TreeFormer: Single-view Plant Skeleton Estimation via Tree-constrained Graph Generation

Xinpeng Liu[1] Hiroaki Santo[1] Yosuke Toda[2,3] Fumio Okura[1]
[1]Osaka University [2]Phytometrics [3]Nagoya University
{liu.xinpeng,santo.hiroaki,okura}@ist.osaka-u.ac.jp yosuke@phytometrics.jp

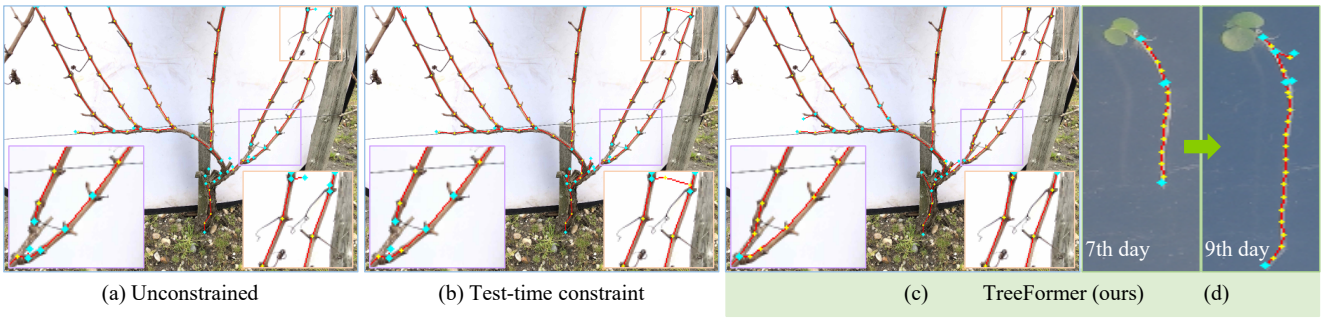


Figure 1. We propose a method for single-image plant skeleton estimation combining learning-based graph generators with traditional graph algorithm (*i.e.*, MST). The red lines show the predicted graph edges. Compared to (a) an *unconstrained* graph generator and (b) a naive tree-graph constraint implementation, (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.

Abstract

Accurate estimation of plant skeletal structure (e.g., *branching structure) 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. While recent graph generation methods successfully infer thin structures from images, it is challenging to constrain the output graph strictly to a tree structure. To this problem, we present TreeFormer, a plant skeleton estimator via tree-constrained graph generation. Our approach combines learning-based graph generation with traditional graph algorithms to impose the constraints during the training loop. Specifically, our method* projects *an unconstrained graph onto a minimum spanning tree (MST) during the training loop and incorporates this prior knowledge into the gradient descent optimization by suppressing unwanted feature values. Experiments show that our method accurately estimates target plant skeletal structures for multiple domains: Synthetic tree patterns, real botanical roots, and grapevine branches. Our implementations are available at* https://github.com/huntorochi/TreeFormer/.

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 [10, 18, 51] and plant organ segmentation [17, 45]. As a similar task, single-view estimation of human poses has been widely studied, *e.g.*, OpenPose [11]. 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 [23, 58, 59]. Recent end-to-end models using recurrent neural networks (RNNs) [2], graph neural networks (GNNs) [35, 36, 41], or transformers [7, 28, 34, 39, 52] 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. 1(a). One reason for this difficulty is that tree graph generation, which

requires finding a set of graph edges that satisfy the constraint defined on the entire graph, naturally falls into combinatorial optimization. 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. 1(b).

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. 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 TreeFormer, which infers tree structures from images capturing plants. We evaluate the effectiveness of TreeFormer on different classes of plant images: Synthetic tree patterns, real-world root, and grapevine branch images. The results show that our constraint-aware graph generator accurately estimates the target tree structures compared to baselines.

Contributions Our contributions are twofold: First, 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. Second, building upon our constrained graph generation method, we develop TreeFormer, the first end-to-end method inferring skeletal structures from a single plant image, which benefits the agriculture and plant science field.

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 [47].

3D plant skeleton estimation Several methods are proposed to derive plant skeletons from 3D observations [47]. These methods often use point clouds acquired by LiDAR [5, 57] or multi-view stereo (MVS) [51, 55]. Regardless of the 3D acquisition method, these works generally use a two-stage pipeline: Skeletonization [9] followed by graph optimization using MST or Dijkstra's algorithm [12, 15, 24, 45, 65], 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 [19]. Similarly, a neural network that converts an input image into a map representing 2D skeleton positions is used to mitigate the occlusions [25]. To reason about the direction of intersecting branches, a recent work [18] proposes to use vector fields representing branch direction instead of mask images, similar to the Part Affinity Fields (PAFs) used in OpenPose [11].

