wang, Lijian Xu, and Shaoting Zhang. 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*, pages 418–428. Springer, 2021. [2](#)
- [8] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755. Springer, 2014. [6](#)
- [9] Gongning Luo, Kuanquan Wang, Jun Liu, Shuo Li, Xinjie Liang, Xiangyu Li, Shaowei Gan, Wei Wang, Suyu Dong, Wenyi Wang, et al. Efficient automatic segmentation for multi-level pulmonary arteries: The parse challenge. *arXiv preprint arXiv:2304.03708*, 2023. [6](#)
- [10] Olivia Miraucourt, Stéphanie Salmon, Marcela Szopos, and Marc Thiriet. Blood flow in the cerebral venous system: modeling and simulation. *Computer methods in biomechanics and biomedical engineering*, 20(5):471–482, 2017. [1](#)
- [11] Sara Moccia, Elena De Momi, Sara El Hadji, and Leonardo S Mattos. Blood vessel segmentation algorithms—review of methods, datasets and evaluation metrics. *Computer methods and programs in biomedicine*, 158:71–91, 2018. [1](#)
- [12] R. Naeem. Trexplorer super datasets (1.0). <https://doi.org/10.5281/zenodo.15888958>, 2025. [6](#)
- [13] Roman Naeem, David Hagerman, Lennart Svensson, and Fredrik Kahl. Trexplorer: Recurrent detr for topologically correct tree centerline tracking. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 744–754. Springer, 2024. [1, 2, 6](#)
- [14] Roman Naeem, David Hagerman, Jennifer Alvén, Lennart Svensson, and Fredrik Kahl. Trexplorer super: Topologically correct centerline tree tracking of tubular objects in ct volumes. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 595–605. Springer, 2025. [1, 2, 3, 5, 6](#)
- [15] Chinmay Prabhakar, Suprosanna Shit, Johannes C Paetzold, Ivan Ezhov, Rajat Koner, Hongwei Li, Florian Sebastian Kofler, and Bjoern Menze. Vesselformer: Towards complete 3d vessel graph generation from images. In *Medical Imaging with Deep Learning*, pages 320–331. PMLR, 2024. [1, 2, 6](#)
- [16] Yulei Qin, Mingjian Chen, Hao Zheng, Yun Gu, Mali Shen, Jie Yang, Xiaolin Huang, Yue-Min Zhu, and Guang-Zhong Yang. Airwaynet: a voxel-connectivity aware approach for accurate airway segmentation using convolutional neural networks. In *International Conference on Medical Image Computing and Computer-Assisted Intervention*, pages 212–220. Springer, 2019. [6](#)
- [17] Suprosanna Shit, Rajat Koner, Bastian Wittmann, Johannes Paetzold, Ivan Ezhov, Hongwei Li, Jiazhen Pan, Sahand Sharifzadeh, Georgios Kaassis, Volker Tresp, et al. Relationformer: A unified framework for image-to-graph generation. In *European Conference on Computer Vision*, pages 422–439. Springer, 2022. [1, 2](#)
- [18] Yucheng Tang, Dong Yang, Wenqi Li, Holger R Roth, Bennett Landman, Daguang Xu, Vishwesh Nath, and Ali Hatamizadeh. 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*, pages 20730–20740, 2022. [3](#)
- [19] Giles Tetteh, Velizar Efremov, Nils D Forkert, Matthias Schneider, Jan Kirschke, Bruno Weber, Claus Zimmer, Marie Piraud, and Björn H Menze. Deepvesselnet: Vessel segmentation, centerline prediction, and bifurcation detection in 3-d angiographic volumes. *Frontiers in Neuroscience*, 14:1285, 2020. [1, 2](#)
- [20] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. [1](#)

