Single Image Tree Reconstruction via Adversarial Network

Zhihao Liu[a,b], Kai Wu[a,b], Jianwei Guo[c,b,*], Yunhai Wang[d], Oliver Deussen[a,e] and Zhanglin Cheng[a,*]

[a] *Shenzhen Institute of Advanced Technology, Chinese Academy of Sciences, China*
[b] *University of Chinese Academy of Sciences, China*
[c] *NLPR, Institute of Automation, Chinese Academy of Sciences, China*
[d] *Shandong University, China*
[e] *University of Konstanz, Germany*

ARTICLE INFO

Keywords:
tree reconstruction
single image reconstruction
adversarial network

ABSTRACT

Realistic 3D tree reconstruction is still a tedious and time-consuming task in the graphics community. In this paper, we propose a simple and efficient method for reconstructing 3D tree models with high fidelity from a single image. The key to single image-based tree reconstruction is to recover 3D shape information of trees via a deep neural network learned from a set of synthetic tree models. We adopted a conditional generative adversarial network (cGAN) to infer the 3D silhouette and skeleton of a tree respectively from edges extracted from the image and simple 2D strokes drawn by the user. Based on the predicted 3D silhouette and skeleton, a realistic tree model that inherits the tree shape in the input image can be generated using a procedural modeling technique. Experiments on varieties of tree examples demonstrate the efficiency and effectiveness of the proposed method in reconstructing realistic 3D tree models from a single image.

1. Introduction

Many tree modeling methods have been developed in the past years, including procedural tree modeling and tree reconstruction from photographs or laser scans. Procedural tree modeling can synthesize virtual tree models, while tree reconstruction methods generate faithful models of real-world trees. In order to reconstruct realistic models for real trees, even collecting input data is costly and time-consuming, e.g., obtaining 3D point clouds using laser scanners or taking a lot of photos carefully from different views around real trees. Single image-based tree reconstruction technique takes only one image as input and drastically simplifies the requirements of input data for real tree reconstruction, which thus provides a feasible solution to generate a realistic tree model only from a single image of the tree.

The key to single image-based tree reconstruction is to recover 3D information from a single image. Previous single image-based tree modeling methods generally infer the 3D tree shape using some prior knowledge. For example, Tan et al. [47] and Guénard et al. [12] transform 2D branches into 3D with the assumption that the distance or projection angle between branches is as large as possible, and Argudo et al. [1] generates symmetrical 3D tree shape by inflating 2D silhouette. However, such 2D to 3D transformations are artificially designed and are too simple to describe the true projection between the 2D image and 3D model of a tree. It's difficult for these methods to generate 3D models with high fidelity from the single image of a tree. Although accurate 3D reconstruction from a single image is ill-posed and impossible in general, we argue that it is possible to learn the prior of 3D tree structures and recover 3D information to some extent from the single image of a tree with the help of a library of 3D tree models. Thus we try to use deep learning methods to learn the underlying 2D to 3D projections and infer the 3D shape of trees from a single image. In this paper, we use a conditional generative adversarial network (cGAN) to recover the depth information of a tree from the skeleton and silhouette image of the tree.

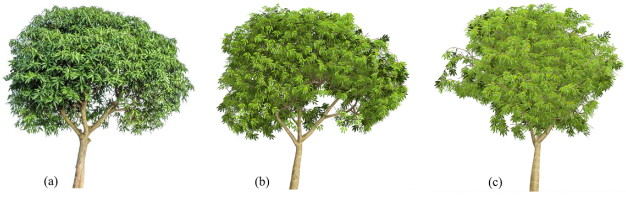


Figure 1: Single image-based tree reconstruction. (a) Single input image of a tree. (b) The reconstructed tree model rendered at the same viewpoint as the input image. (c) The tree model rendered at the side viewpoint.

Most deep learning based object reconstruction methods generally represent and predict 3D shapes as voxels or point clouds, and are limited to reconstruct relatively simple and smooth shapes, like cars or chairs. However, trees have more complex and detailed tiny structures with abundant branches and leaves, so that directly predicting the 3D tree models is quite difficult. In this paper, we combine deep learning and procedural tree modeling techniques to generate high-quality tree models from a single image. A deep neural network is used to predict the basic 3D shape of a tree, and then detailed tree models are generated by procedural modeling based on the predicted shape. We use two representations to describe the basic 3D shape of a tree: one is *skeleton*, which

*Jianwei Guo and Zhanglin Cheng are corresponding authors.
liuzh96@outlook.com (Z. Liu); kai.wu@siat.ac.cn (K. Wu); jianwei.guo@nlpr.ia.ac.cn (J. Guo); cloudseawang@gmail.com (Y. Wang); oliver.deussen@uni-konstanz.de (O. Deussen); zl.cheng@siat.ac.cn (Z. Cheng)
ORCID(s): 0000-0003-1099-5745 (Z. Liu); 0000-0002-1269-9918 (K. Wu); 0000-0002-3376-1725 (J. Guo); 0000-0001-5803-2185 (O. Deussen); 0000-0002-3360-2679 (Z. Cheng)

is used for representing the main visible trunks; the other is *silhouette*, which covers the outer shape of the tree using a point cloud. Such skeleton and silhouette are encoded by two kinds of depth images. Thus, the problem is cast into an image-to-image translation task, instead of estimating the 3D shapes directly. We apply a cGAN architecture to infer such depth images from an input image. We first draw a few strokes to mark up the visible trunks and extract edges from the tree image. The network takes as input the strokes and edges and output corresponding depth images of skeletons and silhouettes. Finally, based on the predicted skeleton of main trunks, we apply the procedural modeling method [31] to generate tree models within the shape defined by silhouette. The procedural modeling method was also adopted to generate numerous realistic tree models automatically as the training set.

The main reason for taking edges instead of real photos as input for generating 3D silhouette is that it is hard and time-consuming to render the synthetic trees as realistically as photographs with detailed textures, and therefore the network trained on synthetic dataset is unable to accurately predict the depths from real photos. However, using edges can solve this problem and also weaken the influence of texture colors and other variable environmental factors such as lighting and shadows, which usually cause the convergence failure of network. Besides, the edges depict the local shapes of trees as well as other important details like the inner holes and the density of branches and leaves. The 3D silhouette provides a relatively rough description for the overall tree crown shape. To reconstruct faithfully the main trunk and visible branches, we mark up the visible trunks using several 2D strokes, and the cGAN network predicts the depth of skeletons directly from such user-drawn strokes.

The main contribution of the paper is 3D tree reconstruction with high fidelity from a single image of the tree by combing a deep learning method and procedural modeling. The reconstruction can be done within seconds, in almost real-time.

2. Related work

Procedural tree modeling. Grammar-based or rule-based procedural modeling techniques provide an efficient way to create complex plant models [7]. The popular L-system is originally introduced by [23] to describe the multi-cellular development, then extended in various ways to model the geometry and structure of different plant types [36, 35, 34]. Another widely used procedural method models trees as recursive branching structures characterized by a small number of geometric rules, such as branching angles and length ratios of consecutive internodes [15], implicit functions [3], fractal models [30], and narrow near-conical tubes [49]. De Reffye et al. [40] also use a collection of rules to simulate branching structures, but these rules are motivated by plant growth models. Instead of directly writing generative rules, Stava et al. [46] use a target polygonal model as constraints and propose an inverse procedural method. They use Monte Carlo Markov Chains (MCMC) to estimate the optimal parameters of a procedural model for producing plants similar to the input. Recently, Guo et al. [13] utilize deep learning to detect branching structures and propose an inverse procedural modeling approach that learns 2D L-system representations of pixel images.

Considering the effects from the environment and between-branch spaces, the space colonization algorithms [43, 2] are proposed to produce more convincing tree forms. Pirk et al. [33] present a dynamic tree modeling and representation technique that allows complex tree models to interact with their environment. Pałubicki et al. [31] use the concept of self-organization to adapt trees to the environment by simulating the competition of buds and branches for light and space. Then Yi et al. [53] extend this method by integrating various growth equations into the procedural tree modeling process. These approaches typically require expert knowledge and are manually intensive, providing indirect means to control the tree modeling process.

