OUKA
The University of Osaka
Institutional Knowledge Archive

Title	PlantPose: Universal Plant Skeleton Estimation via Tree-constrained Graph Generation
Author(s)	Liu, Xinpeng; Santo, Hiroaki; Toda, Yosuke et al.
Citation	International Journal of Computer Vision. 2026, 134, p. 314
Version Type	VoR
URL	https://hdl.handle.net/11094/105906
rights	This article is licensed under a Creative Commons Attribution 4.0 International License.
Note	

The University of Osaka Institutional Knowledge Archive : OUKA

https://ir.library.osaka-u.ac.jp/

The University of Osaka

International Journal of Computer Vision (2026) 134:314
https://doi.org/10.1007/s11263-026-02882-4

PlantPose: Universal Plant Skeleton Estimation via Tree-constrained Graph Generation

Xinpeng Liu[1] · Hiroaki Santo[1] · Yosuke Toda[2,3] · Fumio Okura[1]

Received: 23 December 2024 / Accepted: 5 May 2026
© The Author(s) 2026

Abstract

Accurate estimation of plant skeletal structures (*e.g.*, branching structures) from images is essential for smart agriculture and plant science. Unlike human skeletons with fixed topology, plant skeleton estimation presents a unique challenge, *i.e.*, estimating arbitrary tree graphs from images. To address this problem, we introduce **PlantPose**, a universal plant skeleton estimator via tree-constrained graph generation. PlantPose combines learning-based graph generation with traditional graph algorithms to enforce tree constraints during the training loop. To enhance the model's generalization capability, we curate a large and diverse dataset comprising real-world and synthetic plant images, along with simplified representations (*e.g.*, sketches and abstract drawings). This dataset enables the generalized model to adapt to diverse input styles and categories of plant images while preserving topological consistency. Our approach demonstrates robust and accurate plant skeleton estimation across multiple domains, including previously unseen out-of-domain scenarios. Further analyses highlight the method's strengths and limitations in handling complex, heterogeneous data distributions. All implementations and datasets are available at https://github.com/huntorochi/PlantPose/.

Keywords Plant phenotyping · Agriculture · Image-to-graph generation

1 Introduction

Skeletal structures of plants (*e.g.*, branches and roots) are key information for analyzing plant traits in agriculture and plant science. In particular, single-view estimation of plant skeletons has potential benefits for various downstream tasks, such as high-throughput plant phenotyping (Gentilhomme et al. 2023; Cabrera-Bosquet et al. 2016; Sheng et al. 2019) and plant organ segmentation (Gaillard et al., 2020; Miao et al., 2021). As a similar task, single-view estimation of human poses has been widely studied, *e.g.*, OpenPose (Cao et al., 2017). However, unlike human skeletons, which have a fixed graph topology, the plant skeleton is not organized because the number of joints and their relationships are unknown, posing a unique problem of estimating an arbitrary tree graph from an image.

The estimation of graph structure from images has been studied to extract thin structures such as road networks in satellite images (He et al., 2020; Xu et al., 2023, 2021). Recent end-to-end models using recurrent neural networks (RNNs) (Acuna et al., 2018), graph neural networks (GNNs) (Li et al., 2020, 2021; Ling et al., 2019), or transformers (Belli & Kipf, 2019; Khokhlov et al., 2022; Li et al., 2022; Liang et al., 2020; Shit et al., 2022) show the ability to extract faithful *unconstrained* graph structures from images. However, inferring tree-constrained graphs with the existing graph generators becomes a non-trivial problem, where the output graph often violates the required constraints, as shown in Fig. 1a. One reason for this difficulty is that tree graph generation, which requires finding a set of graph edges that

Communicated by Yasuyuki Matsushita.

✉ Fumio Okura
okura@ist.osaka-u.ac.jp

Xinpeng Liu
liu.xinpeng@ist.osaka-u.ac.jp

Hiroaki Santo
santo.hiroaki@ist.osaka-u.ac.jp

Yosuke Toda
yosuke@phytometrics.jp

[1] Graduate School of Information Science and Technology, The University of Osaka, Suita, Osaka 5650871, Japan

[2] Phytometrics, Hamamatsu, Shizuoka 4350036, Japan

[3] Institute of Transformative Bio-Molecules, Nagoya University, Nagoya, Aichi 4648601, Japan

Published online: 10 June 2026

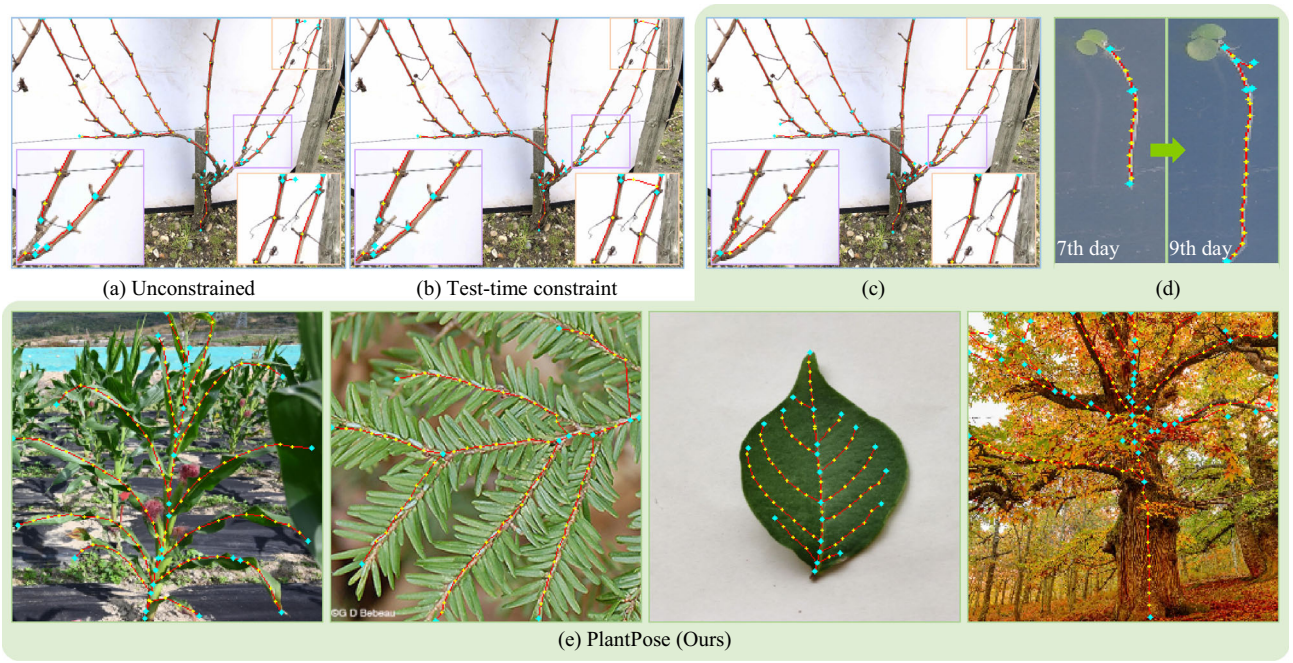


Fig. 1 We propose **PlantPose**, a method for single-image plant skeleton estimation combining learning-based graph generators with traditional graph algorithms (*i.e.*, MST). The red lines show the predicted graph edges. Compared to (a) an *unconstrained* graph generator and (b) a naive tree-graph constraint implementation, (c) our method naturally imposes the constraint during the graph generation models' training. Our method can be directly applied to plant science and agricultural applications, such as (d) time-series reconstruction of botanical roots. Furthermore, the PlantPose with large-scale training demonstrates strong generalization capabilities across diverse datasets, including significantly different out-of-domain plant species, as shown in (e)

satisfy the constraint defined on the entire graph, naturally falls into combinatorial optimization. Although constrained optimization in neural networks has been studied by designing differentiable layers (Kotary et al., 2021), using them to solve our problem is still not straightforward. A simple way to impose the constraints on the graph generation is to convert the inferred unconstrained graphs to the closest graph that satisfies the given constraint using traditional graph algorithms such as Dijkstra's shortest path or minimum spanning tree (MST) algorithms. Such post-processing can work; however, because the graph generators are trained without any constraints, the output may be unrealistic, as shown in Fig. 1b.

For tree graph generation from single images, we propose a simple yet effective way to integrate state-of-the-art learning-based graph generation methods, which achieve high-quality image-based graph estimation, and traditional graph algorithms, which compute strictly constrained tree graphs such as shown in Figs. 1c–d. Specifically, we propose to *project* an unconstrained graph into a tree graph by a non-differentiable MST algorithm during each training loop. Our selective feature suppression (SFS) layer then converts the inferred unconstrained graph to the MST-based tree graph by a differentiable manner, thereby naturally incorporating the constraints into the graph generation.

By integrating our feature suppression layer with a state-of-the-art transformer-based graph generator, we develop **PlantPose**[1], which infers tree structures from images capturing plants. Beyond the majority of plant-oriented vision methods that use domain-specific (*i.e.*, species-specific) datasets for the training, we achieve a generalized model that estimates the plant skeletons for varieties of plants, as highlighted in Fig. 1e. To this goal, we curate or collect 10 datasets spanning real-world, synthetic, and web-sourced plant images, ensuring the model is exposed to various structural patterns and contexts.

We evaluate the effectiveness of PlantPose on different classes of plant images, including out-of-domain datasets that are significantly different from those used for the training. The results show that our constraint-aware graph generator accurately estimates the target tree structures compared to baselines.

This paper is an extended version of a conference paper (Liu et al., 2025) presented in WACV 2025. Beyond (Liu et al., 2025) using the domain-specific training for each plant species, we achieve a significant advancement by focusing on the universal applicability of our method through the

[1] This terminology is, of course, an analogy of OpenPose (Cao et al., 2017) for plants.

large-scale dataset creation and training, resulting in the plant skeleton estimation across various plant species by a single pre-trained model as shown in Fig. 1e.

Contributions

Our contributions are threefold:

- We propose a novel method that tightly integrates learning-based graph generation methods with traditional graph algorithms using the newly proposed SFS layer, which modifies intermediate features in the network, effectively mimicking the behavior of the non-differentiable graph algorithms.
- Building upon our constrained graph generation method, we develop the first end-to-end method inferring skeletal structures from a single plant image, which benefits the agriculture and plant science field.
- We collect or curate plant skeleton datasets, including diverse plant images and skeletons, contributing to the generalization of our method as well as future studies for plant skeleton analysis.

2 Related Work

We propose constraining the graph structures given by image-based graph generators, whose primary goal is plant skeleton estimation. We, therefore, introduce the related work of plant skeleton estimation, graph generation from images, and constrained optimization for neural networks.

2.1 Plant skeleton estimation

Plant skeleton estimation is actively studied since it becomes a fundamental technique for downstream tasks related to plant phenotyping and cultivation (Okura, 2022).

3D plant skeleton estimation

Several methods are proposed to derive plant skeletons from 3D observations (Okura, 2022). These methods often use point clouds acquired by LiDAR (Ai et al., 2020; Xu et al., 2007) or multi-view stereo (MVS) (Sheng et al., 2019; Wu et al., 2020). Regardless of the 3D acquisition method, these works generally use a two-stage pipeline: Skeletonization (Bucksch, 2014) followed by graph optimization using MST or Dijkstra's algorithm (Chaudhury & Godin, 2020; Du et al., 2019; Huang et al., 2013; Miao et al., 2021; Ziamtsov & Navlakha, 2019), where the graph algorithms are required to convert a set of skeleton positions into a graph.

2D plant skeleton estimation

Compared to 3D methods, skeleton estimation from a single 2D image poses significant technical challenges due to the lack of depth information and severe occlusions despite the simplicity of data acquisition. Like 3D methods, existing 2D methods use a two-stage process involving skeletonization and graph optimization. To extract the skeleton regions on 2D images, plant region segmentation is often used for plants with relatively thin leaves (Giuffrida et al., 2015). Similarly, a neural network that converts an input image into a map representing 2D skeleton positions is used to mitigate the occlusions (Isokane et al., 2018). To reason about the direction of intersecting branches, a recent work (Gentilhomme et al., 2023) proposes to use vector fields representing branch direction instead of mask images, similar to the Part Affinity Fields (PAFs) used in OpenPose (Cao et al., 2017).

Unlike existing two-stage methods, we propose an end-to-end method that directly infers a tree graph representing plant skeletons in a single image. Our experiments show that our end-to-end method achieves better accuracy than a recent two-stage method for 2D images.

2.2 Graph generation from images

Graph generation from images, sometimes called image-to-graph generation, is studied for extracting geometric structures (*e.g.*, road networks) or relations (*e.g.*, scene graphs) from images (Acuna et al., 2018; Belli & Kipf, 2019; Cong et al., 2023; He et al., 2020; Khokhlov et al., 2022; Li et al., 2020, 2022, 2021; Liang et al., 2020; Ling et al., 2019; Shit et al., 2022; Xu et al., 2023, 2021).

Geometric image-to-graph generation

Recent learning-based methods for extracting geometric structures often use object detectors, which detect graph nodes (*e.g.*, intersections in road networks) from images, and then aggregate the combinations of node features to predict the edges defined between two nodes as binary (*i.e.*, existence of edges) or categorical (*e.g.*, classification of edge relations) values. Some studies use external knowledge (Abdelkarim et al., 2020; Chiou et al., 2021; Pennington et al., 2014; Sharifzadeh et al., 2022) to improve the results.

Graph generators have usually taken autoregressive methods (*e.g.*, Li et al. (2018); Liu et al. (2018); Luo et al. (2021); Belli and Kipf (2019); Zhou et al. (2023)) that output a graph by starting at an initial node and estimating neighboring graph nodes. Recent GNNs and transformers enable non-autoregressive graph generators (Cong et al., 2023; Kingma & Welling, 2014; Kipf & Welling, 2016; Li et al., 2022; Yang et al., 2022; Zang & Wang, 2020) simultaneously estimating the entire graph. Autoregressive methods are prone to errors during the estimation process, and the state-of-the-art non-autoregressive method, RelationFormer (Shit et al., 2022), performs better than autoregressive methods, especially for medium to large graphs.

A few recent studies consider graph generation with tree-graph constraints in a different context. For the molecule structure estimation (Ahn et al., 2022; Jin et al., 2018, 2020), these methods assume autoregressive graph generation, mak-

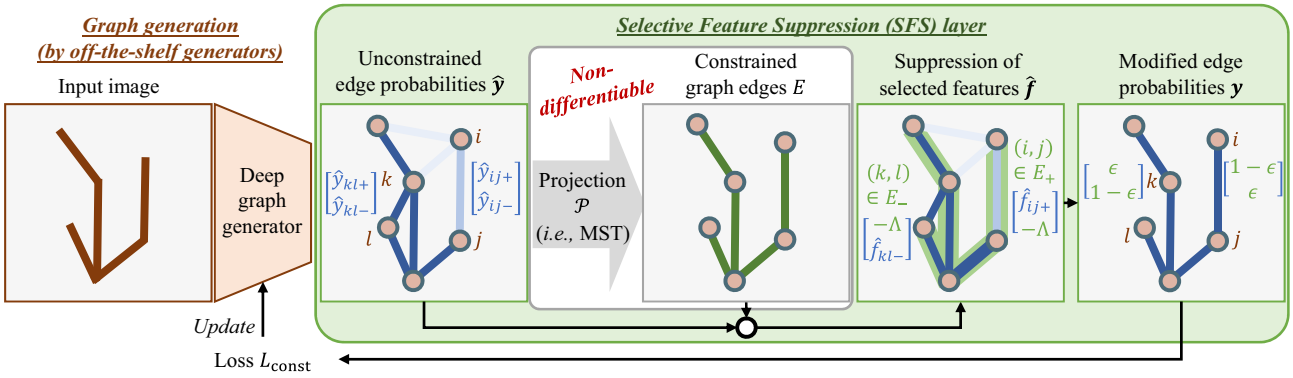


Fig. 2 Overview of reparameterization layer that can be easily plugged into off-the-shelf graph generators. Given unconstrained edge predictions by graph generators, our method *projects* it to the closest constrained graph (*i.e.*, tree) using a non-differentiable MST algorithm. Comparing constrained and unconstrained edges, unwanted edge features are selectively suppressed so that the graph becomes the tree

ing it hard to work with complex and relatively large graphs like botanical plants.

Scene graph generation

Scene graph generation (SGG) represents another line of image-to-graph research, focusing on semantic relations between detected objects. Milestone works such as IMP (Xu et al., 2017) and VCTree (Tang et al., 2019) rely on region-of-interest (RoI) features from bounding boxes and union regions, while Lang-SGG (Zhong et al., 2021) augments RoI features with geometric cues and language-derived word embeddings. More recently, VSS (Zhang et al., 2023) leverages vision–language alignment and text prompts via GLIP (Li et al., 2022) to obtain object nodes. Unlike SGG, our task deals with geometric graphs: Nodes represent junctions or endpoints defined by coordinates, and edges are thin connections between node pairs. SGG methods fundamentally assume region-based objects and semantic predicates; although we can technically use SGG methods on the extracted node positions in our case, the reasoning modules degrade into exhaustive pairwise classification with contextual encoders or become unusable without text prompts. In practice, the pipelines are reduced to those akin to RelationFormer with pairwise reasoning, but with heavier computation or additional semantic dependencies.

2.3 Constrained optimization for neural networks

Constrained optimization is crucial for machine learning. In particular, introducing constraints in neural networks has become a recent trend (Kotary et al., 2021).

Designing differentiable layers

The most direct way to introduce additional constraints to neural networks is to make the constraints differentiable. In the continuous domain, it is known that a convex optimization can be implemented as a differentiable layer (Agrawal et al., 2019). However, the design of differentiable layers for combinatorial optimization poses a significant challenge due to the difficulty of differentiation. Wilder *et al.* Wilder et al. (2019) propose a differentiable layer for linear programming (LP) problems using continuous relaxation. This method is extended to mixed integer linear programming (MILP) (Ferber et al., 2020) by splitting the problem into multiple LPs. MST, which we want to use as constraints, is known to be transformed into the class of MILP (Myung et al., 1995; Pop, 2020). However, using differentiable layers for these complex combinatorial problems requires exponential computation time (Kotary et al., 2021) to obtain the exact solution and is practically unrealistic.

Reparameterization for constrained optimization

If the constraint function is difficult to differentiate, a simple alternative is to *project* unconstrained inferences or model parameters into constrained space, which can be considered a use of reparameterization (Kingma & Welling, 2014).

In gradient descent optimization, methods projecting unconstrained optimization parameters (*e.g.*, model parameters in neural networks) to the closest ones satisfying the given constraint are called projected gradient descent (PGD). PGD is often used for traditional optimization problems, directly optimizing the input variables (Gupta et al., 2018; Xie et al., 2020). While PGD can be used for neural network optimization, such as for generating adversarial examples (Madry et al., 2018), designing projection functions for neural networks is challenging. It requires mapping a large number of model parameters into a space satisfying complex constraints, where the constraints are often more naturally defined on the model output.

Instead of designing a projection function for the model parameters, the model's output can be projected onto the subspace that satisfies the constraints during the training loop. Since reparameterization for model output can be easily integrated with existing neural network models, they are often used for domain-specific applications such as coded aperture

optimization with hardware constraint (Yoshida et al., 2018) and internal organ segmentation with given parametric shape models (Bohlender et al., 2021; Li et al., 2017). We take this approach in our SFS layer, easily plugging it into off-the-shelf end-to-end graph generation methods without preparing the differentiable implementation of the constraints (*i.e.*, MST algorithm).

3 Tree-constrained Graph Generation

We here describe the constrained graph generation method. Figure 2 summarizes the proposed SFS layer, which casts the original unconstrained edge probabilities to the constrained domain.

3.1 Problem statement

We here consider a simple setting of neural-network-based graph generation, where the model outputs the prediction of the edge probabilities (*i.e.*, the edge exists or not) defined for a pair of nodes, while this can be extended to a multi-class classification setting straightforwardly.

Our goal is to design a tree-constrained graph generator $\mathscr{F}$ that converts a given image I to a tree graph G as

$$G = (V, E) = \mathscr{F}(I) \quad \text{s.t.} \quad E \in E_{tree}, \tag{1}$$

where the graph G consists of a set of nodes (or objects) V and edges (or relations) E. Here, E_{tree} denotes all possible edge patterns forming a tree graph given the set of nodes V.

We consider a (non-differentiable) projection function $\mathscr{P}$ that maps an unconstrained graph predicted by graph generators to the constrained graph G. Let the edge probabilities defined between each node pairs as $\{\hat{\mathbf{y}}_{(i,j)}\}_{(i,j)\in V\times V}$.

$$\hat{\mathbf{y}}_{(i,j)} = [\hat{y}^{+}_{(i,j)}, \hat{y}^{-}_{(i,j)}]^{\top} \quad \text{s.t.} \quad \|\hat{\mathbf{y}}_{(i,j)}\|_1 = 1, \tag{2}$$

in which $\hat{y}^{+}_{(i,j)}$ and $\hat{y}^{-}_{(i,j)}$ respectively denote the edge existence and non-existence probabilities. The projection function $\mathscr{P}$ then read as

$$(V, E) = \mathscr{P}_{E\in E_{tree}}(V, \{\hat{\mathbf{y}}_{(i,j)}\}), \tag{3}$$

which are given by traditional graph algorithms with combinatorial optimization. We assume the projection function $\mathscr{P}$ converts the existence probability (or category prediction) of graph edges while leaving the graph nodes V unchanged. A typical example of $\mathscr{P}$ can be designed using the MST algorithm we use in our PlantPose implementation, which projects an arbitrary graph into a tree structure by modifying the existence of graph edges based on the costs defined between each pair of nodes.

We aim to develop a differentiable function $\mathscr{R}$ that mimics the non-differentiable projection $\mathscr{P}$. Plugging with the *unconstrained* graph generator $\hat{\mathscr{F}}$, Eq. (1) is rewritten as

$$G = (V, E) = \mathscr{R}_{E\in E_{tree}}(V, \{\hat{\mathbf{y}}_{(i,j)}\}), \quad (V, \{\hat{\mathbf{y}}_{(i,j)}\}) = \hat{\mathscr{F}}(I), \tag{4}$$

where the whole process is differentiable.

3.2 SFS layer

Overview

We first briefly summarize the intuition of the SFS layer. Naive neural graph generators predict unconstrained edges between node pairs, which may include cycles or disconnected parts. The MST projection converts these predictions into a valid tree by adding or removing edges. Our key idea is to mimic this projection inside the network in a differentiable way.

We here suppose the multi-head binary classification of edge existence between given two nodes, *i.e.*, the features of the last layer outputs both edge and non-edge features, followed by a softmax layer deciding whether the node pair has an edge. Our SFS layer *selectively* suppresses the features in the last layer, *i.e.*, for graph edges that should be added, we suppress the non-edge feature so that the softmax activation function outputs the edge; for edges that should be removed, we suppress the edge feature instead. All other edges remain unchanged. This selective suppression ensures that the resulting edge existence matches the tree structure given by the MST, while still allowing gradients to flow back to the backbone network.

Formulations

Here, we describe a detailed implementation of the SFS layer for constrained graph generation. As described in Eq. (2), the unconstrained graph generator $\hat{\mathscr{F}}$ computes the probability of *unconstrained* edge existence between the i-th and j-th nodes, $\hat{\mathbf{y}}_{(i,j)} = [\hat{y}^{+}_{(i,j)}, \hat{y}^{-}_{(i,j)}]^{\top}$. In neural networks, $\hat{\mathbf{y}}_{(i,j)}$ is usually computed through the softmax activation $\boldsymbol{\sigma}$ applied to the output feature vector of the final layer $\hat{\mathbf{f}}_{(i,j)} = [\hat{f}^{+}_{(i,j)}, \hat{f}^{-}_{(i,j)}]^{\top} \in \mathbb{R}^2$ as

$$\hat{\mathbf{y}}_{(i,j)} = \boldsymbol{\sigma}(\hat{\mathbf{f}}_{(i,j)}). \tag{5}$$

The set of unconstrained graph edges $\hat{E}$ is then obtained by comparing the edge existence probabilities as

$$\hat{E} = \{(i, j) | \hat{y}^{+}_{(i,j)} > \hat{y}^{-}_{(i,j)}\}, \tag{6}$$

in which $\hat{E}$ records node pairs where the edge exists.

Suppose the projection function $\mathscr{P}$ converts the set of unconstrained edge probabilities $\{\hat{\mathbf{y}}_{(i,j)}\}$ to a set of constrained edges E. Let the difference of two sets be $E^{+} =$

$E - \hat{E}$ and $E^- = \hat{E} - E$, denoting the sets of edges newly added and removed by the projection. To mimic discrete (and non-differentiable) inferences by $\mathscr{P}$ in differentiable end-to-end learning, we modify the edge features corresponding to $E^+ \cup E^-$ in the differentiable forward process.