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 thin structures (*e.g.*, road networks) or relations (*e.g.*, scene graphs) from images [2, 7, 14, 23, 28, 34–36, 39, 41, 52, 58, 59]. Recent learning-based methods 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 [1, 13, 48, 50] to improve the results.

Graph generators have usually taken autoregressive methods (*e.g.*, [7, 38, 42, 43, 63]) that output a graph by starting at an initial node and estimating neighboring graph nodes. Recent GNNs and transformers enable non-autoregressive graph generators [14, 29, 31, 34, 60, 62] 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 [52], 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 [4, 26, 27], these methods assume autoregressive graph generation, making it hard to work with complex and relatively large graphs like botanical plants.

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 [32].

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 [3]. However, the design of differentiable layers for combinatorial optimization poses a significant challenge due to the difficulty of differentiation. Wilder *et al*. [54] propose a differentiable layer for linear programming (LP) problems using continuous relaxation. This method is extended to mixed integer linear programming (MILP) [16] 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 [46, 49]. However, using differentiable layers for these complex combinatorial problems requires exponential computation time [32] 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 [30].

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 [21, 56]. While PGD can be used for neural network optimization, such as for generating adversarial examples [44], 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 [61] and internal organ segmentation with given parametric shape models [8, 37]. 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 $\mathcal{F}$ that converts a given image I to a tree graph G as

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

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 $\mathcal{P}$ 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 $\mathcal{P}$ then read as

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

which are given by traditional graph algorithms with combinatorial optimization. We assume the projection function $\mathcal{P}$ converts the existence probability (or category prediction) of graph edges while leaving the graph nodes V unchanged. A typical example of $\mathcal{P}$ can be designed using the MST algorithm we use in our TreeFormer 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 $\mathcal{R}$ that mimics the non-differentiable projection $\mathcal{P}$. Plugging with the *unconstrained* graph generator $\hat{\mathcal{F}}$, Eq. (1) is rewritten as

$$\begin{aligned} G = (V, E) &= \mathcal{R}_{E\in E_{tree}}(V, \{\hat{\mathbf{y}}_{(i,j)}\}), \\ (V, \{\hat{\mathbf{y}}_{(i,j)}\}) &= \hat{\mathcal{F}}(I), \end{aligned} \tag{4}$$

where the whole process is differentiable.

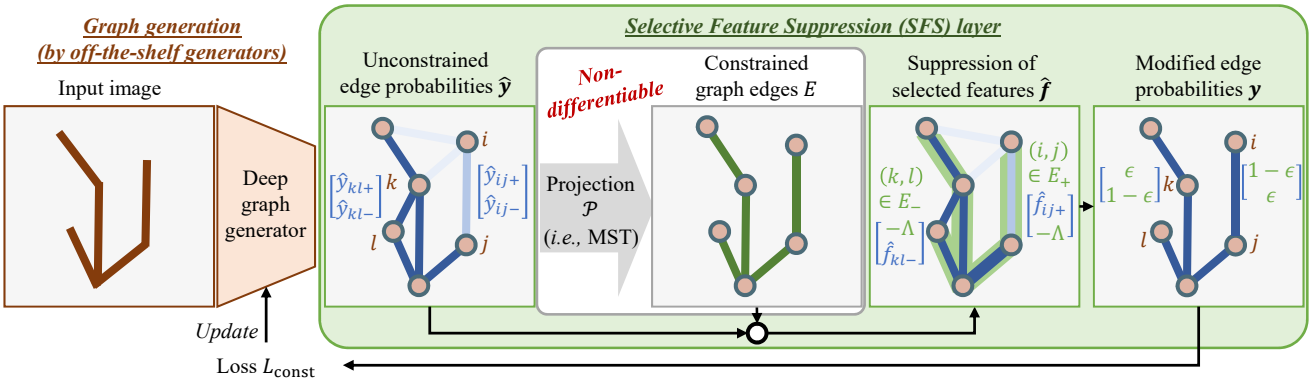


Figure 2. Overview of reparameterization layer that can be easily plugged into off-the-shelf graph generators. Given unconstrained edge predictions by graph generators, our method *projects* it to the closest constrained graph (*i.e.*, tree) using a non-differentiable MST algorithm. Comparing constrained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

3.2. SFS layer

Here, we describe an implementation of the SFS layer for constrained graph generation. As described in Eq. (2), the unconstrained graph generator $\hat{\mathcal{F}}$ 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 σ 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

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

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

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

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

Suppose the projection function $\mathcal{P}$ 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 $\mathcal{P}$ 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 $\mathcal{P}$. 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 σ, it is achieved via the following minimal modification that *selectively* suppresses the feature values by replacing them with a constant[1] as

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

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 σ 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

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

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 $\mathcal{P}$ 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[2]. Here, we briefly analyze the be-

[1] See the supplementary materials for the derivation.