Real-world tree modeling. To reduce the modeling effort, researchers have developed many algorithms to capture and reconstruct trees from real-world data. Laser scanning provides an effective tool for acquiring geometric attributes of trees. From the obtained 3D point cloud, Xu et al. [51] and Livny et al. [25] utilize the minimal spanning graphs to recover the tree branches, then small twigs and leaves are randomly added to form the crown geometry. Livny et al. [24] present a lobe-based representation to approximate the geometry of given data and synthesize a full plant model by instancing each lobe with predefined patches. Yan et al. [52], Raumonen et al. [38] and Zhang et al. [32, 54] propose cylinder marching algorithms by locally fitting or searching cylinders to achieve accurate tree modeling. In other works, Friedman and Stamos [9] infer shape grammars from the wavelet transform of an input point cloud then reproduce the tree structure using the grammars. Li et al. [22] capture a sequence of point clouds to analyze evolving parts of a developing plant.

Image-based techniques usually use multi-view images to extract various representations of the tree geometry to guide the modeling process, such as visual hulls [44], volumetric representation [39] [28], point clouds obtained from Structure From Motion (SFM) [37] [48] [4]. Some researchers also attempt to produce tree models from video [8, 21]. The most related to our approach is the single-image-based modeling methods. In [47] the users draw at least two strokes to identify the crown and the branches. Then the 2D strokes are used to guide 3D tree synthesis by a growth engine. Guénard et al. [12] propose to generate a 3D plant model using an analysis-by-synthesis method that combines information from a single image and a priori knowledge of the plant species. Argudo et al. [1] present a complete pipeline for synthesizing and rendering detailed trees with minimal user effort. The key is to create a rough estimate of the crown shape by solving a thin-plate energy minimization problem, and then add detail through a simplified shape-from-shading approach.

Deep learning-based tree modeling methods have received

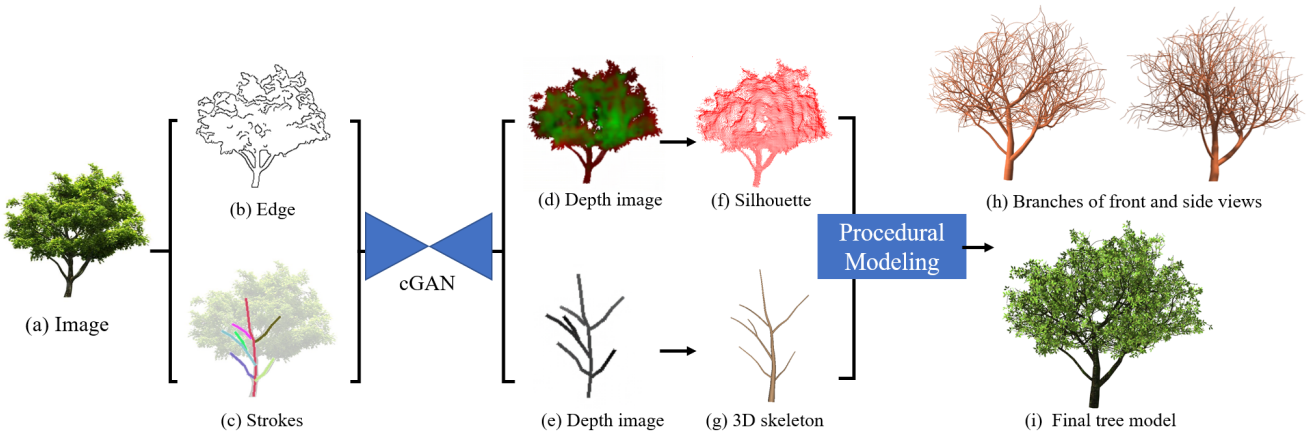


Figure 2: Overview of our tree reconstruction method. (a) The input image. (b) The edges extracted by the Canny detector. (c) The user-drawn strokes for marking up the main trunks. (d) and (e) Two depth images predicted by the cGAN. (f) and (g) The 3D silhouette and skeleton recovered from two depth images. (h) The branching structure generated by a procedural modeling algorithm. (i) The final 3D tree model with leaves.

some attention in the graphics community. Huang et al. [16] propose to synthesize shapes from user sketches, where a deep Convolutional Neural Network (CNN) is trained to map sketches to procedural model parameters. However, this method only works for synthesizing 2D trees. A recent work [17] estimates branch structures of 3D plants from multi-view images via image-to-image translation. They combine a Bayesian extension of image-to-image translation and 3D aggregation to generate a 3D branch probability map. In contrast to them, we generate 3D tree models from a single image, and the inferred depth information represents the branches more precisely than the probability map.

Image-to-image translation via GANs. The first generative adversarial network (GAN) [11] was proposed to solve the image generation problem. After that, many researchers try to solve traditional image-based tasks based on GAN. For example, image-to-image translation is a building block of modern image processing which maps a possible representation of the source image into the target image. Isola et al. [18] present a pix2pix framework based on cGAN [27] for learning image-to-image translation from paired images. CycleGAN [55] and DiscoGAN [19] utilize cycle consistency loss to achieve unpaired image-to-image translation. Choi et al. [6] present starGAN to implement multi-domain image translation problem using a single generator.

Deep learning for 3D objects. Due to the success in generating images, lots of methods try to apply deep learning for 3D generation or reconstruction. Wu et al. [50] generate new 3D objects as voxels in $64 \times 64 \times 64$ based on a generative adversarial network. Girdhar et al. [10] present a TL-embedding network, which contains a series of convolutional and fully connected layers, to predict $20 \times 20 \times 20$ voxels from a 2D image. Rezende et al. [41] adopt a conditional generative model to infer 3D voxels from 2D images. However, the voxel is a low precision representation for 3D objects. Lun et al. [26] reconstruct 3D point cloud from multi-view 2D sketches by a multi-view convolutional network. SurfNet [45] reconstructs 3D shape surfaces directly instead of voxels via a deep residual network. However, such methods are limited to some artificial regular objects, like cars, tables, beds, and chairs. Whereas tree models have more complex structures with no continuous surface, which consist of abundant leaves and branches. Our method can generate high-quality tree models from a single image.

3. Overview

In this paper, we propose to reconstruct a 3D tree model from a single image via deep convolutional networks. Our framework consists of three concrete steps as shown in Fig. 2. The core component of the proposed method is a conditional GAN, which translates an image from one domain to another. To train such a neural network, we automatically produce a large set of synthetic 3D tree models and generate collections of image pairs as the training dataset. Based on the training data, we propose a new loss function to optimize the network parameters.

In the testing stage, given an input image, we first extract the edges using Canny detector [5], and draw several simple strokes to mark up the main trunk, as shown in Fig. 2 (b) and (c). Then the trained cGAN takes as input the edges and strokes and outputs two kinds of depth images (see Fig. 2 (d) and(e)). Such two depth images are particularly designed for storing the geometric shape of a tree in two representations: the 3D silhouette and skeleton (see Fig. 2 (f) and (g)). The 3D silhouette defines the growth space of a tree, while the skeleton represents the 3D visible main trunk. Finally, based on the reconstructed main skeleton, we adopt a procedural modeling method [31] to generate the full 3D tree model within the space defined by the silhouette. In order to generate a more natural tree shape, a direction map is computed from the input image to guide the modeling process. Fig. 2 (h) illustrates the front and side views of the branch-

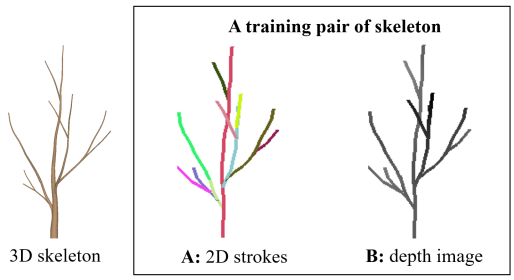


Figure 3: An example of 3D skeleton training pair.

ing structure generated by our method. The final tree model with leaves is rendered in Fig. 2 (i).

4. Algorithm

4.1. Training data representation

To obtain numerous tree samples for training, we first collect a set of 3D tree models automatically by using a procedural modeling approach [31]. The tree dataset contains various tree species by using different modeling parameters. After getting such tree models, we can generate the image pairs for training our network. All images are set to size 256×256 and projected by using orthogonal projection. In the following, we introduce the generation of image pairs for predicting 3D skeleton and silhouette, respectively.