Specifically, what we want to get is the edge probabilities that approximate the constrained edges E, denoted as

$$\mathbf{y}_{(i,j)} = \begin{cases} [1-\epsilon, \quad \epsilon \quad]^\top & ((i,j) \in E^+) \\ [\quad \epsilon \quad, 1-\epsilon]^\top & ((i,j) \in E^-) \\ \left[\hat{y}^+_{(i,j)}, \hat{y}^-_{(i,j)}\right]^\top & (\text{otherwise}). \end{cases} \tag{7}$$

When ϵ is small enough, the constrained output $\mathbf{y}_{(i,j)}$ perfectly mimics the output by the projection function $\mathscr{P}$. However, the direct modification of the edge probabilities naturally disconnects the computation graph. Therefore, we modify the unconstrained feature vector $\hat{\mathbf{f}}_{(i,j)}$ so that the corresponding edge probabilities $\mathbf{y}_{(i,j)}$ follows Eq. (7). Specifically, since $\mathbf{y}_{(i,j)}$ is computed through the softmax function $\boldsymbol{\sigma}$, it is achieved via the following minimal modification that *selectively* suppresses the feature values by replacing them with a constant[2] as

$$\begin{aligned} f^-_{(i,j)} &:= -\Lambda \quad ((i,j) \in E^+) \\ f^+_{(i,j)} &:= -\Lambda \quad ((i,j) \in E^-), \end{aligned} \tag{8}$$

where Λ is assumed to be large enough to make $\exp(-\Lambda) \sim 0$. Given modified features $\mathbf{f}_{(i,j)} = [f^+_{(i,j)}, f^-_{(i,j)}]^\top$, the softmax activation $\boldsymbol{\sigma}$ normalizes and converts them to edge probability $\mathbf{y}_{(i,j)}$.

In summary, from Eqs. (5) and (8), the constrained edge prediction between i-th and j-th nodes, $\mathbf{y}_{ij} = [y_{ij+}, y_{ij-}]^\top$, is obtained as

$$\mathbf{y}_{(i,j)} = \begin{cases} \boldsymbol{\sigma}([\hat{f}^+_{(i,j)}, \quad -\Lambda \quad]^\top) & ((i,j) \in E^+) \\ \boldsymbol{\sigma}([\quad -\Lambda \quad, \hat{f}^-_{(i,j)}]^\top) & ((i,j) \in E^-) \\ \boldsymbol{\sigma}([\hat{f}^+_{(i,j)}, \hat{f}^-_{(i,j)}]^\top) & (\text{otherwise}). \end{cases} \tag{9}$$

After the reparameterization, the set of edges computed from $\{\mathbf{y}_{(i,j)}\}$ in the same way as Eq. (6) is guaranteed to be equal to E inferred by the discrete projection function $\mathscr{P}$ when Λ is large enough.

3.3 Analysis

The common auto differentiation libraries automatically compute the gradient of the SFS layer. Although it can disconnect the computation path at a feature, since we keep at least one of the original features (either $\hat{f}^+_{(i,j)}$ or $\hat{f}^-_{(i,j)}$), the backpropagation path to the backbone graph generation network is not disconnected[3]. Here, we briefly analyze the behavior of the SFS layer. The appendices provide a detailed analysis, including mathematical proofs.

When using the cross-entropy loss $\mathscr{L}_{\text{CE}}$ to evaluate the availability of the graph edges, the derivative to be back-propagated to the backbone graph generator is approximated as[4]

$$\frac{\partial \mathscr{L}_{\text{CE}}}{\partial \hat{\mathbf{f}}} \sim \begin{cases} \left[\begin{array}{ccc} 1-t^+, & 0 & \end{array}\right]^\top & ((i,j) \in E^+) \\ \left[\begin{array}{ccc} 0 & , & 1-t^- \end{array}\right]^\top & ((i,j) \in E^-) \\ \left[y^+ - t^+, y^- - t^-\right]^\top & (\text{otherwise}), \end{cases} \tag{10}$$

where $\mathbf{t} = [t^+, t^-]^\top \in \{0,1\}^2$ denotes the ground truth edge existence and non-existence for the node pair (i,j). Our method modifies the computation graph of the network when the MST algorithm disagrees with the output of graph generation model (*i.e.*, $(i,j) \in E^+ \cup E^-$), but in different ways for derivatives of each feature value $\frac{\partial \mathscr{L}_{\text{CE}}}{\partial \hat{f}^+}$ or $\frac{\partial \mathscr{L}_{\text{CE}}}{\partial \hat{f}^-}$.

Without loss of generality, we consider the case when the MST algorithm *adds* an edge, *i.e.*, $(i,j) \in E^+$. When the MST *correctly* modify the edge availability (*i.e.*, $t^+ = 1$), the gradient vector becomes small, $\frac{\partial \mathscr{L}_{\text{CE}}}{\partial \hat{\mathbf{f}}} \sim \mathbf{0}$, which is the behavior we expect. On the other hand, if the MST *incorrectly* adds the edge (*i.e.*, $t^+ = 0$), the gradient becomes $[1,0]^\top$, which strongly penalizes the positive edge probability, where the norm of the gradient vector is always larger than unconstrained ones[5]. Therefore, the behavior of our simple reparameterization strategy is reasonable in practice.

4 PlantPose: A Plant Skeleton Estimator

We develop PlantPose, an implementation of the SFS layer to a state-of-the-art graph generator. This section first recaps the graph generator we use Shit et al. (2022) and then details how we introduce tree structure constraint.

4.1 RelationFormer: A brief recap

RelationFormer (Shit et al., 2022) is the state-of-the-art non-autoregressive graph generation method. This method uses an end-to-end architecture that combines an object (node) detector and relation (edge) predictor, which shows superior performance for unconstrained graph generation. The object detection part is based on deformable DETR (Zhu et al., 2021), which is trained to extract graph nodes (*e.g.*, objects) and global features from a given image. Specifically, given the extracted image features, the transformer decoder outputs

[2] See the appendices for the derivation.

[3] This is akin to the dropout layer often used in neural networks.

[4] We omit the subscript (i,j) for simplicity.

[5] See appendices for the mathematical proof.

a fixed number of object queries ([obj]-tokens) representing each of the nodes and a relation query ([rtn]-token) describing the global features, including node relations.

The relation prediction head outputs the relationship (*i.e.*, edge existence or category) from the detected pairs of objects (*i.e.*, [obj]-tokens) and the global relation (*i.e.*, [rtn]-tokens). This module is implemented as a multi-layer perceptron (MLP) headed by layer normalization (Ba et al., 2016). RelationFormer is trained using the sum of loss functions related to object detection and edge (relation) estimation, where edge (relation) loss $\mathscr{L}_{\text{edge}}$[6] evaluates the edge existence or category between node pairs using cross-entropy loss.

4.2 Tree-constrained graph generation

To introduce the tree structure constraint, we use Kruskal's MST algorithm (Kruskal, 1956) implemented in NetworkX[7]. To extract a tree from an unconstrained graph predicted by RelationFormer, we use the edge non-existence probabilities $\{\hat{y}^{-}_{(i,j)}\}$ as the edge cost for the MST algorithm to span the tree on edges with higher existence probabilities.

We implement the SFS layer on top of the relation prediction head in the RelationFormer. Specifically, the output features from the MLP after layer normalization are regarded as unconstrained features $\{\hat{\mathbf{f}}\}$. In our experiments, we use $\Lambda = 10$ during training, where $\exp(-\Lambda) = 4.5 \times 10^{-5}$. We show an ablation study changing Λ in the appendices.

Loss function

Our SFS layer affects the evaluation of the edge loss L_{edge} in the graph generator, while the computation of other loss functions, such as for node detection, remains the same as in the original implementation. Our implementation uses both loss functions for original (unconstrained) and constrained edges. Denoting the ground-truth edges as $E_{\text{GT}} = \{(i,j) | t^{+}_{(i,j)} > t^{-}_{(i,j)}\}$, where $\mathbf{t}_{(i,j)} = [t^{+}_{(i,j)}, t^{-}_{(i,j)}]^{\top} \in \{0,1\}^2$, the loss function for edge availability $\mathscr{L}_{\text{edge}}$ is modified as follows

$$\mathscr{L}_{\text{edge}} = \underbrace{\sum_{(i,j)} \mathscr{L}_{\text{CE}}(\hat{\mathbf{y}}_{(i,j)}, \mathbf{t}_{(i,j)})}_{\mathscr{L}_{\text{unconst}}} + \underbrace{\sum_{(i,j)} \mathscr{L}_{\text{CE}}(\mathbf{y}_{(i,j)}, \mathbf{t}_{(i,j)})}_{\mathscr{L}_{\text{const}}}, \tag{11}$$

where $\mathscr{L}_{\text{CE}}$ denotes the cross-entropy loss.

[6] Denoted as $\mathscr{L}_{\text{rln}}$ in the original paper (Shit et al., 2022), we use $\mathscr{L}_{\text{edge}}$ for generality.

[7] https://networkx.org/, last accessed on December 20, 2024.

5 Datasets

To validate the effectiveness of our method, we assess the proposed method using two scenarios, namely, domain-specific and generalized training. The former scenario is still useful for domain or species-specific analysis, which is often used in practical applications in plant science and agriculture, and we use one synthetic and two real-world datasets. Beyond the domain-specific training, we also propose to train our PlantPose model with a large-scale dataset, which comprises six datasets that span various species and attributes. For the validation of the generalized model, we also introduce out-of-domain test datasets sourced from various domains that are significantly different from those used for the training.

5.1 Domain-specific datasets

We use one synthetic and two real datasets, where examples are shown in Fig. 3 and Table 1. Appendices describe the details of the datasets.

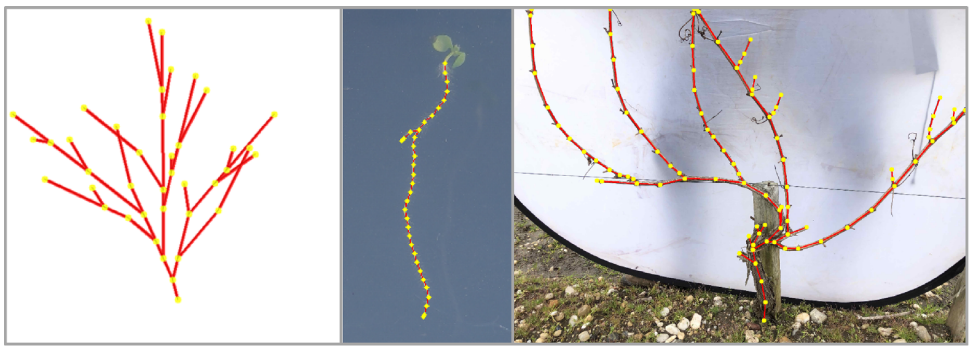
L-system synthetic dataset To systematically demonstrate the performance of our method, we perform an experiment using a large synthetic dataset. We automatically generate images of tree patterns using pre-defined rules of Lindenmayer systems (L-system) (Guo et al., 2020; Lindenmayer, 1968), which generate structural patterns using recursive processes. We add randomness of branching patterns, branch length, and joint angles to increase the dataset variation. The number of nodes in the graph is controlled at less than 100. The resolution of the generated images is 512×512 pixels. We generated 100000 images for training, 1000 for validation, and 20000 for testing.

Root dataset We use photographs of early-growing roots of Arabidopsis, which are often important targets of analysis in plant science. In this dataset, the graph structures are manually annotated. The dataset contains 781 root images, and we randomly divide them into 625 training, 78 validation, and 78 test images. Each graph contains up to 117 nodes. The image resolution is 570×190 pixels. We use data augmentation involving rotation, flipping, and cropping for the training dataset, which collectively expands the training dataset to 62, 500 images.

Grapevine dataset(Gentilhomme et al., 2023) We use the 3D2cut Single Guyot Dataset (Gentilhomme et al., 2023) containing grapevine tree images captured in an agricultural field with annotated branch patterns. The dataset contains relatively complex structures; the graph contains up to 205 nodes. The resolution is 504×378 pixels. The dataset contains 1503 images, and we use the dataset split the same as Gentilhomme et al. (2023), where 1185 images are for training, and 63 and 255 images are for validation and testing, respectively. We use data augmentation in the same manner

Table 1 Domain-specific datasets. The table summarizes the key properties of each dataset used in our experiments

Dataset	Resolution	Max Nodes	Train	Val	Test
L-system (Guo et al., 2020)	512 × 512	100	100, 000	1, 000	20, 000
Root	570 × 190	117	62, 500	78	78
Grapevine (Gentilhomme et al., 2023)	504 × 378	205	118, 500	63	255



(a) Synthetic (b) Root (c) Grapevine

Fig. 3 Example images from the domain-specific dataset. Annotated graphs are superimposed.Yellow dots and red lines indicate nodes and edges

as the root dataset, resulting in 118, 500 involving rotation, flipping, and cropping for the training dataset.

5.2 Generalized datasets

To expand the model's applicability, we curate and process six training datasets and four out-of-domain test datasets. An overview of these datasets is provided in Fig. 4 and Table 2. Across the six training datasets, we collected and processed over 22, 000 individual plant instances (before augmentation), capturing diverse morphological, environmental, and taxonomic variations.

These datasets encompass real-world, synthetic, and web-sourced images, thus covering diverse structural variations and environmental contexts. We increased the maximum node count to 384 and standardized all images to 256 × 256 pixels to ensure consistent training and evaluation conditions. For all datasets, we constructed tree graphs by identifying key branch points and sampling them at uniform intervals of 13 pixels, thereby establishing a standardized representation of structural patterns. The dataset details are described in the appendix.

6 Experiments

We conduct experiments to validate the effectiveness of the proposed method in both domain-specific and generalized scenarios using synthetic and real-world plant image datasets.

6.1 Baselines

Since the constrained graph generation task is new in this paper, there are few established baseline methods. We compare our method with the state-of-the-art methods for 2D plant structure estimation and unconstrained graph generation. Also, as an ablation study, we compare a simpler alternative to our method. Appendices provide additional comparisons with other baseline methods, including autoregressive graph generation.

Two-stage (Gentilhomme et al., 2023) We implement a 2D plant skeleton estimation method based on a two-stage method involving skeletonization and graph optimization with reference to Gentilhomme et al. (2023). Specifically, vector fields of branch directions are generated by a neural network, followed by graph optimization to generate branch structure, in which we find our implementation outperforms the naive re-implementation of the existing method (Gentilhomme et al., 2023). Specific implementations and analyses are described in the appendices.

Unconstrained (Shit et al., 2022) We compare the state-of-the-art (unconstrained) graph generation method, RelationFormer (Shit et al., 2022). This method is identical to our method without applying the tree structure constraint.

Test-time constraint As a straightforward implementation of constrained graph generation, we apply MST only in the inference phase, where the graph generator is trained using the same procedure as the unconstrained method.

6.2 Evaluation metrics

We use different metrics to capture spatial similarity alongside the topological similarity of the predicted graphs.

Table 2 Generalized dataset composition. Training datasets are marked in white, while out-of-domain test datasets are shaded in bold. Augmentation is applied only to training datasets to improve model generalization

Dataset	Max Nodes	Samples (Train/Val/Test)	Aug.	Total
Grapevine (Gentilhomme et al., 2023)	205	1202 / 150 / 151	10x	1503
LVD2021 leaf image (Li et al., 2022)	326	1858 / 232 / 232	10x	2322
LVD2021 vein mask (Li et al., 2022)	326	1858 / 232 / 232	1x	2322
MIPDB (Wang et al., 2024)	210	12992 / 249 / 250	1x	13491
Self-captured	378	652 / 82 / 83	20x	817
Web-sourced	382	1320 / 166 / 168	10x	1654
Thickened L-system	211	– / – / 500	1x	500
LVD w/ background	285	– / – / 500	1x	500
Tree synthetic	384	– / – / 500	1x	500
Root	71	– / – / 782	1x	782

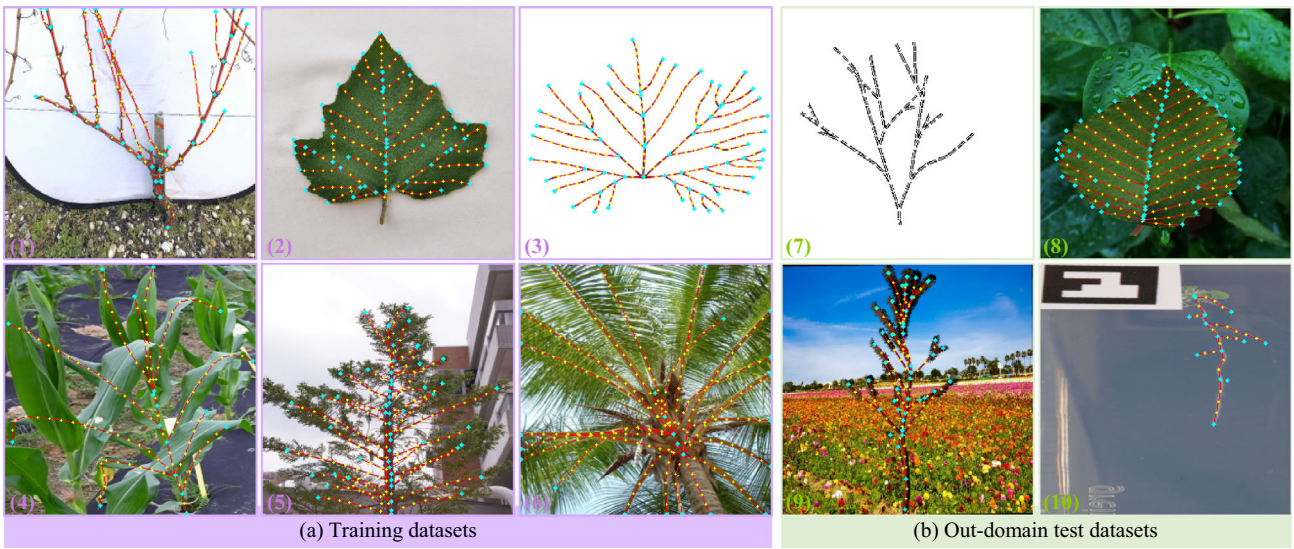


Fig. 4 Sample images from the generalized datasets. Annotated graphs are superimposed, where yellow dots represent nodes and red lines represent edges. (a) **Training datasets**: (1) grapevine, (2) LVD2021 leaf image, (3) LVD2021 vein mask, (4) MIPDB, (5) self-captured, (6) web-sourced. (b) **Out-of-domain test datasets**: (7) thickened L-system, (8) LVD with background, (9) tree synthetic, (10) root. The datasets include diverse plant structures and imaging conditions to improve the model's generalization and robustness across in-domain and out-of-domain scenarios

Street mover's distance (SMD) (Belli & Kipf, 2019) SMD is a metric to assess the accuracy of the positions of graph edges, which is computed as the Wasserstein distance between the predicted and the ground truth edges. In our implementation, the distance is computed between densely sampled points on the edges, which is the same procedure as in the original paper proposing the SMD (Belli & Kipf, 2019).

TOPO score (He et al., 2018) We compute the TOPO scores to evaluate the topological mismatch of the output graph. This metric consists of the precision, recall, and F1 scores of the graph nodes, which are evaluated considering the edge topology. We use the implementation used in the Sat2Graph paper (He et al., 2020), while we only evaluate the nodes with the degree $\neq 2$ that affect the tree structure, *i.e.*, we only evaluate joint and leaf nodes in the graphs.

Tree rate To evaluate how well the output graph satisfies the constraint, we calculate the probability that the output graph forms a tree structure. While it is obvious that the tree rate becomes 100 % for constrained methods, including ours, we are interested in how well the output of the unconstrained graph generation model can reflect the constraint by training on datasets that contain only tree graphs.

6.3 Implementation details

For RelationFormer in our method and the baseline comparison, we use the official implementation[8] on PyTorch.

[8] https://github.com/suprosanna/relationformer, last accessed on December 20, 2024.

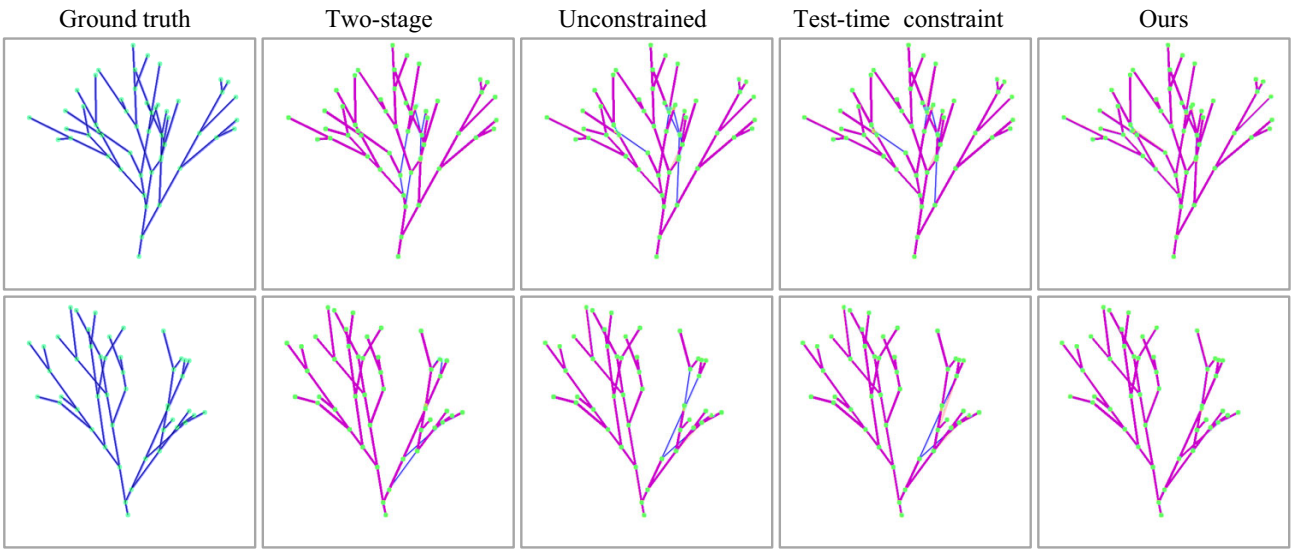


Fig. 5 Visual results for the domain-specific training with the synthetic tree pattern dataset. For the synthetic tree pattern dataset: We translucently overlay the estimated and ground truth edges with red and blue lines, respectively. While all methods accurately detect nodes, only our method accurately predicts the availability of edges from given images compared to the baseline methods

Table 3 Quantitative results on domain-specific training. Our method significantly improves both the shape and topology of the predicted graph while enforcing the given constraints. The best scores are highlighted in **bold**

Dataset	Method	SMD ↓	TOPO score ↑			Tree rate [%]
			Prec.	Rec.	F1	
Synthetic	Two-stage (Gentilhomme et al., 2023)	1.91×10^{-3}	0.940	0.886	0.912	**100.0**
	Unconstrained (Shit et al., 2022)	1.43×10^{-5}	0.978	0.929	0.953	36.2
	Test-time constraint	6.26×10^{-6}	0.977	0.953	0.965	**100.0**
	Ours	$\mathbf{4.78 \times 10^{-6}}$	**0.986**	**0.968**	**0.977**	**100.0**
Root	Two-stage (Gentilhomme et al., 2023)	4.83×10^{-4}	0.767	0.732	0.749	**100.0**
	Unconstrained (Shit et al., 2022)	1.19×10^{-4}	0.831	0.633	0.719	35.9
	Test-time constraint	1.52×10^{-4}	0.829	0.771	0.799	**100.0**
	Ours	$\mathbf{8.82 \times 10^{-5}}$	**0.861**	**0.807**	**0.833**	**100.0**
Grapevine	Two-stage (Gentilhomme et al., 2023)	4.24×10^{-4}	0.677	0.589	0.630	**100.0**
	Unconstrained (Shit et al., 2022)	1.45×10^{-4}	**0.963**	0.559	0.708	0.0
	Test-time constraint	1.47×10^{-4}	0.896	0.840	0.867	**100.0**
	Ours	$\mathbf{1.03 \times 10^{-4}}$	0.899	**0.843**	**0.870**	**100.0**

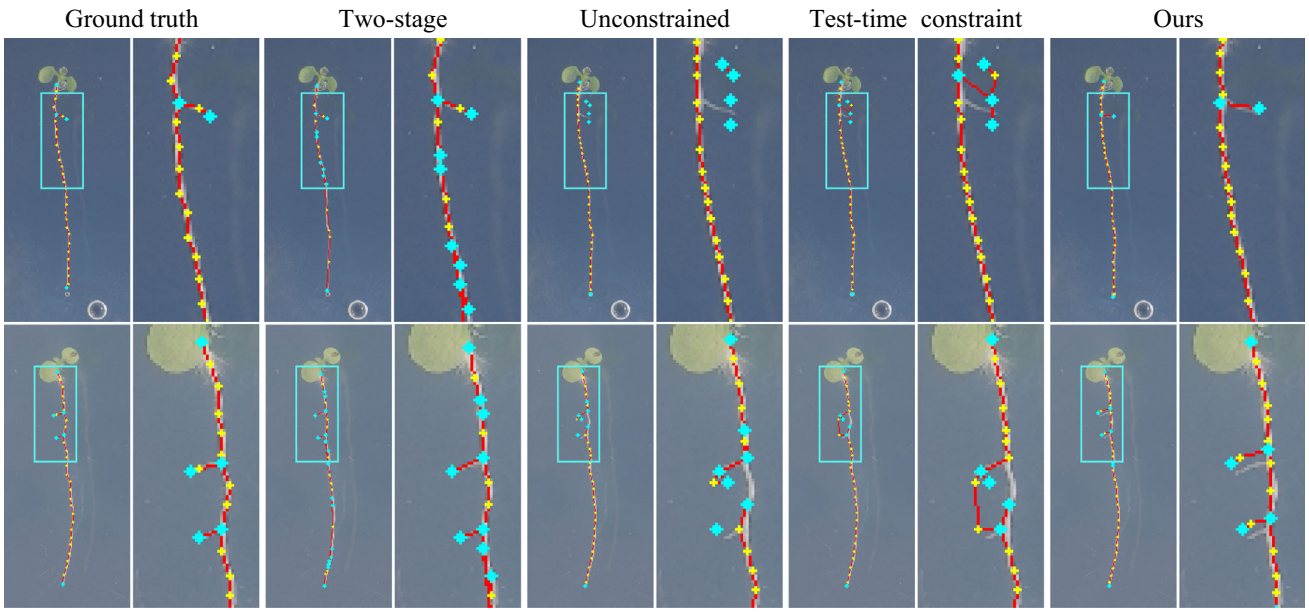
For other hyperparameters, we follow the original RelationFormer implementation used for road network extraction. We used early stopping for all datasets and methods by selecting the model with the best validation performance and terminating training after 30 epochs without improvement. The training of our method takes approximately 141 hours for the synthetic dataset, 10 hours for the root dataset, and 98 hours for the grapevine dataset, all conducted on eight NVIDIA RTX A100 GPUs.