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

havior of the SFS layer. The supplementary materials provide a detailed analysis, including mathematical proofs.

When using the cross-entropy loss $\mathcal{L}_{\text{CE}}$ to evaluate the availability of the graph edges, the derivative to be back-propagated to the backbone graph generator is approximated as[3]

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

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 $\frac{\partial \mathcal{L}_{\text{CE}}}{\partial \hat{f}^+}$ or $\frac{\partial \mathcal{L}_{\text{CE}}}{\partial \hat{f}^-}$.

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, $\frac{\partial \mathcal{L}_{\text{CE}}}{\partial \hat{\mathbf{f}}} \sim \mathbf{0}$, 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[4]. Therefore, the behavior of our simple reparameterization strategy is reasonable in practice.

4. TreeFormer: A Plant Skeleton Estimator

We develop TreeFormer, an implementation of the SFS layer to a state-of-the-art graph generator. This section first recaps the graph generator [52] and then details how we introduce tree structure constraint.

4.1. RelationFormer: A brief recap

RelationFormer [52] 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 [64], 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 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 [6]. RelationFormer is trained using the sum of loss functions related to object detection and edge (relation) estimation, where edge (relation) loss $\mathcal{L}_{\text{edge}}$[5] 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 [33] implemented in NetworkX[6]. 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 supplementary materials.

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 $E_{\text{GT}} = \{(i,j) \mid t^+_{(i,j)} > t^-_{(i,j)}\}$, where $\mathbf{t}_{(i,j)} = [t^+_{(i,j)}, t^-_{(i,j)}]^\top \in \{0,1\}^2$, the loss function for edge availability $\mathcal{L}_{\text{edge}}$ is modified as follows

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

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

5. Experiments

To assess the effectiveness of the proposed method and TreeFormer implementation, we perform experiments using synthetic and real image datasets.

5.1. Datasets

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

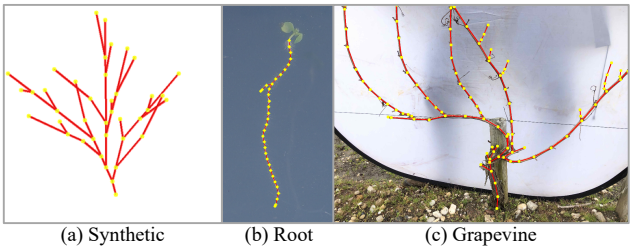
Synthetic dataset To systematically demonstrate the performance of our method, we perform an experiment using a large synthetic dataset. We automatically generate images

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

[4] See supplementary materials for the mathematical proof.

[5] Denoted as $\mathcal{L}_{\text{rln}}$ in the original paper [52], we use $\mathcal{L}_{\text{edge}}$ for generality.

[6] https://networkx.org/, last accessed on July 15, 2024.



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

Figure 3. Example images from the dataset we used for our experiments. Annotated graphs are superimposed. Yellow dots and red lines indicate nodes and edges.

of tree patterns using pre-defined rules of Lindenmayer systems (L-system) [20, 40], 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, 20000 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 [18] We use 3D2cut Single Guyot Dataset [18] 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 [18], 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 as the root dataset, resulting in $118,500$ training images.

5.2. Evaluation metrics

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

Street mover's distance (SMD) [7] 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 [7].

TOPO score [22] 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 Sat2Graph paper [23], 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.

5.3. 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. Supplementary materials provide additional comparisons with other baseline methods, including autoregressive graph generation.

Two-stage [18] We implement a 2D plant skeleton estimation method based on a two-stage method involving skeletonization and graph optimization with reference to [18]. 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 [18]. Specific implementations and analyses are described in the supplementary materials.

Unconstrained [52] We compare the state-of-the-art (unconstrained) graph generation method, RelationFormer [52]. 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.

5.4. Implementation details

For RelationFormer in our method and the baseline comparison, we use the official implementation[7] on PyTorch. For other hyperparameters, we follow the original RelationFormer implementation used for road network extraction.

[7] https://github.com/suprosanna/relationformer, last accessed on July 15, 2024.