Skeleton training pairs. For predicting the 3D skeleton, the input to our network is a collection of 2D strokes and the output is the corresponding depth image (see Fig. 3). Given a 3D tree model, the 2D strokes can be obtained directly by rendering each branch with different colors, while the depth image can be directly extracted from the depth buffer. Here we use different colors to encode the branches because the colors can indicate the relative positions of branches clearly according to the intersections of different strokes. This approach also helps to improve the convergence of the neural network. In our framework, the colors used to mark up different branches are fixed for each tree. Besides, all strokes are set to 4 pixel width, because thinner strokes will make the convergence difficult while wider strokes may cause heavy occlusion between line segments. Finally, we set the rendering window size to 256×256 and use orthogonal projection to render each image pair. In this situation, the depth value lies in range $[0, 1]$, where 0 is for the nearest plane and 1 is for the farthest plane.

Silhouette training pairs. To estimate the 3D silhouette, the network takes as input an edge image and outputs a special depth image which is used for depicting the 3D shape of the tree volume (see Fig. 4). In order to obtain edge images, we first apply a Gaussian filter (kernel size = 5) to smooth the image, then extract edges using a Canny detector, the high and low thresholds of which are set to 0.8 and 0.4. To store the 3D silhouette into a single image, we depict silhouette using a dual depth image, encoding depth values for front and back sides of the tree at the same time. As shown in Fig. 5, the depths of two opposite views D_1 and D_2 are obtained under orthogonal projection. To match the pixel pairs correctly, D_2 should be mirrored horizontally first. For the final dual depth image, we use the red channel (R) to store the depths of front view, and use green channel (G) to store the depth intervals between front and back views, that is

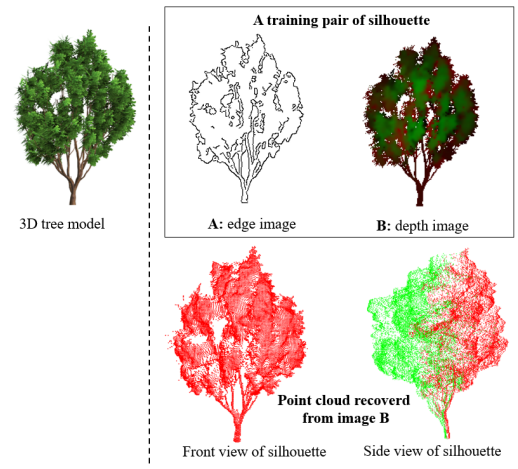


Figure 4: An example of 3D silhouette training pair. In the bottom row, the red and green points represent the 3D depths of front and back views, respectively.

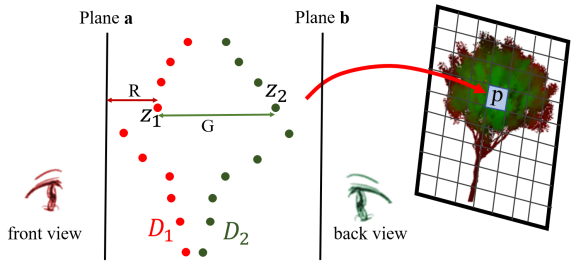


Figure 5: Illustration of casting 3D silhouette to a depth image.

$$\begin{cases} R_{x,y} = z_{1,x,y} \\ G_{x,y} = \left(1 - z_{2,x,y}\right) - z_{1,x,y} \end{cases} \tag{1}$$

where $R_{x,y}$ and $G_{x,y}$ denote the values of red and green channels at pixel (x, y) in the final dual depth image. $z_{1,x,y}$ and $z_{2,x,y} \in [0, 1]$ are respectively the depths in images D_1 and D_2. The term $(1 - z_{2,x,y})$ is to make two depths in the same coordinate system since they are obtained from opposite views. Note that a way that seems more intuitive is to directly use the depths of back view as G channel. However, the output of neural network usually contains noise so that the predicted back points might be located before the front points. By comparison, since the predicted values of depth intervals are always positive, it ensures the back points must be always behind the front points. Finally, we apply Gaussian filter (kernel size = 5) to each channel of the dual depth image to smooth the silhouette, which can effectively improve the convergence of network. Fig. 4 shows an example of the silhouette training pair (top) and its corresponding point cloud (bottom).

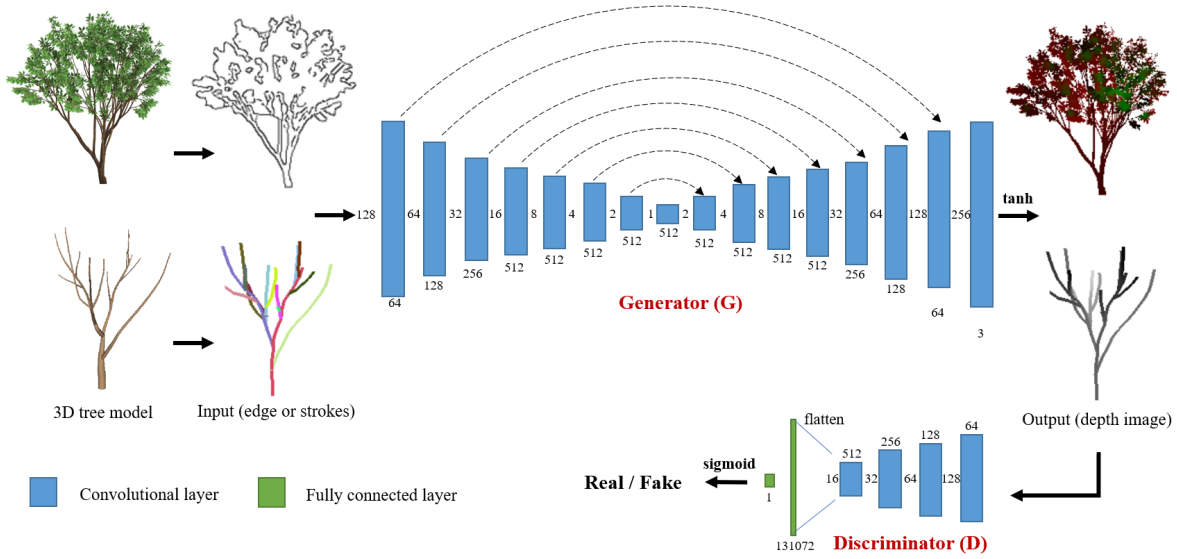


Figure 6: Adversarial network architecture for recovering depth information.

4.2. Network for predicting depth

In this paper, we extend the neural network proposed by [18] that successfully solves the image-to-image translation problem using a conditional GAN (cGAN). Our cGAN takes as input the edges (or strokes) extracted from a single image, and outputs the depth image for silhouette (or skeleton). Fig. 6 illustrates the detailed cGAN architecture. As described above, the input for predicting 3D silhouette is a 1-channel edge image and the output is a 2-channel silhouette image. For skeleton, the network inputs a 3-channel stroke image and outputs a 1-channel depth image.

Generator G. The core of our cGAN is a generator G, which takes as input an image of one domain and translates it into another domain. We adopt U-Net [42] as the generator, which is a popular choice in many 2D image based tasks. The U-Net is an encoder-decoder structure with skip connections between mirrored layers in the encoder and decoder stacks, as shown in Fig. 6. The structure of the encoder and decoder are symmetrical, both consisting of a series of convolutional layers. All convolutions use 4×4 filters with stride of 2. The intermediate layers use batch normalization and adopt leaky ReLU as activation function. For the last layer in decoder, a convolution is performed to get the final output image, followed by a tanh activation.

Discriminator D. The discriminator D is trained to distinguish the fake images produced by generator G from the real depth images as well as possible. The architecture of discriminator is also composed of a series of convolutional layers. The intermediate convolutions also use batch normalization and leaky ReLU. All convolutions use kernel size of 4 and stride of 2. After the last convolutional layer, we adopt a fully connected layer and sigmoid activation function to map to a 1 dimensional output, which indicates the probability how the output image looks like the real ones.

Loss function. We adopt a cGAN loss to describe the rivalry game between the generator G and discriminator D, which can be defined as:

$$L_{\text{cGAN}}(G, D) = E_{x,y}[\log D(y|x)] - E_x[\log(D(G(x)|x)], \quad (2)$$

where x is the input image, y is the corresponding ground-truth depth image. During the training process, generator G tries to minimize this function whereas discriminator D tries to maximize it.