6.4 Results

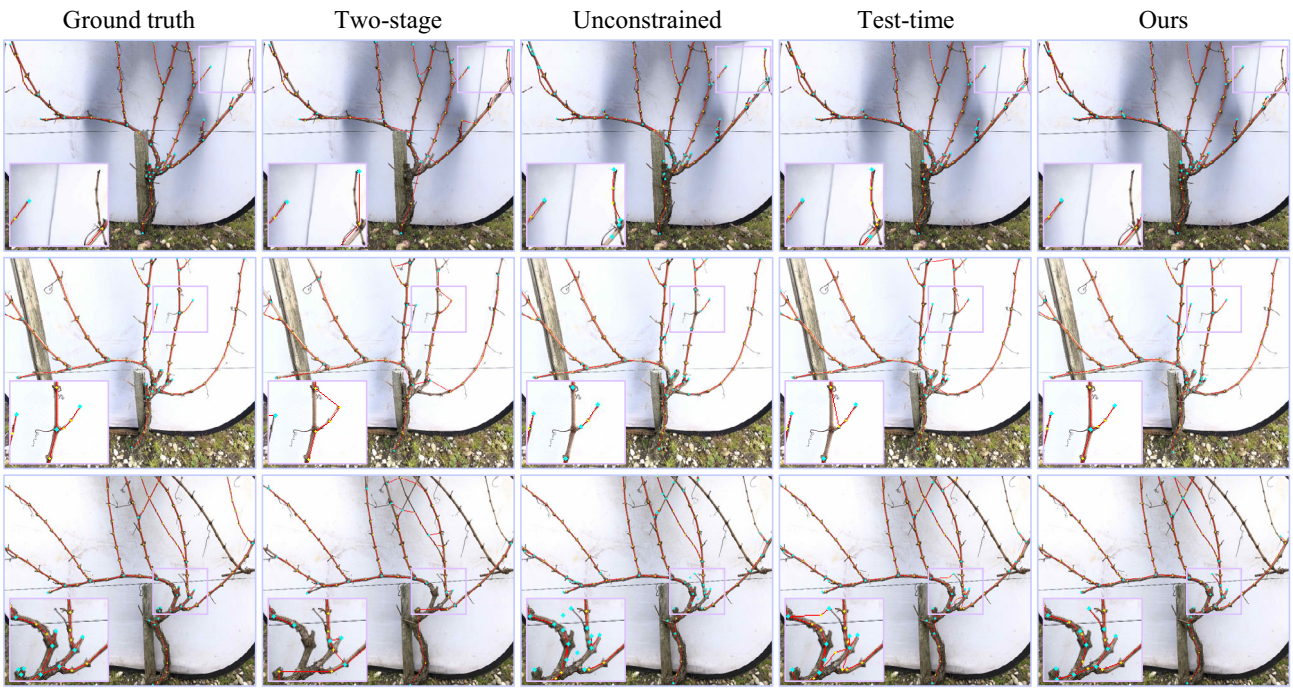
6.4.1 Results on domain-specific training

Results on synthetic dataset

Figure 5 shows visual results for the synthetic dataset, where the red and blue lines indicate the predicted and ground truth edges, respectively. Since they are shown translucently, if the estimated edge overlaps the true edge, it is displayed in purple. Similarly, cyan and yellow dots indicate the nodes, which merge into green if correctly estimated. From the



(a) Visual results for the root dataset.



(b) Visual results for the grapevine dataset.

Fig. 6 Visual results for the domain-specific training with the real image datasets: Red lines, yellow dots, and cyan dots indicate the predicted graph edges, nodes, and keypoints (*i.e.*, joints and leaf nodes). Our method accurately estimates the target plant structures compared with baseline methods, demonstrating the applicability of our method for practical uses in plant science and agriculture

results, all methods correctly estimate the node positions. The existing unconstrained method outputs isolated edges and cycles. Although the two-stage and test-time constraint methods enforce the tree structure constraint, they often produce incorrect edges. Compared to the baselines, our method accurately generates the graph edges.

The above trend can be quantitatively confirmed in Table 3. The unconstrained method produces tree structures with

Table 4 Results on generalized training. Our method significantly improves both the shape and topology of the predicted graph while enforcing the given constraints, especially for out-of-domain scenarios

Inner-domain datasets	Method	SMD ↓	TOPO score ↑ Prec.	Rec.	F1	Tree rate [%]
Grapevine	Two-stage	1.52×10^{-2}	**0.898**	*0.817*	*0.855*	*100.0*
	Unconstrained	3.58×10^{-4}	*0.956*	0.398	0.562	0.0
	Test-time constraint	*1.29×10^{-4}*	0.876	0.792	0.832	*100.0*
	Ours	**1.57×10^{-4}**	0.885	**0.798**	**0.839**	*100.0*
LVD2021 leaf image	Two-stage	6.36×10^{-3}	**0.924**	*0.772*	*0.842*	*100.0*
	Unconstrained	5.62×10^{-4}	*0.963*	0.269	0.421	0.0
	Test-time constraint	**5.44×10^{-4}**	0.881	0.731	0.799	*100.0*
	Ours	*5.41×10^{-4}*	0.890	**0.755**	**0.817**	*100.0*
LVD2021 vein mask	Two-stage	1.81×10^{-2}	0.943	0.802	**0.867**	*100.0*
	Unconstrained	4.20×10^{-4}	*0.979*	0.309	0.470	0.0
	Test-time constraint	**3.86×10^{-4}**	0.931	**0.803**	0.862	*100.0*
	Ours	*3.75×10^{-4}*	**0.948**	*0.836*	*0.888*	*100.0*
MIPDB	Two-stage	7.45×10^{-1}	0.011	0.012	0.011	39.2
	Unconstrained	1.74×10^{-3}	*0.843*	0.357	0.502	0.0
	Test-time constraint	*1.55×10^{-3}*	0.703	**0.623**	**0.660**	*100.0*
	Ours	**1.61×10^{-3}**	**0.723**	*0.653*	*0.686*	*100.0*
Self-captured	Two-stage	8.55×10^{-2}	0.486	0.358	0.413	95.2
	Unconstrained	2.12×10^{-3}	*0.912*	0.354	0.510	1.2
	Test-time constraint	*1.68×10^{-3}*	**0.747**	*0.648*	*0.694*	*100.0*
	Ours	**1.92×10^{-3}**	**0.747**	**0.642**	**0.691**	*100.0*
Web-sourced	Two-stage	3.36×10^{-1}	0.331	0.188	0.240	72.0
	Unconstrained	9.00×10^{-3}	*0.821*	0.304	0.443	1.8
	Test-time constraint	*7.92×10^{-3}*	**0.656**	*0.564*	*0.606*	*100.0*
	Ours	**8.26×10^{-3}**	0.634	**0.559**	**0.594**	*100.0*
Out-of-domain datasets	**Method**	**SMD ↓**	**TOPO score ↑ Prec.**	**Rec.**	**F1**	**Tree rate [%]**
Thickened L-system	Two-stage	2.02×10^{-2}	0.716	0.578	0.640	*100.0*
	Unconstrained	6.61×10^{-5}	*0.955*	0.389	0.553	0.0
	Test-time constraint	**5.61×10^{-5}**	0.843	**0.754**	**0.796**	*100.0*
	Ours	*5.59×10^{-5}*	**0.892**	*0.795*	*0.841*	*100.0*
LVD w/ background	Two-stage	2.99×10^{-2}	*0.892*	*0.596*	*0.715*	98.8
	Unconstrained	2.40×10^{-2}	**0.834**	0.213	0.340	0.8
	Test-time constraint	**2.34×10^{-2}**	0.678	0.450	0.541	*100.0*
	Ours	*2.22×10^{-2}*	0.725	**0.502**	**0.594**	**99.8**
Tree synthetic	Two-stage	4.97×10^{-2}	**0.796**	0.216	0.340	97.4
	Unconstrained	1.16×10^{-2}	*0.929*	0.085	0.156	0.1
	Test-time constraint	**8.57×10^{-3}**	0.767	**0.347**	**0.478**	**99.8**
	Ours	*7.34×10^{-3}*	0.772	*0.352*	*0.483*	*100.0*
Root	Two-stage	9.50×10^{-1}	0.032	0.005	0.008	5.0
	Unconstrained	6.17×10^{-2}	*0.694*	0.482	**0.569**	8.2
	Test-time constraint	**6.06×10^{-2}**	**0.585**	*0.599*	*0.592*	**94.1**
	Ours	*2.70×10^{-2}*	0.563	**0.548**	0.555	*98.2*

Italic values indicate the best score, and Bold values indicate the second best

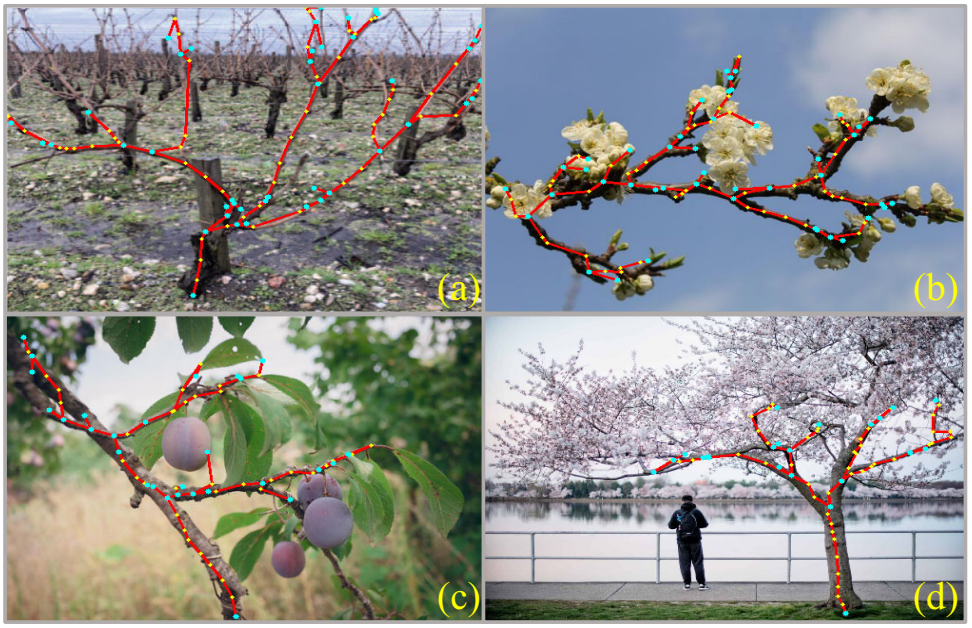


Fig. 7 Visual results highlighting generalization ability of domain-specific training using the grapevine dataset for (a) a grapevine tree under a natural background and (b–d) other tree species

only about 30 % probability, even though all the training graphs form tree structures. Although introducing the test-time constraint and two-stage methods improves the shape and topology, there are still many incorrect estimates. Compared to those baseline methods, our method significantly improves both edge positions and graph topology.

Results on real datasets

Figure 6 shows the results of skeleton estimation for two real-world datasets. For these figures, red lines, yellow dots, and cyan dots indicate the edges, nodes, and keypoints (*i.e.*, joints and leaf nodes), respectively. In agreement with the synthetic results, our method predicts visually better structures, while the unconstrained model hardly produces tree structures. The method with the test-time constraint clearly produces false edges, as shown in the results for the grapevine images. The two-stage method is often sensitive to the node detection error, leading to unnecessary (*cf*. Fig. 6a) or missing (*cf*. Fig. 6b) keypoints. In these practical settings, our end-to-end pipeline especially benefits from the simultaneous optimization of edge and node detection, resulting in faithful predictions at both nodes and edges for real-world datasets.

The quantitative results in Table 3 confirm the advantage of our method for real-world scenes. Our method shows particularly compelling results on grapevine datasets with relatively complex branching structures, outperforming the second-best method (test-time constraint) by approximately 30 % improvement on the edge accuracy evaluated by SMD.

Generalization ability of domain-specific training

Although the model is trained in a domain-specific manner, our model has a certain ability of generalization. Although the model is trained with the Grapevine dataset with few background textures, it successfully works for grapevine images with background textures (Fig. 7(a)) and for other tree species (Fig. 7(b–d)). These results highlight the generalizability of our method.

6.4.2 Results on generalized datasets

The evaluation of generalized datasets examines the model's performance across diverse plant structures. We group these results into three aspects: (1) *Inner-domain results*, evaluating performance on datasets resembling the training distribution; (2) *Out-of-domain results*, assessing adaptability to novel conditions; and (3) *Generalization ability*, highlighting performance on unseen and diverse data distributions.

Inner-domain results

Table 4 and Fig. 8 compares results across training-related datasets, including grapevine, LVD2021 leaf image, LVD2021 vein mask, MIPDB, self-captured, and web-sourced datasets. We evaluate each method using the SMD, TOPO, and tree rate metrics.

The **two-stage** method often generates a small number of redundant nodes near actual branching points during the node detection stage. These redundant nodes are typically close to the actual keypoints. Since TOPO evaluates correctness by searching within a neighborhood around ground truth keypoints, these redundant nodes increase the likelihood of matching. As a result, both precision and recall remain high. However, these redundant nodes introduce short edges that misalign with the actual graph structure, significantly increasing the SMD value.

In contrast, the **unconstrained** method generates numerous independent nodes in areas where keypoints might exist.

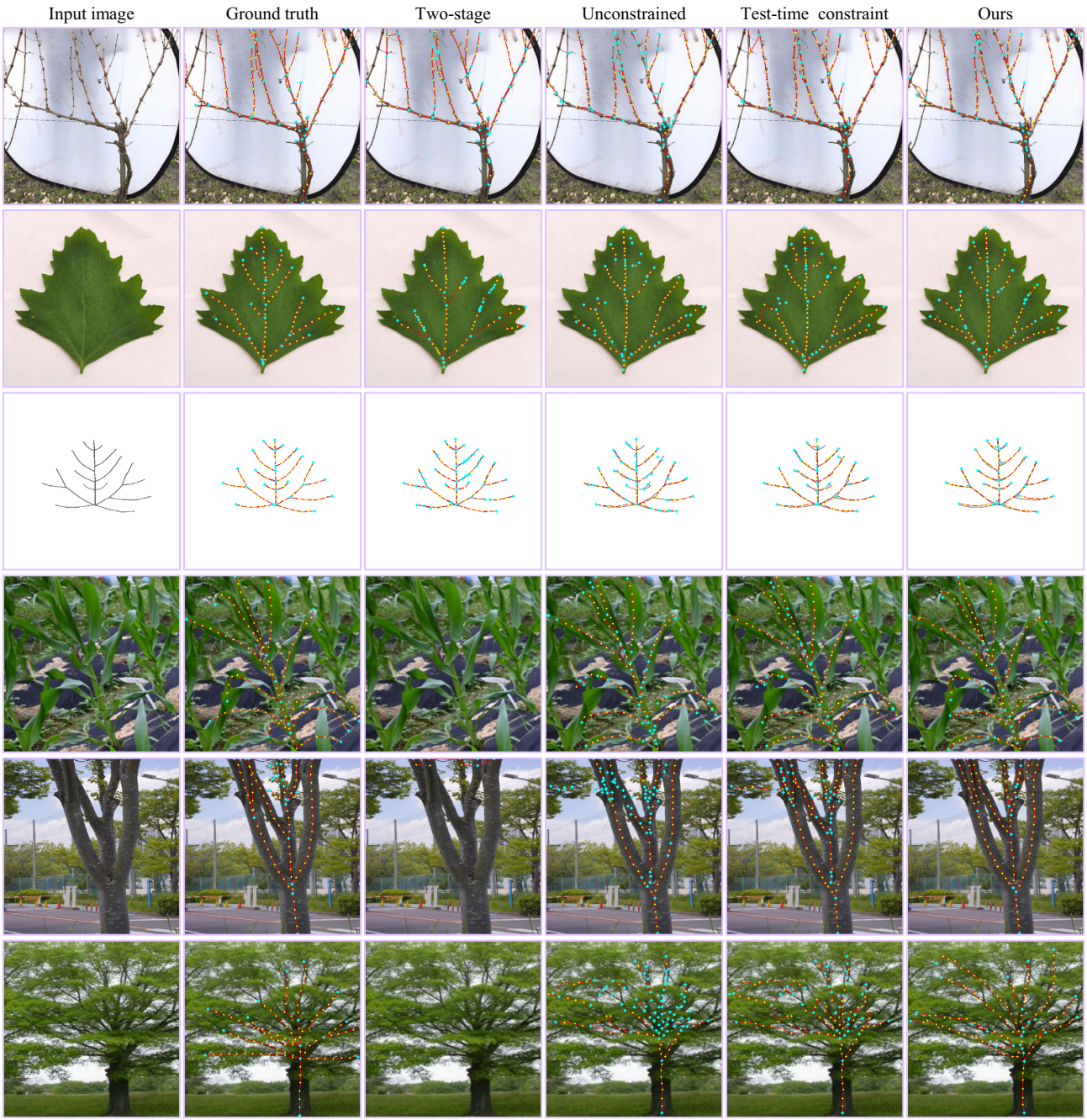


Fig. 8 Visual results on the inner-domain datasets. From top to bottom: grapevine, LVD2021 leaf image, LVD2021 vein mask, MIPDB, self-captured, and web-sourced datasets. The two-stage method sometimes produces very few nodes (*e.g.*, on the Self-captured dataset) or even fails to generate any nodes (*e.g.*, on MIPDB and web-sourced dataset). On the grapevine and LVD2021 datasets, it frequently outputs multiple redundant keypoints around actual nodes (visible as dense clusters of blue markers), thereby inflating TOPO scores but also increasing SMD. In contrast, our method achieves a more balanced performance, maintaining stable TOPO scores and low SMD values

While this increases the chances of matching predicted nodes to ground truth keypoints, many of these nodes fall outside TOPO's search range. Consequently, TOPO precision appears high due to the excess of predicted nodes, but recall remains low as not all ground truth keypoints are matched.

The **test-time constraint method** provides a tree structure at inference, improving TOPO scores and spatial coherence compared to unconstrained predictions. However, because it applies constraints *only at inference time*, it does not fully leverage structural information to refine the model's ability.

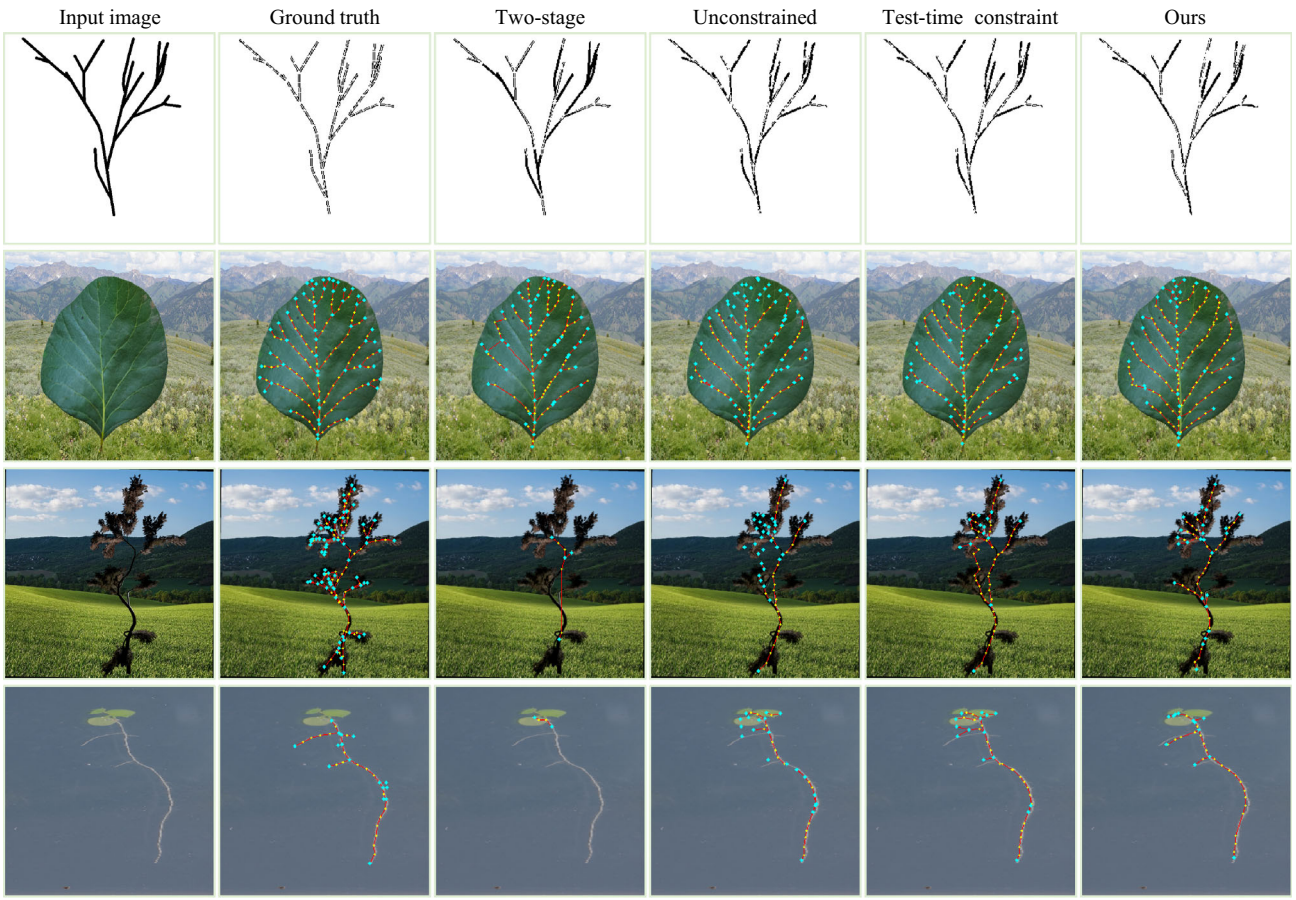


Fig. 9 Visual results on the out-of-domain test datasets: thickened L-system, LVD with background, tree synthetic, and root datasets (top to bottom). The unconstrained method generates excessive isolated points and disconnected components, inflating TOPO precision but reducing recall due to poor keypoint coverage. In contrast, our method preserves structural consistency, balancing TOPO precision and recall, and produces coherent tree structures. The root dataset highlights the difficulty of detecting short and thin lateral roots, which remains challenging for all methods

Still, the test-time constraint method is a strong baseline that ensures tree structure while maintaining reasonable topological fidelity.

Our method (ours) combines structural priors directly into the learning process. Our method consistently outperforms or closely matches the performance of the test-time constraint method in a wide range of datasets.

In summary, ours and the test-time constraint method are the top contenders on the training datasets. The test-time constraint method provides a simple baseline solution that ensures tree structure by design. At the same time, ours further refine the predictions by integrating tree constraints into the inference process more naturally.

Out-of-domain results

The out-of-domain test datasets, including thickened L-system, LVD with background, tree synthetic, and root datasets, introduce additional complexity and distributional shifts, testing the model's generalization beyond training conditions. Visual examples are shown in Fig. 9, and detailed metrics are provided in Table 4.

On the **thickened L-system** dataset, both the test-time constraint method and ours achieve near-perfect tree rates. Our method achieves a higher TOPO F1 score and lower SMD, showing that internally integrated structural priors can better handle such domain shifts, producing more topologically coherent and spatially accurate outputs.

For the **LVD with background** dataset, which merges natural leaves and environmental variations, as discussed, the two-stage method achieves a relatively high TOPO F1 but suffers from poor SMD due to redundant nodes near branching points. Both the test-time constraint and our method significantly improve SMD, with our approach achieving the lowest value. This proves our method provides the most geometrically accurate tree structures while balancing spatial precision and topological correctness.