- [21] Weihao Yu, Hao Zheng, Minghui Zhang, Hanxiao Zhang, Jiayuan Sun, and Jie Yang. Break: Bronchi reconstruction by geodesic transformation and skeleton embedding. In *2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI)*, pages 1–5. IEEE, 2022. [6](#)
- [22] Minghui Zhang, Hanxiao Zhang, Guang-Zhong Yang, and Yun Gu. Cfda: Collaborative feature disentanglement and augmentation for pulmonary airway tree modeling of covid-19 cts. In *Medical Image Computing and Computer Assisted Intervention–MICCAI 2022: 25th International Conference, Singapore, September 18–22, 2022, Proceedings, Part I*, pages 506–516. Springer, 2022.
- [23] Minghui Zhang, Yangqian Wu, Hanxiao Zhang, Yulei Qin, Hao Zheng, Wen Tang, Corey Arnold, Chenhao Pei, Pengxin Yu, Yang Nan, et al. Multi-site, multi-domain airway tree modeling. *Medical Image Analysis*, 90:102957, 2023. [6](#)
- [24] Pengyue Zhang, Fusheng Wang, and Yefeng Zheng. 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*, pages 755–763. Springer, 2018. [2](#)
- [25] Yuyang Zhang, Gongning Luo, Wei Wang, and Kuanquan Wang. 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*, pages 35–44. Springer, 2020. [2](#)
- [26] Zhongwen Zhang, Dmitrii Marin, Egor Chesakov, Marc Moreno Maza, Maria Drangova, and Yuri Boykov. Divergence prior and vessel-tree reconstruction. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10216–10224, 2019. [2](#)
- [27] Zhongwen Zhang, Dmitrii Marin, Maria Drangova, and Yuri Boykov. Confluent vessel trees with accurate bifurcations. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9573–9582, 2021. [2](#)
- [28] Hao Zheng, Yulei Qin, Yun Gu, Fangfang Xie, Jie Yang, Jiayuan Sun, and Guang-Zhong Yang. Alleviating class-wise gradient imbalance for pulmonary airway segmentation. *IEEE Transactions on Medical Imaging*, 40(9):2452–2462, 2021. [6](#)

RefTr: Recurrent Refinement of Confluent Trajectories for 3D Vascular Tree Centerline Graphs

Supplementary Material

7. Qualitative examples

In this section, we present qualitative examples from the PARSE 2022 dataset for various cases. While these observations also apply to the ATM’22 dataset to some extent, we focus on PARSE 2022 because the patterns are clearer and easier to analyze in detail.

7.1. Missing ground truth annotations

We present qualitative examples of branches missing from the ground truth annotations. As shown in examples in Figure 6, a bifurcation occurs along the selected vessel in the CT slice sequence and a new branch emerges, but the ground truth does not follow this branch, whereas RefTr does. Because these correctly tracked vessels are counted as false positives in both point-level metrics (rAP, rF1) and branch-level metrics (rBAP, rBF1), they reduce the reported precision and overall F1 score.

Figure 7 shows examples of early termination in the ground truth annotations. In each case, the vessel is initially tracked by both the ground truth and RefTr, but the ground truth annotation stops after a few slices, even though the CT data clearly shows the vessel continuing. RefTr correctly continues to follow the vessel. However, these valid RefTr predictions beyond the prematurely terminated ground truth path are also treated as false positives in both point-level and branch-level metrics, further lowering the reported precision and F1 score.

These instances of missing ground truth annotations have been confirmed by an expert radiologist. The authors of PARSE 2022 do not specify any criteria for excluding certain vessels from annotation. The selection may be based on criteria of clinical importance defined by the expert annotating clinicians. While many datasets omit vessels below a certain diameter, this does not appear to be the case for PARSE 2022, since the authors emphasize that thinner pulmonary arteries are both more challenging to detect and more clinically important. Depending on the task, the predicted centerline graphs can be readily post-processed to filter branches using length or radius thresholds.

7.2. Earlier predicted bifurcations

In Figure 8, we show several examples of vessel bifurcations in CT image patches. RefTr often bifurcates earlier than the ground truth annotation, likely to avoid missing a branch that could be overlooked if divergence is predicted too late. Both the predicted and ground truth centerlines remain spatially and topologically valid, since there is in-

herent ambiguity in the exact positions of centerline nodes and bifurcations. This ambiguity increases as vessel size increases.

