Three Dimensional Reconstruction of Botanical Trees with Simulatable Geometry

ED QUIGLEY, NVIDIA Corporation, USA
WINNIE LIN, Stanford University, USA
YILIN ZHU, Stanford University, USA
RONALD FEDKIW, Stanford University, USA

We tackle the challenging problem of creating full and accurate three dimensional reconstructions of botanical trees with the topological and geometric accuracy required for subsequent physical simulation, e.g. in response to wind forces. Although certain aspects of our approach would benefit from various improvements, our results exceed the state of the art especially in geometric and topological complexity and accuracy. Starting with two dimensional RGB image data acquired from cameras attached to drones, we create point clouds, textured triangle meshes, and a simulatable and skinned cylindrical articulated rigid body model. We discuss the pros and cons of each step of our pipeline, and in order to stimulate future research we make the raw and processed data from every step of the pipeline as well as the final geometric reconstructions publicly available.

Additional Key Words and Phrases: 3D reconstruction, semantic segmentation, botanical trees

ACM Reference Format:
Ed Quigley, Winnie Lin, Yilin Zhu, and Ronald Fedkiw. 2021. Three Dimensional Reconstruction of Botanical Trees with Simulatable Geometry . *Proc. ACM Comput. Graph. Interact. Tech.* 4, 3 (September 2021), 16 pages. https://doi.org/10.1145/3480146

1 INTRODUCTION

Human-inhabited outdoor environments typically contain ground surfaces such as grass and roads, transportation vehicles such as cars and bikes, buildings and structures, and humans themselves, but are also typically intentionally populated by a large number of trees and shrubbery; most of the motion in such environments comes from humans, their vehicles, and wind-driven plants/trees. Tree reconstruction and simulation are obviously useful for AR/VR, architectural design and modeling, film special effects, etc. For example, when filming actors running through trees, one would like to create virtual versions of those trees with which a chasing dinosaur could interact. Other uses include studying roots and plants for agriculture [Estrada et al. 2015; Fuentes et al. 2017; Zheng et al. 2011] or assessing the health of trees especially in remote locations (similar in spirit to [Zuffi et al. 2018]). 2.5D data, i.e. 2D images with some depth information, is typically sufficient for robotic navigation, etc.; however, there are many problems that require true 3D scene understanding to the extent one could 3D print objects and have accurate geodesics. Whereas navigating around objects might readily generalize into categories or strategies such as 'move left,' 'move right,' 'step up,' 'go under,' etc., the 3D object understanding required for picking up a cup, knocking down

Authors' addresses: Ed Quigley, equigley@nvidia.com, NVIDIA Corporation, Santa Clara, CA, USA; Winnie Lin, winnielin@stanford.edu, Stanford University, Stanford, CA, USA; Yilin Zhu, yilinzhu@stanford.edu, Stanford University, Stanford, CA, USA; Ronald Fedkiw, rfedkiw@stanford.edu, Stanford University, Stanford, CA, USA.

2577-6193/2021/9-ART $15.00
https://doi.org/10.1145/3480146

a building, moving a stack of bricks or a pile of dirt, or simulating a tree moving in the wind requires significantly higher fidelity. As opposed to random trial and error, humans often use mental simulations to better complete a task, e.g. consider stacking a card tower, avoiding a falling object, or hitting a baseball (visualization is quite important in sports); thus, physical simulation can play an important role in end-to-end tasks, e.g. see [Jiang and Liu 2018; Kloss et al. 2017; Peng et al. 2017] for examples of combining simulation and learning.

Accurate 3D shape reconstruction is still quite challenging. Recently, Malik argued[1] that one should not apply general purpose reconstruction algorithms to say a car and a tree and expect both reconstructions to be of high quality. Rather, he said that one should use domain-specific knowledge as he has done for example in [Kanazawa et al. 2018]. Another example of this specialization strategy is to rely on the prior that many indoor surfaces are planar in order to reconstruct office spaces [Huang et al. 2017] or entire buildings [Armeni et al. 2017, 2016]. Along the same lines, [Zuffi et al. 2018] uses a base animal shape as a prior for their reconstructions of wild animals. Thus, we similarly take a specialized approach using a generalized cylinder prior for both large and medium scale features.

In Section 3, we discuss our constraints on data collection as well as the logistics behind the choices we made for the hardware (cameras and drones) and software (structure from motion, multi-view stereo, inverse rendering, etc.) used to obtain our raw and processed data. Sections 4, 5, and 6 then describe how we create geometry from that data with enough efficacy for physical simulation. Section 7 discusses our use of machine learning, and Section 8 presents a number of experimental results.

2 PREVIOUS WORK

Tree Modeling and Reconstruction: Researchers in computer graphics have been interested in modeling trees and plants for decades [Bloomenthal 1985; Lindenmayer 1968; Prusinkiewicz et al. 1997; Stava et al. 2014; Weber and Penn 1995]. SpeedTree[2] is probably the most popular software utilized, and their group has begun to consider the incorporation of data-driven methods. Amongst the data-driven approaches, [Tan et al. 2007] is most similar to ours combining point cloud and image segmentation data to build coarse-scale details of a tree; however, they generate fine-scale details procedurally using a self-similarity assumption and image-space growth constraints, whereas we aim to capture more accurate finer structures from the image data. Other data-driven approaches include [Livny et al. 2010] which automatically estimates skeletal structure of trees from point cloud data, [Xie et al. 2015] which builds tree models by assembling pieces from a database of scanned tree parts, [Li et al. 2013] which tracks plant structure over time as the plant grows, etc.

Many of these specialized, data-driven approaches for trees are built upon more general techniques such as the traditional combination of structure from motion (see e.g. [Wu 2013]) and multi-view stereo (see e.g. [Furukawa and Ponce 2010]). In the past, researchers studying 3D reconstruction have engineered general approaches to reconstruct fine details of small objects captured by sensors in highly controlled environments [Seitz et al. 2006]. At the other end of the spectrum, researchers have developed approaches for reconstructing building- or even city-scale objects using large amounts of image data available online [Agarwal et al. 2009]. Our goal is to obtain a 3D model of a tree with elements from both of these approaches: the scale of a large structure with the fine details of its many branches and twigs. However, unlike in general reconstruction approaches, we cannot simply collect images online or capture data using a high-end camera.