Our method outperforms all baselines on the **tree synthetic** dataset, achieving the highest F1, lowest SMD, and



Fig. 10 Visual results on the unlabeled web-sourced dataset, including flowers, rare plants, potted plants, and hand-drawn trees. These results highlight the diversity of test cases and demonstrate our method's generalization ability across different data domains and styles

a 100 % tree rate. The lowest SMD further highlights the robustness of our combined structural constraints and MST-based pipeline in generating predictions that closely match the ground truth.

Finally, on the **root** dataset, ours and the test-time constraint method perform well, achieving high tree rates. Notably, our method achieves the lowest SMD, signifying better spatial alignment with the ground truth while maintaining comparable F1 scores and consistent topological accuracy.

Generalization ability

To evaluate the generalization ability of our method, we qualitatively test our method on the unlabeled web-sourced test dataset. We search and download diverse datasets using an extensive list of keywords, which include various plant species, unique ferns, crops, and plants with distinct appearances. These datasets feature significant variability, ranging from rare species like *Dionaea muscipula* to major crops such as soybean and sugarcane, as well as abstract hand-drawn tree structures.

Some visual results are shown in Fig. 10. Our method consistently performs well across these datasets, achieving accurate graph reconstruction even in unusual morphological structures or abstract representations. For instance, the model performed well on flowers, rare species, potted plants, and hand-drawn trees. The appendix shows more examples.

7 Conclusion

We present the first attempt at tree-constrained graph generation from a single image, especially useful for plant skeleton estimation. Our method combines modern learning-based graph generators and traditional graph algorithms, where we project the unconstrained predictions onto constrained graphs during the training loop. Our SFS layer behaves to selectively suppress the gradient to be backpropagated, which exploits the constraint cues from the non-differentiable graph algorithm (*i.e.*, MST). Our method can be easily integrated with off-the-shelf graph generators, as we develop PlantPose based on RelationFormer.

Through extensive experiments on six training datasets and four out-of-domain test datasets, we demonstrate that PlantPose achieves state-of-the-art results on a wide range of plant images. The strong performance of our model across generalized datasets highlights the effectiveness and versatility of the SFS layer in promoting robust generalization to diverse and previously unseen data distributions. The compelling results on both quantitative metrics and qualitative evaluations underscore its practical utility for plant science and agriculture.

Limitations

We use graph algorithms during each training iteration, taking a longer training time than unconstrained methods, where fast GPU-based MST implementations (*e.g.*, Vineet et al. (2009)) can improve computational performance. The success of our method depends on the accuracy of the underlying graph generation model, as we see a few undetected nodes in the visual results.

Appendix A Details of SFS layer

A.1 Motivation

Our method infers a tree graph via MST as formulated in Eq. (1)–Eq. (4) in the main paper. Our task's goal is to optimize the graph generation network so that the **final output (*i.e.*, tree graph via MST) becomes similar to the ground-truth tree graph**. Since MST modifies the edge availability in *unconstrained* inferences, the unconstrained methods evaluating the unconstrained graph edges are indirect. Instead, our method **directly evaluates the quality of the final output tree** by mimicking MST. While experiments highlight our method's benefit, the following theoretical analysis also supports this intuition.

A.2 Derivation

This section details the derivation of Eq. (10) in the main paper. To make this material self-contained, we repeat several descriptions in the main paper.

As discussed in the main paper, we consider the edge probabilities $\hat{\mathbf{y}}_{(i,j)} = [\hat{y}^+_{(i,j)}, \hat{y}^-_{(i,j)}]^\top$ is usually computed through the softmax activation $\boldsymbol{\sigma}$ applied to the output feature vector of the final layer $\hat{\mathbf{f}}_{(i,j)} = [\hat{f}^+_{(i,j)}, \hat{f}^-_{(i,j)}]^\top$ as

$$\begin{aligned}\hat{\mathbf{y}}_{(i,j)} &= \boldsymbol{\sigma}(\hat{\mathbf{f}}_{(i,j)}) \\ &= \left[\frac{\exp(\hat{f}^+_{(i,j)})}{\exp(\hat{f}^+_{(i,j)})+\exp(\hat{f}^-_{(i,j)})}, \frac{\exp(\hat{f}^-_{(i,j)})}{\exp(\hat{f}^+_{(i,j)})+\exp(\hat{f}^-_{(i,j)})}\right]^\top.\end{aligned} \tag{A1}$$

The set of unconstrained graph edges $\hat{E}$ is then obtained by comparing the edge existence probabilities as

$$\hat{E} = \{(i,j) | \hat{y}^+_{(i,j)} > \hat{y}^-_{(i,j)}\}, \tag{A2}$$

in which $\hat{E}$ records node pairs where the edge exists.

Suppose the projection function $\mathscr{P}$ converts the set of unconstrained edge probabilities $\{\hat{\mathbf{y}}_{(i,j)}\}$ to a set of constrained edges E. Let the difference of two sets be $E^+ = E - \hat{E}$ and $E^- = \hat{E} - E$, denoting the sets of edges newly added and removed by the projection. To mimic the discrete (and non-differentiable) inferences by $\mathscr{P}$ in the differentiable

end-to-end learning, we modify the edge features corresponding to $E^+ \cup E^-$ in the differentiable forward process. Here, we want to get the edge probabilities that approximate the constrained edges E, which can be denoted as

$$\mathbf{y}_{(i,j)} = \begin{cases} [\ 1\ ,\ 0\ ]^\top & ((i,j) \in E^+) \\ [\ 0\ ,\ 1\ ]^\top & ((i,j) \in E^-) \\ \left[\hat{y}^+_{(i,j)}, \hat{y}^-_{(i,j)}\right]^\top & \text{(otherwise)}. \end{cases} \tag{A3}$$

$$\sim \begin{cases} [1-\epsilon,\ \epsilon\ ]^\top & ((i,j) \in E^+) \\ [\ \epsilon\ , 1-\epsilon]^\top & ((i,j) \in E^-) \\ \left[\hat{y}^+_{(i,j)}, \hat{y}^-_{(i,j)}\right]^\top & \text{(otherwise)}. \end{cases} \tag{A4}$$

When ϵ is small enough, the constrained output $\mathbf{y}_{(i,j)}$ perfectly mimics the output by the projection function $\mathscr{P}$. Our goal is to modify the feature vector $\hat{\mathbf{f}}_{(i,j)}$ so that it makes the probabilities as Eq. (A4) through the softmax activation.

In the SFS layer, we replace the features as

$$\begin{aligned} f^-_{(i,j)} &:= -\Lambda \quad ((i,j) \in E^+) \\ f^+_{(i,j)} &:= -\Lambda \quad ((i,j) \in E^-), \end{aligned} \tag{A5}$$

where Λ is assumed to be large enough. Given modified features $\mathbf{f}_{(i,j)} = [f^+_{(i,j)}, f^-_{(i,j)}]^\top$, the softmax activation $\boldsymbol{\sigma}$ normalizes and converts them to edge probability $\mathbf{y}_{(i,j)}$ as

$$\mathbf{y}_{(i,j)} = \begin{cases} \boldsymbol{\sigma}([\hat{f}^+_{(i,j)},\ -\Lambda\ ]^\top) & ((i,j) \in E^+) \\ \boldsymbol{\sigma}([\ -\Lambda\ , \hat{f}^-_{(i,j)}]^\top) & ((i,j) \in E^-) \\ \boldsymbol{\sigma}([\hat{f}^+_{(i,j)}, \hat{f}^-_{(i,j)}]^\top) & \text{(otherwise)}. \end{cases} \tag{A6}$$

Without loss of generality, we discuss the case in $(i,j) \in E^+$. Substituting Eq. (A6) into Eq. (A1) yields

$$\begin{aligned} \mathbf{y}_{(i,j)} &= \left[\frac{\exp(\hat{f}^+_{(i,j)})}{\exp(\hat{f}^+_{(i,j)})+\exp(-\Lambda)}, \frac{\exp(-\Lambda)}{\exp(\hat{f}^+_{(i,j)})+\exp(-\Lambda)}\right]^\top \\ &= \left[\frac{\exp(\hat{f}^+_{(i,j)})}{\exp(\hat{f}^+_{(i,j)})+\epsilon'}, \frac{\epsilon'}{\exp(\hat{f}^+_{(i,j)})+\epsilon'}\right]^\top \quad ((i,j) \in E^+), \end{aligned}$$

where $\epsilon' = \exp(-\Lambda) \sim 0$ when Λ is large enough, leading to $\mathbf{y}_{(i,j)} = [1-\epsilon, \epsilon]^\top$ as in Eq. (A4) by denoting $\epsilon = \frac{\epsilon'}{\exp(\hat{f}^+_{(i,j)})+\epsilon'} \sim 0$.

A.3 Detailed analysis

We describe a detailed analysis of our reparameterization layer. As described in (9), the unconstrained edge feature between i and j-th nodes $\hat{\mathbf{f}}_{(i,j)} = [\hat{f}^+_{(i,j)}, \hat{f}^-_{(i,j)}]^\top$ is converted to constrained prediction of the edge availability $\mathbf{y}_{(i,j)} = [y^+_{(i,j)}, y^-_{(i,j)}]^\top$ by selectively suppressing unwanted feature values.

When using the cross-entropy loss $\mathscr{L}_{\text{CE}}$ to evaluate the availability of the graph edges, the derivative to be back-propagated to the backbone graph generator is[9]

$$\frac{\partial \mathscr{L}_{\text{CE}}}{\partial \hat{\mathbf{f}}} = \begin{cases} \left[(1-\epsilon)-t^+,\ 0\ \right]^\top & ((i,j) \in E^+) \\ \left[\ 0\ ,\ (1-\epsilon)-t^-\right]^\top & ((i,j) \in E^-) \\ \left[\ y^+-t^+\ ,\ y^--t^-\ \right]^\top & \text{(otherwise)}, \end{cases} \tag{A7}$$

$$\sim \begin{cases} \left[\ 1-t^+\ ,\ 0\ \right]^\top & ((i,j) \in E^+) \\ \left[\ 0\ ,\ 1-t^-\ \right]^\top & ((i,j) \in E^-) \\ \left[y^+-t^+, y^--t^-\right]^\top & \text{(otherwise)}, \end{cases} \tag{A8}$$

where $\mathbf{t} = [t^+, t^-]^\top$ denotes the ground truth edge existence and non-existence for the node pair (i,j). Our method modifies the computation graph of the network when the MST algorithm does not agree with the output of the graph generation model (*i.e.*, $(i,j) \in E^+ \cup E^-$), but in different ways for derivatives of each feature value $\frac{\partial \mathscr{L}_{\text{CE}}}{\partial \hat{f}^+}$ or $\frac{\partial \mathscr{L}_{\text{CE}}}{\partial \hat{f}^-}$.

Table 5 summarizes the case-by-case behavior, in which we can categorize the behaviors of the SFS layer into eight cases. Hereafter, we use $E^* \triangleq E^+ \cup E^-$.

Case $(i,j) \notin E^*$ *(Cases 1, 2, 5, 6)* When the graph algorithm (*i.e.*, MST) does not modify the edge availability (*i.e.*, cases 1, 2, 5, and 6 in the table), the behavior is the same as the usual cross-entropy loss for unconstrained edges.

Case $(i,j) \in E^*$ & $\mathbf{y} \sim \mathbf{t}$ *(Cases 4, 7)* In these cases, the MST algorithm *correctly* suppresses the unwanted features, where the constrained prediction $\mathbf{y}$ becomes the approximation of the ground-truth edge availability $\mathbf{t}$. The loss value becomes small, and the derivative is $\frac{\partial \mathscr{L}_{\text{CE}}}{\partial \hat{\mathbf{f}}} \sim \mathbf{0}$. This is natural since our constrained graph generator produces correct predictions.

Case $(i,j) \in E^*$ & $\mathbf{y} \nsim \mathbf{t}$ *(Cases 3, 8)* In these cases, MST *incorrectly* modifies the edge availability, *i.e.*, the node pair (i,j) belongs to E^+ or E^-, but the constrained prediction $\mathbf{y}$ does not fit the ground truth $\mathbf{t}$. Here, we mathematically discuss the behavior in these cases. Without loss of generality, we focus on Case 3, where MST *incorrectly* removes an edge and compares the methods with and without tree-graph constraints using the SFS layer.

Case 3 (MST incorrectly removes an edge)

The following discussions can be straightforwardly extended to Case 8, where MST *incorrectly* adds an edge.

Conditions:

- Unconstrained prediction (edge exists): $\hat{f}^+ > \hat{f}^-$,
- MST *removes* the edge: $(i,j) \in E^-$,
- GT edge availability (edge exists): $[t^+, t^-] = [1, 0]$.

[9] We omit the subscript (i,j) for simplicity.

Table 5 Case-by-case analyses of our reparameterization layer

Case	Feats & probs			GT	Loss	Approx. derivatives				Modified by MST
	$\hat{\mathbf{f}}$	(i, j)	$\mathbf{y}$	$\mathbf{t}$	$\mathscr{L}_{\text{CE}}$	$\frac{\partial \mathscr{L}_{\text{CE}}}{\partial \hat{f}^+}$	$\frac{\partial \mathscr{L}_{\text{CE}}}{\partial \hat{f}^-}$	$\left\lvert \frac{\partial \mathscr{L}_{\text{CE}}}{\partial \hat{f}^+} \right\rvert$	$\left\lvert \frac{\partial \mathscr{L}_{\text{CE}}}{\partial \hat{f}^-} \right\rvert$	
1	$+$		$+$	$[1, 0]^\top$	$\downarrow$	y^{+-1}	y^-	$\downarrow$	$\downarrow$	Unmodified
2	$+$		$+$	$[0, 1]^\top$	$\uparrow$	y^+	y^{--1}	$\uparrow$	$\uparrow$	Unmodified
3	$\mathbf{+}$	$\boldsymbol{E^-}$	$\mathbf{-}$	$\mathbf{[1, 0]^\top}$	$\uparrow$	**0**	**1**	$\downarrow$	$\uparrow$	**Incorrect**
4	$\mathbf{+}$	$\boldsymbol{E^-}$	$\mathbf{-}$	$\mathbf{[0, 1]^\top}$	$\downarrow$	**0**	**0**	$\downarrow$	$\downarrow$	**Correct**
5	$-$		$-$	$[1, 0]^\top$	$\uparrow$	y^{+-1}	y^-	$\uparrow$	$\uparrow$	Unmodified
6	$-$		$-$	$[0, 1]^\top$	$\downarrow$	y^+	y^{--1}	$\downarrow$	$\downarrow$	Unmodified
7	$\mathbf{-}$	$\boldsymbol{E^+}$	$\mathbf{+}$	$\mathbf{[1, 0]^\top}$	$\downarrow$	**0**	**0**	$\downarrow$	$\downarrow$	**Correct**
8	$\mathbf{-}$	$\boldsymbol{E^+}$	$\mathbf{+}$	$\mathbf{[0, 1]^\top}$	$\uparrow$	**1**	**0**	$\uparrow$	$\downarrow$	**Incorrect**

For the columns of unconstrained features $\hat{\mathbf{f}}$, constrained prediction $\mathbf{y}$, and the ground truth edge availability $\mathbf{t}$, the table shows the index of the larger element. For example, the column $\hat{\mathbf{f}}$ will be $+$ when the edge feature for the positive edge availability is larger, *i.e.*, $\hat{f}^+ > \hat{f}^-$. The column (i, j) displays E^+ or E^- if the MST algorithm modifies the edge availability (in which the rows are also highlighted). For the remaining columns, $\uparrow$ and $\downarrow$ denote each value becoming (relatively) large or small, respectively

Unconstrained method (without SFS layer)

The gradient at $\hat{f}^+$ by the unconstrained method is

$$\frac{\partial \mathscr{L}_{\text{unconst}}}{\partial \hat{f}^+} = \frac{\exp(\hat{f}^+)}{\exp(\hat{f}^+) + \exp(\hat{f}^-)} - 1. \tag{A9}$$

Since $\hat{f}^+ > \hat{f}^-$, it takes the value in the range of $(-0.5, 0)$. Similarly,

$$\frac{\partial \mathscr{L}_{\text{unconst}}}{\partial \hat{f}^-} = \frac{\exp(f^-)}{\exp(f^-) + \exp(f^+)} - 0, \tag{A10}$$

thus the gradient at $\hat{f}^-$ is inside $(0, 0.5)$. Thus, the gradient vector $\frac{\partial \mathscr{L}_{\text{unconst}}}{\partial \hat{\mathbf{f}}}$ is always shorter than $[-0.5, 0.5]^\top$ (corresponding to the special case $\hat{f}^+ = \hat{f}^-$).

Constrained method (Ours)

As described in Eqs. (A7) and (A8), our method yields the gradient as

$$\frac{\partial \mathscr{L}_{\text{const}}}{\partial \hat{f}^+} = 0, \qquad \frac{\partial \mathscr{L}_{\text{const}}}{\partial \hat{f}^-} = 1 - \epsilon \sim 1,$$

i.e., $\frac{\partial \mathscr{L}_{\text{const}}}{\partial \hat{\mathbf{f}}} \sim [0, 1]^\top$.

Comparisons

While both methods control the features to increase the edge availability, the relation of gradient vectors $\| \frac{\partial \mathscr{L}_{\text{const}}}{\partial \hat{\mathbf{f}}} \| > \| \frac{\partial \mathscr{L}_{\text{unconst}}}{\partial \hat{\mathbf{f}}} \|$ always holds, which means our method strongly penalizes the incorrect estimates by MST by directly comparing the final estimation (*i.e.*, tree graph) with the ground-truth edge availability, which highlights our key motivation—a **direct** control of the tree-constrained graph generation.

Appendix B Dataset Details

We describe the details of the datasets used in our experiment.

B.1 Domain-specific datasets

Synthetic tree pattern dataset

To prepare the synthetic dataset, we implement a generator of two-dimensional tree patterns based on the L-system (Lindenmayer, 1968), a formal language for describing the growth of the structural form. The L-system recursively applies rewriting rules to the current structure to simulate the growth of branching structures.

Figure 11 shows the initial structures and the rewriting rules we used. At the beginning of the tree generation, an initial sequence is randomly chosen from the pre-defined sequences marked with a purple frame in the figure. At each iteration during the tree generation, the leaf edges ("**A**" in the sequences) are replaced by a randomly chosen pattern from eight pre-defined ones. A simple example is shown in Fig. 12. We iterate the rewriting process a maximum of three times to generate a tree pattern. We also add randomness to the branch length and joint angles in our dataset. We randomly choose a branch length of scaling [0.5, 2.5] and joint angles of $[10°, 35°]$.

Root dataset

For the root dataset, the structure of the early-growing roots of Arabidopsis is manually annotated. The structures are annotated by placing points (*i.e.*, graph nodes) on the root path, where the distance between neighboring points may vary depending on the annotator and the images. We, therefore, resample the graph nodes with the same intervals. Starting from keypoints with the degree $\neq 2$ (*i.e.*, joints and leaf

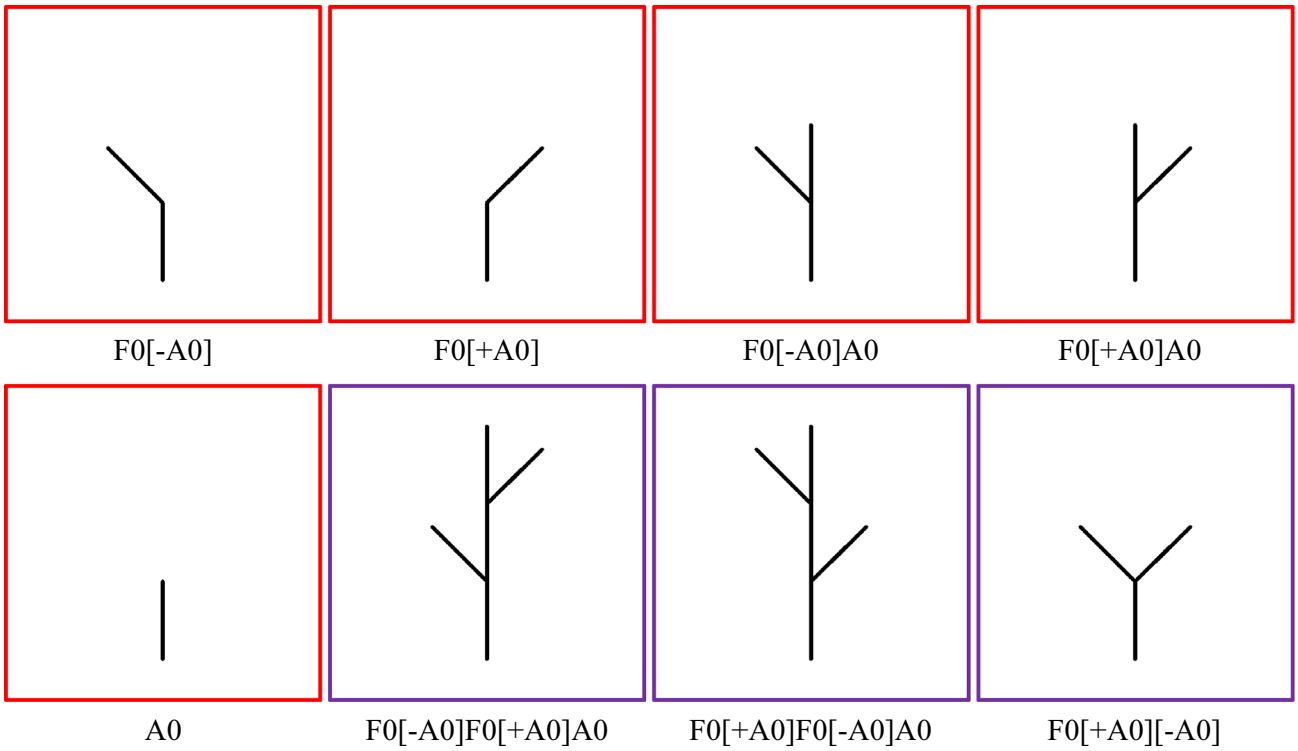


Fig. 11 Atomic structures used for synthetic dataset generation. Three pre-defined initial structures are highlighted in purple. Eight pre-defined rewriting rules are used during the generation

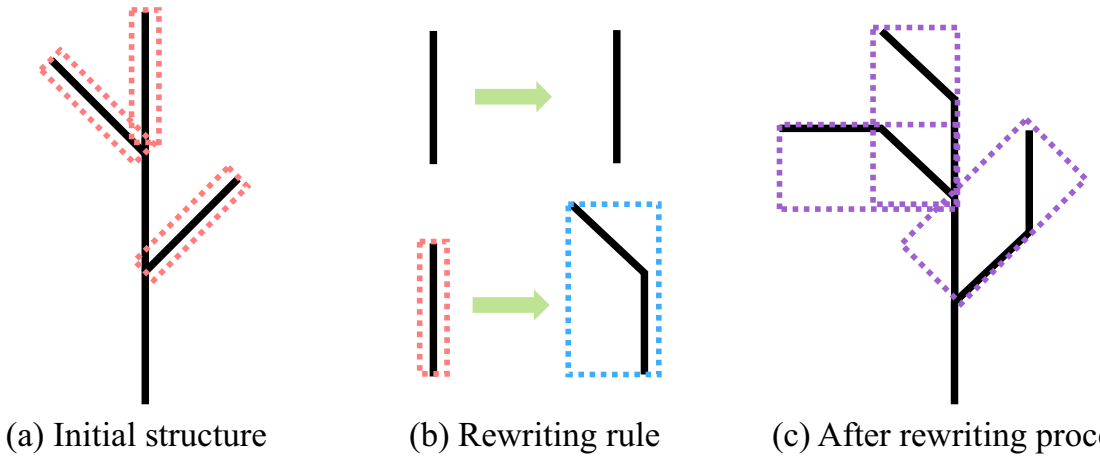


Fig. 12 An example of the rewriting process. Suppose the initial structure is represented as **F0[+A0]F0[-A0]A0**. If a rewrite rule **F**→**F**; **A**→**F[-A]** is applied, *i.e.*, **F** remains unchanged and **A** becomes **F[-A]**, the result of the rewrite process is **F0[+F1[-A1]]F0[-F1[-A1]]F1[-A1]**. The digits in the sequences indicate the number of times the rewrite is applied

nodes), we sample nodes at intervals of 8 pixels along continuous branch segments.

For data augmentation, we apply flipping, rotation, cropping, noise, lighting, and scaling on the original images. Supposing the roots are almost aligned at seeding, we limit the range of rotation angles in $[-9°, +9°]$.

Grapevine dataset

We use the 3D2cut Single Guyot Dataset (Gentilhomme et al., 2023) containing manual annotations on branch structures. We perform data augmentation with rotation angles in $[-15°, +15°]$ in the same manner as Gentilhomme et al. (2023). This dataset also contains the classification of nodes (four classes) and edges (five classes) related to biological meanings. Since the existing two-stage method (Gentilhomme et al., 2023) estimates these categories, we follow

the same setup for the two-stage baseline method (refer to the next section for detailed discussions). For the other methods, including our PlantPose implementation, we use only the binary class information (*i.e.*, branch availability) for generalizability.

