

# THREE DIMENSIONAL RECONSTRUCTION OF BOTANICAL TREES WITH SIMULATABLE GEOMETRY: SUPPLEMENTARY MATERIAL

**Anonymous authors**

Paper under double-blind review

## A 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. Unfortunately, point clouds, triangle soups with holes, and other similar 3D representations are not readily amenable to such an approach. 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 1.

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

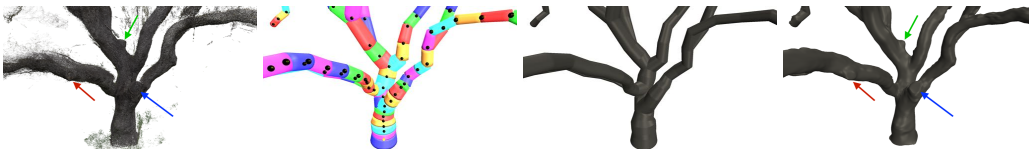


Figure 1: Point cloud data (far left) is used as a guide for interactively placing generalized cylinders which are readily simulatable as articulated rigid bodies (middle left). The generalized cylinders are connected together into a skinned mesh (middle right), which is then perturbed based on the point 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).

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. Taubin (1995); Desbrun et al. (1999)), 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.

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.

## B 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 2). 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 Appendix A), 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 5.1) 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 5.4). Then, we again create a contiguous skin mesh for these newly recovered 3D branches. In order to boost our ability to capture geometric



Figure 2: (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.



Figure 3: (Left) Triangle meshes recovered from point cloud data (Appendix A) and image annotations (Section 5). (Right) A close-up view of medium scale branches.

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 3.

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.

## C 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 4 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 4.

## D VIDEO SYNCING

We collect data using a pair of drone-mounted cameras. In order to synchronize the two video feeds, we calculate a sequence of the L1 distances between adjacent video frames for each camera as a rough estimate of how much motion occurred between frames. We then use the maximum cross correlation of these sequences to find the integer frame offset that best aligns the two videos. Note that since the videos are recorded at 30 fps, even the optimal offset could leave as much as a 1/60 second temporal error, in addition to the other sources of error mentioned in Appendix B.



Figure 4: (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.

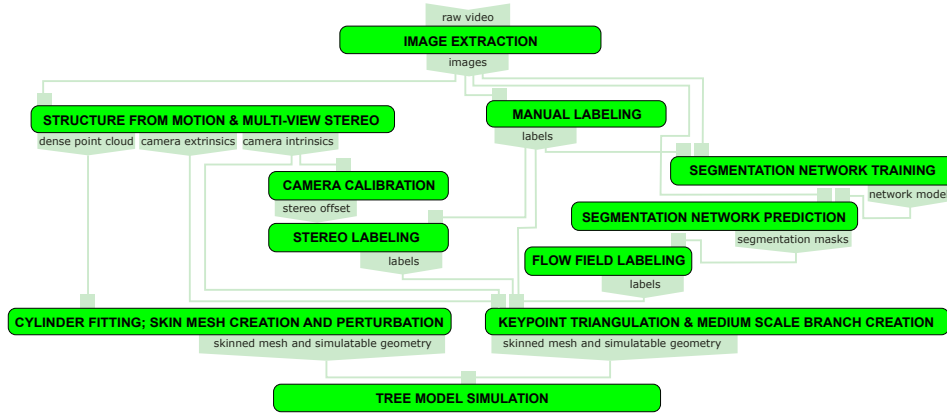


Figure 5: System diagram of our reconstruction pipeline, in which nodes represent processing steps and edges represent the data passed between processing steps.

## E CAMERA CALIBRATION

For each structure from motion problem, we withhold images from camera 2 and obtain estimates of camera 1’s intrinsics as well as extrinsics for individual image samples. Then, we hold the camera intrinsics constant and solve for the stereo offset between cameras 1 and 2 using a standard checkerboard pattern (Zhang, 2000). See Figure 5. The scale of the stereo transform relative to the structure from motion scene is unknown, but this scale may be found using image correspondences or even estimated visually.