[1] Jitendra Malik, Stanford cs231n guest lecture, 29 May 2018
[2] `https://speedtree.com`

Fig. 1. We target a California oak for reconstruction and simulation. (Inset) The drone and camera setup used to collect video data of the tree.

To address similar challenges in specialized cases, researchers take advantage of domain-specific prior knowledge. [Zhou et al. 2008] uses a generalized cylinder prior (similar to us) for reconstructing tubular structures observed during medical procedures and illustrates that this approach performs better than simple structure from motion. The process of creating a mesh that faithfully reflects topology and subsequently refining its geometry is similar in spirit to [Xu et al. 2018], which poses a human model first via its skeleton and then by applying fine-scale deformations.

Learning and Networks: So far, our use of networks is limited to segmentation tasks, where we rely on segmentation masks for semi-automated tree branch labeling. Due to difficulties in getting sharp details from convolutional networks, the study of network-based segmentation of thin structures is still an active field in itself; there has been recent work on designing specialized multiscale architectures [Lin et al. 2017; Qu et al. 2018; Ronneberger et al. 2015] and also on incorporating perceptual losses [Johnson et al. 2016] during network training [Mosinska et al. 2018].

3 RAW AND PROCESSED DATA

As a case study, we select a California oak (*quercus agrifolia*) as our subject for tree reconstruction and simulation (see Figure 1). The mere size of this tree imposes a number of restrictions on our data capture: one has to deal with an outdoor, unconstrained environment, wind and branch motion will be an issue, it will be quite difficult to observe higher up portions of the tree especially at close proximities, there will be an immense number of occluded regions because of the large number of branches that one cannot see from any feasible viewpoint, etc.

In an outdoor setting, commodity structured light sensors that use infrared light (e.g. the Kinect) fail to produce reliable depth maps as their projected pattern is washed out by sunlight; thus, we opted to use standard RGB cameras. Because we want good coverage of the tree, we cannot simply capture images from the ground; instead, we mounted our cameras on a quadcopter drone that was piloted around the tree. The decision to use a drone introduces additional constraints: the cameras must be lightweight, the camera locations cannot be known *a priori*, the drone creates its own air currents which can affect the tree's motion, etc. Balancing the weight constraint with the benefits of using cameras with a global shutter and minimal distortion, we mounted a pair of Sony rx100 v

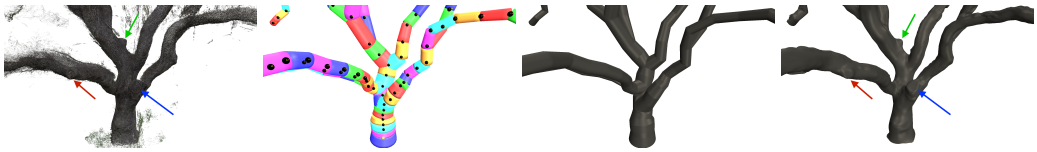

Fig. 2. Point cloud data (far left) is used as a guide for interactively placing generalized cylinders which are readily simulatable as articulated rigid bodies (middle left). The generalized cylinders are connected together into a skinned mesh (middle right), which is then perturbed based on the point cloud data to capture finer scale geometric details (far right). Notice the stumps that appear in the perturbed mesh reflecting the point cloud data (marked with arrows).

cameras to a DJI Matrice 100 drone. We calibrated the stereo offset between the cameras before flight, and during flight each camera records a video with 4K resolution at 30 fps.

Data captured in this manner is subject to a number of limitations. Compression artifacts in the recorded videos may make features harder to track than when captured in a RAW format. Because the drone must keep a safe distance from the tree, complete 360° coverage of a given branch is often infeasible. This lack of coverage is compounded by occlusions caused by other branches and leaves (in seasons when the latter are present). Furthermore, the fact that the tree may be swaying slightly in the wind even on a calm day violates the rigidity assumption upon which many multi-view reconstruction algorithms rely. Since we know from the data collection phase that our data coverage will be incomplete, we will need to rely on procedural generation, inpainting, "hallucinating" structure, etc. in order to complete the model.

After capturing the raw data, we augment it to begin to estimate the 3D structure of the environment. We subsample the videos at a sparse 1 or 2 fps and use the Agisoft PhotoScan tool[3] to run structure from motion and multi-view stereo on those images, yielding a set of estimated camera frames and a dense point cloud. We align cameras and point clouds from separate structure from motion problems by performing a rigid fit on a sparse set of control points. This is a standard workflow also supported by open-source tools [Moulon et al. 2016; Schönberger and Frahm 2016; Wu 2011]. Some cameras may be poorly aligned (or in some cases, so severely incorrect that they require manual correction). Once the cameras are relatively close, one can utilize an inverse rendering approach like that of [Loper and Black 2014] adjusting the misaligned cameras' parameters relative to the point cloud. In the case of more severely misaligned cameras, one may select correspondences between 3D points and points in the misaligned image and then find the camera's extrinsics by solving a perspective-n-point problem [Fischler and Bolles 1981].

4 BUILDING A SIMULATABLE TREE

Recent work enables the simulation of highly detailed trees modeled as articulated rigid bodies at real-time or interactive speeds (9.5k rigid bodies simulated at 86 frames/sec, or 3 million rigid bodies simulated at 2.3 sec/frame) [Quigley et al. 2018]. The scalability of this method makes the simulation of rich, highly detailed geometric models of real-world trees feasible. In order to apply such a method to our reconstruction, we need to create articulated rigid bodies with masses and inertia tensors connected via springs with stiffnesses and damping coefficients (of course, such a representation could also be readily adapted to other solvers such as that of [Müller and Chentanez 2011]). Unfortunately, point clouds, triangle soups with holes, and other similar 3D representations are not readily amenable to such an approach (although one can of course build tools to organize existing geometry into simulatable structures as in [Zhao and Barbič 2013]).