B.2 Generalized datasets

B.2.1 Training datasets

Grapevine dataset (Gentilhomme et al., 2023) For the grapevine dataset, we used the same data described in the domain-specific datasets (Gentilhomme et al., 2023). Notably, the grapevine dataset has an average edge length of approximately 12.8 pixels, closely matching our 13-pixel sampling interval. This dataset also contains highly detailed annotations of plant biological joints, which are critical for capturing the structural semantics of plants. To preserve the dataset's biological accuracy and annotation quality, we did not apply additional resampling to the grapevine dataset.

We applied extensive augmentations for the grapevine images, including translation, flipping, rotation, brightness adjustment, aspect ratio modification, and scaling. This process expanded the training set tenfold, resulting in 12, 020 training images alongside 150 validation and 151 test images.

LVD2021 leaf image dataset (Li et al., 2022) The Leaf Vein Dataset 2021 (LVD2021) (Li et al., 2022) provides high-resolution RGB images of leaf veins with pixel-level annotations. The leaf images were captured against a white background, offering a clear visual contrast for leaf structure analysis. We applied the same augmentation techniques used for the grapevine dataset. This process expanded the training set tenfold, resulting in 18, 580 training images alongside 232 validation and 232 test images. The class distribution of the leaf species in the train, validation, and test splits is summarized in Table 6.

LVD2021 vein mask dataset (Li et al., 2022) The LVD2021 dataset also includes highly detailed vein masks for structural segmentation. From these vein masks, we extract three graphs by converting the masks into skeletons and identifying endpoints and joints based on the degree of each pixel. The vein masks are particularly valuable because real-world tree structures are not limited to captured or synthetic plant images but also exist in simplified forms, such as sketches or abstract drawings. Including vein masks in the training regimen enhances the model's robustness to simplified structural inputs like sketches or abstract drawings. The class distribution of the leaf species in the train, validation, and test splits is the same as the LVD2021 leaf image dataset. Unlike the leaf images, the vein masks were treated as synthetic data without augmentation, yielding 1, 858 training images, 232 validation images, and 232 test images.

MIPDB dataset (Wang et al., 2024) The MIPDB dataset (Wang et al., 2024), a maize image-phenotype database, contains over 28, 000 high-resolution images annotated using a precise point-line method. To prepare the dataset for graph-based analysis, we segmented the field images into single-plant views by clustering annotated points, ensuring that each resulting image contained only one complete plant. To prepare the dataset for graph-based analysis, we applied preprocessing steps (detailed in later sections) to convert the field images into single-plant tree-structured graphs. The final dataset consists of 12, 992 training images, 249 validation images, and 250 test images. Due to the dataset's large size and diversity, we did not apply data augmentation.

Self-captured dataset We collected images of common plants, including cherry blossoms, ginkgo trees, pine trees, and a few flowers and grasses. The dataset includes 817 images, divided into 652 training images, 82 validation images, and 83 test images. Data augmentation expanded the training set to 13, 040 images.

Web-sourced dataset We collected and organized additional data from publicly available images of various plants, including potted, coconut, and lemon trees. The dataset comprises 1, 654 images, divided into 1, 320 training images, 166 validation images, and 168 test images. The training set was augmented tenfold to 13, 200 images. We summarize the class distribution of the web-sourced dataset across the train, validation, and test splits in Table 7.

Additional preprocessing details

For the newly captured or collected images, namely the self-captured and web-sourced datasets, we manually annotate the branch paths.

For the MIPDB dataset, the original annotations were not provided as direct graphs but as sequential points along each branch. We convert them to a tree graph in a greedy manner to ensure a coherent tree structure. We first identify the longest branch as the initial tree structure. For every other branch, we compute the distances from its endpoints to each segment of the pre-computed tree structure and connect the branch endpoint closest to the pre-computed tree. By iteratively adding the newly connected branches to the tree and repeating this process, we ensure that the final graph becomes a single connected tree.

For the LVD2021 leaf image and vein mask datasets, we first skeletonize the vein masks to extract veins as lines. By treating each pixel as a node and connecting adjacent nodes, we convert the masks to tree graphs that preserve the leaf vein topology.

B.2.2 Out-of-domain test datasets

The out-of-domain test datasets consist of four distinct sources; images are processed to have their nodes at 13-pixel

Table 6 Leaf category distribution of the LVD2021 leaf image and vein mask datasets across train, validation, and test splits

Leaf Category	Train	Val	Test	Leaf Category	Train	Val	Test
Forsythia Suspensa	83	13	7	Sycamores	48	5	8
Fructus Xanthii	32	1	6	Lilac	50	10	12
Grape	70	7	12	Persimmon	7	2	2
Morning Glory	80	5	8	Smoke Tree	105	8	6
Apricot	85	7	8	Vitex Negundo Var	96	13	10
Chenopodium Album	94	14	13	Elm	77	14	12
Callistephus Chinensis	50	6	12	Holly	45	6	2
Walnut	69	9	10	Poplar	144	17	18
Maple Tree	109	16	9	Chinese Redbud	83	16	15
Amaranth	104	10	10	Hackberry	55	6	8
Honeysuckle	52	3	6	Crataegus Pinnatifida	68	8	4
Sweet Potato	87	10	11	Mirabilis Jalapa	47	8	10
Cedar	81	16	13				
Mulberry	26	2	–	Virginia Creeper	1	–	–
Thistle	9	–	–	Hibiscus	1	–	–

Table 7 Plant category distribution of the Web-sourced dataset across train, validation, and test splits

Plant Category	Train	Val	Test	Plant Category	Train	Val	Test
Pomegranate	144	16	21	Mango	156	20	19
Plum	125	12	14	Almond	2	–	–
Coconut	525	67	69	Mulberry	1	–	–
Apricot	192	24	25	Cherry	1	–	–
Lemon	173	27	20	Fig	1	–	–

intervals same as the training datasets for consistency. Details of each dataset are provided below.

Thickened L-system dataset The thickened L-system dataset shares the same generation rules as the L-system synthetic dataset from the domain-specific datasets. To distinguish this dataset from the fine, curved lines in the LVD2021 vein mask dataset, we increased line thickness and employed straight branches.

Root dataset The root dataset is identical to the root dataset used in the domain-specific datasets. Here, we combine all the data into a single test set for out-of-domain evaluation. To ensure consistency, we applied the same 13-pixel resampling process, reducing the maximum number of nodes from 117 to 71 per graph. This adjustment ensures uniform spacing between nodes, aligning with the structural standards of other datasets.

Tree synthetic dataset We employed a more advanced modular tree framework (mtree)[10], which generates more complex, irregularly branching trees with variable thickness and realistic growth patterns. We rendered these synthetic tree images onto 1,408 natural backgrounds curated by environment-based keyword searches (*e.g.*, forests, meadows, deserts, wetlands, savannahs) under varying conditions (*e.g.*, sunny, shaded, foggy, or rainy). This diverse environmental context enhances the dataset's complexity, allowing the model to adapt to lighting, density, and environmental features.

LVD with background dataset The LVD with background dataset augments the LVD2021 leaf images by placing them against 627 diverse natural backgrounds sourced from publicly available images. We applied various transformations (translation, flipping, rotation, brightness/scale adjustments) to introduce random variation in leaf appearance and positioning. The resulting dataset comprises 500 test images, providing an out-of-domain challenge for the model to adapt to varying backgrounds and leaf distortions.

Unlabeled web-sourced test dataset We also test our PlantPose qualitatively for a variety of plant images sourced online, but drawn from plant species not present in the web-sourced training dataset, to assess our method for a completely unseen distribution. This dataset does not have ground-truth annotations. This dataset was curated specifically to evaluate the generalization capability of our model on out-of-training-distribution data.

[10] https://github.com/MaximeHerpin/modular_tree, last accessed on December 20, 2024.

Table 8 Quantitative comparisons between our re-implementation of Gentilhomme et al. (2023) and our two-stage baseline implementation

Method	SMD ↓	TOPO score ↑			MSE ↓	
		Prec.	Rec.	F1	Node confidence	Edge direction
Re-implementation of Gentilhomme et al. (2023)	3.84×10^{-3}	0.459	0.365	0.406	6.95×10^{-3}	1.01×10^{-2}
Our implementation of two-stage method	$\mathbf{4.24 \times 10^{-4}}$	**0.677**	**0.589**	**0.630**	$\mathbf{1.19 \times 10^{-3}}$	$\mathbf{2.67 \times 10^{-3}}$

Appendix C Details of Baseline Methods

We describe the implementation details for the baseline methods: The *two-stage* method and the method with the *test-time constraint*. Note the implementation for the other baseline, the *unconstrained* method, is identical to the original RelationFormer (Shit et al., 2022).

C.1 Two-stage baseline

Our experiment implements a two-stage baseline involving skeletonization and graph optimization. This baseline implementation is based on ViNet (Gentilhomme et al., 2023), a state-of-the-art plant skeleton estimation method. Since the implementation of Gentilhomme et al. (2023) is not publicly available, we re-implement the method with reference to the descriptions in the paper. Through the re-implementation, we find room for improvement in the two-stage baseline method. Table 8 compares the performance of our two-stage implementation with a naive re-implementation of Gentilhomme et al. (2023). The SMD and TOPO scores are the same metrics used in the main paper, and we also compare the mean squared error (MSE) of the first-stage output of both methods. Our implementation achieves a better performance; thus, we use the improved version for our experiment. In the following, we describe the implementation details.

First stage: Skeletonization

Similar to ViNet (Gentilhomme et al., 2023), the first stage of our implementation outputs the prediction of node and edge positions as single-channel confidence maps and two-channel vector fields (hereafter referred to as node confidence maps and edge direction maps, respectively). This step is similar to a widespread human pose estimation method, *i.e.*, OpenPose (Cao et al., 2017), which jointly estimates the confidence of person keypoints and the Part Affinity Fields (*i.e.*, two-channel vector fields).

While ViNet (Gentilhomme et al., 2023) uses a sequence of residual blocks followed by the Stacked Hourglass Network (Newell et al., 2016) for this stage, we use a pre-trained ResNet50 (He et al., 2016) for image feature extraction. This is for a fair comparison to our PlantPose implementation, which also uses ResNet50 as the backbone[11]. We implement an architecture like the Feature Pyramid Network (FPN) (Lin et al., 2017), illustrated in Fig. 13, to decode the node & edge maps from the image features. Figure 14 visually compares the estimated node & edge maps, showing a better accuracy by our two-stage implementation.

The original ViNet estimates multiple classes of nodes (four classes) and edges (five classes) as different maps for the grapevine dataset. Compared to just using binary classes (*i.e.*, a branch exists or not), our two-stage implementation also yields better estimation accuracies using the multiple classes (SMD in 4.2×10^{-4} with multi-class and 1.4×10^{-2} with binary classes). Therefore, we use the multi-class setup for our two-stage implementation of the grapevine dataset. For the other dataset, we use binary classification since we do not have specific class information.

Second stage: Graph algorithm

Given the node confidence and edge direction maps, ViNet (Gentilhomme et al., 2023) first extracts the node positions, followed by the computation of the *resistivity* between each node pair, defined using the edge directions and the Euclidean distance between nodes. The final estimates of the graph structure are generated using the Dijkstra algorithm, where the tree structure is obtained by computing the shortest paths from all nodes to the detected root crown. The resistivity is used as the edge cost for the Dijkstra algorithm.

For the second stage, we follow the method in Gentilhomme et al. (2023) except for the graph algorithm used; namely, we compute MST instead of the shortest paths given by the Dijkstra algorithm, since using MST reduces the SMD metric to 4.2×10^{-4}, compared to 5.9×10^{-4} using the Dijkstra algorithm for the grapevine dataset.

Detailed parameter settings

The two-stage method involves heuristic parameters for node and edge detection. Therefore, we empirically select the best parameter sets for each dataset. Table 9 lists the detailed parameters. In particular, for the root dataset, we need to carefully tune some hyperparameters (namely, d, τ_m, and τ_n in the table) to yield reasonable estimates, where the configurations yielding the best SMD scores are reported in the main paper.

[11] ResNet50 is actually used as the node detection module in RelationFormer (that is based on Deformable DETR (Zhu et al., 2021)), which is the basis of our PlantPose, and we inherited its implementation.

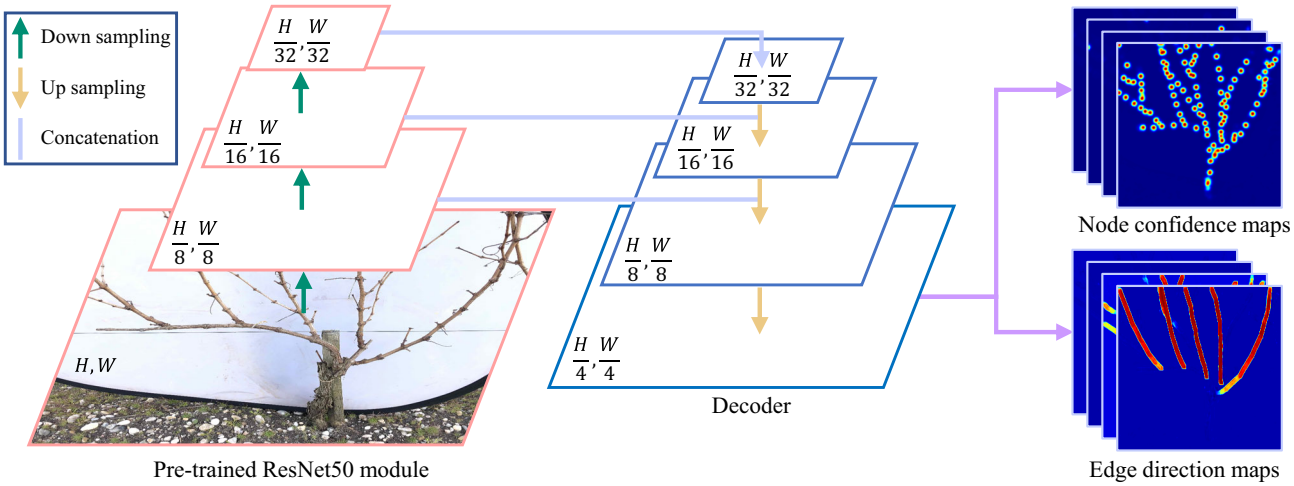


Fig. 13 Network architecture for the first (skeletonization) stage of our two-stage baseline method

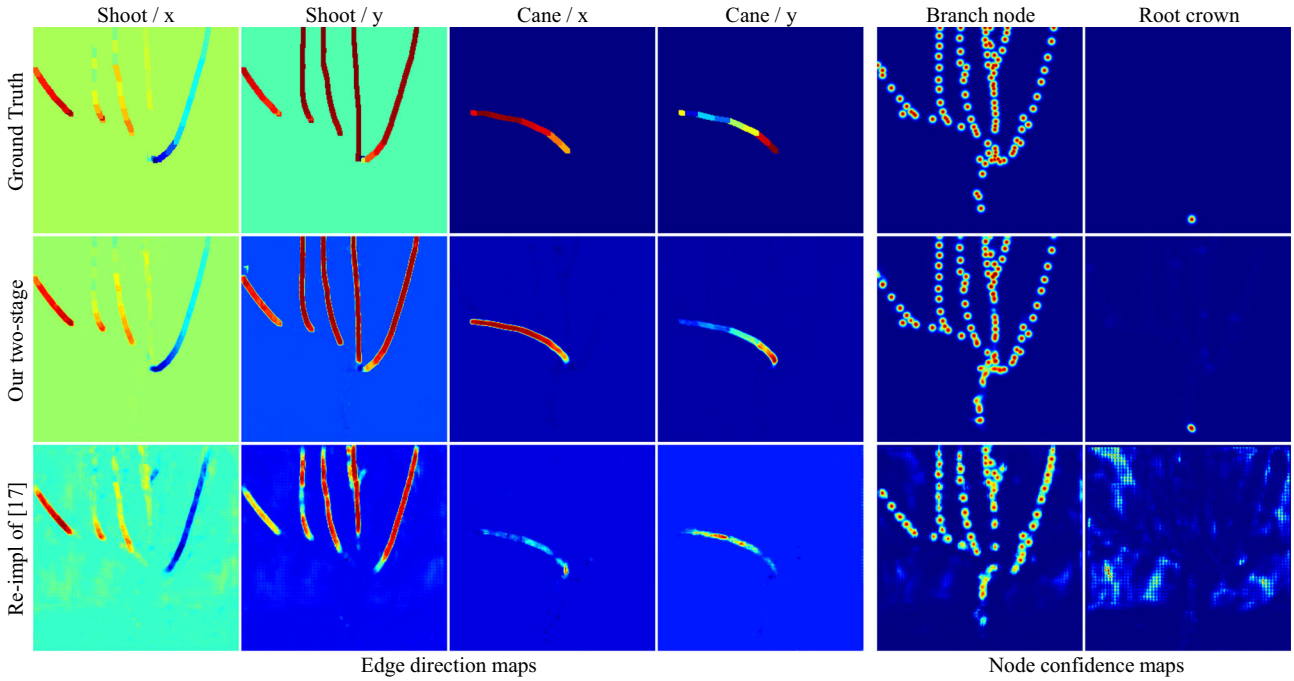


Fig. 14 Visual comparisons between a re-implementation of Gentilhomme et al. (2023) and our two-stage baseline implementation. Our implementation yields better node confidence and edge direction maps, which are the outputs of the first stage of these methods

Table 9 Parameters used for the two-stage baseline method

Dataset	Image size (W, H) [px]	Map size (W, H) [px]	Node confidence diameter [px]	Edge direction width [px]	Node search distance d [px]	Thresholds for node detection τ_m	τ_n
Synthetic	512, 512	512, 512	4	5	9	0.97	0.5
Root	570, 190	570, 190	3	5	7	0.99	0.3
Grapevine	1008, 756	256, 256	3	10	25	0.97	0.5

d denotes the distance threshold for the local maximum value search (*i.e.*, non-maximum suppression) of node candidates. τ_m and τ_n are used as the thresholds for node detection from the confidence maps. For the detailed definitions, refer to the original paper (Gentilhomme et al., 2023)

C.2 Test-time constraint baseline

For the test-time constraint baseline, we apply MST only in the inference phase, where the graph generator is trained using the same procedure as the unconstrained method. The MST used in this baseline method is identical to our proposed one.

Appendix D Performance Analysis

In this section, we present a detailed analysis of the performance of our proposed method. The effectiveness of our method is evaluated through comprehensive experiments in different scenarios. Specifically, we compare our method with the Auto-regressive (AR) model, and we also analyze the performance when our method is applied solely during the training process.

D.1 Comparison with auto-regressive method

While we implement the tree-graph constraint on the state-of-the-art non-autoregressive graph generator, RelationFormer (Shit et al., 2022), other choices of constrained graph generation are viable. Existing works aiming for tree-constrained graph generation, such as in molecule structure estimation (Ahn et al., 2022; Jin et al., 2018, 2020), use auto-regressive graph generation. AR methods are a simpler choice for imposing the constraint since it is relatively straightforward to implement the tree-graph constraint in their graph development process. However, since the auto-regressive methods generate graph nodes and edges progressively, they are prone to breakdowns due to changes in the output order or errors during the generation. This tendency is particularly pronounced for relatively large graphs, including our setup.

To assess the potential of auto-regressive methods, we test the state-of-the-art transformer-based auto-regressive graph generator, Generative Graph Transformer (GGT) (Belli & Kipf, 2019). Table 10 compares our method and several variances of GGT on the synthetic tree pattern dataset. The results show that the accuracy of GGT falls short compared to our method, although the vanilla GGT (the top row) mostly outputs tree graphs (92 %) without explicitly imposing the tree-graph constraint. We identified that errors by GGT occurring at a particular step in the auto-regressive generation process continuously cause errors in the sequence of the following generations. The GGT was initially designed for small datasets, specifically for graphs with $|V| \leq 10$. For our setup, where $|V| \geq 100$, generating these long sequences in a specific order presents a significant challenge.

D.2 Effectiveness of tree-constraint during training

To assess whether our SFS layer (positively) affects the training process itself or not, we evaluate our method *without* using the SFS layer and the MST algorithm during the inference phase, *i.e.*, introducing constraint only during the training phase (called *train-time constraint* hereafter). Table 11 summarizes the performances. Inducting the tree constraint during the training phase mostly outperforms the methods without constraints, meaning that the improvement by our method is based on network improvement by the loss propagated via the SFS layer. We also checked the change in the accuracy metric during training, and found our method consistently achieved better accuracy from the beginning of the training.

Appendix E Other Design Choices

The experiments in the main paper already provide some ablation studies, namely, comparisons of our method with 1) graph generation without constraint (*unconstrained*), and 2) a method without using MST in the training loop (*test-time constraint*). Here, we delve further into the potential design choices of our PlantPose model.

E.1 Other graph generators

Although the proposed module, the SFS layer, can be easily integrated into graph generators other than RelationFormer (Shit et al., 2022), we found that no methods but our PlantPose implementation achieve satisfactory results. Here, we discuss the results of the implementation of our method to the AR graph generator, GGT (Belli & Kipf, 2019), which achieves the second-best accuracy for multiple datasets following the state-of-the-art RelationFormer.

Table 10 in the last section compares the GGT with and without the tree-graph constraint. Compared to the GGT with test-time MST, using our SFS layer on top of GGT improves both SMD and TOPO scores[12], although the accuracy is insufficient in practice due to the drawback of AR-based generation processes discussed above. Using the newer RelationFormer model significantly improves the estimation accuracy, which implies that our SFS layer will benefit from the future development of graph generation models.

E.2 Using node distances for edge cost in MST

Although our proposed method uses the edge non-existence probabilities $\{\hat{y}^{-}_{(i,j)}\}$ as the edge cost for the MST algorithm,

[12] GGT w/ SFS layer does not achieve 100 [%] tree rate because it sometimes fails to generate any graphs.

Table 10 Comparisons of different graph generation models (*i.e.*, RelationFormer (Shit et al., 2022) and GGT (Belli & Kipf, 2019)) on the synthetic dataset

Method	SMD ↓	TOPO score ↑			Tree rate [%]
		Prec.	Rec.	F1	
GGT (Belli & Kipf, 2019)	2.71×10^{-3}	0.635	0.537	0.582	92.06
GGT w/ test-time constraint	4.13×10^{-3}	0.620	0.545	0.580	98.10
GGT w/ SFS layer	2.80×10^{-3}	0.652	0.584	0.616	99.63
RelationFormer (Shit et al., 2022) w/ SFS layer (Ours)	4.78×10^{-6}	0.986	0.968	0.977	100.0

Table 11 Quantitative results with additional baseline, *train-time constraint*

Dataset	Method	SFS	SMD ↓	TOPO score ↑			Tree rate [%]
				Prec.	Rec.	F1	
Synthetic	Unconstrained		1.43×10^{-5}	0.978	0.929	0.953	36.2
	Test-time		6.26×10^{-6}	0.977	0.953	0.965	**100.0**
	Train-time	✓	8.44×10^{-6}	**0.987**	0.954	0.970	56.5
	Ours	✓	$\mathbf{4.78 \times 10^{-6}}$	0.986	**0.968**	**0.977**	**100.0**
Root	Unconstrained		1.19×10^{-4}	0.831	0.633	0.719	35.9
	Test-time		1.52×10^{-4}	0.829	0.771	0.799	**100.0**
	Train-time	✓	$\mathbf{7.81 \times 10^{-5}}$	0.853	0.619	0.718	37.2
	Ours	✓	8.82×10^{-5}	**0.861**	**0.807**	**0.833**	**100.0**
Grapevine	Unconstrained		1.45×10^{-4}	0.963	0.559	0.708	0.0
	Test-time		1.47×10^{-4}	0.896	0.840	0.867	**100.0**
	Train-time	✓	1.30×10^{-4}	**0.965**	0.566	0.713	0.0
	Ours	✓	$\mathbf{1.03 \times 10^{-4}}$	0.899	**0.843**	**0.870**	**100.0**

inspired by the two-stage method that uses node distance for the edge cost computation, we multiply the Euclidean distance between nodes by our original edge cost.

As a result, SMD with modified edge cost does not improve accuracy (it achieves the same SMD as our method in the Grapevine dataset). A possible reason for this is that the graph generator itself can take the node distance into account when estimating graph edges. Therefore, we simply use the edge non-existence probabilities $\{\hat{y}^{-}_{(i,j)}\}$ as the edge cost for our method.

E.3 Ablation for Λ

An important hyperparameter in our method is Λ, which controls the level of suppression for unwanted features. Here, we report an ablation study for this parameter using the grapevine dataset. Table 12 shows that our choice ($\Lambda = 10$) achieves better, while the changes in Λ do not significantly affect the overall accuracy as long as $\exp(-\Lambda)$ is close enough to zero. This result indicates that our method is robust to the hyperparameter setting.

Appendix F Additional Visual Results

F.1 Results on domain-specific training

We first present the results of domain-specific training. Figures 15 and 16 show the additional results for the synthetic and root datasets, respectively. Figures 17 and 18 show the results for the grapevine dataset. Figure 19 shows the results for out-of-domain testing.