## F SEMANTIC SEGMENTATION

The branches and twigs in the image data are often either clustered together or silhouetted against the ground. Hypothesizing that two models trained on identical network architectures might perform better than a single monolithic model, we divided the training data into a ‘brown’ set and a ‘green’ set using  $k$ -means clustering approximately capturing the ‘clustered’ and ‘silhouetted’ arrangements of twigs, respectively. Because color saturation is a defining feature in green vs. brown, we chose to cluster the image crops based on the saturation channel in HSV color space. However, using the average saturation over the entire image crop is not discriminatory enough due to the high amounts of brown that are present even in crops that contain branches silhouetted against grass; the median saturation value is also not sufficiently robust to varying amounts of grass in the background. Thus, for each image crop we took the 30<sup>th</sup> and 90<sup>th</sup> percentile of per-pixel saturations as a feature vector. We performed  $k$ -means clustering on these 2D features to split the image crops into two groups. Finally, we trained two models with identical U-Net network architectures, each with image crops of the clusters.

To segment an image, we pass the image through both models to obtain two candidate segmentation masks. We then composite the candidate segmentations by taking a per-pixel weighted sum. To set the per-pixel weights, we compute a feature vector of the 30<sup>th</sup> and 90<sup>th</sup> percentile saturation values for a small neighborhood around each pixel; then, we calculate the feature vector’s distance to each of the two cluster centers previously obtained in the  $k$ -means clustering step and use those distances to assign the corresponding pixel a weight between 0 and 1. See Figures 6 and 7 for examples.

Note that while our experiments with  $k$ -means clustering on training data for network models were reasonably successful, experiments with using clustering algorithms directly on images to perform segmentation were distinctively less so, as the color-based segmentation methods had trouble segmenting out tangled branches that are not clearly silhouetted against the grass or the sky. See Figure 8 for several examples.

## G FLOW FIELD EXTRACTION

Our handcrafted scale-aware directional kernels are piecewise sinusoidal with a linear falloff, and are designed to have a maximum activation when convolved with an image patch that contains a line passing through the patch center with some specified thickness  $r$  and orientation  $\theta$ . A falloff threshold  $\sigma$  controls the maximum detectable thickness; convolving any line thicker than  $(1 + \sigma)r$  with one of the kernels will result in an activation value of zero. See Figure 9. The formulation of the kernel  $k : S \rightarrow \mathbb{R}$  defined on the discrete pixel grid  $S = \{-17, -16, \dots, 16, 17\}^2$  is

$$k(p; \theta, r, \sigma) = \begin{cases} w_{p;r} \cos\left(\frac{d_{p;\theta,r}}{2} \pi\right) & \text{if } d_{p;\theta,r} < 1 \\ -w_{p;r} \frac{1}{\sigma} \sin\left(\frac{(d_{p;\theta,r}-1)}{\sigma} \pi\right) & \text{if } 1 < d_{p;\theta,r} < 1 + \sigma \\ 0 & \text{if } 1 + \sigma < d_{p;\theta,r} \end{cases}$$

where  $w_{p;r} = 1 - \frac{1}{r} \|p\|$ ,

$$v_\theta = \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix},$$

$$d_{p;\theta,r} = \frac{1}{r} \|p - (p \cdot v_\theta) v_\theta\|.$$

We use a bank of 18 filters, with  $\theta$  ranging from  $0^\circ$  to  $170^\circ$  in  $10^\circ$  increments. Instead of varying  $r$  and  $\sigma$ , for more efficient computation we use fixed, empirically chosen values for  $r$  and  $\sigma$  and convolve these filters with a number of rescaled instances of the input image. These convolutions yield a set of directional activations (see Figure 10b and Figure 11b).