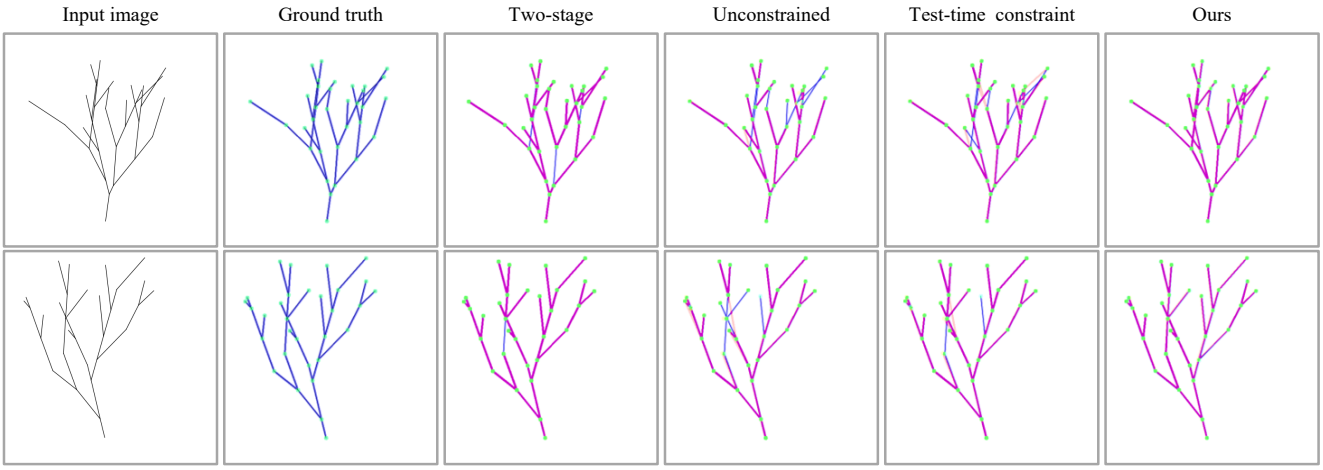


Figure 4. Visual results for the synthetic tree pattern dataset. From left to right: Input images, results of the two-stage method (similar to [18]), the unconstrained method (identical to RelationFormer [52]), a naive implementation with test-time constraint, and ours are shown. We translucently overlay 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 1. Quantitative results. Our method significantly improves both the shape and topology of the predicted graph while enforcing the given constraints. The best scores are highlighted **bold**.

Dataset	Method	SMD ↓	TOPO score ↑			Tree rate
			Prec.	Rec.	F1	[%]
Synthetic	Two-stage [18]	1.91×10^{-3}	0.940	0.886	0.912	**100.0**
	Unconstrained [52]	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 [18]	4.83×10^{-4}	0.767	0.732	0.749	**100.0**
	Unconstrained [52]	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 [18]	4.24×10^{-4}	0.677	0.589	0.630	**100.0**
	Unconstrained [52]	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**

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.

5.5. Results on synthetic dataset

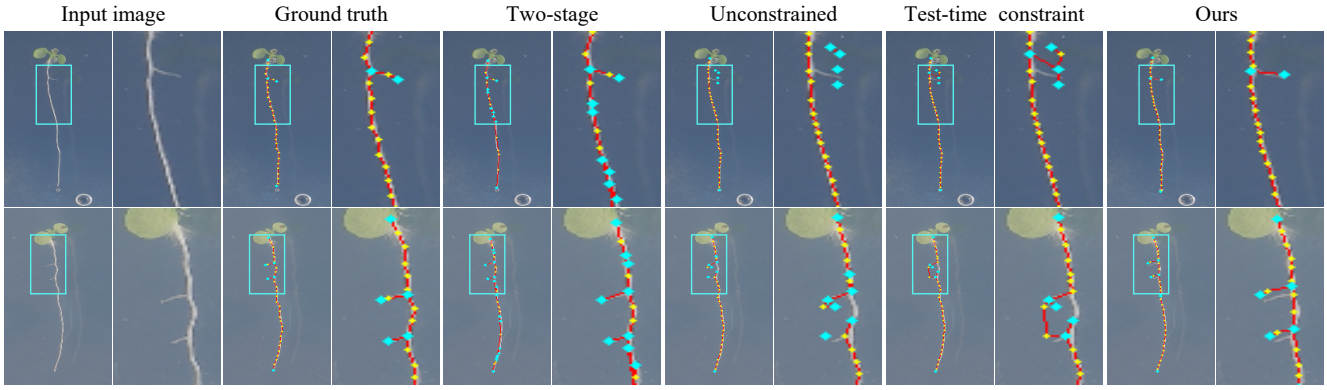
Figure 4 shows visual results for the synthetic dataset, where the red and blue lines indicate the predicted and ground truth edges, respectively. Since they are shown translucently, if the estimated edge overlaps the true edge, it is displayed in purple. Similarly, cyan and yellow dots indicate the nodes, which merge into green if correctly estimated. From the 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 1. The unconstrained method produces tree structures with 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.