Moreover, we add a depth loss L_{depth} to measure the per-pixel differences between output depth and ground-truth using L-1 distance. Unlike most of image translation tasks, we do not care about the pixels in background. Thus, we add a binary mask to compute the depth loss as:

$$L_{\text{depth}}(G) = \sum_p m_p \left\| d_p - \tilde{d}_p \right\|_1, \quad (3)$$

where m_p is the binary mask value for pixel p, which is 1 for foreground and 0 for background. d_p and $\tilde{d}_p$ denote the ground-truth and predicted depth respectively. To sum up, our final loss function L can be expressed as:

$$L = L_{\text{cGAN}}(G, D) + \lambda L_{\text{depth}}(G). \quad (4)$$

The typical value of λ is set to 100. For efficient network training, we alternately optimize the parameters of discriminator and generator using Adam optimizer [20].

4.3. Tree reconstruction

Depth image denoising. After generating the depth images by using our network, we can easily recover 3D point clouds of the silhouette and skeleton. However, the direct predicted point clouds are usually not clean and contain noise. By observing the zoomed-in depth images and predicted point clouds, we find that most of noise points are located at the boundary between foreground and background, e.g., the light-colored pixels in Fig. 7. Thus we perform denoising on the inferred depth images.

We first mark all the pixels located at the boundary as noises because their depth values are not accurate. Next, we process each channel of the depth image as follows. For each pixel p^k ($k = R, G, B$) in foreground, we compute the average distance d^k between p^k and all of its neighbors p_i^k:

$$d^k = \frac{1}{N(p^k)} \sum_{i \in N(p^k)} |p^k - p_i^k|, \tag{5}$$

where $i \in N(p^k)$ and $N(p^k)$ denotes the 2-ring neighbors of p^k. For each neighbor pixel, we also compute its average distance d_i^k using the same way. Then, the pixel p^k is considered as a noise if the variance of all average distances d_i^k in its neighborhood is larger than a given threshold:

$$\delta^2 = \frac{\sum_{i \in N(p^k)} \left(d_i^k - d_{avg}^k \right)^2}{N\left(p^k\right)} > \epsilon, \tag{6}$$

where $d_{avg}^k = \frac{1}{N(p^k)} \sum_{i \in N(p^k)} (d^k - d_i^k)$ represents the mean of d_i^k, and ϵ is the threshold with default value 0.019. Then, all of noise pixels are filled with the average depth value of its neighbors. Fig. 8 (a) shows the optimized point cloud of the silhouette after depth image denoising.

Visible branches tracing. To reconstruct the visible branches, we organize the skeleton points as a connected graph structure. We apply the breadth-first search for traversing pixels of the skeleton depth image from the root which is the lowest position. We use a queue q to store the pixels to be visited and a list l to record the branches that have been found. The branches are identified by their colors, and each maintains a list of nodes. In the beginning, q is initialized with the root position, and a start branch is added into l. We then search pixels along eight directions in the image with a step of 4 (which is the default stroke width). For each pixel in q, we form a new node by computing the centroid position of its neighbors with radius of 4, and push the node back into q and its corresponding branch in l. If encountering a new color, we add a new branch into l. The algorithm ends when the queue q becomes empty. Finally, the connected tree structure can be formed according to the pixel adjacency and the 3D distances between the endpoints of each branch. Fig. 8 (b) shows the optimized connected tree skeleton after depth image denoising and branches tracing.

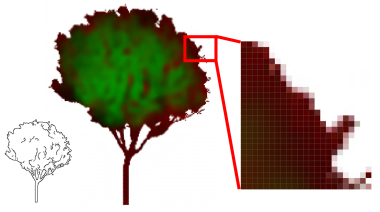


Figure 7: Noises are often located at the boundary between foreground and background.

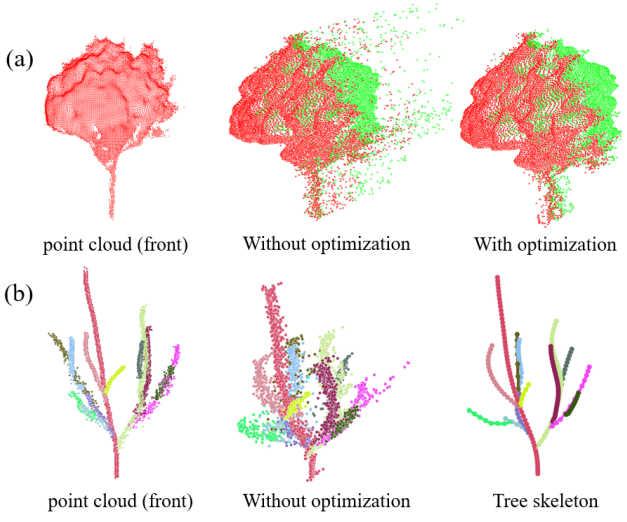


Figure 8: Data processing for point cloud denoising and visible branches tracing.

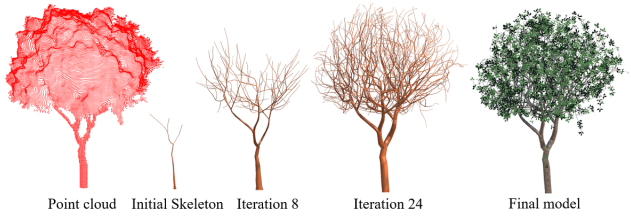
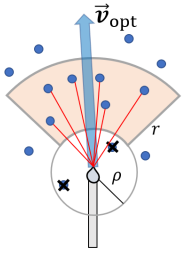


Figure 9: Procedural growth guided by the 3D silhouette and skeleton.

Procedural modeling with direction field. Constrained by the 3D silhouette and skeleton, we grow trees automatically using the procedural modeling method [31]. The core of the method is space colonization, which simulates the competition of growing branches for space. The space for growth is represented by a set of marker points. In our work, markers are generated within the tree volume defined by the predicted 3D silhouette with uniform random distribution. Beginning with the reconstructed visible skeleton, we attach buds to the last few branches. Whether a bud will produce a new shoot depends on the availability of space. The method assumes that each bud is surrounded by a spherical occupancy zone of radius ρ and has a conical perception volume characterized by the perception angle θ and radius r (see the insert above). At the beginning of each growth iteration, all buds remove markers within their occupancy zones. A bud can develop into a shoot if there remain markers within its perception volume. The optimal growth direction of a bud $\vec{v}_{\text{opt}}$ can be written as:



$$\vec{v}_{\text{opt}} = \frac{1}{N} \sum_{i}^{N} \vec{v}_i \tag{7}$$

where $\vec{v}_i$ is a normalized vector formed by the bud and its neighboring marker i. N is the number of the markers within

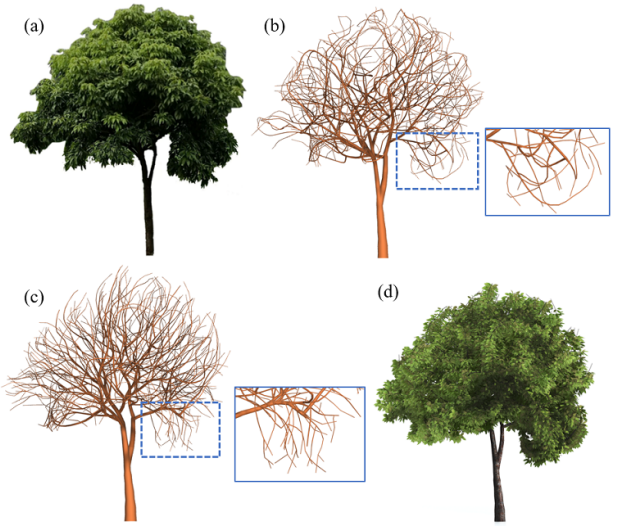


Figure 10: The effect of using attraction direction. (a) The input image. (b) Branches without attraction. (c) Branches with attraction. (d) Final tree model.