To extract flow field vectors from the directional activations, we first determine whether a meaningful nonzero flow direction exists for each pixel. First, we filter out noise by zeroing out all activation values that are below an empirically chosen threshold (see Figure 11c). Next, for each pixel and for each  $\theta$  sample, we reduce the activation maps for different scales to a single map by taking the maximum activation across all scales. This yields a per-pixel histogram of activation values as a function of  $\theta$ . See Figure 11d.

Pixels with all zero activations and pixels with more than half their activations nonzero are ignored; the latter case is for filtering out pixels corresponding to roughly isotropic ‘blobs’ in the segmentation mask. We determine primary directions for the remaining pixels by clustering their activation values. Observing that an elongated structure in the segmentation mask may cause large activations in multiple kernels with similar  $\theta$  values, we sort the directional activations by  $\theta$  and find blocks of activations caused by adjacent  $\theta$  samples bounded by local minima in the distribution. See Figures 11d and 11e. For a given activation block  $b$  with per-angle activations  $a_\theta$ , we use the weighted sum of the directions  $\sum_{\theta \in b} a_\theta \begin{pmatrix} \cos \theta \\ \sin \theta \end{pmatrix}$  as a single vector representing the block. After finding all such blocks and computing the corresponding vectors, we designate the vectors with the largest magnitudes as the primary flow directions for the corresponding pixel. In practice, selecting the top two vectors as primary flow directions has proven sufficient so far. See Figure 11a and Figure 10.

## H PROJECTION-ADVECTION SCHEME

For a point  $p$  with nonzero flow  $\mathbf{n}$ , we consider the line  $L$  perpendicular to  $\mathbf{n}$  that passes through  $p$ . We then find the line segment  $\overline{AB}$  on  $L$  which includes  $p$  and for which every point on the segment has a principal flow direction within 15 degrees of  $\mathbf{n}$ . The center point  $(A + B)/2$  of the line segment, referred to as  $p'$ , is now the projection of  $p$  onto the medial axis, and the length  $\|\overline{AB}\|$  is the thickness of the branch at point  $p'$ .  $p'$  is added to the set of medial axis vertices, and we next consider the flow field at point  $p'$ . If the flow field at  $p'$  is nonzero, we choose the principle direction  $\mathbf{n}'$  that is closest to  $\mathbf{n}$ .  $p$  is then updated to be  $p' + t\mathbf{n}'$ , where  $t$  is a user-specified step size, and the projection-advection scheme is repeated until we find a zero flow field at  $p'$ , i.e. where a nonzero  $\mathbf{n}'$  does not exist.

Note that when estimating a medial axis within our annotation tool we additionally have labeler-specified curve endpoints  $p_0$  and  $p_1$ . To account for this we add an additional attraction force,

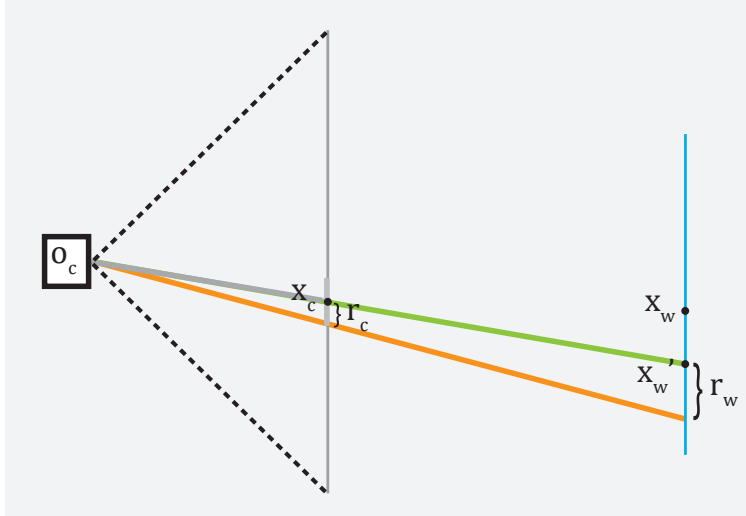
picking the advection direction  $\mathbf{n}'$  as  $w\mathbf{n}_f + (1 - w)\mathbf{n}_d$ , where  $\mathbf{n}_f$  is the normalized flow field direction,  $\mathbf{n}_d$  is the normalized direction from  $p'$  to  $p_1$ , and  $w = \frac{\|p' - p_1\|}{\|p_0 - p_1\|}$ .