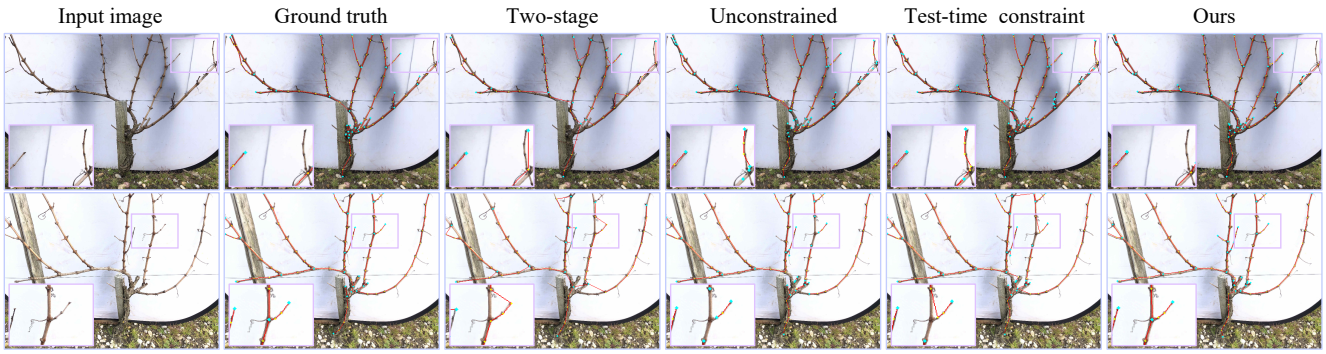
5.6. Results on real datasets

Figure 5 show 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 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. 5a) or missing (*cf.* Fig. 5b) 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 1 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.



(a) Visual results for the root dataset.



(b) Visual results for the grapevine dataset.

Figure 5. Visual results for 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.

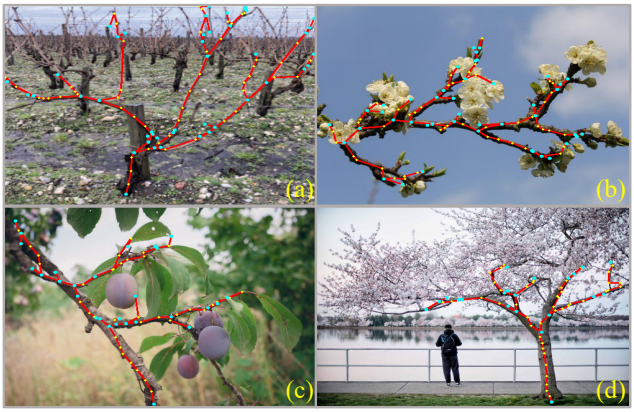


Figure 6. Results for the additional test images of (a) a grapevine tree under a natural background and (b–d) other tree species.

Generalization ability We test our model on additional test datasets to validate the out-of-domain performance of our method, using the model trained on the Grapevine dataset. Although the model is trained with grapevine trees with few background textures, it successfully works for grapevine images with background textures (Fig. 6(a)) and for other tree species (Fig. 6(b–d)). These results highlight the generalizability of our method.

6. Conclusion

We present the first attempt at tree-constrained graph generation from a single image, especially for plant skeleton estimation. We combine modern learning-based graph generators and traditional graph algorithms via the SFS layer, easily integrated with off-the-shelf graph generators.

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.*, [53]) 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. Unlike *universal* human skeleton estimation such as OpenPose [11], our method requires domain-specific training due to the excessive variety of real-world plant appearances and structures, although we show certain generalizability in our experiments.

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. This work was partly supported by JSPS KAKENHI Grant Numbers JP22K17910, JP23H05491, and JP21H03466, and JST FOREST Grant Number JPMJFR206F.

References

[1] Sherif Abdelkarim, Aniket Agarwal, Panos Achlioptas, Jun Chen, Jiaji Huang, Boyang Li, Kenneth Ward Church, and Mohamed Elhoseiny. Exploring long tail visual relationship recognition with large vocabulary. In *Proceedings of IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 15901–15910, 2020. 2

[2] David Acuna, Huan Ling, Amlan Kar, and Sanja Fidler. Efficient interactive annotation of segmentation datasets with polygon-RNN++. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 859–868, 2018. 1, 2

[3] Akshay Agrawal, Brandon Amos, Shane Barratt, Stephen Boyd, Steven Diamond, and J Zico Kolter. Differentiable convex optimization layers. In *Advances in Neural Information Processing Systems (NeurIPS)*, volume 32, 2019. 3