The presence of this ambiguity can reduce branch-level metrics, as correctly predicted branches may be counted as false positives and ground truth branches as false negatives, lowering both precision and recall and resulting in a lower overall F1 score, especially for higher branch matching thresholds ($\tau^{\text{match}} > 0.8$). We report branch-level metrics for $\tau^{\text{match}} = [0.5 : 0.05 : 0.9]$. By reducing the upper threshold, the metrics can be adjusted to avoid penalizing such cases. The choice of threshold may vary for different datasets, depending on whether strict adherence to the ground truth centerlines is required.

7.3. Failure cases

In Figure 9, we show cases where another vessel in the CT image comes very close to the currently tracked vessel, either touching or nearly touching, before diverging. At the divergence point, RefTr interprets the nearby vessel as a new branch originating from the tracked vessel, predicts a bifurcation, and begins tracking it. This newly predicted branch is counted as a false positive, reducing both point-level and branch-level precision.

Figure 10 shows additional failure cases where RefTr fails to predict a bifurcation and misses a new branch. In these examples, the bifurcation is often ambiguous. In the first and second examples, the bifurcating vessels remain partially connected across several slices before fully separating. In such cases, RefTr treats the diverging vessel as an anomaly and does not predict it as a new branch, likely due to limited examples of such cases in the training dataset. In the third example, RefTr misses a relatively thin new branch compared to the original tracked vessel. This is uncommon, as vessels typically bifurcate into child branches with radii that are more similar and smaller than the parent branch. These cases result in lower point-level and branch-level recall.

These types of failures could be mitigated by increasing the diversity and amount of training data, applying hard-mining strategies for difficult vessels, or using data augmentation techniques. Exploring these approaches could improve performance in future work.

Figure 11 illustrates another failure mode, where RefTr prematurely terminates a currently tracked branch. In some cases, RefTr is uncertain whether a branch continues past an ambiguous region or represents a different structure, as

in the first example. In others, RefTr appears to stop based on an internal estimate of where the ground-truth branch should end. In the second example, the ground-truth lower-leg branch ends at a later bifurcation, but RefTr terminates at an earlier bifurcation. These cases also result in lower recall. Failures of the first type could be addressed using the techniques mentioned for previous failure cases, while the second type may benefit from explicitly enforcing specific branch exclusion criteria.

8. Building a tree from confluent trajectories

RefTr predicts a set of confluent trajectories, along with an end position for each trajectory and a divergence position for each pair of trajectories. The predicted positions are continuous values between 0 and 1. These values are first unnormalized to match the length of the trajectories and then discretized by rounding to the nearest integer, which gives the ID of the trajectory node where the end or divergence occurs.

To build a tree from the confluent trajectories, we follow these steps:

1. Cluster trajectories into groups based on their divergence positions, so that trajectories that do not diverge (i.e., divergence occurs at the end of the trajectory) are grouped together.
2. Merge the trajectories within each cluster into a representative trajectory by averaging positions, radii, and end positions.
3. Compute pairwise divergence positions between the representative trajectories using the average divergence positions.
4. Starting from the shared root of the confluent trajectories, construct the tree level by level using a breadth-first approach.
5. At each level, cluster the nodes of all representative branches at that level based on divergence positions. If multiple clusters are present, new branches are created for each cluster. Clusters may further subdivide at subsequent levels, creating additional branches.
6. At each level, terminate branches that have reached their end positions.

Building the tree in this way ensures a single connected component with no cycles, preserving topological correctness.

9. Experimental Setup

We provide additional model hyperparameters in Table 5.

Table 5. Additional training hyperparameters

Hyperparameter	Value
3D patch size	$64 \times 64 \times 64$
Producer blocks	1
Refiner blocks	1
Decoder heads	16
Token dimension	512
Decoder FF dim	2048
Position head dim	[512,512,512]
Radius head dim	[512,512,512]
Divergence head dim	[1024,512,256,128]
End head dim	[512,256,128,64]
Encoder feat. dim	24
Encoder depths	[2,2,2,2]
Encoder heads	[3,6,12,24]
Encoder patch size	2
Encoder window size	7