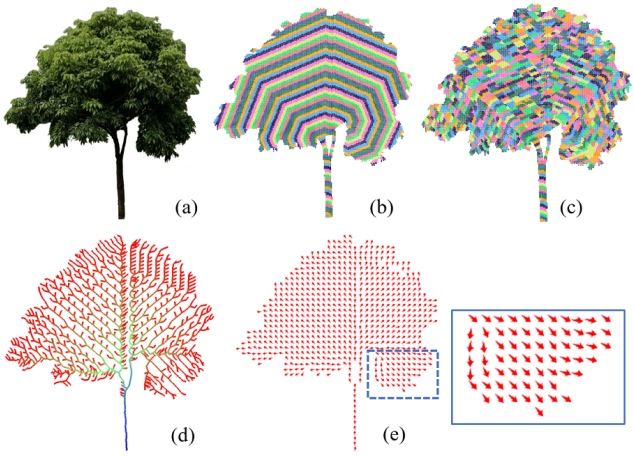


Figure 11: The process of computing the direction map. (a) The input image. (b) Preliminary bins. (c) Bins after the second time division. (d) The path formed by connecting the centroid points of adjacent bins. (e) The direction map.

its perception volume. The tree grows iteratively within the silhouette until there is no space available for buds to compete. Fig. 9 shows the resultant tree models at different iterations guided by the point cloud.

However, the distribution of above branches may be not consistent with the real photographs. As shown in Fig. 10 (b), the reconstructed branches are intricately filled in the dashed box area rather than growing downward as the input image. To solve this problem, we add an additional attraction direction $\vec{v}_{\text{trac}}$ for each marker based on a direction map. Given a tree image, we first treat each foreground pixel of this image as a 2D point so that a sparse 2D point cloud can be obtained. We then apply Dijkstra's shortest path algorithm to compute the shortest paths from the root to all other points. These points can be clustered into preliminary bins based on the length of the shortest paths (Fig. 11 (b)). We subsequently divide each bin for the second time (Fig. 11 (c)) based on the graph adjacency and a given interclass minimal spacing. A 2D path can be formed by connecting the centroids of adjacent bins (Fig. 11 (d)). Finally, the 2D direction of each pixel can be computed by averaging the directions of its neighbor branches based on the 2D skeleton. Fig. 11 (e) shows the final 2D direction map. To generate a 3D attraction direction for each marker, we simply rotate the 2D direction map around the central axis of the image to the plane where the marker is located. The attraction direction $\vec{v}_{\text{trac}}$ of each marker can be represented by the average direction of its neighborhood. After considering both the direction $\vec{v}_{\text{opt}}$ and $\vec{v}_{\text{trac}}$, the final growth direction of each bud can be expressed as:

$$\vec{v}_{\text{final}} = \vec{v}_{\text{opt}} + \alpha \frac{1}{N} \sum_{i}^{N} \vec{v}_{\text{trac}} + \beta \vec{v}_{\text{gravity}} \tag{8}$$

where $\vec{v}_{\text{gravity}} = (0, -1, 0)$ is a tropism vector. The default values of α and β are set to 0.6 and 0.2. Fig. 10 (c) and (d) shows the effect of using the attraction direction. The branches in dashed box area tend to grow downward as the input image.

Geometry construction and leaf population. The polygonal meshes of branches are reconstructed by a set of generalized cylinders. The branch diameters are computed basipetally and accumulated along the stem axes [31]. All branches are finally smoothed by cubic Hermite interpolation. To obtain the full tree models, we attach leaves on branches according to specified tree species. The users can also adjust the leaf/trunk textures and the parameters of procedural generation to enhance visual appearance according to the photos.

5. Experimental Results

In this section, we conduct a number of experiments on various examples to demonstrate the efficacy of our approach. We first evaluate the result of 3D shapes predicted by cGAN and then conduct experiments by reconstructing tree models from real images. We finally provide a comparison to the state-of-the-art approaches. All shown results are obtained on a desktop computer equipped with an Intel i7-7700HQ processor clocked at 2.8GHz, 8GB of RAM.

Training settings. Offline training runs on an NVIDIA GeForce GTX 1080 Ti (11GB memory) GPU. We implemented our depth inference algorithm in PyTorch and Python. We trained our network using $20k$ image pairs for both skeletons and silhouettes. By using different parameters of procedural modeling [31], we generate 20 species of trees, such as pines, maples, oaks, etc. Each tree species contains 1000 models with rich structural variations. 90% of the tree examples are deciduous trees while the rest are coniferous. Some species that cannot be synthesized by our procedural modeling algorithm, such as palms are not included in the dataset. The training process takes about 4 days, while in the testing phase, generating one depth image needs only about 0.07 seconds through our network.

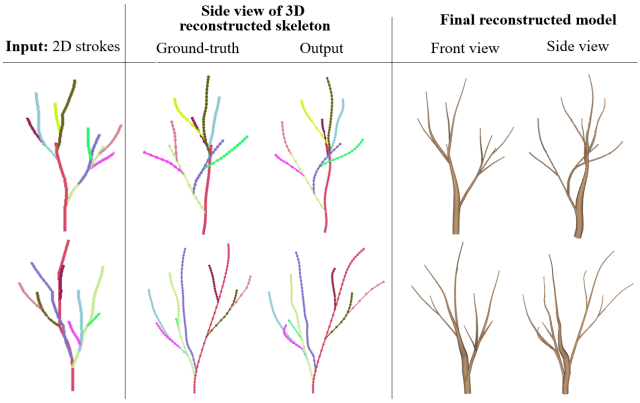


Figure 12: Examples of reconstructing 3D skeletons and models from input 2D strokes.

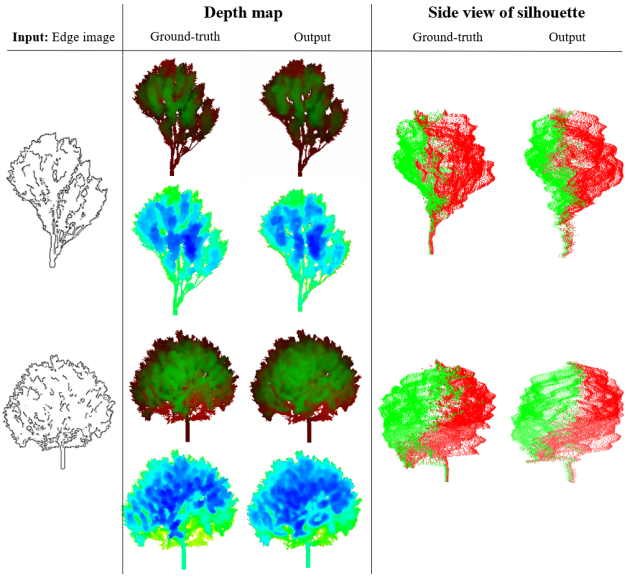


Figure 13: 3D silhouettes reconstruction (from left to right): the input edges, depth images (top is our two view representations and bottom is the depth image of front view with color coding), side view of reconstructed 3D silhouettes.

5.1. Evaluation

To evaluate the accuracy of the inferred 3D structure, we also generate $2K$ testing images from above 20 species (100 images per species) where the branching parameters are different from those used in the training dataset. These testing examples provide the ground truth branches for both qualitative and quantitative evaluation.

Fig. 12 shows two examples of 2D input strokes and their reconstructed skeletons. By observing the branches from side view (second column), we find that the output branches faithfully keep their original relative positions as the ground truth. Fig. 13 illustrates some results of the 3D silhouettes predicted from input edges. The second column shows the depth images of silhouettes (top) and the color coding depth images of front view (bottom). The bottom color coding depth images only visualize the depths of red channel (front part of silhouette), which can help to compare the depth differences more clearly. The color coding depth image and the side view of silhouettes demonstrate that the fluctuation of the silhouette can be kept roughly by our algorithm. That is to say, the network can keep the skeletons and silhouettes consistent well with ground truth from the front view, and give a reasonable and approximate predicted depth. Besides, the holes in the silhouette can be recovered and maintained, and the branches and foliage regions are easy to distinguish.

Next, we evaluate the outputs against the ground truth using two quantitative measures: depth error and Hausdorff distance. The depth error is the average depth differences between the output depth images and ground truth depths. The average depth error over $2K$ testing images for skeleton and silhouette are 0.032 and 0.041, respectively (The depths lie in range $[0, 1]$). In addition, we adopt Hausdorff distance to measure the similarity between the reconstructed skeleton and ground-truth, where we take as input two point sets formed by the tree nodes of reconstructed and real skeletons. The average Hausdorff distance over the testing dataset is 0.098. The values of depth error and Hausdorff distance indicate our method is effective to maintain the branching fidelity of trees.

Figure 14: Reconstructing trees with a few leaves (a) and even no leaves (b). From left to right: photograph, input strokes, reconstructed models under front view and side view, and rendered models under the front view.