[4] Sungsoo Ahn, Binghong Chen, Tianzhe Wang, and Le Song. Spanning tree-based graph generation for molecules. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2022. 3

[5] Mingyao Ai, Yuan Yao, Qingwu Hu, Yue Wang, and Wei Wang. An automatic tree skeleton extraction approach based on multi-view slicing using terrestrial LiDAR scans data. *Remote Sensing*, 12(22), 2020. 2

[6] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016. 5

[7] Davide Belli and Thomas Kipf. Image-conditioned graph generation for road network extraction. In *Proceedings of NeurIPS Workshop on Graph Representation Learning*, 2019. 1, 2, 6

[8] Simon Bohlender, Ilkay Öksüz, and A. Mukhopadhyay. A survey on shape-constraint deep learning for medical image segmentation. *IEEE Reviews in Biomedical Engineering*, 16:225–240, 2021. 3

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

[10] Llorenç Cabrera-Bosquet, Christian Fournier, Nicolas Brichet, Claude Welcker, Benoît Suard, and François Tardieu. 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, 2016. 1

[11] Zhe Cao, Tomas Simon, Shih-En Wei, and Yaser Sheikh. Realtime multi-person 2D pose estimation using part affinity fields. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2, 8

[12] Ayan Chaudhury and Christophe Godin. Skeletonization of plant point cloud data using stochastic optimization framework. *Frontiers in Plant Science*, 11, 2020. 2

[13] Meng-Jiun Chiou, Henghui Ding, Hanshu Yan, Changhu Wang, Roger Zimmermann, and Jiashi Feng. Recovering the unbiased scene graphs from the biased ones. In *Proceedings of ACM International Conference on Multimedia (MM)*, pages 1581–1590, 2021. 2

[14] Yuren Cong, Michael Ying Yang, and Bodo Rosenhahn. RelTR: Relation transformer for scene graph generation. *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 45(9):11169–11183, 2023. 2

[15] Shenglan Du, Roderik Lindenbergh, Hugo Ledoux, Jantien Stoter, and Liangliang Nan. AdTree: Accurate, detailed, and automatic modelling of laser-scanned trees. *Remote Sensing*, 11(18), 2019. 2

[16] Aaron Ferber, Bryan Wilder, Bistra Dilkina, and Milind Tambe. Mipaal: Mixed integer program as a layer. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 1504–1511, 2020. 3

[17] Mathieu Gaillard, Chenyong Miao, James c. Schnable, and Bedrich Benes. Sorghum segmentation by skeleton extraction. In *Proceedings of European Conference on Computer Vision (ECCV) Workshops*, 2020. 1

[18] Theophile Gentilhomme, Michael Villamizar, Jerome Corre, and Jean-Marc Odobez. Towards smart pruning: ViNet, a deep-learning approach for grapevine structure estimation. *Computers and Electronics in Agriculture*, 207:107736, 2023. 1, 2, 6, 7

[19] Valerio Giuffrida, Massimo Minervini, and Sotirios Tsaftaris. Learning to count leaves in rosette plants. In *Proceedings of Workshop on Computer Vision Problems in Plant Phenotyping (CVPPP)*, pages 1.1–1.13, 01 2015. 2

[20] Jianwei Guo, Haiyong Jiang, Bedrich Benes, Oliver Deussen, Xiaopeng Zhang, Dani Lischinski, and Hui Huang. Inverse procedural modeling of branching structures by inferring L-systems. *ACM Transactions on Graphics (TOG)*, 39(5):1–13, 2020. 6

[21] Harshit Gupta, Kyong Hwan Jin, Ha Q Nguyen, Michael T McCann, and Michael Unser. CNN-based projected gradient descent for consistent CT image reconstruction. *IEEE Transactions on Medical Imaging*, 37(6):1440–1453, 2018. 3

[22] Songtao He, Favyen Bastani, Sofiane Abbar, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, and Sam Madden. RoadRunner: Improving the precision of road network inference from gps trajectories. In *Proceedings of ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*, pages 3–12, 2018. 6

[23] Songtao He, Favyen Bastani, Satvat Jagwani, Mohammad Alizadeh, Hari Balakrishnan, Sanjay Chawla, Mohamed M. Elshrif, Samuel Madden, and Mohammad Amin Sadeghi. Sat2Graph: Road graph extraction through graph-tensor encoding. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 51–67, 2020. 1, 2, 6

[24] Hui Huang, Shihao Wu, Daniel Cohen-Or, Minglun Gong, Hao Zhang, Guiqing Li, and Baoquan Chen. L1-medial skeleton of point cloud. *ACM Transactions on Graphics (TOG)*, 32(4):65, 2013. 2