[3] Agisoft PhotoScan, http://www.agisoft.com/

Commonly used techniques such as Poisson surface reconstruction [Kazhdan et al. 2006] produce potentially disconnected meshes that do not respect the topology of the underlying tree, and are thus not well-suited for simulation. In order to create a simulatable reconstructed tree, we make a strong prior assumption that the tree reconstruction consists of a number of generalized cylinders as underlying building blocks, appropriately skinned to provide smooth interconnections, and subsequently modified to provide the desired geometric detail.

We create the generalized cylinders interactively using the point cloud data obtained via multi-view stereo as a guide. An initial cylinder is positioned at the base of the tree's trunk, then a second cylinder is attached to the first by a common endpoint, and so on, progressively "growing" the generalized cylinder model of the tree. Each cylinder endpoint may be connected to zero, one, or two subsequent cylinders to model a branch ending, curving, or bifurcating, respectively. A radius is also specified for each endpoint of the generalized cylinders. After approximating the trunk and branches using this generalized cylinder basis, the surfaces of the generalized cylinders are skinned together into a single contiguous triangle mesh representing the exterior surface of the tree. Although this model only roughly captures the geometry using the radii of the generalized cylinders as estimates of the tree's cross-sectional thicknesses, the advantage of this representation is that the model has a topology consistent with the real tree. The generalized cylinders are also readily simulated in order to drive deformations of the skinned mesh. See Figure 2.

Although topologically accurate, the skinned generalized cylinders miss much of the rich geometric structure of the tree that is captured in part by the point cloud data. Thus we augment the generalized cylinder representation in a manner informed by the point cloud. For each vertex on the contiguous skinned mesh, we construct a cylindrical sampling region normal to the mesh; then, the average position of the point cloud points that fall within this sampling region is used to perturb the vertex in its normal direction, essentially creating a 2D height field with respect to the skinned mesh. That is, the point cloud informs a displacement map [Cook 1984].

Some vertices may have no point cloud data within their respective sampling regions; often the point cloud data only models one side of a branch, so only vertices on that side of the skin mesh are perturbed. To avoid sharp discontinuities in the perturbed mesh, we solve Laplace's equation for the heightfield displacements of the vertices with empty sampling regions using the adequately perturbed heights as Dirichlet boundary conditions. To obtain a visually desirable mesh, one can additionally utilize Laplacian smoothing (e.g. [Desbrun et al. 1999; Taubin 1995]), vertex normal smoothing, Loop subdivision [Loop 2001], etc., as well as point cloud subset selection interleaved with additional perturbations along the resulting vertex normal directions. In fact, extending image inpainting ideas [Bertalmio et al. 2000] to the height fields on the two dimensional mesh surface (i.e. geometric inpainting) would likely give the best results, especially if informed from other areas of the tree where the geometry is more readily ascertained.

Fig. 3. (Left) We successfully reconstruct twig geometry using a traditional structure from motion and multi-view stereo pipeline under favorable conditions: complete coverage, indoor lighting, rigid geometry, etc. (Right) In the wild, difficult conditions preclude the ability for such reconstructions.

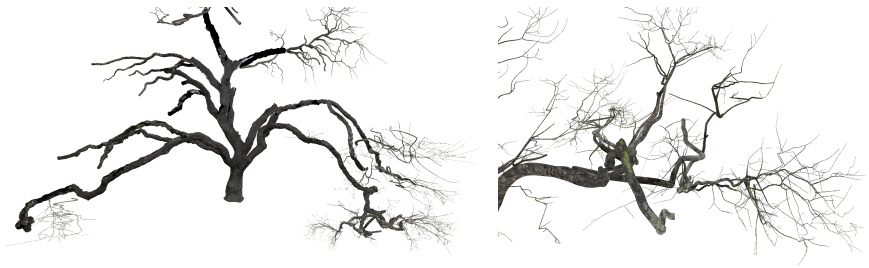

Fig. 4. (Left) Triangle meshes recovered from point cloud data (Section 4) and image annotations (Section 8). (Right) A close-up view of medium scale branches.

Finally, the mesh is textured by assigning each post-perturbation vertex the color of its nearest point cloud point. Note that we do not use an average of the nearby data as this tends to wash out the texture details. Here, image inpainting can also be used to fill in regions that have no point cloud data for textures.

5 MEDIUM SCALE BRANCHES

The aforementioned process fails on parts of the tree for which there is insufficient point cloud data (or no point cloud data at all). Although traditional structure from motion is sufficient for recovering fine twig details under favorable conditions, the process of capturing data from a real tree accrues errors that necessitate a specialized approach (see Figure 3). These errors may be attributed to many sources: some branches are heavily occluded by others, the drone cannot perform a full 360° sweep of most branches without other branches acting as obstacles, twig features are often only a few pixels wide when maintaining a safe distance between the drone and the tree, the tree may be nonrigidly deforming in the breeze even on a relatively calm day (or even due to the air currents generated by the drone itself), etc. The net effect of these sources of error is that our approach for creating the trunk and thicker branches of the tree is insufficient for the tree's finer structures that are not well-resolved by the point cloud data.

Thus, we switch to an image-based approach for finer structures. Our 3D generalized cylinder prior can be extended to 2D images by choosing projected radii and projected lengths of hypothetical 3D generalized cylinders. These 2D projections of generalized cylinders can be extended back to three spatial dimensions using multiple images. Whereas significant geometric detail on the thicker parts of the tree comes from geometric roughness on the skin of the cylinder as caused by knots, bark, etc. and is captured by our aforementioned perturbation process (see Section 4), the most significant geometric detail on thinner branches is often simple bending of their centerline. Thus, the fact that thinner branches lack adequate representation in the 3D point cloud is less consequential.