## I RECOVERING MEDIUM SCALE BRANCHES

Here we provide additional implementation details for the approach described in Section 5.4. See Figure 12 for additional views of the resulting meshed geometry.

**Misaligned Cameras:** Some images may fail to be aligned properly after running structure from motion. There are few enough of these that it is feasible to visually inspect each image and mark those with incorrect extrinsics as ‘misaligned.’ During medium scale branch reconstruction, we perform an initial pass of keypoint triangulation while withholding annotations from misaligned images. We then use the resulting 3D triangulated keypoints to solve a perspective- $n$ -point problem for each misaligned image. If this fails, then the annotations from that image are ignored; otherwise, the image is marked as ‘aligned.’ We then repeat the keypoint triangulation step using annotations from the newly aligned images along with those of the originally aligned images, thus potentially modifying the 3D positions of some keypoints while also potentially adding new keypoints.

### Keypoint Thickness Estimation:



For a keypoint with triangulated position  $x_w$ , we estimate the world space radius by considering the set of cameras  $C$  in which the keypoint is labeled. For each such camera we know the camera’s optical center  $o_c$ , the annotated point  $x_c$ , and the annotated radius  $r_c$ . Let  $p$  be the plane that contains  $x_w$  and is parallel to the camera’s image plane. We intersect the ray  $\overrightarrow{o_c x_c}$  with  $p$  obtaining a point  $x'_w$ , then estimate the keypoint’s radius  $r_w$  using a ratio of distances:  $\frac{r_w}{r_c} = \frac{\|x'_w - o_c\|}{\|x_c - o_c\|}$ . Finally, we average the radius estimates over each camera in which the keypoint is annotated so that the final radius estimate is  $\frac{1}{|C|} \sum_{c \in C} \frac{\|x'_w - o_c\|}{\|x_c - o_c\|} r_c$ .



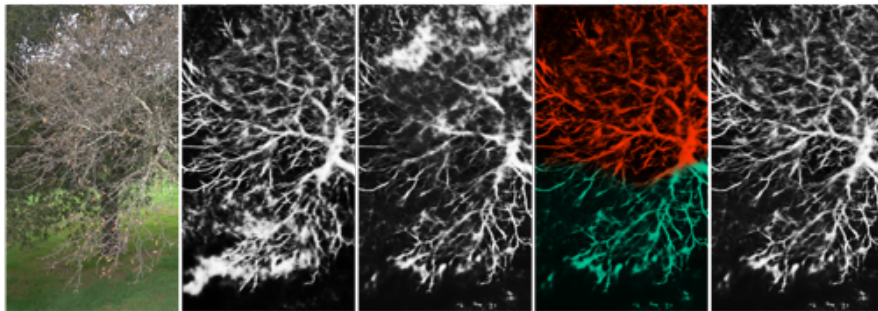


Figure 6: From left to right: original image, output of model trained on cluster 1 ('brown'), output of model trained on cluster 2 ('green'), visualization of contributions from each model in the composited output (orange for cluster 1 and green for cluster 2), and the final composited segmentation.