[25] Takahiro Isokane, Fumio Okura, Ayaka Ide, Yasuyuki Matsushita, and Yasushi Yagi. Probabilistic plant modeling via multi-view image-to-image translation. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 2906–2915, 2018. 2

[26] Wengong Jin, Regina Barzilay, and Tommi Jaakkola. Junction tree variational autoencoder for molecular graph genera-

tion. In *Proceedings of International Conference on Machine Learning (ICML)*, pages 2323–2332, 2018. 3

[27] Wengong Jin, Regina Barzilay, and T. Jaakkola. Hierarchical generation of molecular graphs using structural motifs. In *Proceedings of International Conference on Machine Learning (ICML)*, 2020. 3

[28] Ivan Khokhlov, Lev Krasnov, Maxim V. Fedorov, and Sergey Sosnin. Image2SMILES: Transformer-based molecular optical recognition engine. *Chemistry-Methods*, 2(1):e202100069, 2022. 1, 2

[29] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In Yoshua Bengio and Yann LeCun, editors, *Proceedings of International Conference on Learning Representations (ICLR)*, 2014. 2

[30] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2014. 3

[31] Thomas N Kipf and Max Welling. Variational graph auto-encoders. In *Proceedings on NeurIPS Workshop on Bayesian Deep Learning*, 2016. 2

[32] James Kotary, Ferdinando Fioretto, Pascal Van Hentenryck, and Bryan Wilder. End-to-end constrained optimization learning: A survey. In *Proceedings of International Joint Conferences on Artificial Intelligence (IJCAI)*, 2021. 3

[33] Joseph B Kruskal. On the shortest spanning subtree of a graph and the traveling salesman problem. *Proceedings of the American Mathematical Society*, 7(1):48–50, 1956. 5

[34] Rongjie Li, Songyang Zhang, and Xuming He. SGTR: End-to-end scene graph generation with transformer. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 19486–19496, 2022. 1, 2

[35] Weijian Li, Yuhang Lu, Kang Zheng, Haofu Liao, Chihung Lin, Jiebo Luo, Chi-Tung Cheng, Jing Xiao, Le Lu, Chang-Fu Kuo, et al. Structured landmark detection via topology-adapting deep graph learning. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 266–283, 2020. 1, 2

[36] Weijia Li, Wenqian Zhao, Huaping Zhong, Conghui He, and Dahua Lin. Joint semantic-geometric learning for polygonal building segmentation. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 1958–1965, 2021. 1, 2

[37] Yuanwei Li, Chin Pang Ho, Matthieu Toulemonde, Navtej Chahal, Roxy Senior, and Meng-Xing Tang. Fully automatic myocardial segmentation of contrast echocardiography sequence using random forests guided by shape model. *IEEE Transactions on Medical Imaging*, 37(5):1081–1091, 2017. 3

[38] Yujia Li, Oriol Vinyals, Chris Dyer, Razvan Pascanu, and Peter Battaglia. Learning deep generative models of graphs. *arXiv preprint arXiv:1803.03324*, 2018. 2

[39] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Yuwen Xiong, Rui Hu, and Raquel Urtasun. PolyTransform: Deep polygon transformer for instance segmentation. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 9128–9137, 2020. 1, 2

[40] Aristid Lindenmayer. Mathematical models for cellular interactions in development I. Filaments with one-sided inputs. *Journal of Theoretical Biology*, 18(3):280–299, 1968. 6

[41] Huan Ling, Jun Gao, Amlan Kar, Wenzheng Chen, and Sanja Fidler. Fast interactive object annotation with curve-GCN. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5252–5261, 2019. 1, 2

[42] Qi Liu, Miltiadis Allamanis, Marc Brockschmidt, and Alexander L. Gaunt. Constrained graph variational autoencoders for molecule design. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2018. 2

[43] Youzhi Luo, Keqiang Yan, and Shuiwang Ji. Graphdf: A discrete flow model for molecular graph generation. In *Proceedings of International Conference on Machine Learning (ICML)*, 2021. 2

[44] Aleksander Madry, Aleksandar Makelov, Ludwig Schmidt, Dimitris Tsipras, and Adrian Vladu. Towards deep learning models resistant to adversarial attacks. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2018. 3

[45] Teng Miao, Chao Zhu, Tongyu Xu, Tao Yang, Na Li, Yuncheng Zhou, and Hanbing Deng. Automatic stem-leaf segmentation of maize shoots using three-dimensional point cloud. *Computers and Electronics in Agriculture*, 187:106310, 2021. 1, 2