We employ an image annotation approach (see Section 8.1) to obtain image space labelings of branch and twig curves and "keypoints," or features that can be identified across multiple images. After annotating a number of images, we use the annotation data to recover 3D structures by triangulating keypoint positions, connecting 3D keypoints to match the topology of image space curves, and estimating the tree thickness for each 3D point (see Section 8.4). Then, we again create a contiguous skin mesh for these newly recovered 3D branches. In order to boost our ability to capture geometric changes in the centerline, we use the 3D positions as control points for a b-spline curve rather than directly meshing the piecewise linear segments. See Figure 4.

Fig. 5. (Left) Final simulatable geometry: articulated rigid bodies skinned and textured with the aid of the point cloud, along with thinner branch reconstructions with geometry and topology that follow the image data, all of which inform the motion of any reconstructed points and triangles that lack high enough fidelity reconstructions to form coherent structures. (Right) An RGB image of the actual tree from the same view.

Finally, we project texture information from the annotated images onto the skinned branches. For each vertex on the skinned mesh, we estimate its corresponding position within each corresponding annotated curve by measuring its fractional length along the curve's medial axis and its fractional thickness measured by projecting the vertex's distance from its 3D segment onto a plane parallel to the current image plane. For each such annotated curve we compute a quality estimate based on how close the corresponding camera is to the vertex and how closely aligned the vertex's surface normal is to the direction from the vertex to the annotated point. Since averaging smears out texture information, we assign each vertex the color with the maximum quality score.

6 UNRESOLVED STRUCTURE

Because the image annotations depend on human labelers, many of the tree's branches and twigs remain unmodeled even as more images are progressively covered; the automated and semi-automated approaches considered in Section 7 can help with this. In order to avoid discarding data, we additionally constrain the unstructured point cloud data obtained in Section 3 to the nearest generalized cylinders of the reconstructed model so that the point cloud deforms as the tree's rigid bodies move during simulation. This allows points from leaves, branches, and other structures that remain "orphaned" even after all possible generalized cylinders are created to contribute to the virtual tree's appearance and motion. See Figure 5.

7 ANNOTATION AND LEARNING

Annotating images is a challenging task for human labelers and automated methods alike. Branches and twigs heavily occlude one another, connectivity can be difficult to infer, and the path of even a relatively large branch can often not be traced visually from a single view. Thus it is desirable to augment the image data during annotation to aid human labelers.

One method for aiding the labeler is to automatically extract a "flow field" of vectors tracking the anisotropy of the branches in image space (see Figure 10). The flow field is overlaid on the image in the annotation tool, and the labeler may select endpoints to be automatically connected using the projection-advection scheme discussed in Section 8.3. Section 8.3 also discusses how we generate the flow field itself, after first creating a segmentation mask. Note that segmentation (i.e. discerning *tree* or *not tree* for each pixel in the image) is a simpler problem than annotation (i.e. discerning medial axes, topology, and thickness in image space).

Obtaining segmentation masks is straightforward under certain conditions, e.g. in areas where branches and twigs are clearly silhouetted against the grass or sky, but segmentation can be difficult

Fig. 6. Human labelers use our annotation tool to draw curves with positions, thicknesses, connectivities, and unique identifiers on images of the tree.

in visually dense regions of an image. Thus, we explore deep learning-based approaches for performing semantic segmentation on images from our dataset. In particular, we use U-Net [Ronneberger et al. 2015], a state-of-the-art fully convolutional architecture for segmentation; the strength of this model lies in its many residual connections, which give the model the capacity to retain sharp edges despite its hourglass structure. Note that U-Net has its origins in biomedical literature, where similar problems involving segmenting vascular structures in images are closely studied (see e.g. [Krissian et al. 2006]). See Section 8.2 for further discussion.

8 EXPERIMENTS

Since the approach to large scale structure discussed in Section 4 works well, we focus here on medium scale branches.

8.1 Image Annotation

We present a human labeler with an interface for drawing piecewise linear curves on an overlay of a tree image. User annotations consist of vertices with 2D positions in image space, per-vertex branch thicknesses, and edges connecting the vertices. Degree-1 vertices are curve endpoints, degree-2 vertices lie on the interior of a curve, and degree-3 vertices exist where curves connect. A subset of the annotated vertices are additionally given unique identifiers that are used to match common points between images; these will be referred to as “keypoints" and are typically chosen as bifurcation points or points on the tree that are easy to identify in multiple images. See Figure 6.

We take advantage of our estimated 3D knowledge of the tree’s environment in order to aid human labelers and move towards automatic labeling. After some annotations have been created, their corresponding 3D structures are generated and projected back into each image, providing rough visual cues for annotating additional images. Additionally, since we capture stereo information, we augment our labeling interface to be aware of stereo pairs: users annotate one image, copy those annotations to the stereo image, and translate the curve endpoints along their corresponding epipolar lines to the correct location in the stereo image. This curve translation constrained to epipolar lines (with additional unconstrained translation if necessary to account for error) is much less time consuming than labeling the stereo image from scratch.

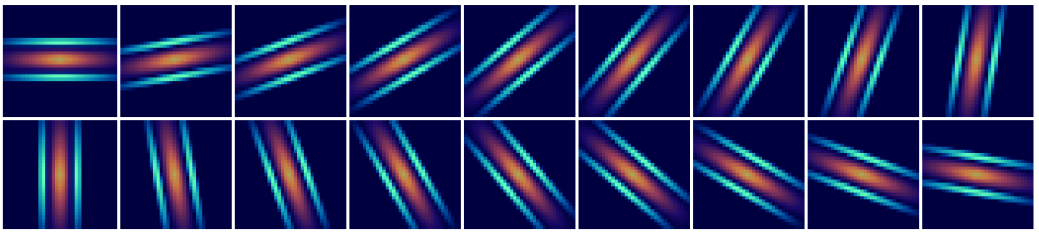

Fig. 7. A set of anisotropic kernels is used to obtrain directional activations in segmentation masks for both perceptual loss and flow field generation.