5.2. Modeling abilities

To further demonstrate the modeling abilities of our method, we conduct experiments on a set of real-world images by using the same cGAN model trained on the synthetic dataset. Our method can be effectively applied to reconstruct a variety of tree species (see Figs. 14 and 15). Two trees with a few or even no leaves are shown in Fig. 14. The network only takes as input 2D user-drawn strokes to recover plausible branching structures. Fig. 15 shows the reconstructed results of various species of leafy trees such as pines, maple trees, etc. All the resulting trees resemble the target photos faithfully. Fig. 15 (b) shows a maple tree with uneven density of foliage, where the leaves in the left part are denser than the right. Our method produces rightful distribution of leaves and branches since the messy lines in the left of edge image can reflect this feature well. Fig. 15 (d) shows a Sophora japonica tree, the side view of which preserves a

Figure 15: Modeling results of our method using real-world images. From left to right: photographs, extracted 2D edges and strokes, reconstructed branch structures, and the final full tree models rendered from front and side views.

Table 1
Time (in second) for modeling trees in Figs. 15 and 14. We report the time cost in each of the following steps: stroke drawing by users, skeleton and silhouette depth image generation, procedural tree modeling.

No.	Stroke Drawing	Skeleton Depth Gen.	Silhouette Depth Gen.	Tree Modeling
Fig. 14(a)	28.43	0.065	N/A	0.01
Fig. 14(b)	19.20	0.071	N/A	0.01
Fig. 15(a)	2.17	0.073	0.066	1.62
Fig. 15(b)	10.80	0.069	0.061	1.53
Fig. 15(c)	17.13	0.083	0.058	1.22
Fig. 15(d)	11.80	0.076	0.072	1.07
Fig. 15(e)	9.85	0.066	0.077	1.29
Fig. 15(f)	6.01	0.070	0.066	1.04

similar bent shape as its front view. That indicates our network is capable of extracting the underlying shape information from a single image. For the oak tree in Fig. 15 (e), an obvious inner hole is also well kept in the final tree model. Moreover, in Table 1 we report the computational time of each step for modeling the trees in Figs. 14 and 15. Inferring 3D skeleton and silhouette via cGAN is quite efficient, and procedural modeling to generate a full tree only needs about $1 \sim 2$ seconds. The supplemental video shows all the 3D reconstructed results in detail.

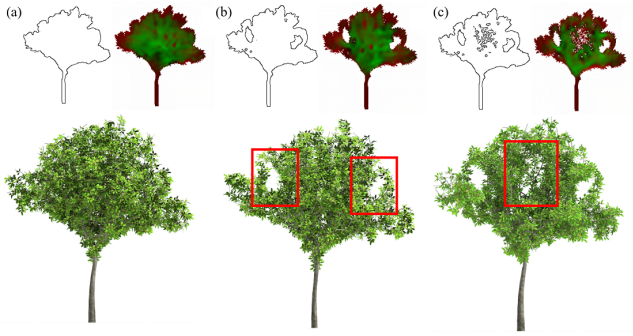


Figure 16: Different inner details can make effect on the tree structure. The trees are respectively with (a) no inner strokes, (b) two large holes, (c) some messy strokes. For each tree, we show the extracted edges, inferred depth image, and reconstructed model.

Fig. 16 illustrates the effect of different inner edge details on the tree structure. The clean outline edges will produce a leafy tree model as shown in Fig. 16 (a). Then we gradually add details into the edge image to observe the changes of reconstructed tree models. For the tree in Fig. 16 (b), two large holes in the edge image cause no branch or leaves generated in the corresponding region of the tree. In Fig. 16 (c), the density of branches and leaves is affected by the small messy lines in the middle of the edge image, since such lines infer discontinuous depth values which cause no adequate space (markers) for branches to grow.

In addition to the real-world images, our method can be used to model more complicated examples, as shown in Fig. 17. Our output trees faithfully reflect the shapes designed by the user. It demonstrates the generalization ability of our method.

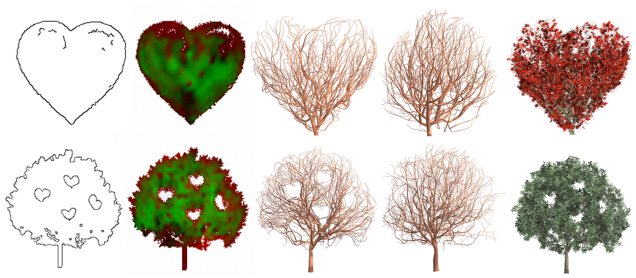


Figure 17: Ornamental tree models created from the shapes designed by a user.

5.3. Comparison to the state-of-the-art

In Fig. 18 we compare our method with a recent work [14]. This method applies procedural modeling approach to generate tree models from the real point cloud that are reconstructed from multiple view images. First, to generate the modeling result shown in Fig. 18 (b), they take a long time (about 15 minutes) to compute 3D dense point cloud by exploiting a binocular stereovision approach. Our method takes as input the strokes and edges shown in Fig. 18 (c) and can reconstruct the corresponding tree models (Fig. 18 (d)) within seconds. Besides, due to lacking explicit branch control, [14] gets a wrong branch structure in the red box region. By comparison, our method can recover the visible main branches as the real image from the input strokes. We note that, since the strokes are encoded in different colors, the relative positions of reconstructed branches in the red box region are also consistent well with the input image.

We now compare with a single image based tree modeling method [47], which takes about 20 minutes to reconstruct the branches on a PC with 2.4GHz CPU, whereas our method only takes a few seconds with a 2.8GHz CPU. Since [47] only takes the outer boundary of a tree into account, their method doesn't perform well for the trees with sparse leaves or inner holes, while our method can adapt to such complex cases better. Fig. 19 compares the modeling result of a sparse tree between [47] and our method. Our method estimates the 3D silhouette and skeleton by using a neural network trained on a dataset, so that the side view is changeable. However, the tree generated by [47] always looks similar between front and side views. Moreover, to get 3D visible branches from 2D strokes, the methods of [47] and [29] simply rotate the branches with assumption that the distance between branches is as large as possible, which is too simple to describe the real skeletons. Our method can infer more accurate 3D skeleton from 2D strokes by using the network.Our method can also handle harder cases as shown in Fig. 17. The first row shows the tree reconstructed from a heart shape. The second row shows the tree with several inner heart holes. It is difficult for [47] to model such cases.

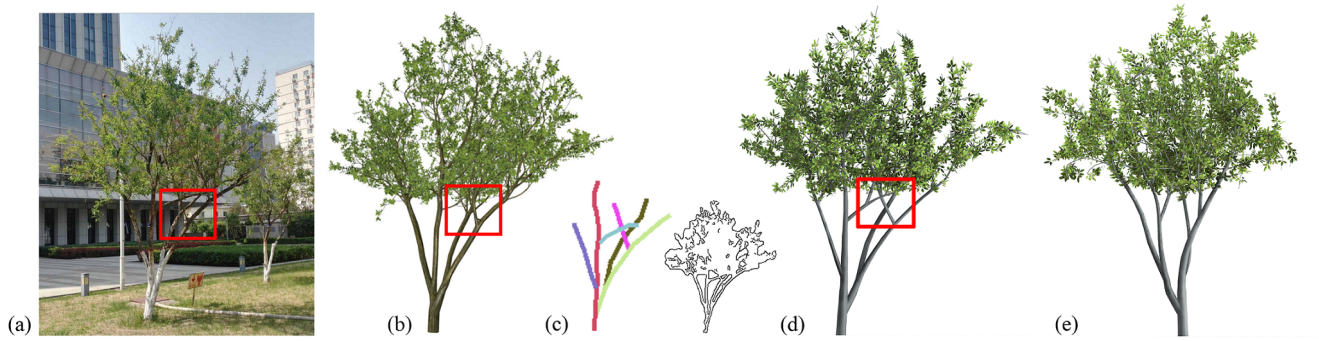


Figure 18: Comparison with Guo et al. [14] which generates tree models from multiple view images. Note that our method can more accurately recover the relative positions of the visible branches in the red box. (a) The input image. (b) Tree model generated by [14]. (c) The input strokes and edges of our method. (d) Our modeling result from front view. (e) Side view of our result.

Figure 19: Comparison with Tan et al. [47]. (a) Real tree image. (b) Modeling result of [47]. (c) Edges and strokes. (d) Front view of our result. (e) Side view of our result.