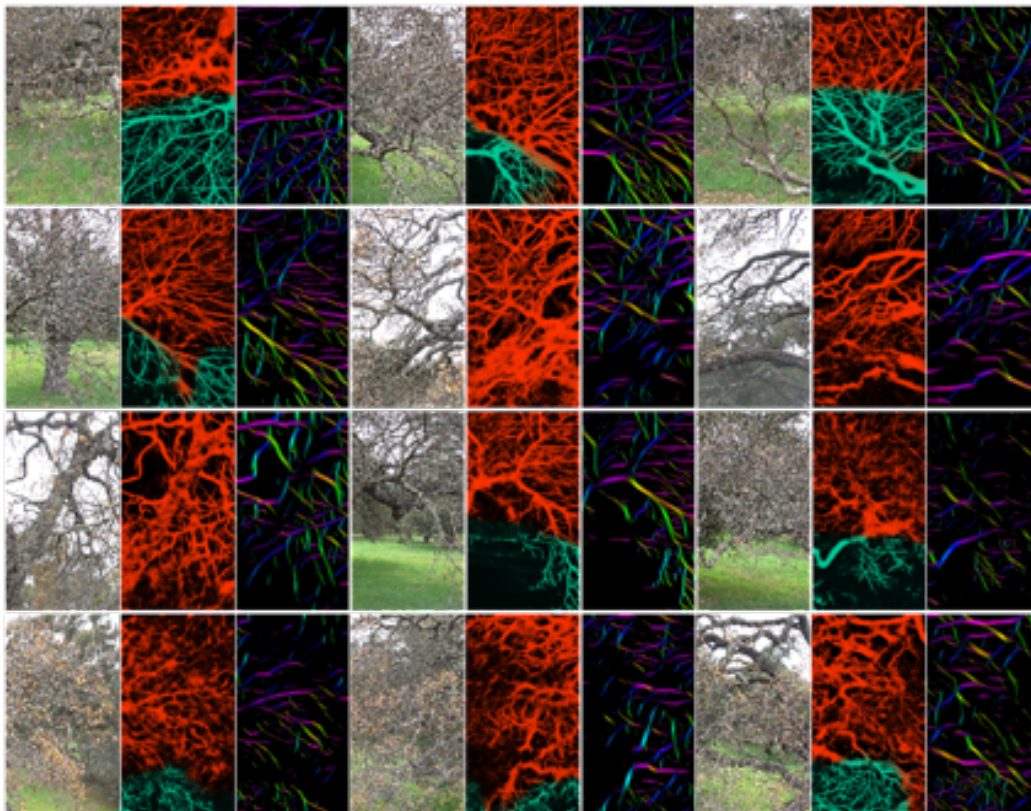


Figure 7: Examples of network-predicted segmentations and subsequent flow field extractions using images not seen during network training.

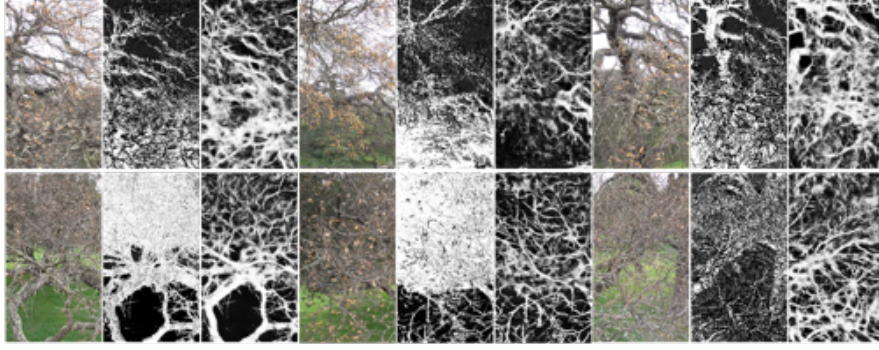


Figure 8: Examples of  $k$ -means clustering versus our trained segmentation network. From left to right for each image triplet: input image,  $k$ -means result, network output. For this visualization, we performed  $k$ -means clustering with 2 clusters in  $L^*a^*b^*$  color space, which has a Euclidean distance that closely corresponds to color differences perceived by humans.