Human labelers often identify matching branches and twigs across images by performing human optical flow, toggling between adjacent frames of the source video and using the parallax effect to determine branch connectivity. This practice is an obvious candidate for automation, e.g. by annotating an initial frame then automatically carrying the annotated curves through subsequent frames via optical flow. Unfortunately, the features of interest are often extremely small and thin and the image data contains compression artifacts, making automatic optical flow approaches quite difficult. However, it is our hope that in future work the same tools that aid human labelers can be applied to automatic approaches making them more effective for image annotation.

8.2 Deep Learning

In order to generate flow fields for assisting the human labeler as discussed in Section 7, we first obtain semantic segmentations of *tree* and *not tree* using a deep learning approach. To train a network for semantic segmentation, we generate a training dataset by rasterizing the image annotations as binary segmentation masks of the labeled branches. From these 4K masks, we then generate a dataset of 512×512 crops containing more than 4000 images. The crop centers are guaranteed to be at least 50 pixels away from one another, and each crop is guaranteed to correspond to a segmentation mask containing both binary values. The segmentation problem on

Fig. 8. (Left) Image masks generated from image annotations and used as training data. (Right) Outputs of the segmentation network.

Fig. 9. (Top left) A ground truth mask of the tree taken by flattening the image annotation data into a simple binary mask. (Bottom left) A visualization of flow directions estimated by applying directional filters to the ground truth mask. (Right) Medial axes of the tree branches estimated from the flow field.

the raw 4K images must work on image patches with distinctly different characteristics: the more straightforward case of branches silhouetted against the grass, and the more complex case of highly dense branch regions. Therefore, we split the image patches into two sets via k-means clustering, and train two different models to segment the two different cases. For the same number of training epochs, our two-model approach yields qualitatively better results than the single-model approach.

Instead of directly using the standard binary cross entropy loss, the sparseness and incompleteness of our data led us to use a weighted variant, in order to penalize false negatives more than false positives. As a further step to induce smoothness and sparsity in our results, we introduce a second order regularizer through the L2 difference of the output and ground truth masks' gradients. We also experiment with an auxiliary loss similar to the VGG perceptual loss described in [Mosinska et al. 2018], but instead of using arbitrary feature layers of a pretrained network, we look at the L1 difference of hand-crafted multiscale directional activation maps. These activation maps are produced by convolving the segmentation mask with a series of Gabor filter-esque [Jain and Farrokhnia 1991] feature kernels $\{k(\theta, r, \sigma) : \mathbb{R}^2 \rightarrow [0, \ldots, N]^2\}$, where each kernel is scale-aware and piecewise sinusoidal (see Figure 7). A given kernel $k(\theta, r, \sigma)$ detects branches that are at an angle θ and have thicknesses within the interval $[r, \sigma r]$. For our experiments, we generate 18 kernels spaced 10 degrees apart and use $N = 35$, $r = 4$, and $\sigma = 1.8$.

Figure 8 illustrates two annotated images used in training and the corresponding learned semantic segmentations. Note that areas of the semantic segmentation that are not part of the labeled annotation may correspond to true branches or may be erroneous; for the time being a human must still choose which pieces of the semantic segmentation to use in adding further annotations.

8.3 Learning-Assisted Annotation

To generate a flow field, we create directional activation maps as in Section 8.2 again using the kernels from Figure 7, then perform a clustering step on the resulting per-pixel histograms of

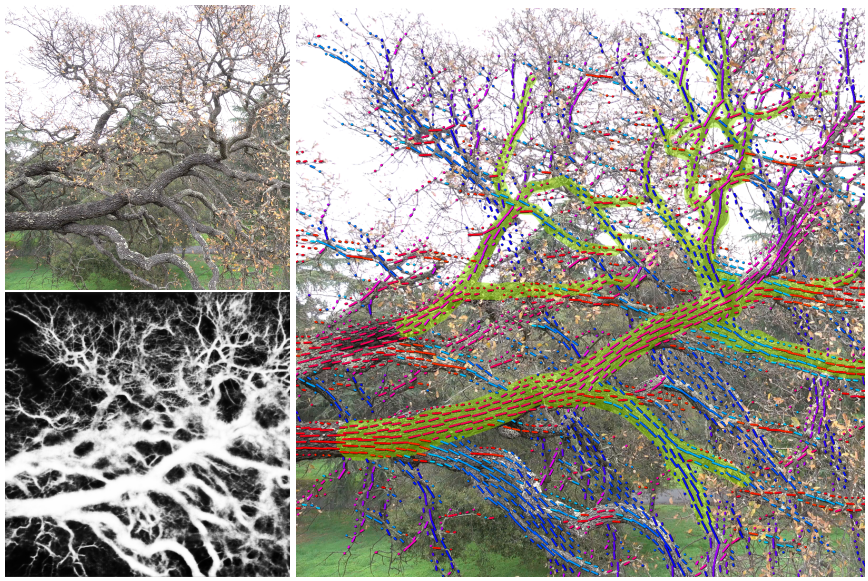

Fig. 10. The trained network infers a segmentation mask (bottom left) from an input image (top left). We then estimate a flow field (right) by applying anisotropic filters to the segmentation mask. A labeler can specify endpoints between which a medial axis and thickness values are automatically estimated (right, in green).

gradients [Dalal and Triggs 2005] to obtain flow vectors. Each pixel labeled as *tree* with sufficient confidence is assigned one or more principal directions; pixels with more than one direction are potentially branching points. We find the principal directions by detecting clusters in each pixel's activation weights; for each cluster, we take the sum of all relevant directional slopes weighted by their corresponding activation values.

Having generated a flow field of sparse image space vectors, we trace approximate medial axes through the image via an alternating projection-advection scheme. From a given point on a branch, we estimate the thickness of the branch by examining the surrounding flow field and project the point to the estimated center of the branch. We then advect the point through the flow field and repeat this process. In areas with multiple directional activations (e.g. at branch crossings or bifurcations), our advection scheme prefers the direction that deviates least from the previous direction. More details about this scheme may be found in the supplemental material. By applying this strategy to flow fields generated from ground truth image segmentations, we are able to recover visually plausible medial axes (see Figure 9). However, medial axes automatically extracted from images without ground truth labels are error prone. Thus, we overlay the flow field on the annotation interface and rely on the human labeler. The labeler may select curve endpoints in areas where the flow field is visually plausible, and these endpoints are used to guide the medial axis generation. See Figure 10 for an example flow field generated from the learned segmentation mask and the supplemental material for a demonstration of semi-automated medial axis generation.