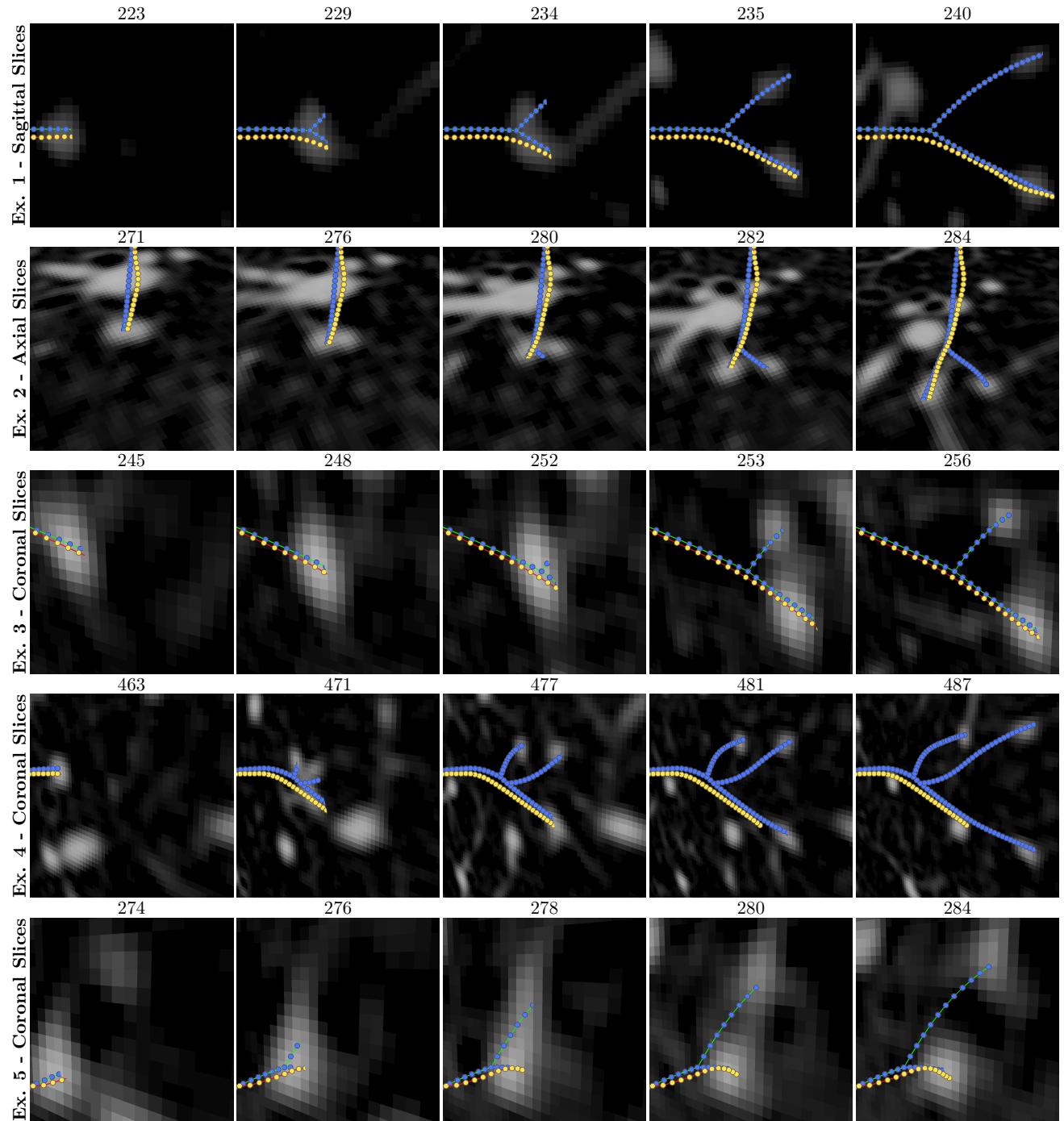


Figure 6. Qualitative examples of vessel bifurcations missing in the ground truth (yellow) but correctly predicted by RefTr (blue). Each row shows one example across a sequence of CT image slices. Only vessels branching from the selected vessel in the first slice are shown.