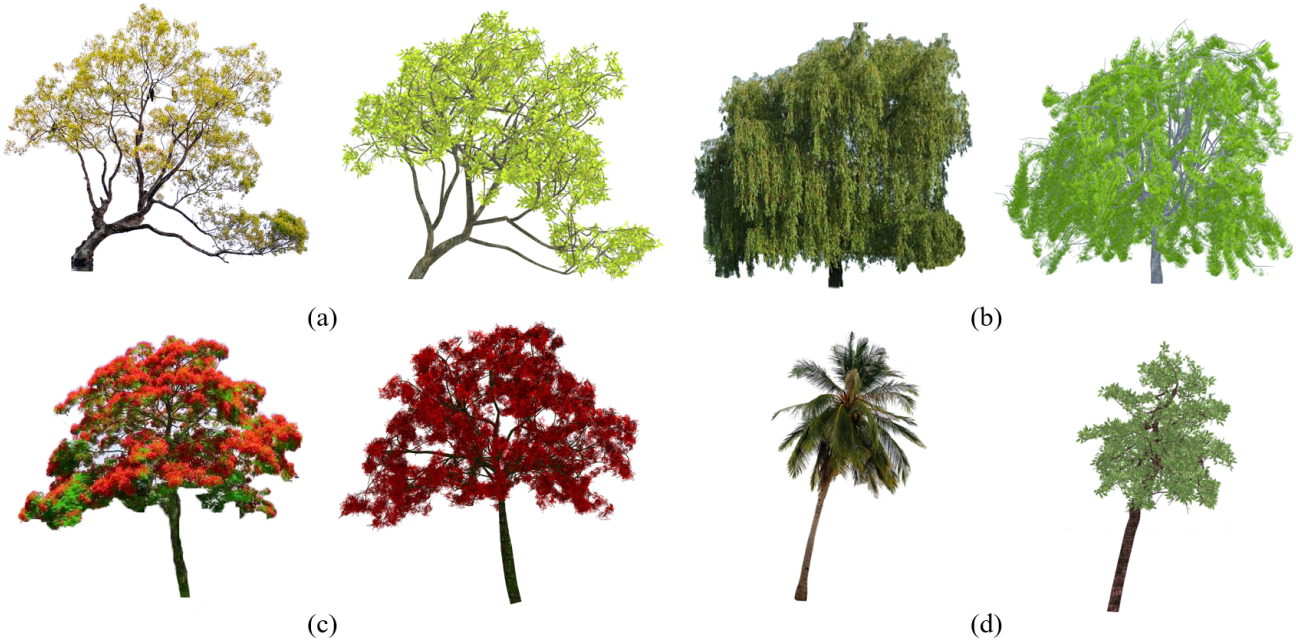


Figure 20: Modeling results of some difficult cases. (a) A tree with very sparse leaves and twigs. (b) A weeping willow. (c) A Delonix tree composed of various types of leaves. (d) A failure case of a palm tree, the species of which wasn't included in the training set.

5.4. Limitations

We successfully reconstruct large sets of tree models comprising a high variation of species from a single image using the proposed approach. However, our method also has some limitations. First, the procedural modeling algorithm is hard to recover the full details of some trees with special forms, especially for those that do not exist in the training set. Fig. 20 (a) shows a tree with very sparse foliage. In this example, the procedural generation cannot faithfully propagate small twigs and leaves in accordance with the target image. Moreover, for the weeping tree in Fig. 20 (b), it is difficult to make the drooping branches grow entirely accorded with the input image.

Second, the cGAN network only predicts the overall 3D tree shapes, but no more further details on tree species and foliage textures. For the Delonix tree in Fig. 20 (c), which has both red and green leaves, our method cannot effectively recover the actual textures of its foliage. In addition, Fig. 20 (d) also shows a failure case when trying to reconstruct a palm tree, however, the species of which is not included in the training set.

6. Conclusion and Future Work

We have presented a new approach to reconstruct realistic tree models from a single image. The core idea of our method is the combination of deep learning and procedural modeling. We utilize two parallel representations to describe the geometric shape of one tree: a 3D silhouette defines an approximate space that covers the whole tree, while a 3D skeleton represents the visible trunk precisely. By representing such two shapes as two depth images, we formulate the 2D-to-3D inference problem as solving an image-to-image translation task using a cGAN network. Based on the predicted 3D skeleton, the full tree model can be generated in the space defined by the 3D silhouette using procedural modeling techniques. What's more, it is easy to extend our approach into an interactive sketch-based tree modeling system, i.e., users can generate various tree models by drawing the silhouette or skeleton strokes interactively without input images. We have shown the effectiveness of our method in reconstructing convincing 3D tree models by various input tree images. Compared with previous single image based modeling methods, our approach is much faster and more robust for modeling trees with different types.

Several problems remain open for future research. We would like to develop an integrated neural network that can predict not only the 3D shapes but some information on tree species. Furthermore, an important extension would be automatic extraction of the textures and visible trunks.

Acknowledgment

This work is partially funded by the NSFC (61972388, 61761003, 61802406), Shenzhen Basic Research Program (JCYJ20180507182222355), the Leading Talents of Guangdong Program (00201509), and the CAS grant (GJHZ1862).

References

[1] Argudo, O., Chica, A., Andujar, C., 2016. Single-picture reconstruction and rendering of trees for plausible vegetation synthesis. Computers & Graphics 57, 55–67.

[2] Beneš, B., Andrysco, N., Šťava, O., 2009. Interactive modeling of virtual ecosystems, in: Proceedings of the Fifth Eurographics conference on Natural Phenomena, pp. 9–16.

[3] Bloomenthal, J., 1985. Modeling the mighty maple. SIGGRAPH Comput. Graph. 19, 305–311.

[4] Bradley, D., Nowrouzezahrai, D., Beardsley, P., 2013. Image-based reconstruction and synthesis of dense foliage. ACM Trans. on Graphics (Proc. SIGGRAPH) 32, 74.

[5] Canny, J., 1986. A computational approach to edge detection. IEEE Trans. on Pattern Analysis and Machine Intelligence , 679–698.

[6] Choi, Y., Choi, M., Kim, M., Ha, J.W., Kim, S., Choo, J., 2018. Stargan: Unified generative adversarial networks for multi-domain image-to-image translation, in: IEEE Computer Vision and Pattern Recognition (CVPR), pp. 8789–8797.

[7] Deussen, O., Lintermann, B., 2006. Digital design of nature: computer generated plants and organics. Springer Science & Business Media.

[8] Diener, J., Reveret, L., Fiume, E., 2006. Hierarchical retargetting of 2d motion fields to the animation of 3d plant models, in: Proceedings of the 2006 ACM SIGGRAPH/Eurographics symposium on Computer animation, Eurographics Association. pp. 187–195.

[9] Friedman, S., Stamos, I., 2013. Automatic procedural modeling of tree structures in point clouds using wavelets, in: International Conference on 3D Vision, IEEE. pp. 215–222.

[10] Girdhar, R., Fouhey, D.F., Rodriguez, M., Gupta, A., 2016. Learning a predictable and generative vector representation for objects, in: European Conference on Computer Vision, Springer. pp. 484–499.

[11] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y., 2014. Generative adversarial nets, in: Advances in neural information processing systems, pp. 2672–2680.

[12] Guénard, J., Morin, G., Boudon, F., Charvillat, V., 2013. Reconstructing Plants in 3D from a Single Image Using Analysis-by-Synthesis, in: Advances in Visual Computing. volume 8033 of *Lecture Notes in Computer Science*, pp. 322–332.

[13] 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 Trans. on Graphics 39, 1–13.

[14] Guo, J., Xu, S., Yan, D.M., Cheng, Z., Jaeger, M., Zhang, X., 2018. Realistic procedural plant modeling from multiple view images. IEEE Trans. on Vis. and Comp. Graphics 26, 1372–1384.

[15] Honda, H., 1971. Description of the form of trees by the parameters of the tree-like body: Effects of the branching angle and the branch length on the shape of the tree-like body. Journal of Theoretical Biology 31, 331 – 338.

[16] Huang, H., Kalogerakis, E., Yumer, E., Mech, R., 2017. Shape synthesis from sketches via procedural models and convolutional networks. IEEE Trans. on Vis. and Comp. Graphics 23, 2003–2013.