[46] Young-Soo Myung, Chang-Ho Lee, and Dong-Wan Tcha. On the generalized minimum spanning tree problem. *Networks*, 26(4):231–241, 1995. 3

[47] Fumio Okura. 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, 2022. 2

[48] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, 2014. 2

[49] Petrică C Pop. The generalized minimum spanning tree problem: An overview of formulations, solution procedures and latest advances. *European Journal of Operational Research*, 283(1):1–15, 2020. 3

[50] Sahand Sharifzadeh, Sina Moayed Baharlou, Martin Schmitt, Hinrich Schütze, and Volker Tresp. Improving scene graph classification by exploiting knowledge from texts. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 2189–2197, 2022. 2

[51] wu Sheng, Weiliang Wen, Boxiang Xiao, Xinyu Guo, Jian Jun Du, Chuanyu Wang, and Yongjian Wang. An accurate skeleton extraction approach from 3d point clouds of maize plants. *Frontiers in Plant Science*, 10:248, 2019. 1, 2

[52] Suprosanna Shit, Rajat Koner, Bastian Wittmann, Johannes Paetzold, Ivan Ezhov, Hongwei Li, Jiazhen Pan, Sahand Sharifzadeh, Georgios Kaissis, Volker Tresp, et al. Relationformer: A unified framework for image-to-graph generation. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 422–439, 2022. 1, 2, 5, 6, 7

[53] Vibhav Vineet, Pawan Harish, Suryakant Patidar, and PJ Narayanan. Fast minimum spanning tree for large graphs

on the GPU. In *Proceedings of Conference on High Performance Graphics (HPG)*, pages 167–171, 2009. 8

[54] Bryan Wilder, Bistra Dilkina, and Milind Tambe. Melding the data-decisions pipeline: Decision-focused learning for combinatorial optimization. In *Proceedings of AAAI Conference on Artificial Intelligence (AAAI)*, pages 1658–1665, 2019. 3

[55] Sheng Wu, Weiliang Wen, Yongjian Wang, Jiangchuan Fan, Chuanyu Wang, Wenbo Gou, and Xinyu Guo. MVS-Pheno: A portable and low-cost phenotyping platform for maize shoots using multiview stereo 3D reconstruction. *Plant Phenomics*, 2020:1848437, 2020. 2

[56] Cihang Xie, Mingxing Tan, Boqing Gong, Jiang Wang, Alan L. Yuille, and Quoc V. Le. Adversarial examples improve image recognition. In *Proceedings of IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[57] Hui Xu, Nathan Gossett, and Baoquan Chen. Knowledge and heuristic-based modeling of laser-scanned trees. *ACM Transactions on Graphics (TOG)*, 26(4):19, 2007. 2

[58] Zhenhua Xu, Yuxuan Liu, Yuxiang Sun, Ming Liu, and Lujia Wang. RNGDet++: Road network graph detection by transformer with instance segmentation and multi-scale features enhancement. *IEEE Robotics and Automation Letters*, pages 1–8, 2023. 1, 2

[59] Zhenhua Xu, Yuxiang Sun, and Ming Liu. iCurb: Imitation learning-based detection of road curbs using aerial images for autonomous driving. *IEEE Robotics and Automation Letters*, 6:1097–1104, 2021. 1, 2

[60] Jingkang Yang, Yi Zhe Ang, Zujin Guo, Kaiyang Zhou, Wayne Zhang, and Ziwei Liu. Panoptic scene graph generation. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 178–196, 2022. 2

[61] Michitaka Yoshida, Akihiko Torii, Masatoshi Okutomi, Kenta Endo, Yukinobu Sugiyama, Rin-ichiro Taniguchi, and Hajime Nagahara. Joint optimization for compressive video sensing and reconstruction under hardware constraints. In *Proceedings of European Conference on Computer Vision (ECCV)*, pages 634–649, 2018. 3

[62] Chengxi Zang and Fei Wang. Moflow: A invertible flow model for generating molecular graphs. In *Proceedings of ACM SIGKDD International Conference on Knowledge Discovery & Data Mining (KDD)*, pages 617–626, 2020. 2

[63] Xiaochen Zhou, Bosheng Li, Bedrich Benes, Songlin Fei, and Sören Pirk. Deeptree: Modeling trees with situated latents. *IEEE Transactions on Visualization and Computer Graphics*, 30(8):2795–5809, 2023. 2

[64] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: Deformable transformers for end-to-end object detection. In *Proceedings of International Conference on Learning Representations (ICLR)*, 2021. 5

[65] Illia Ziamtsov and Saket Navlakha. Machine learning approaches to improve three basic plant phenotyping tasks using three-dimensional point clouds. *Plant Physiology*, 181(4):1425–1440, 2019. 2