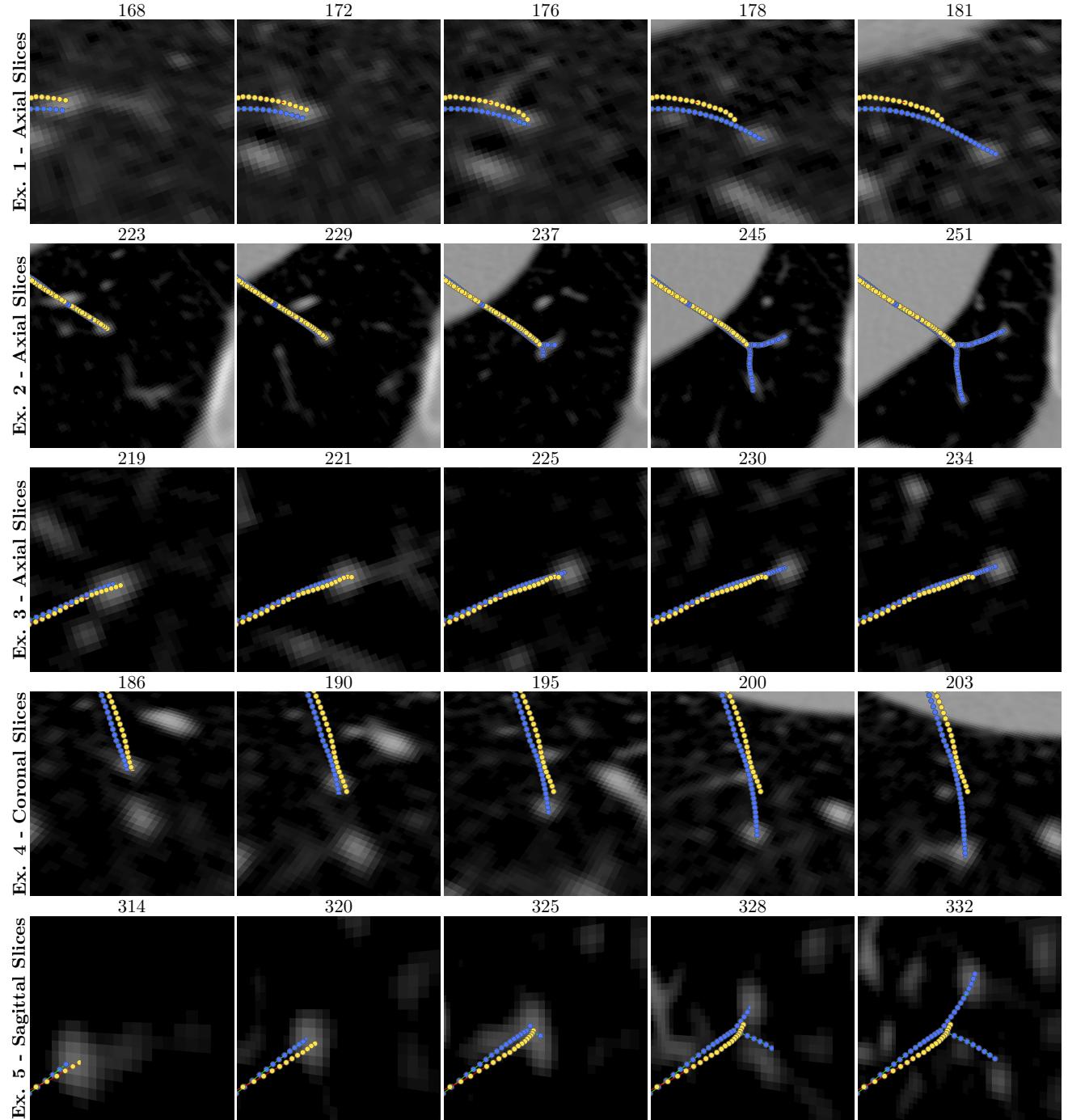


Figure 7. Qualitative examples of vessels terminated early in the ground truth (yellow) but correctly predicted by RefTr (blue). Each row shows one example across a sequence of CT image slices. Only vessels branching from the selected vessel in the first slice are shown.