[17] Isokane, T., Okura, F., Ide, A., Matsushita, Y., Yagi, Y., 2018. Probabilistic plant modeling via multi-view image-to-image translation, in: IEEE Computer Vision and Pattern Recognition (CVPR).

[18] Isola, P., Zhu, J.Y., Zhou, T., Efros, A.A., 2017. Image-to-image translation with conditional adversarial networks, in: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 1125–1134.

[19] Kim, T., Cha, M., Kim, H., Lee, J.K., Kim, J., 2017. Learning to discover cross-domain relations with generative adversarial networks, in: Proceedings of the 34th International Conference on Machine Learning-Volume 70, JMLR. org. pp. 1857–1865.

[20] Kingma, D.P., Ba, J., 2014. Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 .

[21] Li, C., Deussen, O., Song, Y.Z., Willis, P., Hall, P., 2011. Modeling and generating moving trees from video. ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 30, 127:1–127:12.

[22] Li, Y., Fan, X., Mitra, N.J., Chamovitz, D., Cohen-Or, D., Chen, B., 2013. Analyzing growing plants from 4d point cloud data. ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 32, 157:1–157:10.

[23] Lindenmayer, A., 1968. Mathematical models for cellular interactions in development. ii. simple and branching filaments with two-sided inputs. Journal of Theoretical Biology 18, 300–315.

[24] Livny, Y., Pirk, S., Cheng, Z., Yan, F., Deussen, O., Cohen-Or, D., Chen, B., 2011. Texture-lobes for tree modeling, in: ACM Trans. on Graphics (Proc. SIGGRAPH), p. 1.

[25] Livny, Y., Yan, F., Olson, M., Chen, B., Zhang, H., El-Sana, J., 2010. Automatic reconstruction of tree skeletal structures from point clouds. ACM Trans. on Graphics (Proc. SIGGRAPH) 29, 151.

[26] Lun, Z., Gadelha, M., Kalogerakis, E., Maji, S., Wang, R., 2017. 3d shape reconstruction from sketches via multi-view convolutional networks, in: International Conference on 3D Vision (3DV), pp. 67–77.

[27] Mirza, M., Osindero, S., 2014. Conditional generative adversarial nets. Computer Science , 2672–2680.

[28] Neubert, B., Franken, T., Deussen, O., 2007. Approximate image-based tree-modeling using particle flows. ACM Trans. on Graphics (Proc. SIGGRAPH) 26.

[29] Okabe, M., Owada, S., Igarash, T., 2005. Interactive design of botanical trees using freehand sketches and example-based editing. Computer Graphics Forum (Proc. EUROGRAPHICS) 24, 487–496.

[30] Oppenheimer, P.E., 1986. Real time design and animation of fractal plants and trees. SIGGRAPH Comput. Graph. 20, 55–64.

[31] Palubicki, W., Horel, K., Longay, S., Runions, A., Mech, R., Prusinkiewicz, P., 2009. Self-organizing tree models for image synthesis. ACM Trans. on Graphics (Proc. SIGGRAPH) 28.

[32] Pfeifer, N., Gorte, B., Winterhalder, D., et al., 2004. Automatic reconstruction of single trees from terrestrial laser scanner data, in: Proceedings of 20th ISPRS Congress, pp. 114–119.

[33] Pirk, S., Stava, O., Kratt, J., Said, M.A.M., Neubert, B., Měch, R., Benes, B., Deussen, O., 2012. Plastic trees: Interactive self-adapting botanical tree models. ACM Trans. on Graphics (Proc. SIGGRAPH) 31, 50:1–50:10.

[34] Prusinkiewicz, P., Hammel, M., Hanan, J., Mech, R., 1996. L-systems: from the theory to visual models of plants, in: Proceedings of the 2nd CSIRO Symposium on Computational Challenges in Life Sciences, pp. 1–32.

[35] Prusinkiewicz, P., James, M., Měch, R., 1994. Synthetic topiary, in: Proceedings of the 21st Annual Conference on Computer Graphics and Interactive Techniques, pp. 351–358.

[36] Prusinkiewicz, P., Lindenmayer, A., 1990. The Algorithmic Beauty of Plants. Springer-Verlag New York, Inc., New York, USA.

[37] Quan, L., Tan, P., Zeng, G., Yuan, L., Wang, J., Kang, S.B., 2006. Image-based plant modeling, in: ACM Trans. on Graphics (Proc. SIGGRAPH), pp. 599–604.

[38] Raumonen, P., Kaasalainen, M., Åkerblom, M., Kaasalainen, S., Kaartinen, H., Vastaranta, M., Holopainen, M., Disney, M., Lewis, P., 2013. Fast automatic precision tree models from terrestrial laser scanner data. Remote Sensing 5, 491.

[39] Reche-Martinez, A., Martin, I., Drettakis, G., 2004. Volumetric reconstruction and interactive rendering of trees from photographs. ACM Trans. on Graphics (Proc. SIGGRAPH) 23, 720–727.

[40] de Reffye, P., Edelin, C., Françon, J., Jaegerl, M., Puech, C., 1988. Plant models faithful to botanical structure and development. ACM SIGGRAPH 22, 151–158.

[41] Rezende, D.J., Eslami, S.A., Mohamed, S., Battaglia, P., Jaderberg, M., Heess, N., 2016. Unsupervised learning of 3d structure from images, in: Advances in Neural Information Processing Systems, pp. 4996–5004.

[42] Ronneberger, O., Fischer, P., Brox, T., 2015. U-net: Convolutional networks for biomedical image segmentation, in: International Conference on Medical image computing and computer-assisted intervention, Springer. pp. 234–241.

[43] Runions, A., Lane, B., Prusinkiewicz, P., 2007. Modeling trees with a space colonization algorithm, in: Proceedings of the Third Eurographics Conference on Natural Phenomena, pp. 63–70.

[44] Shlyakhter, I., Rozenoer, M., Dorsey, J., Teller, S., 2001. Reconstructing 3d tree models from instrumented photographs. IEEE Comput. Graph. Appl. 21, 53–61.

[45] Sinha, A., Unmesh, A., Huang, Q., Ramani, K., 2017. Surfnet: Generating 3d shape surfaces using deep residual networks, in: IEEE Computer Vision and Pattern Recognition (CVPR), pp. 6040–6049.

[46] Stava, O., Pirk, S., Kratt, J., Chen, B., Mech, R., Deussen, O., Benes, B., 2014. Inverse procedural modelling of trees. Computer Graphics Forum 33, 118–131.

[47] Tan, P., Fang, T., Xiao, J., Zhao, P., Quan, L., 2008. Single image tree modeling. ACM Trans. on Graphics (Proc. SIGGRAPH Asia) 27, 108:1–108:7.

[48] Tan, P., Zeng, G., Wang, J., Kang, S.B., Quan, L., 2007. Image-based tree modeling, in: ACM Trans. on Graphics (Proc. SIGGRAPH), p. 87.

[49] Weber, J., Penn, J., 1995. Creation and rendering of realistic trees, ACM. pp. 119–128.

[50] Wu, J., Zhang, C., Xue, T., Freeman, B., Tenenbaum, J., 2016. Learning a probabilistic latent space of object shapes via 3d generative-adversarial modeling, in: Advances in neural information processing systems, pp. 82–90.

[51] Xu, H., Gossett, N., Chen, B., 2007. Knowledge and heuristic-based modeling of laser-scanned trees. ACM Trans. on Graphics 26, 19.

[52] Yan, D.M., Wintz, J., Mourrain, B., Wang, W., Boudon, F., Godin, C., 2009. Efficient and robust reconstruction of botanical branching structure from laser scanned points, in: Proc. 11th IEEE Int. Conf. CAD/Graph. Comput., pp. 572–575.

[53] Yi, L., Li, H., Guo, J., Deussen, O., Zhang, X., 2018. Tree growth modelling constrained by growth equations, in: Computer Graphics Forum, Wiley Online Library. pp. 239–253.

[54] Zhang, X., Li, H., Dai, M., Ma, W., Quan, L., 2014. Data-driven synthetic modeling of trees. IEEE Trans. on Vis. and Comp. Graphics 20, 1214–1226.

[55] Zhu, J.Y., Park, T., Isola, P., Efros, A.A., 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks, in: IEEE International Conference on Computer Vision (ICCV), pp. 2223–2232.