These results consistently demonstrate the high-fidelity estimation of plant skeletons by our PlantPose, which uses the SFS layer that incorporates the constraints while training graph generation models.

F.2 Results on generalized training

We present the results of the training using generalized datasets. Figures 20 to 25 illustrate the visual outputs for the six training datasets, showcasing the diversity and structural complexity captured by our model. Figures 26 to 29 display the results for the four out-of-domain test datasets, which assess the model's ability to generalize to unseen structure and contexts. Finally, Figures 30 to 31 demonstrate the performance on the unlabeled web test dataset, which further

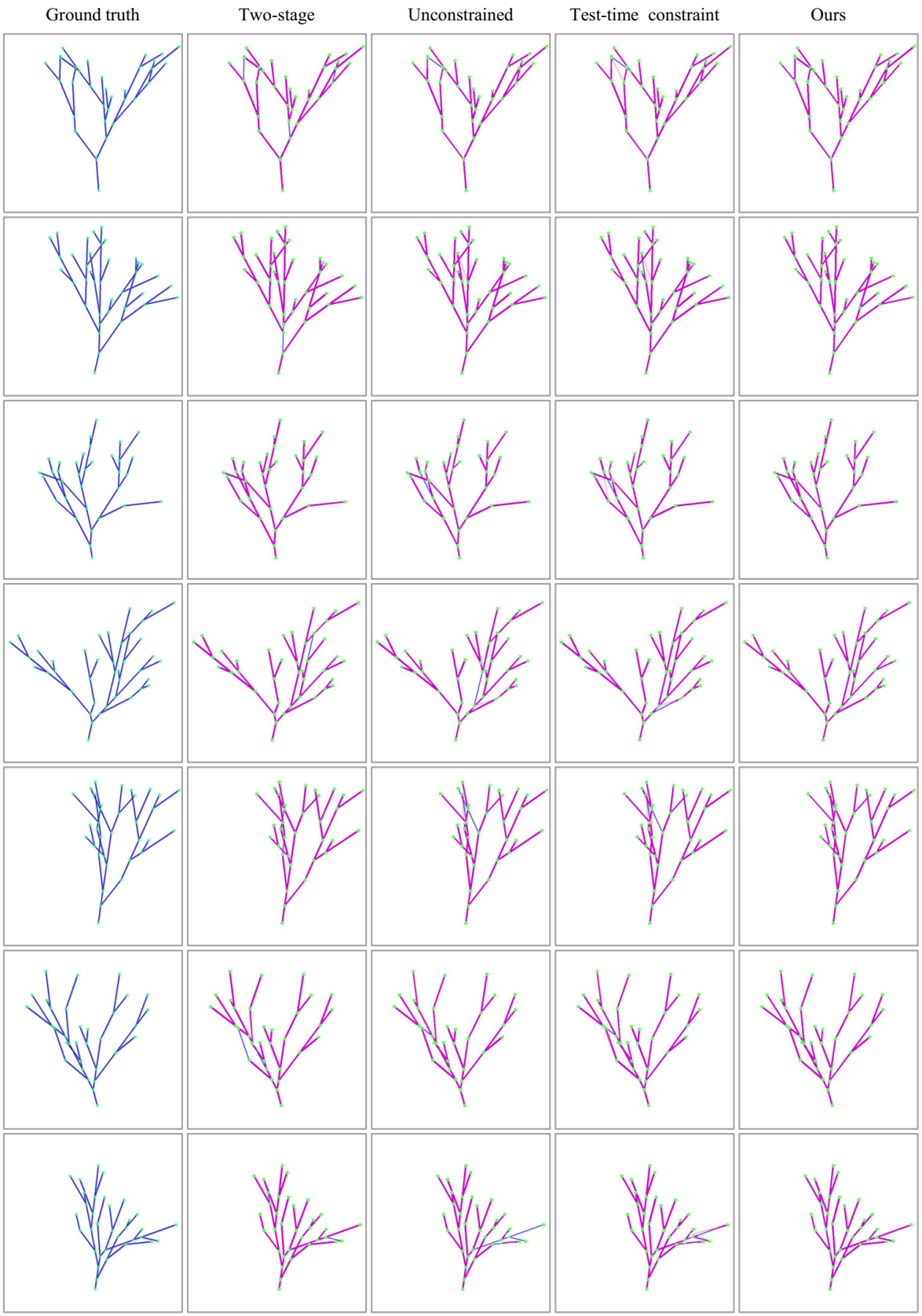


Fig. 15 Additional results for the synthetic branch pattern dataset (domain-specific training)

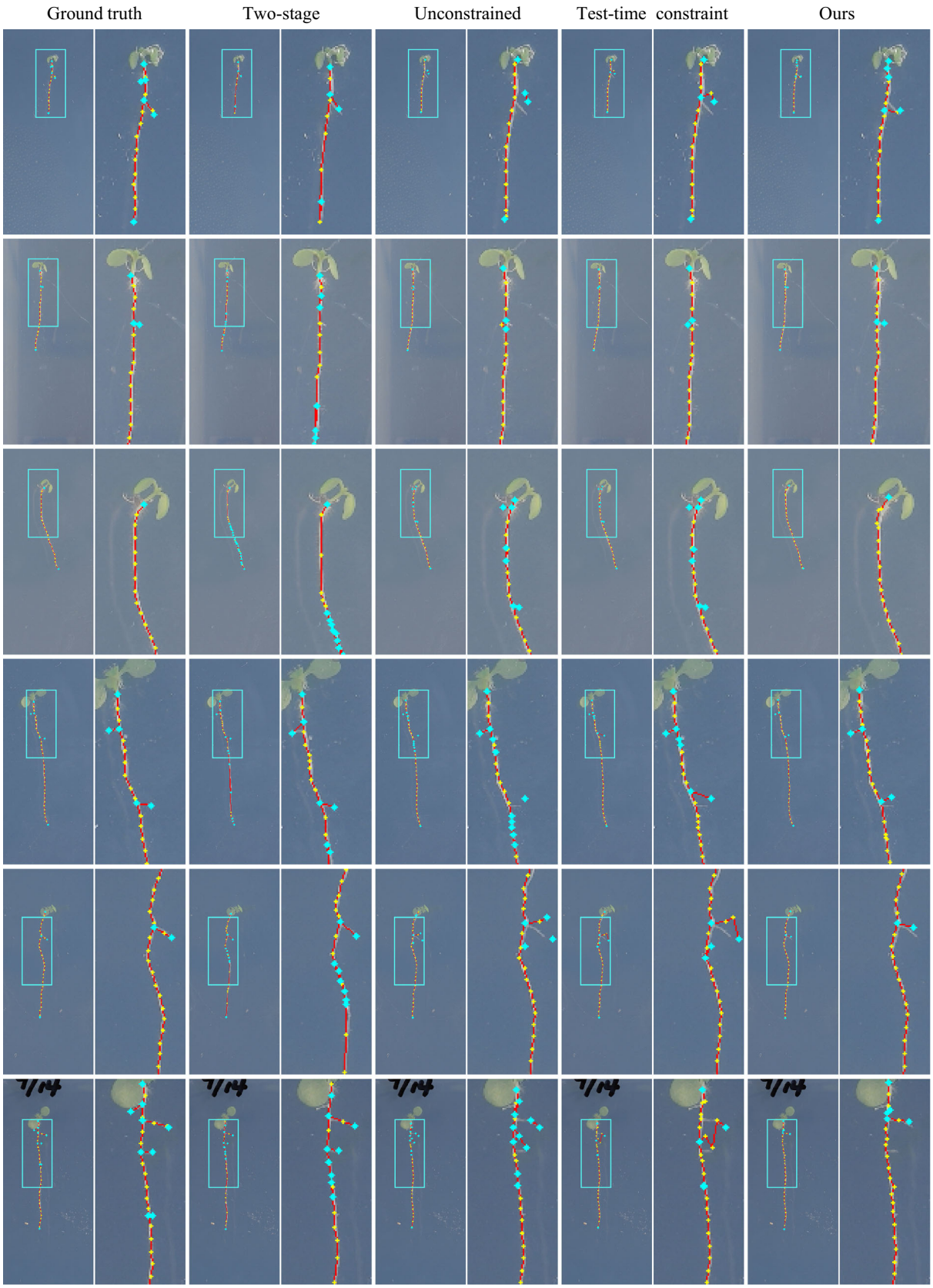


Fig. 16 Additional results for the root dataset (domain-specific training)



Fig. 17 Additional results for the grapevine dataset (domain-specific training)

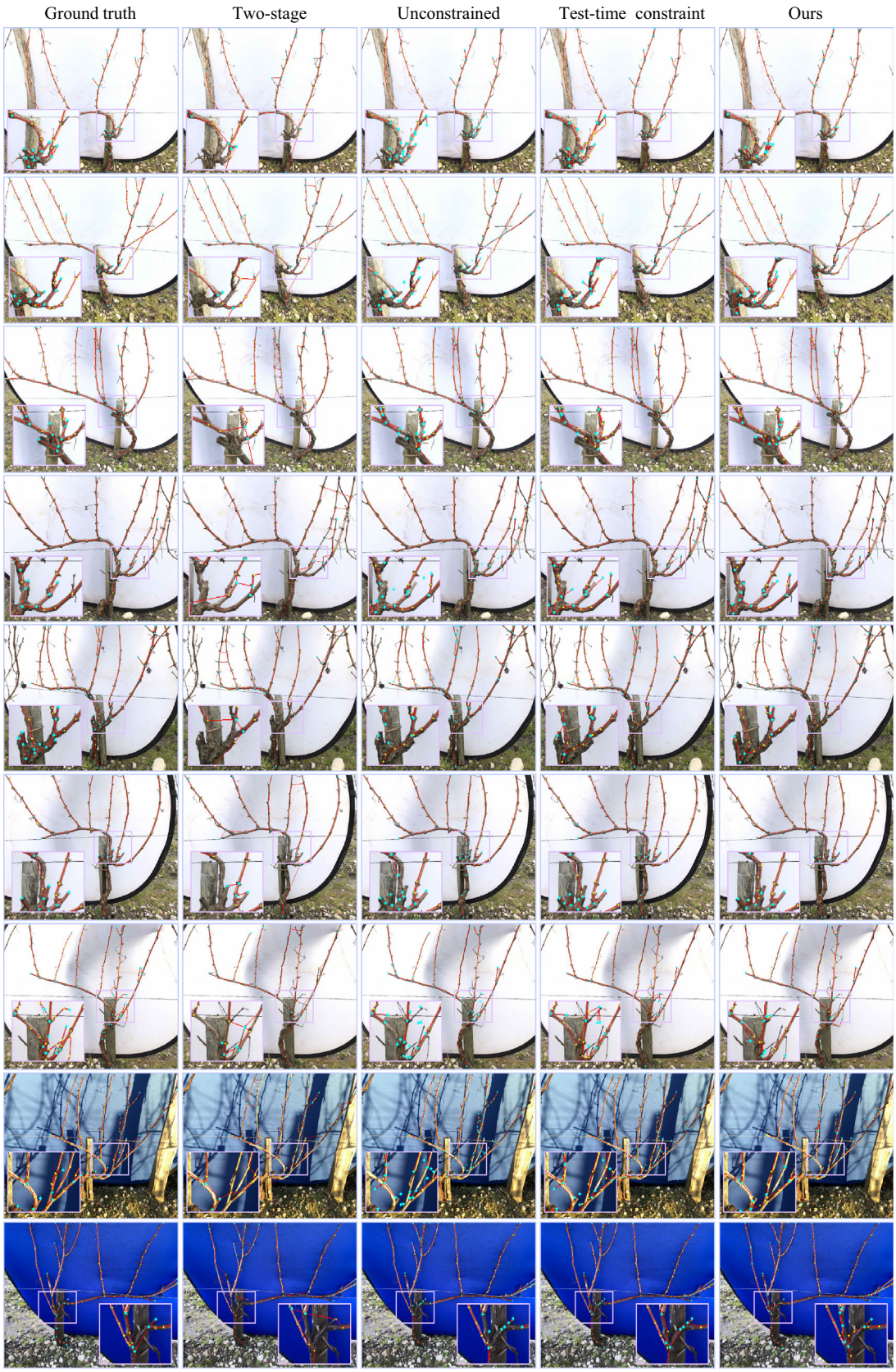


Fig. 18 Additional results for the grapevine dataset (domain-specific training, cont'd)

Table 12 Ablation for Λ

Λ	SMD ↓	TOPO score ↑		
		Prec.	Rec.	F1
2 ($\exp(-\Lambda) = 1.4 \times 10^{-1}$)	1.51×10^{-4}	0.871	0.803	0.836
5 ($\exp(-\Lambda) = 6.7 \times 10^{-3}$)	1.27×10^{-4}	0.866	0.799	0.831
10 ($\exp(-\Lambda) = 4.5 \times 10^{-5}$)	$\mathbf{1.03 \times 10^{-4}}$	**0.899**	**0.843**	**0.870**
100 ($\exp(-\Lambda) = 3.7 \times 10^{-44}$)	1.07×10^{-4}	0.886	0.830	0.857



Fig. 19 Additional results for the out-of-domain test dataset (trained on the grapevine dataset)

evaluates the model's capability to handle out-of-training-distribution data.

These results consistently demonstrate the high-fidelity estimation of plant skeletons by our PlantPose, which generalizes effectively across diverse datasets, including real-world, synthetic, and out-of-training-distribution images. Notably, the model accurately captures structural variations and environmental contexts, highlighting its robustness and generalization capability.



Fig. 20 Additional results for the grapevine dataset (generalized training)

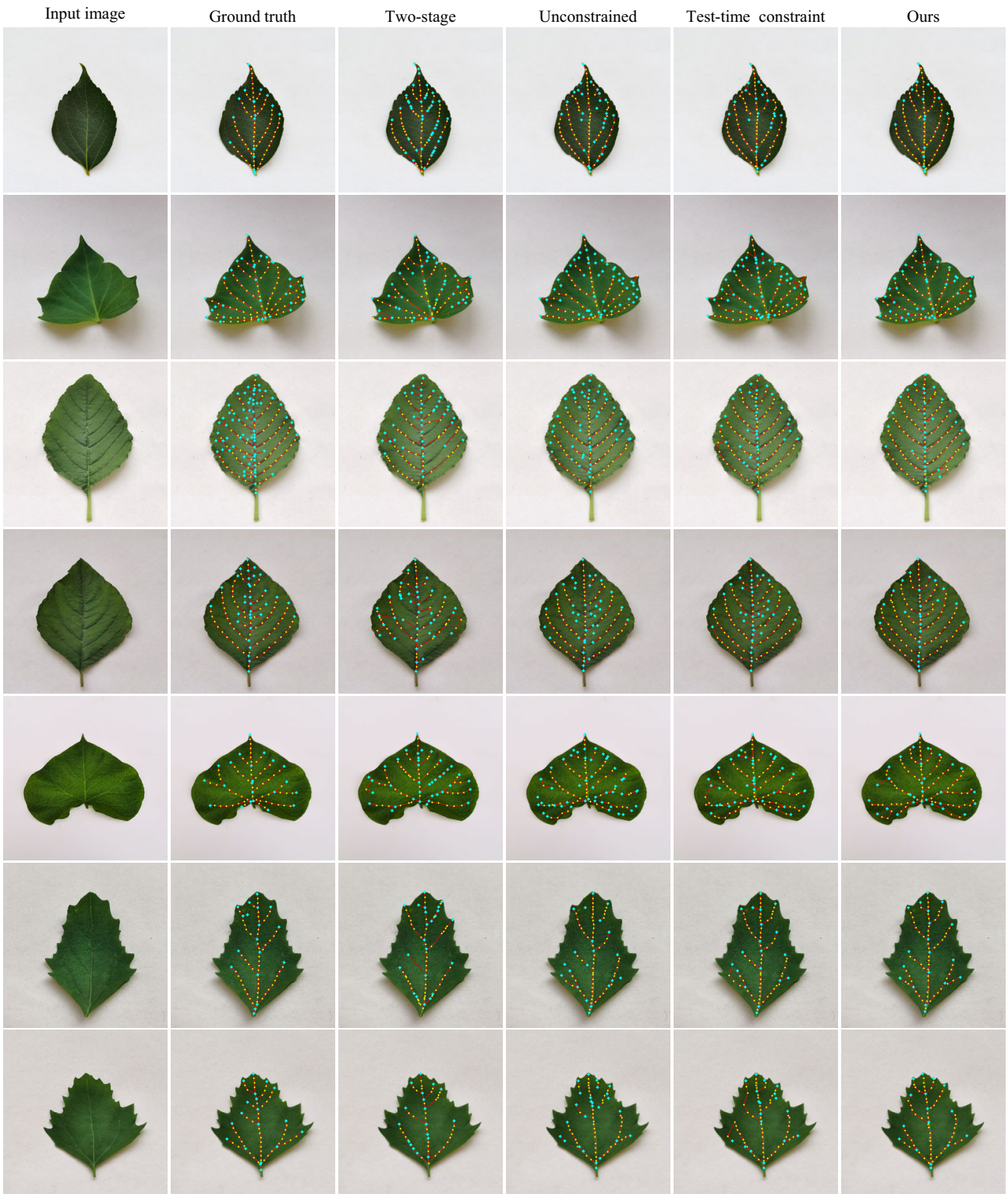


Fig. 21 Additional results for the LVD2021 leaf image dataset (generalized training)

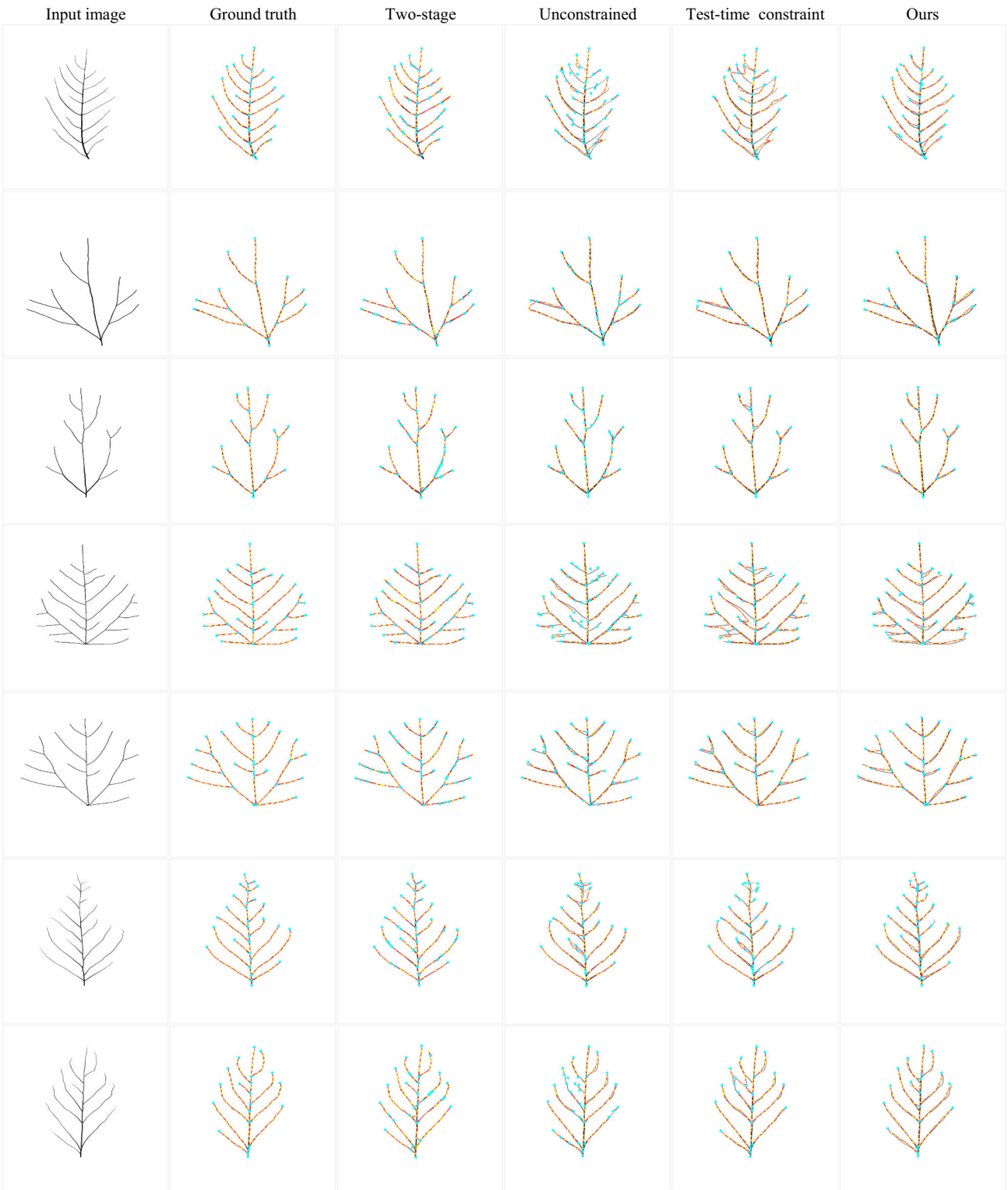


Fig. 22 Additional results for the LVD2021 leaf vein mask dataset (generalized training)

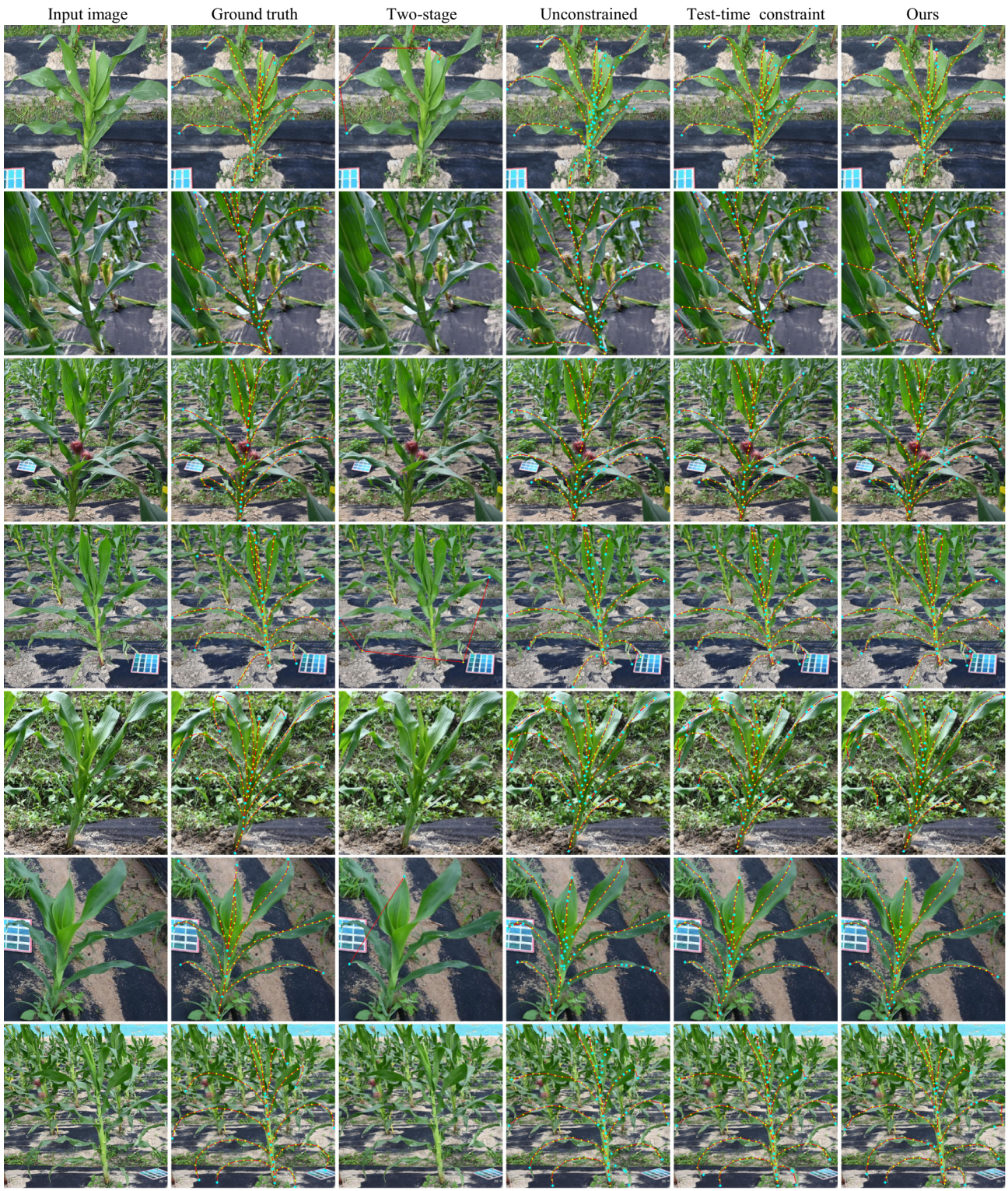


Fig. 23 Additional results for the MIPDB dataset (generalized training)