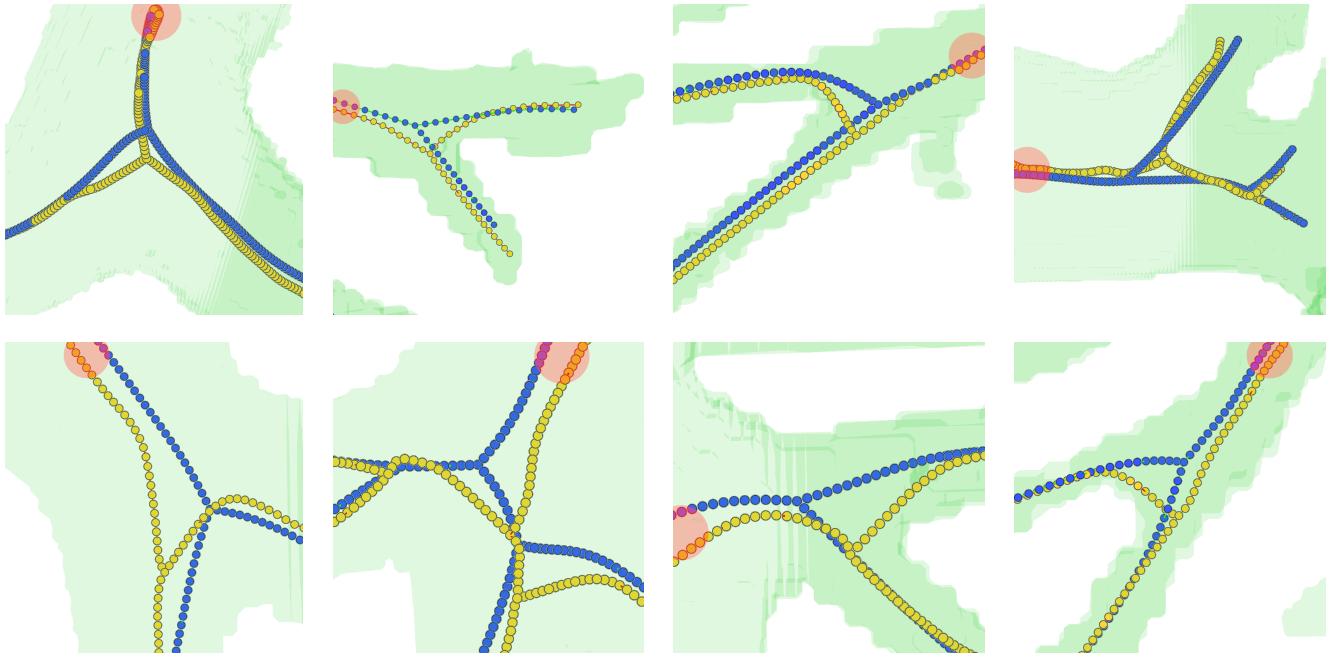


Figure 8. Qualitative examples of differing bifurcation locations between the ground truth (yellow) and RefTr predictions (blue). The red marker denotes the starting point of the highlighted vessel within each patch. Each grid cell shows a separate case. RefTr generally bifurcates earlier than the ground truth to reduce the chance of missing branches. The vessel mask (green) is shown only for visualization and is not used during training or evaluation.