8.4 Recovering Medium Scale Branches

Given a set of image annotations and camera extrinsics obtained via structure from motion and stereo calibration, we first construct piecewise linear branches in 3D. We triangulate keypoints that have been labeled in multiple images, obtaining 3D positions by solving for the point that minimizes the sum of squared distances to the rays originating at each camera's optical center and

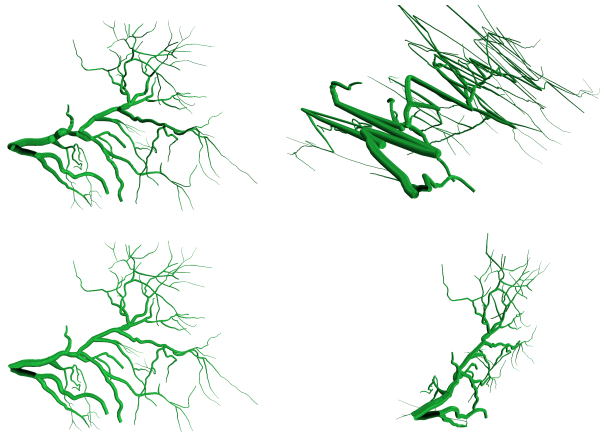

Fig. 11. Branches labeled from a stereo pair of cameras are visually plausible from the perspective of those cameras (top left), but they can exhibit severe error when viewed from a different angle (top right). By clamping these branch positions, one can achieve a virtually identical projection to the original cameras (bottom left) while maintaining a nondegenerate albeit “flattened" appearance from a different angle (bottom right).

passing through the camera’s annotated keypoint. We then transfer the topology of the annotations to the 3D model by connecting each pair of 3D keypoints with a line segment if a curve exists between the corresponding keypoint pair in any image annotation.

Next, we subdivide and perturb the linear segments connecting the 3D keypoints to match the curvature of the annotated data. Each segment between two keypoints is subdivided by introducing additional vertices evenly spaced along the length of the segment. For each newly introduced vertex, we consider the point that is the same fractional length along the image-space curve between the corresponding annotated keypoints in each image for which such a curve exists. We trace rays through these intra-curve points to triangulate the position of each new vertex in the same way that we triangulated the original keypoints.

Finally, we estimate the thickness of each 3D vertex beginning with the 3D keypoints. We estimate the world space thickness of each keypoint by considering the corresponding thickness in all annotated camera frames. For each camera in which the keypoint is labeled, we estimate world space thickness using similar triangles, then average these estimates to get the final thickness value. We then set the branch thickness of each of the vertices obtained through subdivision simply by interpolating between the thicknesses of the keypoints at either end of the 3D curve. Using this strategy, we recover a set of 3D positions with local cross-sectional thicknesses connected by edges, which is equivalent to the generalized cylinder representation employed in Section 4.

The human users of our annotation tools encounter the traditional trade-off of stereo vision: it is easy to identify common features in images with a small baseline, but these features triangulate poorly exhibiting potentially extreme variance in the look-at directions of the corresponding cameras. Conversely, cameras whose look-at directions are close to orthogonal yield more stable triangulations, but common features between such images are more difficult to identify. One heuristic approach is to label each keypoint three times: twice in similar images and once from a more diverse viewpoint. However, it may be the case that some branches are only labeled in two images with a small baseline (e.g. a stereo pair). In this case, we propose a clamping strategy based on the topological prior of the tree. Designating a “root" vertex of a subtree for such annotations, we

Fig. 12. The tree model is deformed from its rest pose (left) to an exaggerated pose (right) via simulation.

triangulate the annotated keypoints as usual obtaining noisy positions in the look-at directions of the stereo cameras. We then march from the root vertex to the leaf vertices. For each vertex p with location p_x, we consider each outboard child vertex c with location c_x. For each camera in which the point c is labeled, we consider the intersection of the ray from the location of c's annotation to c_x with the plane parallel to the image plane that contains p_x; let c'_x be the intersection point. We then clamp the location of c between c'_x and the original location c_x based on a user-specified fraction. This process is repeated for each camera in which c is annotated, and we obtain the final location for c by averaging the clamped location from each camera. See Figure 11.

9 LIMITATIONS

The human annotation process is by far the most time-intensive stage of the reconstruction pipeline, requiring many user-hours. However, having done this work, we have built a data set of image annotations as a side effect of obtaining the 3D simulatable reconstruction. We expect this pipeline to generalize well to other trees since the structural prior of a tree (in the graph sense) is inherent to any type of botanical tree. The use of a drone also generalizes well to much taller trees. The drone-mounted camera approach encounters difficulty, however, in sweeping out a full circle about features in the interior of trees with dense branches, as is the case to some extent in our subject tree. Furthermore, the occlusion of branches due to foliage presents a challenge in reconstructing a one-to-one match of the tree's structure. Of course, one could procedurally generate branches within the volume contained within the leaves (see e.g. [Tan et al. 2007; Wither et al. 2009]). For accurate data in the case of deciduous trees, another option is to collect data while the tree is without leaves, as we have done in this work. While we use our hand-annotated images to train segmentation networks for further annotation of the subject tree, we have not tested how well networks trained on this data generalize to segmenting trees of other species or in other environments. We expect, however, that at the very least the data will be useful for transfer learning or few-shot learning applications for other tree species.

10 CONCLUSION AND FUTURE WORK

We presented an end-to-end pipeline for reconstructing a 3D model of a botanical tree from RGB image data. Our reconstructed model may be readily simulated to create motion in response to external forces, e.g. to model the tree blowing in the wind (see Figure 12). We use generalized cylinders to initialize an articulated rigid body system, noting that one could subdivide these primitives as desired for additional bending degrees of freedom, or decrease their resolution for faster performance on mobile devices. The simulated bodies drive the motion of the textured triangulated surfaces and/or the point cloud data as desired.