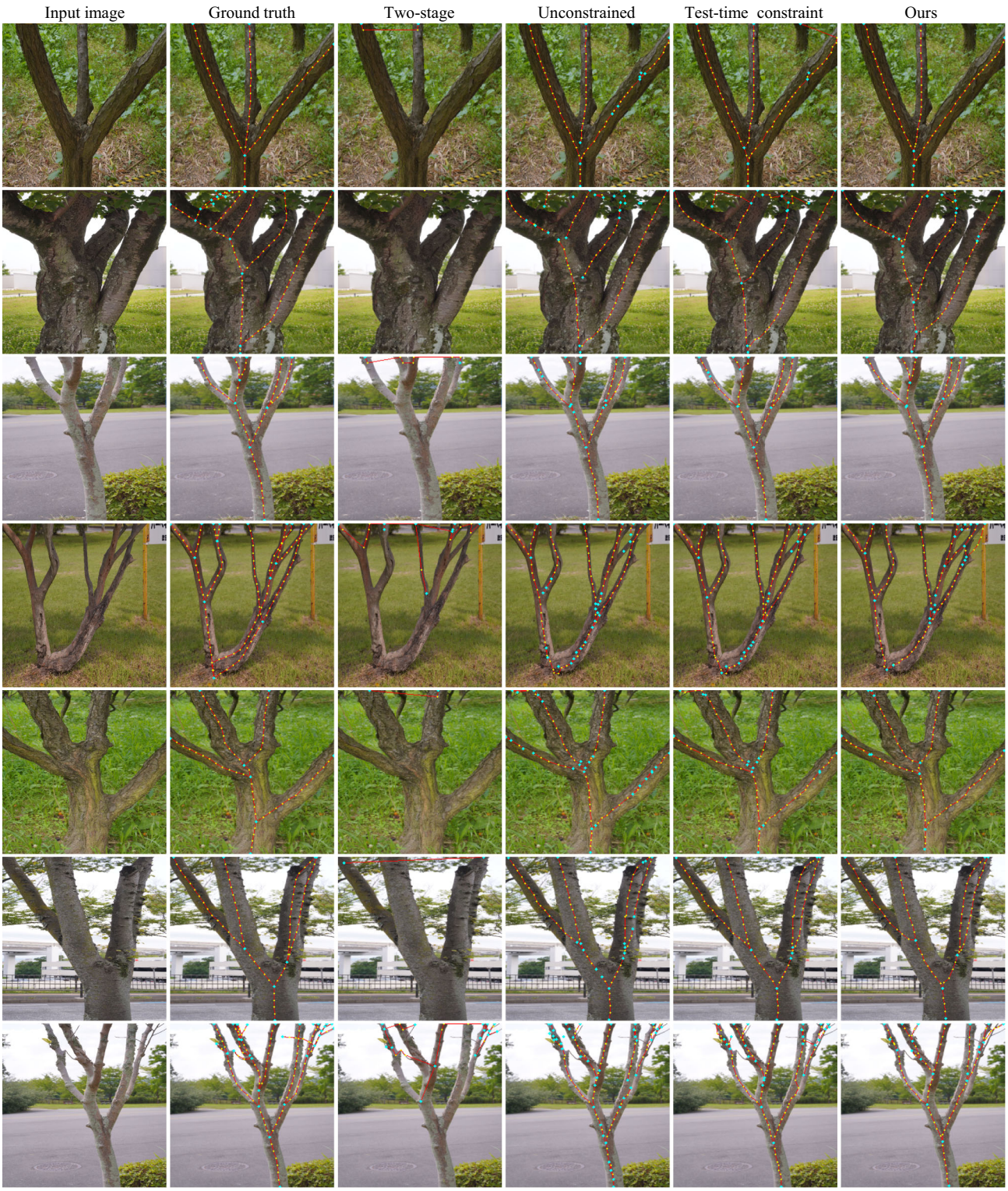


Fig. 24 Additional results for the self-captured dataset (generalized training)

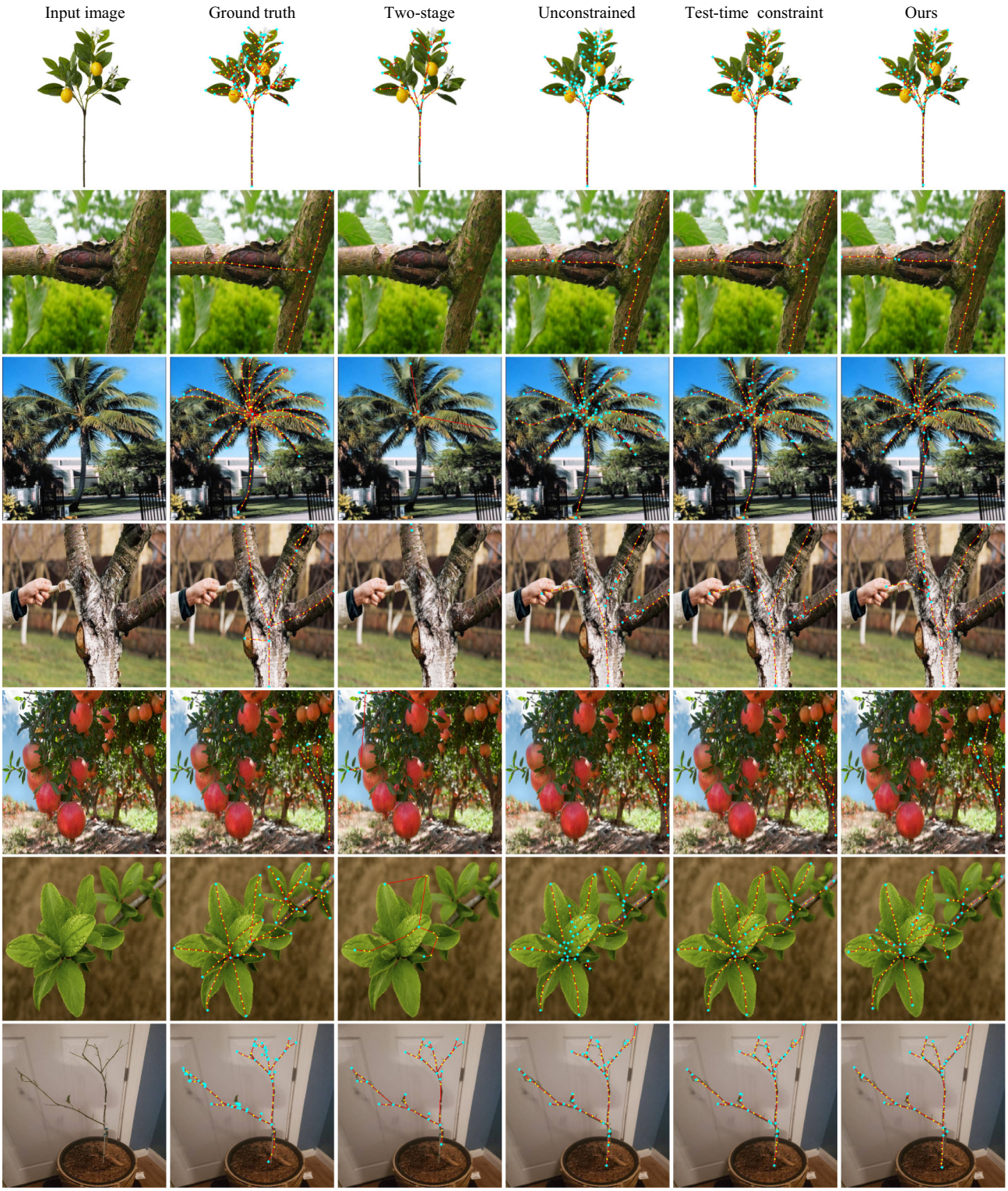


Fig. 25 Additional results for the web-sourced dataset (generalized training)

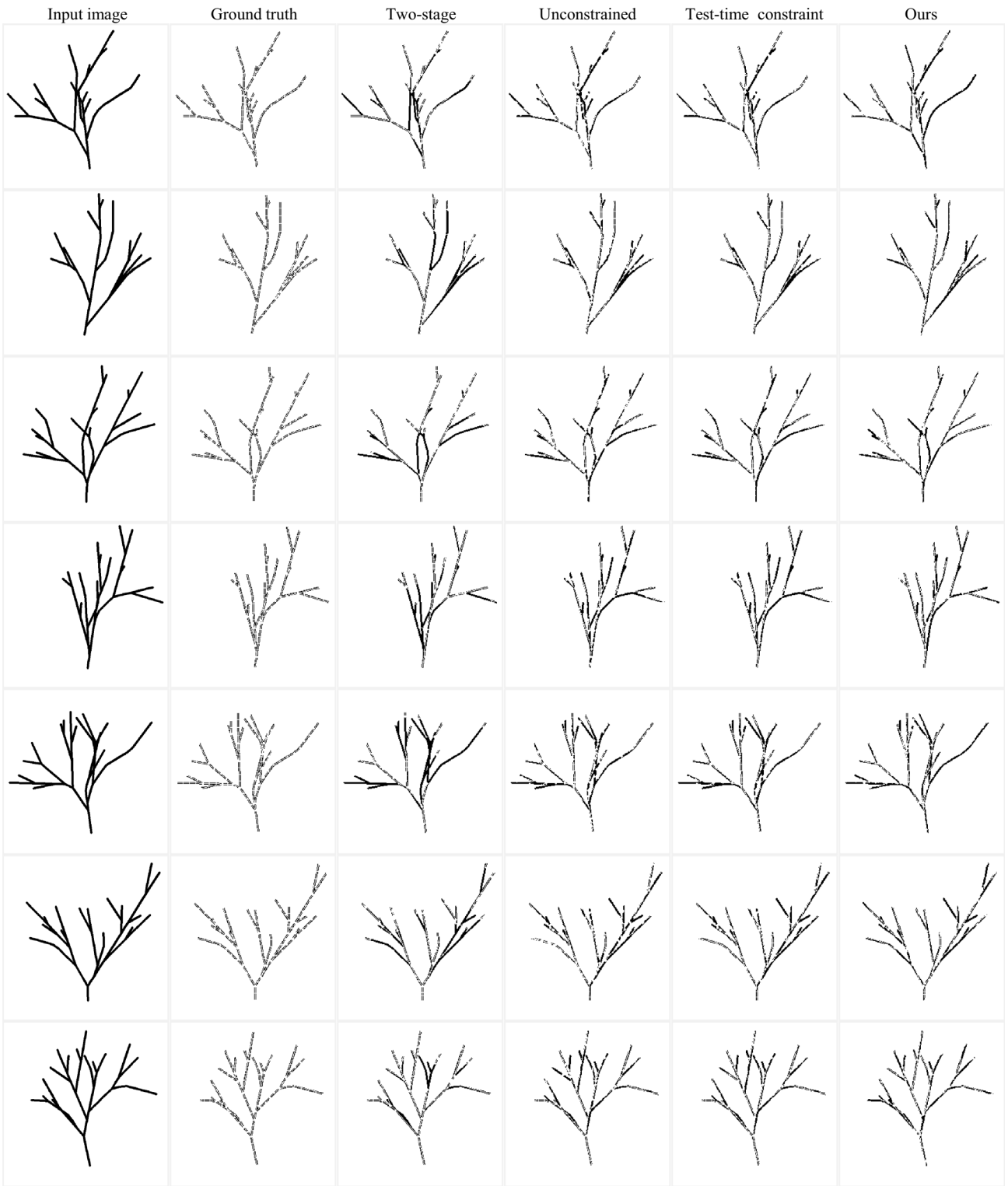


Fig. 26 Additional results for the thickened L-system dataset (generalized training)

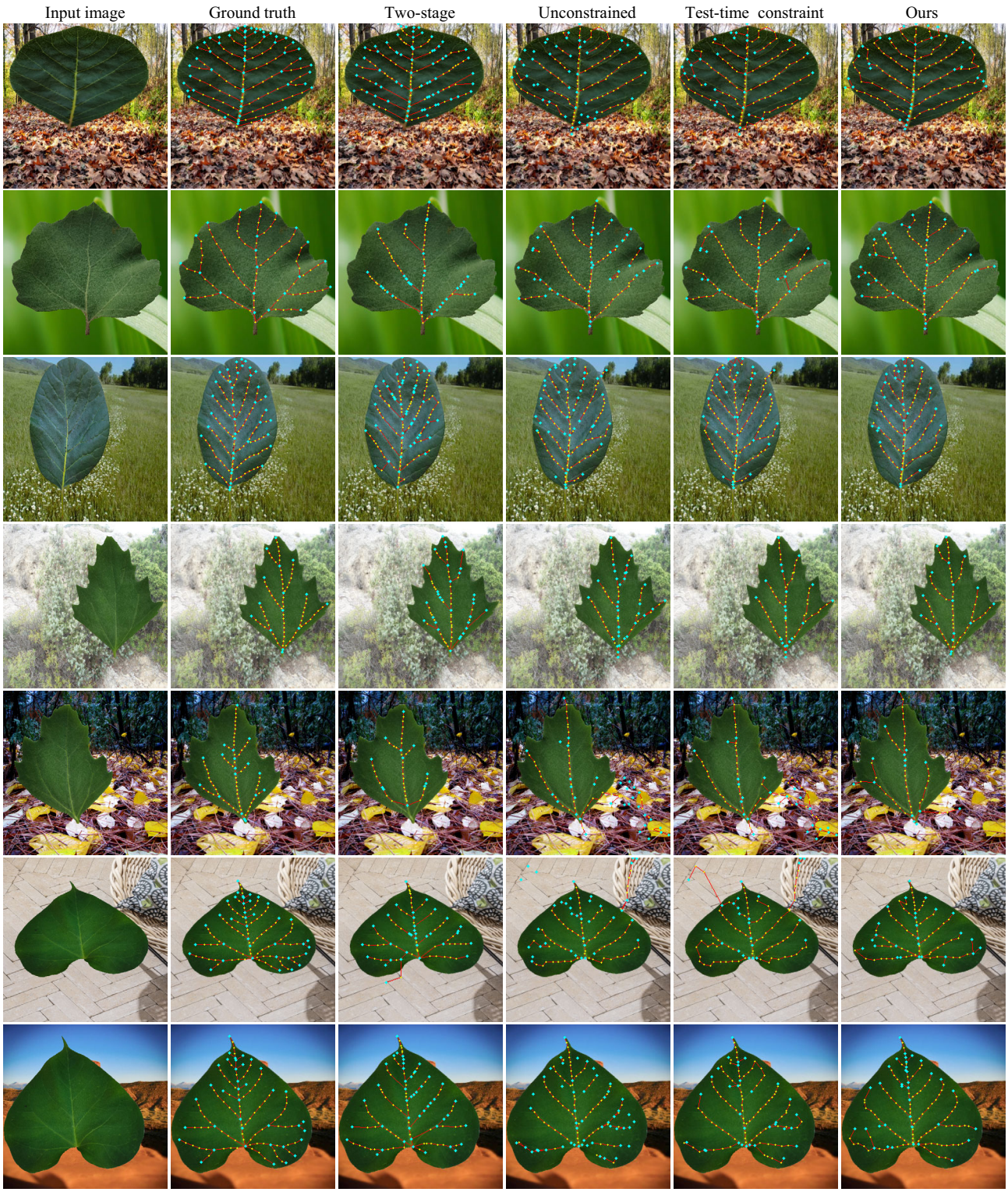


Fig. 27 Additional results for the LVD with background dataset (generalized training)

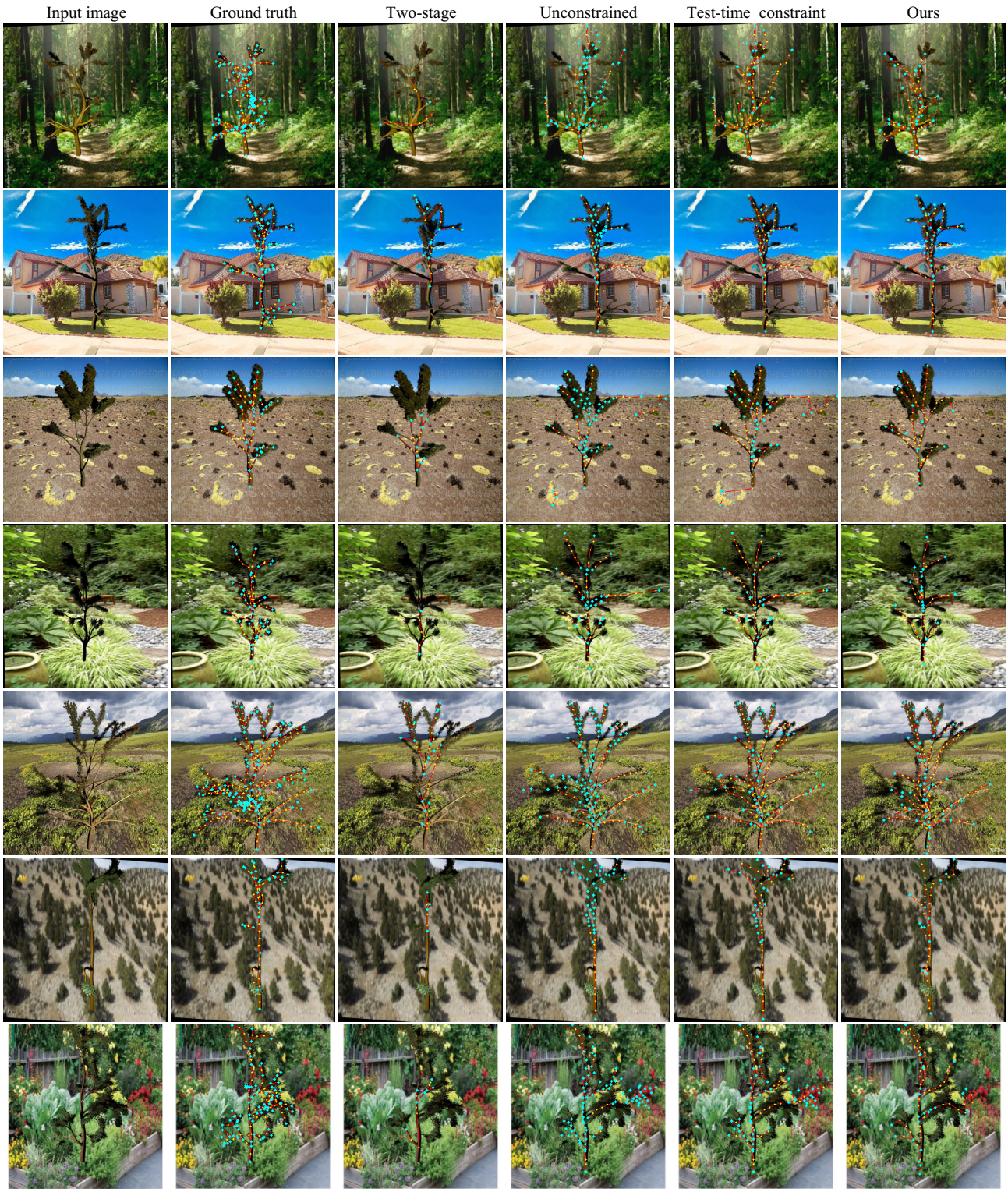


Fig. 28 Additional results for the tree synthetic dataset (generalized training)



Fig. 29 Additional results for the root dataset (generalized training)

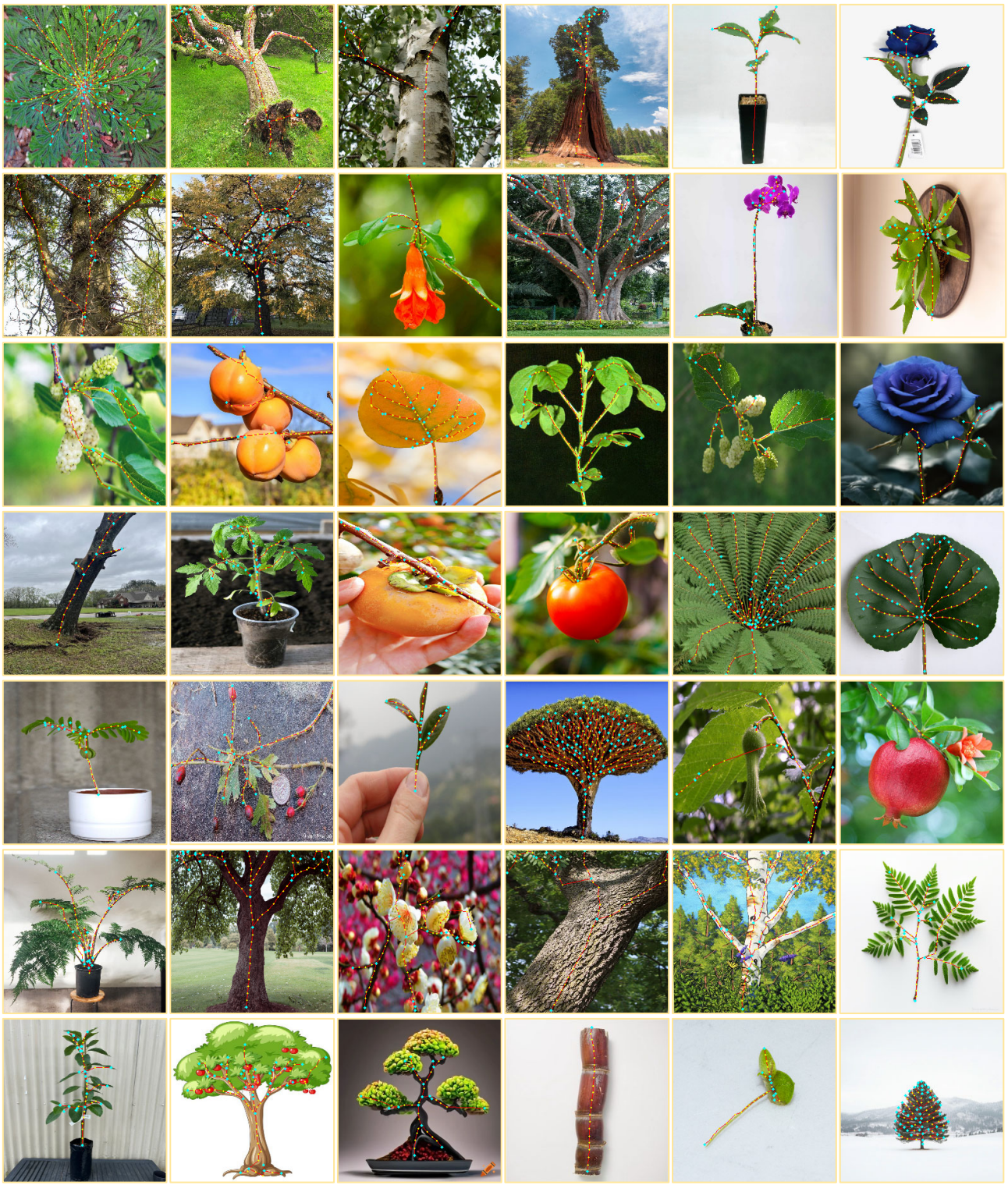


Fig. 30 Additional results for the unlabeled web test dataset (generalized training)



Fig. 31 Additional results for the unlabeled web test dataset (generalized training, cont'd)

F.3 Generalization beyond plants

While the scope is beyond the plant skeleton estimation, we further assess the generalization ability of our model for images other than plants. We tested the pre-trained generalized PlantPose model on retinal vessel images (RETA (Lyu et al., 2022) dataset). The visual results in Fig. 32 show that the majority of vessels were reconstructed as valid tree graphs, while a few thin vessels were only partially predicted. These results show the potential usability of our method and the pre-trained model for broader domains that require tree graphs, possibly with or without fine-tuning.

Appendix G Analysis of Evaluation Metrics

In the experimental results, we report SMD and TOPO scores commonly used in image-to-graph generation studies. While our method achieves promising results, it is not intuitive what each metric evaluates, which has not been investigated so far.

To briefly investigate the SMD and TOPO scores, we show an illustrative example in Fig. 33. The figure compares the cases where TOPO is similar, but SMD reveals large differences between our method and the test-time constraint baseline. In this case, the TOPO score overlooks the apparent mistake in the structure. On the other hand, SMD decreases approximately 40% by our method, which reflects the difference in the shape (implicitly including structure and position) of the reconstructed graph.

Overall, SMD is considered the key indicator of skeleton fidelity in our setup. While TOPO provides complementary evidence, it may not always evaluate structural errors adequately. We hope that providing better evaluation metrics would be an important open problem for image-to-graph generation studies.

Appendix H Details of Graph Size

We use the maximum cap of 384 nodes for the experiments. To further provide the justification and investigate the scalability of our method, we describe the details of how the graph sizes have been determined, and discuss the potential of extending our method for larger scales.

Input Resolution

We adopt a fixed input resolution of 256 × 256 across all 10 datasets to ensure reproducibility and training stability, while emphasizing *topological fidelity* (connectivity, junctions, and endpoints) rather than fine texture details. This choice is consistent with prior image-to-graph works: RelationFormer (Shit et al., 2022) uses 64 × 64 for the Toulouse road dataset and 128 × 128 for the 20 US Cities dataset, RoadTracer (Bastani et al., 2018) slices city-scale 4096 × 4096 maps into 256 × 256 tiles, and RNGDet (Xu et al., 2022), RNGDet++ Xu et al. (2023), as well as SAM-Road++ Yin et al. (2025) adopt 128–256 pixels patches as their default input.