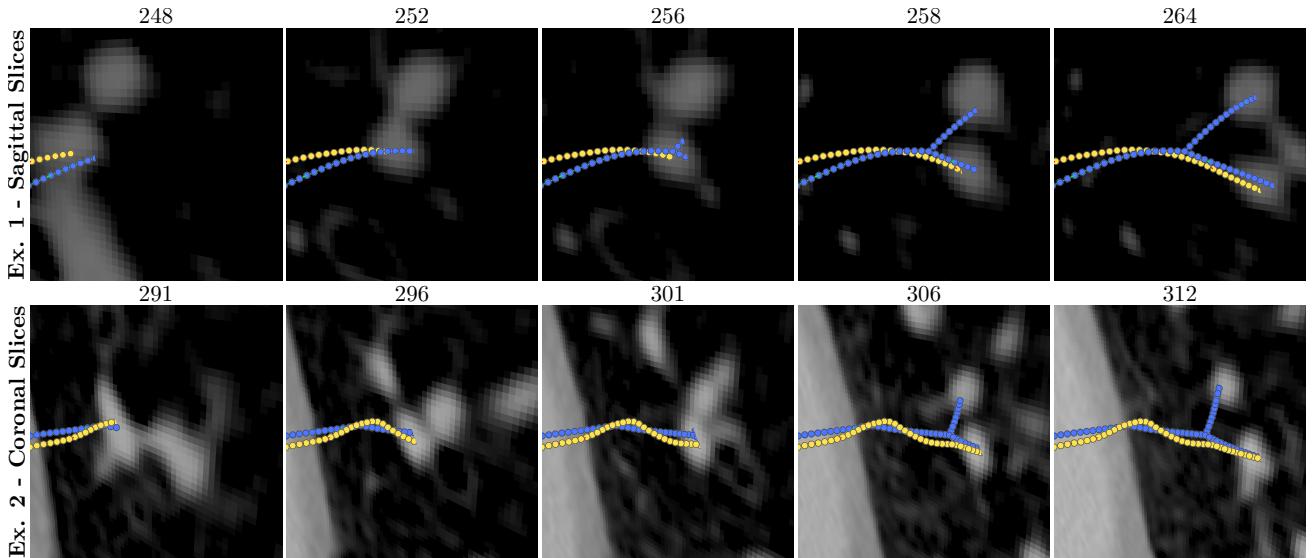


Figure 9. Qualitative examples of failure cases where two nearby branches run close together before diverging. At the divergence point, RefTr incorrectly begins to track the diverging branch as a new branch.