Although we presented one set of strategies to go all the way from the raw data to a simulatable mesh, it is our hope that various researchers will choose to apply their expertise to and improve upon various stages of this pipeline, yielding progressively better results. In particular, the rich

topological information in the annotations has great potential for additional deep learning applications, particularly for topology extraction [Máttyus et al. 2017; Ventura et al. 2018; Xue et al. 2018] and 2D to 3D topology generation [Estrada et al. 2015].

ACKNOWLEDGMENTS

Research supported in part by ONR N000014-13-1-0346, ONR N00014-17-1-2174, ARL AHPCRC W911NF-07-0027, and generous gifts from Amazon and Toyota. In addition, we would like to thank both Reza and Behzad at ONR for supporting our efforts into computer vision and machine learning, as well as Michael Black for many interesting suggestions especially regarding drones. E.Q. is supported by NDSEG. E.Q. would also like to thank David Hyde and Michael Bao for donating cameras, Pilot AI Labs for the use of their drone, and Richard Heru, Martin Mbuthia, Katherine Liu, and Riley Wilson for their data annotation work.

REFERENCES

Sameer Agarwal, Noah Snavely, Ian Simon, Steven M Seitz, and Richard Szeliski. 2009. Building rome in a day. In *Computer Vision, 2009 IEEE 12th Int. Conf. on*. IEEE, 72–79.

Iro Armeni, Sasha Sax, Amir R Zamir, and Silvio Savarese. 2017. Joint 2d-3d-semantic data for indoor scene understanding. *arXiv preprint arXiv:1702.01105* (2017).

Iro Armeni, Ozan Sener, Amir R Zamir, Helen Jiang, Ioannis Brilakis, Martin Fischer, and Silvio Savarese. 2016. 3d semantic parsing of large-scale indoor spaces. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 1534–1543.

Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. 2000. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*. ACM Press/Addison-Wesley Publishing Co., 417–424.

Jules Bloomenthal. 1985. Modeling the mighty maple. In *ACM SIGGRAPH Computer Graphics*, Vol. 19. ACM, 305–311.

Robert L Cook. 1984. Shade trees. *ACM Siggraph Comput. Graph.* 18, 3 (1984), 223–231.

Navneet Dalal and Bill Triggs. 2005. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, Vol. 1. IEEE, 886–893.

M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. 1999. Implicit Fairing of Irregular Meshes using Diffusion and Curvature Flow. *Comput. Graph. (SIGGRAPH Proc.)* (1999), 317–324.

Rolando Estrada, Carlo Tomasi, Scott C Schmidler, and Sina Farsiu. 2015. Tree topology estimation. *IEEE Transactions on Pattern Analysis & Machine Intelligence* 1 (2015), 1–1.

Martin A Fischler and Robert C Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981), 381–395.

Alvaro Fuentes, Sook Yoon, Sang Cheol Kim, and Dong Sun Park. 2017. A robust deep-learning-based detector for real-time tomato plant diseases and pests recognition. *Sensors* 17, 9 (2017), 2022.

Yasutaka Furukawa and Jean Ponce. 2010. Accurate, dense, and robust multiview stereopsis. *IEEE transactions on pattern analysis and machine intelligence* 32, 8 (2010), 1362–1376.

Jingwei Huang, Angela Dai, Leonidas J Guibas, and Matthias Nießner. 2017. 3DLite: towards commodity 3D scanning for content creation. *ACM Trans. Graph.* 36, 6 (2017), 203–1.

Anil K Jain and Farshid Farrokhnia. 1991. Unsupervised texture segmentation using Gabor filters. *Pattern recognition* 24, 12 (1991), 1167–1186.

Yifeng Jiang and C Karen Liu. 2018. Data-Augmented Contact Model for Rigid Body Simulation. *arXiv preprint arXiv:1803.04019* (2018).

Justin Johnson, Alexandre Alahi, and Li Fei-Fei. 2016. Perceptual losses for real-time style transfer and super-resolution. In *European Conference on Computer Vision*. Springer, 694–711.

Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. 2018. Learning Category-Specific Mesh Reconstruction from Image Collections. In *The European Conf. on Comput. Vision (ECCV)*. 371–386.

Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. 2006. Poisson Surface Reconstruction. In *Proc. of the Fourth Eurographics Symp. on Geom. Processing* (Cagliari, Sardinia, Italy) *(SGP '06)*. Eurographics Association, Aire-la-Ville, Switzerland, Switzerland, 61–70. http://dl.acm.org/citation.cfm?id=1281957.1281965

Alina Kloss, Stefan Schaal, and Jeannette Bohg. 2017. Combining learned and analytical models for predicting action effects. *arXiv preprint arXiv:1710.04102* (2017).

Karl Krissian, Xunlei Wu, and Vincent Luboz. 2006. Smooth vasculature reconstruction with circular and elliptic cross sections. *Medicine Meets Virtual Reality 14: Accelerating Change in Healthcare: Next Medical Toolkit* 119 (2006), 273.

Yangyan Li, Xiaochen Fan, Niloy J Mitra, Daniel Chamovitz, Daniel Cohen-Or, and Baoquan Chen. 2013. Analyzing growing plants from 4D point cloud data. *ACM Transactions on Graphics (TOG)* 32, 6 (2013), 1–10.

Guosheng Lin, Anton Milan, Chunhua Shen, and Ian Reid. 2017. RefineNet: Multi-Path Refinement Networks for High-Resolution Semantic Segmentation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

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

Yotam Livny, Feilong Yan, Matt Olson, Baoquan Chen, Hao Zhang, and Jihad El-sana. 2010. Automatic Reconstruction of Tree Skeletal Structures from Point Clouds. *Proc. SIGGRAPH Asia 2010* 29 (2010), 151:1–151:8. Issue 6.

C. Loop. 2001. *Triangle mesh subdivision with bounded curvature and the convex hull property.* Technical Report MSR-TR-2001-24. Microsoft Research.