Node Cap

We cap the maximum number of nodes per graph at 384, which is not a fundamental limitation but a standardized setting under our sampling interval of 13 pixels at 256 × 256. This interval was derived from the observed average inter-junction spacing in grapevine images (≈ 12.8 px ≈ 1/20 of the width). It corresponds to a sampling interval $s \approx 13$ pixels derived from the observed mean inter-junction spacing in grapevine (≈ 12.8 px ≈ 1/20 of the width). Under uniform density, the capacity per tile scales as

$$N_{max}(H, W, s) \approx \left\lfloor \frac{H}{s} \right\rfloor \cdot \left\lfloor \frac{W}{s} \right\rfloor, \tag{H11}$$

resulting in s=13 at H=W=256, giving ≈ 3.88×10^2 (rounded to 384 for scheduling).

Scalability

Figure 34 shows how graphs of different sizes appear in our datasets. From left to right, the examples become progressively more complex: Starting with simple grass-like plants or single trunks (∼50 nodes), then adding visible full plant structures (∼100–200 nodes), and finally reaching fully developed trees with dense branches (∼300–350 nodes). This trend reflects our dataset statistics: The average node count in the *Self-captured dataset* is 156, while in the *Web-sourced dataset* is 96. When node count grows, the cost of pairwise edge classification increases quadratically ($O(N^2)$), leading to higher memory and runtime. Future work may address this by (i) pruning candidate edges to local neighborhoods, (ii) tiling and stitching subgraphs, or (iii) using memory-efficient attention mechanisms (*e.g.*, FlashAttention (Dao, 2024)) to handle denser graphs without altering our formulation.

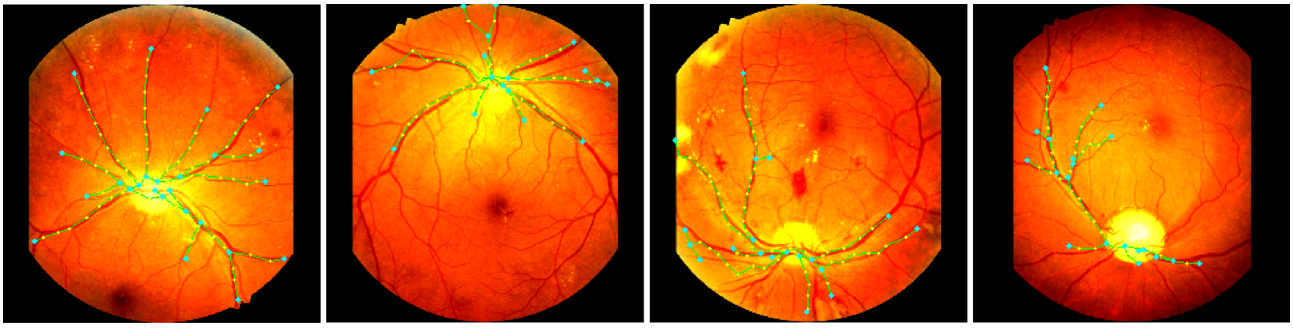


Fig. 32 Qualitative results on retinal vessel images. PlantPose, trained only on plant skeletons, recovers most major vessel trees after simple preprocessing (contrast adjustment and rotation)

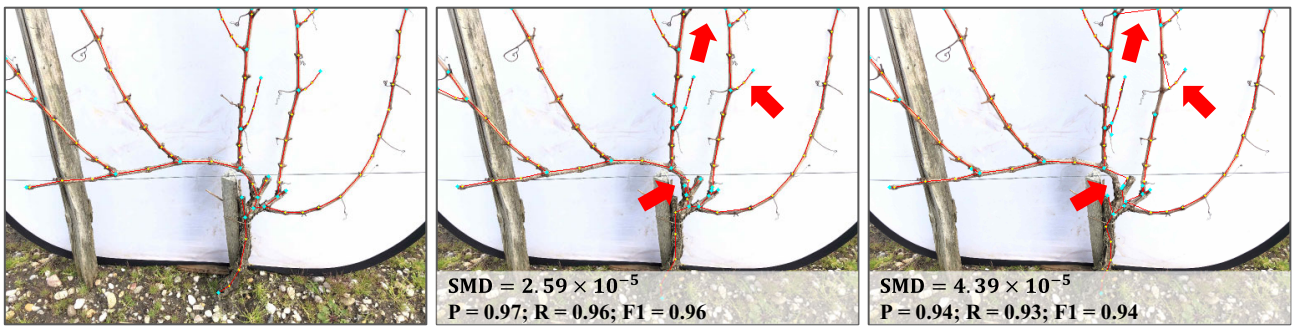


Fig. 33 Comparison of our method and test-time MST constraint under generalized training. Both predictions obtain similar TOPO scores ($F1 = 0.94$ vs. 0.96), indicating comparable reachability. However, as highlighted by red arrows, the baseline introduces several spurious or misplaced edges, which are not penalized by TOPO but cause a substantial increase in SMD (almost 2× higher). This example demonstrates that SMD is a more discriminative measure of structural fidelity, capturing global geometric deviations that TOPO may overlook

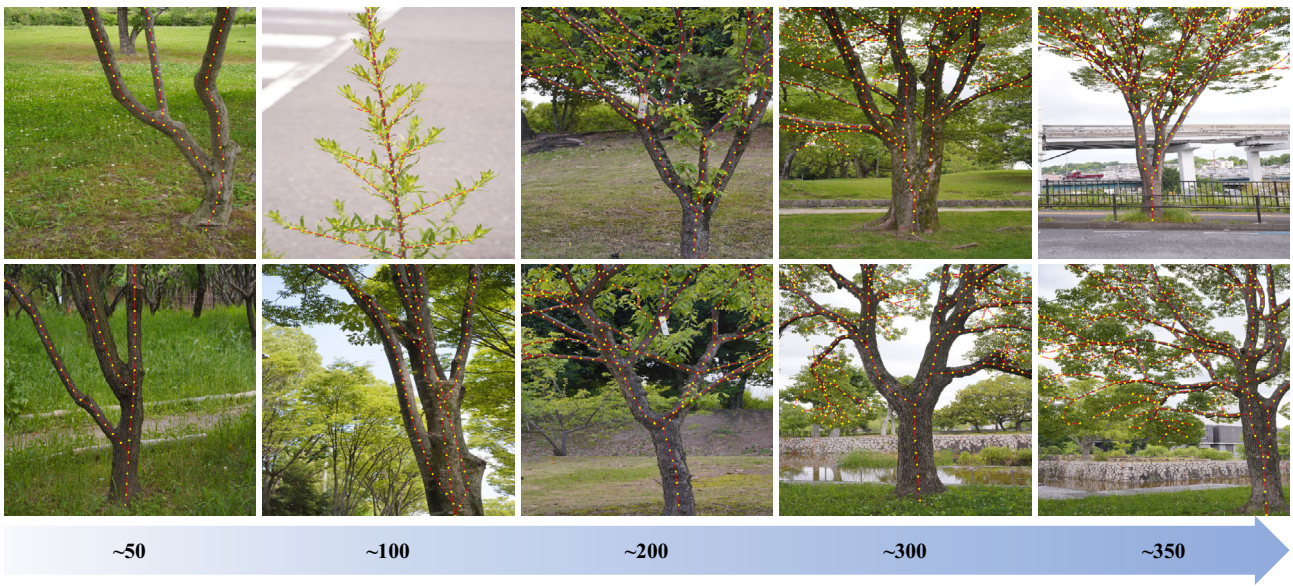


Fig. 34 Illustration of scalability with respect to node count. From left to right: examples with ∼50, ∼100, ∼200, ∼300, and ∼350 nodes. Simpler graphs (grass-like or single-trunk plants) have sparse nodes, while more complex trees include dense branching structures

Acknowledgements We thank Professor Yasuyuki Matsushita for insightful discussions throughout the study. We also thank Momoko Takagi, Manami Okazaki, and Professor Kei Hiruma for providing us with root images.

Author Contributions **Conceptualization:** Xinpeng Liu, Fumio Okura. **Methodology:** Xinpeng Liu, Fumio Okura. **Formal analysis and investigation:** Xinpeng Liu. **Writing – original draft preparation:** Xinpeng Liu, Hiroaki Santo, Fumio Okura. **Writing – review and editing:** Xinpeng Liu, Fumio Okura. **Funding acquisition:** Fumio Okura. **Resources:** Yosuke Toda. **Supervision:** Fumio Okura.

Funding Open Access funding provided by The University of Osaka This work was partly supported by JSPS KAKENHI Grant Numbers JP23H05491, JP25K03140, and JP25K00142, and JST FOREST Grant Number JPMJFR206F.

Data Availability Our own data used and/or analyzed during the current study is publicly available in the repository at https://github.com/huntorochi/PlantPose. The curated datasets can be provided by original authors.

Code Availability The source code supporting the findings of this study is publicly available in the repository at https://github.com/huntorochi/PlantPose.

Declarations

Competing interests The authors declare that they have no competing interests.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

Abdelkarim, S., Agarwal, A., Achlioptas, P., Chen, J., Huang, J., Li, B., Church, K. W., & Elhoseiny, M. (2020). Exploring long tail visual relationship recognition with large vocabulary. *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)* pp. 15901–15910.

Acuna, D., Ling, H., Kar, A., & Fidler, S. (2018). Efficient interactive annotation of segmentation datasets with polygon-RNN. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* (pp. 859–868)

Agrawal, A., Amos, B., Barratt, S., Boyd, S., Diamond, S., & Kolter, J. Z. (2019). Differentiable convex optimization layers. Advances in Neural Information Processing Systems (NeurIPS) vol. 32.

Ahn, S., Chen, B., Wang, T., & Song, L. (2022). Spanning tree-based graph generation for molecules. Proceedings of International Conference on Learning Representations (ICLR).

Ai, M., Yao, Y., Hu, Q., Wang, Y., & Wang, W. (2020). An automatic tree skeleton extraction approach based on multi-view slicing using terrestrial LiDAR scans data. Remote Sensing **12**(22).

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization arXiv preprint arXiv:1607.06450.

Bastani, F., He, S., Abbar, S., Alizadeh, M., Balakrishnan, H., Chawla, S., Madden, S., & DeWitt, D. (2018). Roadtracer: Automatic extraction of road networks from aerial images. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 4720–4728.

Belli, D., & Kipf, T. (2019). Image-conditioned graph generation for road network extraction. Proceedings of NeurIPS Workshop on Graph Representation Learning.

Bohlender, S., Öksüz, I., & Mukhopadhyay, A. (2021). A survey on shape-constraint deep learning for medical image segmentation. *IEEE Reviews in Biomedical Engineering, 16*, 225–240.

Bucksch, A. (2014). A practical introduction to skeletons for the plant sciences. *Applications in Plant Sciences, 2*(8), 1400005.

Cabrera-Bosquet, L., Fournier, C., Brichet, N., Welcker, C., Suard, B., & Tardieu, F. (2016). High-throughput estimation of incident light, light interception and radiation-use efficiency of thousands of plants in a phenotyping platform. *New Phytologist, 212*(1), 269–281.

Cao, Z., Simon, T., Wei, S.-E., & Sheikh, Y. (2017). Realtime multi-person 2D pose estimation using part affinity fields. Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)

Chaudhury, A., & Godin, C. (2020). Skeletonization of plant point cloud data using stochastic optimization framework. Frontiers in Plant Science, *11*.

Chiou, M.-J., Ding, H., Yan, H., Wang, C., Zimmermann, R., & Feng, J. (2021). Recovering the unbiased scene graphs from the biased ones. *Proceedings of ACM International Conference on Multimedia (MM)* pp. 1581–1590.

Cong, Y., Yang, M., & Rosenhahn, B. (2023). RelTR: Relation transformer for scene graph generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI), 45*(9), 11169–11183.

Dao, T. (2024). FlashAttention-2: Faster attention with better parallelism and work partitioning. Proceedings of International Conference on Learning Representations (ICLR).

Du, S., Lindenbergh, R., Ledoux, H., Stoter, J., & Nan, L. (2019). AdTree: Accurate, detailed, and automatic modelling of laser-scanned trees. Remote Sensing, *11*(18).

Ferber, A., Wilder, B., Dilkina, B., & Tambe, M. (2020). Mipaal: Mixed integer program as a layer. *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)* pp. 1504–1511.

Gaillard, M., Miao, C., Schnable, J., & Benes, B. (2020). Sorghum segmentation by skeleton extraction. Proceedings of European Conference on Computer Vision (ECCV) Workshops

Gentilhomme, T., Villamizar, M., Corre, J., & Odobez, J.-M. (2023). Towards smart pruning: ViNet, a deep-learning approach for grapevine structure estimation. *Computers and Electronics in Agriculture, 207*, Article 107736.

Giuffrida, V., Minervini, M., & Tsaftaris, S. (2015). Learning to count leaves in rosette plants. *Proceedings of Workshop on Computer Vision Problems in Plant Phenotyping (CVPPP)* pp. 1–1113.

Guo, J., Jiang, H., Benes, B., Deussen, O., Zhang, X., Lischinski, D., & Huang, H. (2020). Inverse procedural modeling of branching structures by inferring L-systems. *ACM Transactions on Graphics (TOG), 39*(5), 1–13.

Gupta, H., Jin, K. H., Nguyen, H. Q., McCann, M. T., & Unser, M. (2018). CNN-based projected gradient descent for consistent CT image reconstruction. *IEEE Transactions on Medical Imaging, 37*(6), 1440–1453.

He, S., Bastani, F., Abbar, S., Alizadeh, M., Balakrishnan, H., Chawla, S., & Madden, S. (2018). RoadRunner: Improving the precision

of road network inference from gps trajectories. *Proceedings of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems* pp. 3–12.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 770–778.

He, S., Bastani, F., Jagwani, S., Alizadeh, M., Balakrishnan, H., Chawla, S., Elshrif, M. M., Madden, S., & Sadeghi, M. A. (2020). Sat2Graph: Road graph extraction through graph-tensor encoding. *Proceedings of European Conference on Computer Vision (ECCV)* (pp. 51–67)

Huang, H., Wu, S., Cohen-Or, D., Gong, M., Zhang, H., Li, G., & Chen, B. (2013). L1-medial skeleton of point cloud. *ACM Transactions on Graphics (TOG), 32*(4), 65.

Isokane, T., Okura, F., Ide, A., Matsushita, Y., & Yagi, Y. (2018). Probabilistic plant modeling via multi-view image-to-image translation. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 2906–2915.

Jin, W., Barzilay, R., & Jaakkola, T. (2018). Junction tree variational autoencoder for molecular graph generation. *Proceedings of International Conference on Machine Learning (ICML)* pp. 2323–2332.

Jin, W., Barzilay, R., & Jaakkola, T. (2020). Hierarchical generation of molecular graphs using structural motifs. Proceedings of International Conference on Machine Learning (ICML).

Khokhlov, I., Krasnov, L., Fedorov, M. V., & Sosnin, S. (2022). Image2SMILES: Transformer-based molecular optical recognition engine. *Chemistry-Methods, 2*(1), 202100069.

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. In Y. Bengio & Y. LeCun (Eds.), Proceedings of International Conference on Learning Representations (ICLR).

Kingma, D. P., & Welling, M. (2014). Auto-encoding variational bayes. Proceedings of International Conference on Learning Representations (ICLR).

Kipf, T. N., & Welling, M. (2016). Variational graph auto-encoders. Proceedings on NeurIPS Workshop on Bayesian Deep Learning.

Kotary, J., Fioretto, F., Van Hentenryck, P., & Wilder, B. (2021). End-to-end constrained optimization learning: A survey. Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI)

Kruskal, J. B. (1956). On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society, 7*(1), 48–50.

Li, L., Hu, W., Lu, J., & Zhang, C. (2022). Leaf vein segmentation with self-supervision. Computers and Electronics in Agriculture.

Li, W., Lu, Y., Zheng, K., Liao, H., Lin, C., Luo, J., Cheng, C.-T., Xiao, J., Lu, L., Kuo, C.-F., et al. (2020). Structured landmark detection via topology-adapting deep graph learning. In: Proceedings of European Conference on Computer Vision (ECCV), pp. 266–283.

Li, Y., Vinyals, O., Dyer, C., Pascanu, R., & Battaglia, P. (2018). Learning deep generative models of graphs arXiv preprint arXiv:1803.03324.

Li, R., Zhang, S., & He, X. (2022). SGTR: End-to-end scene graph generation with transformer. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 19486–19496.

Li, L.H., Zhang, P., Zhang, H., Yang, J., Li, C., Zhong, Y., Wang, L., Yuan, L., Zhang, L., Hwang, J.-N., et al. (2022). Grounded language-image pre-training. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 10965–10975.

Li, W., Zhao, W., Zhong, H., He, C., & Lin, D. (2021). Joint semantic-geometric learning for polygonal building segmentation. In: Proceedings of AAAI Conference on Artificial Intelligence (AAAI), pp. 1958–1965.

Liang, J., Homayounfar, N., Ma, W.-C., Xiong, Y., Hu, R., & Urtasun, R. (2020). PolyTransform: Deep polygon transformer for instance segmentation. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 9128–9137.

Li, Y., Ho, C. P., Toulemonde, M., Chahal, N., Senior, R., & Tang, M.-X. (2017). Fully automatic myocardial segmentation of contrast echocardiography sequence using random forests guided by shape model. *IEEE Transactions on Medical Imaging, 37*(5), 1081–1091.

Lin, T.-Y., Dollár, P., Girshick, R., He, K., Hariharan, B., & Belongie, S. (2017). Feature pyramid networks for object detection. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 2117–2125.

Lindenmayer, A. (1968). Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology, 18*(3), 280–299.

Ling, H., Gao, J., Kar, A., Chen, W., & Fidler, S. (2019). Fast interactive object annotation with curve-GCN. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 5252–5261.

Liu, Q., Allamanis, M., Brockschmidt, M., & Gaunt, A. L. (2018). Constrained graph variational autoencoders for molecule design. Advances in Neural Information Processing Systems (NeurIPS).

Liu, X., Santo, H., Toda, Y., & Okura, F. (2025). TreeFormer: Single-view plant skeleton estimation via tree-constrained graph generation. *Proceedings of IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)* pp. 8154–8164. IEEE.

Luo, Y., Yan, K., & Ji, S. (2021). Graphdf: A discrete flow model for molecular graph generation. Proceedings of International Conference on Machine Learning (ICML).

Lyu, X., Cheng, L., & Zhang, S. (2022). The reta benchmark for retinal vascular tree analysis. *Scientific Data, 9*(1), 397.

Madry, A., Makelov, A., Schmidt, L., Tsipras, D., & Vladu, A. (2018). Towards deep learning models resistant to adversarial attacks. Proceedings of International Conference on Learning Representations (ICLR).

Miao, T., Zhu, C., Xu, T., Yang, T., Li, N., Zhou, Y., & Deng, H. (2021). Automatic stem-leaf segmentation of maize shoots using three-dimensional point cloud. *Computers and Electronics in Agriculture, 187*, Article 106310.

Myung, Y.-S., Lee, C.-H., & Tcha, D.-W. (1995). On the generalized minimum spanning tree problem. *Networks, 26*(4), 231–241.

Newell, A., Yang, K., & Deng, J. (2016). Stacked hourglass networks for human pose estimation. *Proceedings of European Conference on Computer Vision (ECCV)* pp. 11–16.

Okura, F. (2022). 3D modeling and reconstruction of plants and trees: A cross-cutting review across computer graphics, vision, and plant phenotyping. *Breeding Science, 72*(1), 31–47.

Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global vectors for word representation. *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)* pp. 1532–1543.

Pop, P. C. (2020). The generalized minimum spanning tree problem: An overview of formulations, solution procedures and latest advances. *European Journal of Operational Research, 283*(1), 1–15.

Sharifzadeh, S., Baharlou, S. M., Schmitt, M., Schütze, H., & Tresp, V. (2022). Improving scene graph classification by exploiting knowledge from texts. *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)* pp. 2189–2197.

Sheng, w., Wen, W., Xiao, B., Guo, X., Du, J.J., Wang, C., & Wang, Y. (2019). An accurate skeleton extraction approach from 3d point clouds of maize plants. Frontiers in Plant Science 10, 248.

Shit, S., Koner, R., Wittmann, B., Paetzold, J., Ezhov, I., Li, H., Pan, J., Sharifzadeh, S., Kaissis, G., & Tresp, V. (2022). Relationformer: A unified framework for image-to-graph generation. *Proceedings of European Conference on Computer Vision (ECCV)* pp. 422–439.

Tang, K., Zhang, H., Wu, B., Luo, W., & Liu, W. (2019). Learning to compose dynamic tree structures for visual contexts. *Proceedings*

of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 6619–6628.

Vineet, V., Harish, P., Patidar, S., & Narayanan, P. (2009). Fast minimum spanning tree for large graphs on the GPU. *Proceedings of Conference on High Performance Graphics (HPG)* pp. 167–171.

Wang, P., Chang, J., Deng, W., Liu, B., Lai, H., Hou, Z., Dong, L., Chen, Q., Zhou, Y., Zhang, Z., Liu, H., & Ruan, J. (2024). MIPDB: A maize image-phenotype database with multi-angle and multi-time characteristics. bioRxiv.

Wilder, B., Dilkina, B., & Tambe, M. (2019). Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)* pp. 1658–1665.

Wu, S., Wen, W., Wang, Y., Fan, J., Wang, C., Gou, W., & Guo, X. (2020). MVS-Pheno: A portable and low-cost phenotyping platform for maize shoots using multiview stereo 3D reconstruction. Plant Phenomics *2020*.

Xie, C., Tan, M., Gong, B., Wang, J., Yuille, A. L., & Le, Q. V. (2020). Adversarial examples improve image recognition. Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR).

Xu, Z., Liu, Y., Sun, Y., Liu, M., & Wang, L. (2023). Rngdet++: Road network graph detection by transformer with instance segmentation and multi-scale features enhancement. arXiv preprint arXiv:2209.10150.

Xu, Z., Liu, Y., Sun, Y., Liu, M., & Wang, L. (2023). RNGDet++: Road network graph detection by transformer with instance segmentation and multi-scale features enhancement. IEEE Robotics and Automation Letters, 1–8.

Xu, D., Zhu, Y., Choy, C. B., & Fei-Fei, L. (2017). Scene graph generation by iterative message passing. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 5410–5419.

Xu, H., Gossett, N., & Chen, B. (2007). Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Transactions on Graphics (TOG), 26*(4), 19.

Xu, Z., Liu, Y., Gan, L., Sun, Y., Wu, X., Liu, M., & Wang, L. (2022). Rngdet: Road network graph detection by transformer in aerial images. *IEEE Transactions on Geoscience and Remote Sensing, 60*, 1–12.

Xu, Z., Sun, Y., & Liu, M. (2021). iCurb: Imitation learning-based detection of road curbs using aerial images for autonomous driving. *IEEE Robotics and Automation Letters, 6*, 1097–1104.

Yang, J., Ang, Y. Z., Guo, Z., Zhou, K., Zhang, W., & Liu, Z. (2022). Panoptic scene graph generation. *Proceedings of European Conference on Computer Vision (ECCV)* pp. 178–196.

Yin, P., Li, K., Cao, X., Yao, J., Liu, L., Bai, X., Zhou, F., & Meng, D. (2025). Towards satellite image road graph extraction: A global-scale dataset and a novel method. In: Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), pp. 1527–1537.

Yoshida, M., Torii, A., Okutomi, M., Endo, K., Sugiyama, Y., Taniguchi, R.-I., & Nagahara, H. (2018). Joint optimization for compressive video sensing and reconstruction under hardware constraints. *Proceedings of European Conference on Computer Vision (ECCV)* pp. 634–649.

Zang, C., & Wang, F. (2020). Moflow: A invertible flow model for generating molecular graphs. In: Proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD), pp. 617–626.

Zhang, Y., Pan, Y., Yao, T., Huang, R., Mei, T., & Chen, C.-W. (2023). Learning to generate language-supervised and open-vocabulary scene graph using pre-trained visual-semantic space. *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)* pp. 2915–2924.

Zhong, Y., Shi, J., Yang, J., Xu, C., & Li, Y. (2021). Learning to generate scene graph from natural language supervision. *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)* pp. 1823–1834.

Zhou, X., Li, B., Benes, B., Fei, S., & Pirk, S. (2023). Deeptree: Modeling trees with situated latents. IEEE Transactions on Visualization and Computer Graphics.

Zhu, X., Su, W., Lu, L., Li, B., Wang, X., & Dai, J. (2021). Deformable DETR: Deformable transformers for end-to-end object detection. Proceedings of International Conference on Learning Representations (ICLR).

Ziamtsov, I., & Navlakha, S. (2019). Machine learning approaches to improve three basic plant phenotyping tasks using three-dimensional point clouds. *Plant Physiology, 181*(4), 1425–1440.

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.