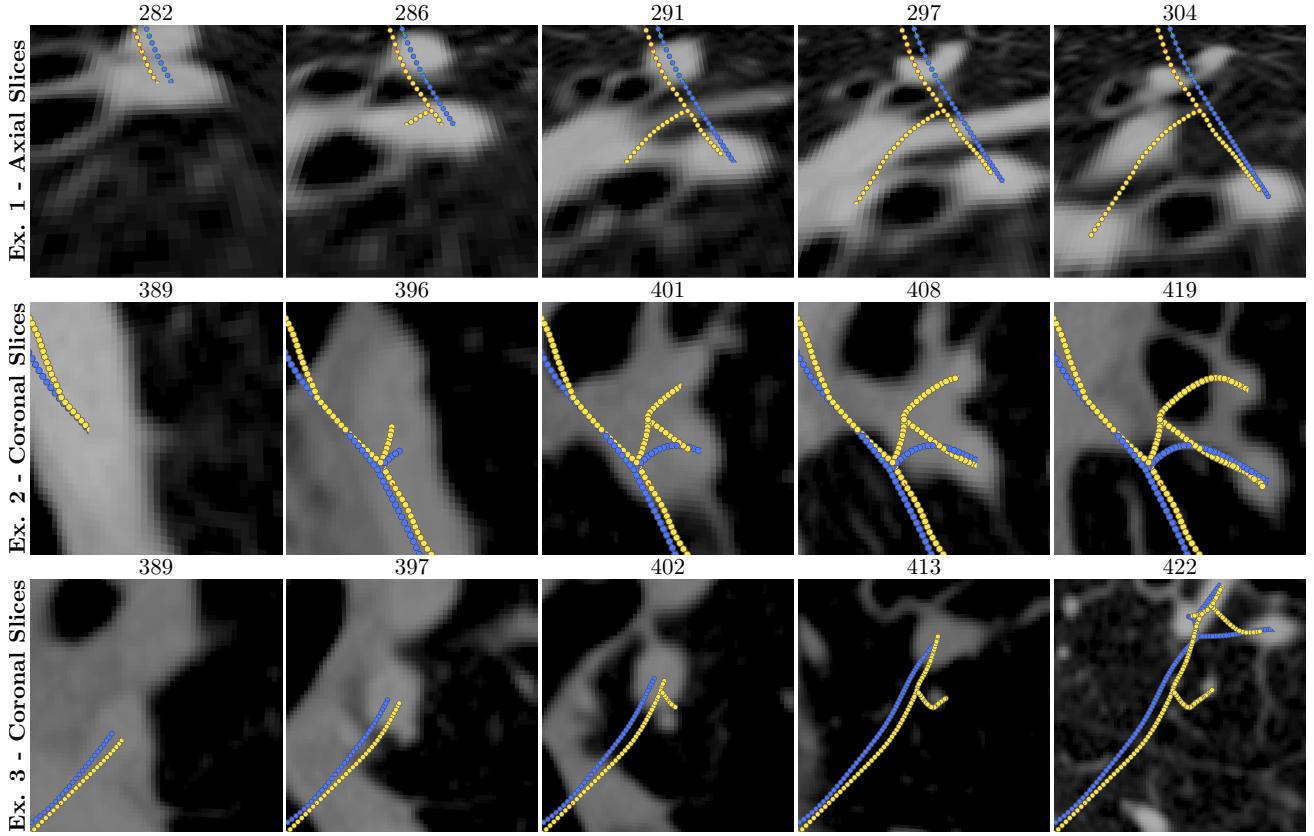


Figure 10. Qualitative examples of failure cases where RefTr (blue) misses a vessel bifurcation that leads to a new ground truth vessel (yellow). Each row shows one example across a sequence of CT image slices. Only vessels branching from the selected vessel in the first slice are shown.

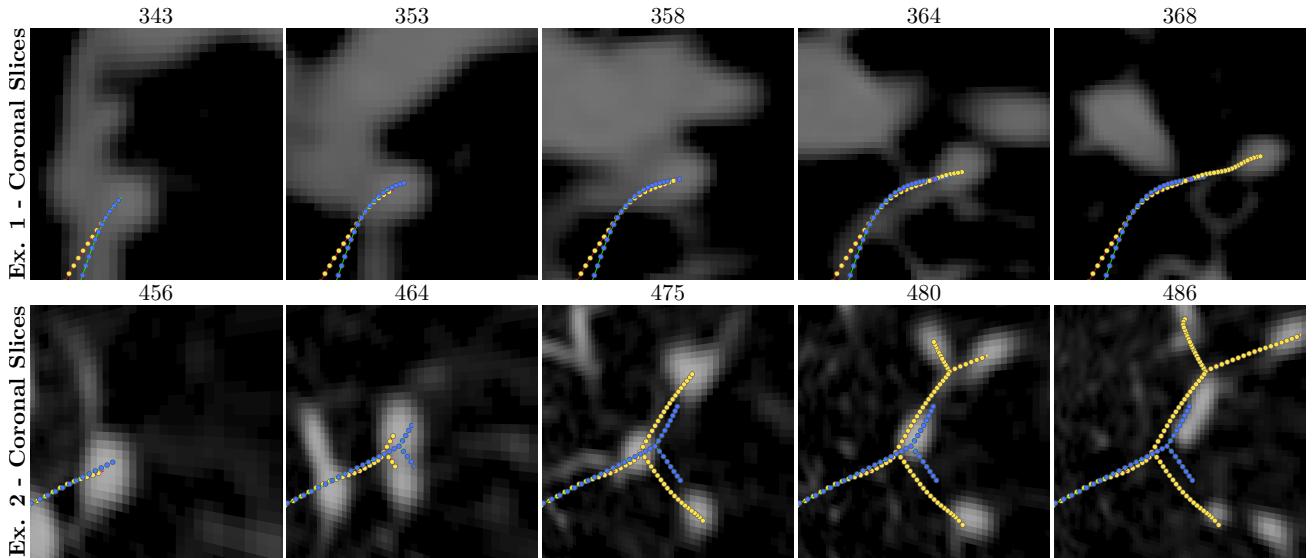


Figure 11. Qualitative examples of failure cases where RefTr (blue) terminates a branch earlier than the ground truth (yellow). Each row shows one example across a sequence of CT image slices. Only vessels branching from the selected vessel in the first slice are shown.