Matthew M Loper and Michael J Black. 2014. OpenDR: An approximate differentiable renderer. In *European Conf. on Comput. Vision*. Springer, 154–169.

Gellért Máttyus, Wenjie Luo, and Raquel Urtasun. 2017. Deeproadmapper: Extracting road topology from aerial images. In *The IEEE International Conference on Computer Vision (ICCV)*.

Agata Mosinska, Pablo MÃąrquez-Neila, Mateusz KoziÅĎski, and Pascal Fua. 2018. Beyond the Pixel-Wise Loss for Topology-Aware Delineation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Pierre Moulon, Pascal Monasse, Romuald Perrot, and Renaud Marlet. 2016. OpenMVG: Open multiple view geometry. In *International Workshop on Reproducible Research in Pattern Recognition*. Springer, 60–74.

Matthias Müller and Nuttapong Chentanez. 2011. Solid Simulation with Oriented Particles. *ACM TOG* 30, 4, Article 92 (2011), 10 pages.

Xue Bin Peng, Glen Berseth, KangKang Yin, and Michiel Van De Panne. 2017. Deeploco: Dynamic locomotion skills using hierarchical deep reinforcement learning. *ACM Trans. Graph.* 36, 4 (2017), 41.

Przemyslaw Prusinkiewicz, Mark Hammel, Jim Hanan, and Radomír Měch. 1997. Visual models of plant development. In *Handbook of formal languages*. Springer, 535–597.

Guoxiang Qu, Wenwei Zhang, Zhe Wang, Xing Dai, Jianping Shi, Junjun He, Fei Li, Xiulan Zhang, and Yu Qiao. 2018. StripNet: Towards Topology Consistent Strip Structure Segmentation. In *Proceedings of the 26th ACM International Conference on Multimedia* (Seoul, Republic of Korea) *(MM '18)*. ACM, New York, NY, USA, 283–291. https://doi.org/10.1145/3240508.3240553

Ed Quigley, Yue Yu, Jingwei Huang, Winnie Lin, and Ronald Fedkiw. 2018. Real-time Interactive Tree Animation. *IEEE transactions on visualization and computer graphics* 24, 5 (2018), 1717–1727.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. 2015. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*. Springer, 234–241.

Johannes L Schönberger and Jan-Michael Frahm. 2016. Structure-from-motion revisited. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 4104–4113.

Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. 2006. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*. IEEE, 519–528.

Ondrej Stava, Sören Pirk, Julian Kratt, Baoquan Chen, Radomír Měch, Oliver Deussen, and Bedrich Benes. 2014. Inverse procedural modelling of trees. In *Computer Graphics Forum*, Vol. 33. Wiley Online Library, 118–131.

Ping Tan, Gang Zeng, Jingdong Wang, Sing Bing Kang, and Long Quan. 2007. Image-based tree modeling. In *ACM Trans. Graph.*, Vol. 26. ACM, 87.

Gabriel Taubin. 1995. A signal processing approach to fair surface design. In *Proc. of the 22nd annual conf. on Comput. graphics and interactive techniques*. ACM, 351–358.

Carles Ventura, Jordi Pont-Tuset, Sergi Caelles, Kevis-Kokitsi Maninis, and Luc Van Gool. 2018. Iterative Deep Learning for Road Topology Extraction. In *Proc. of the British Machine Vision Conf. (BMVC)*.

Jason Weber and Joseph Penn. 1995. Creation and rendering of realistic trees. In *Proc. 22nd Ann. Conf. Comput. Graph. Int. Tech.* ACM, 119–128.

Jamie Wither, Frédéric Boudon, M-P Cani, and Christophe Godin. 2009. Structure from silhouettes: a new paradigm for fast sketch-based design of trees. In *Computer Graphics Forum*, Vol. 28. Wiley Online Library, 541–550.

Changchang Wu. 2011. VisualSFM: A visual structure from motion system. http://ccwu.me/vsfm/.

Changchang Wu. 2013. Towards linear-time incremental structure from motion. In *3D Vision-3DV 2013, 2013 International Conference on*. IEEE, 127–134.

Ke Xie, Feilong Yan, Andrei Sharf, Oliver Deussen, Baoquan Chen, and Hui Huang. 2015. Tree Modeling with Real Tree-Parts Examples. *IEEE TVCG* 22, 12 (Dec 2015), 2608–2618.

Weipeng Xu, Avishek Chatterjee, Michael Zollhöfer, Helge Rhodin, Dushyant Mehta, Hans-Peter Seidel, and Christian Theobalt. 2018. MonoPerfCap: Human performance capture from monocular video. *ACM Transactions on Graphics (TOG)* 37, 2 (2018), 27.

Tianfan Xue, Jiajun Wu, Zhoutong Zhang, Chengkai Zhang, Joshua B. Tenenbaum, and William T. Freeman. 2018. Seeing Tree Structure from Vibration. In *The European Conf. on Comput. Vision (ECCV)*. 748–764.

Yili Zhao and Jernej Barbič. 2013. Interactive Authoring of Simulation-Ready Plants. *ACM Trans. Graph.* 32, 4 (2013), 84:1–84:12.

Ying Zheng, Steve Gu, Herbert Edelsbrunner, Carlo Tomasi, and Philip Benfey. 2011. Detailed reconstruction of 3D plant root shape. In *Comput. Vision (ICCV), 2011 IEEE Int. Conf. on*. IEEE, 2026–2033.

Jin Zhou, Ananya Das, Feng Li, and Baoxin Li. 2008. Circular generalized cylinder fitting for 3D reconstruction in endoscopic imaging based on MRF. In *Comput. Vision and Pattern Recognition Workshops, 2008. CVPRW'08. IEEE Comput. Society Conf. on*. IEEE, 1–8.

Silvia Zuffi, Angjoo Kanazawa, and Michael J Black. 2018. Lions and Tigers and Bears: Capturing Non-Rigid, 3D, Articulated Shape From Images. In *Proc. of the IEEE Conf. on Comput. Vision and Pattern Recognition (CVPR)*. 3955–3963.