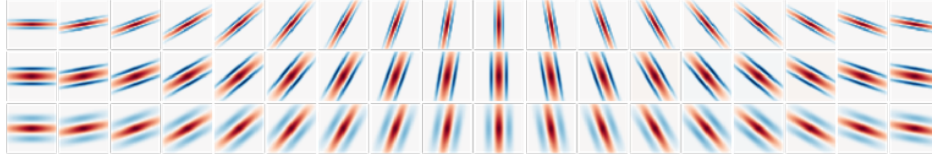


Figure 9: Heatmap visualizations of  $35 \times 35$  kernels, where red indicates positive weights and blue indicates negative weights. The middle row shows kernels with  $r = 3.8$ ,  $\sigma = 0.8$ , and  $\theta$  increasing in  $10^\circ$  increments; these are the values we use in practice. To illustrate the effect of the kernel parameters, the top and bottom rows depict kernels with  $r = 2.0$ ,  $\sigma = 0.8$  and  $r = 3.8$ ,  $\sigma = 2.0$ , respectively.

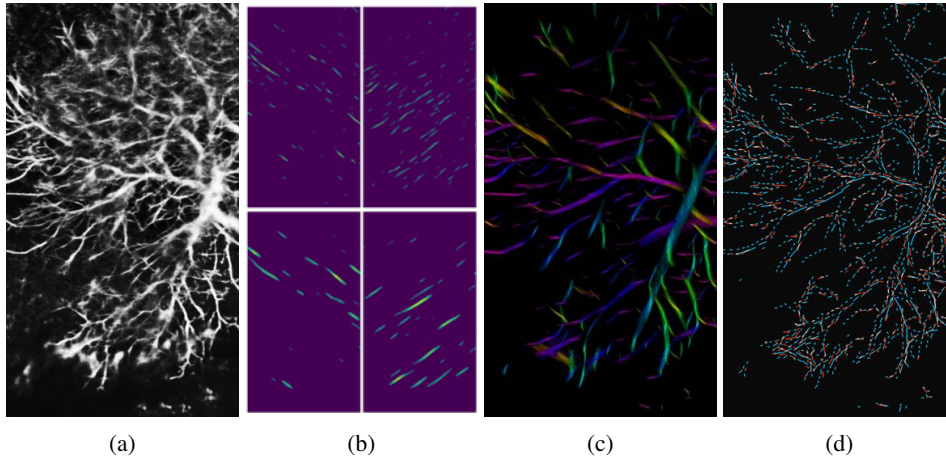
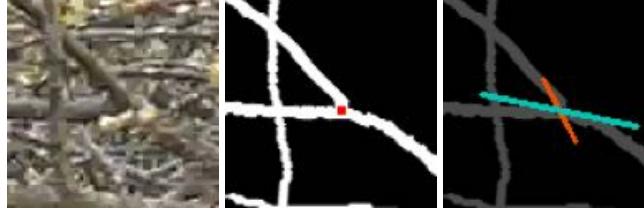
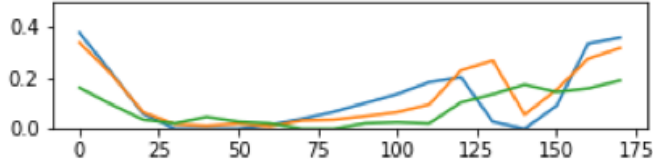


Figure 10: From left to right: input segmentation mask, a selected subset of multiscale directional activation maps, an HSV visualization of primary flow directions (in which hue represents angle and value represents magnitude), and a flow field visualization of primary and secondary flow directions (visualized in blue and orange, respectively).

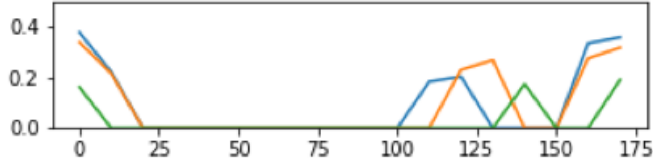




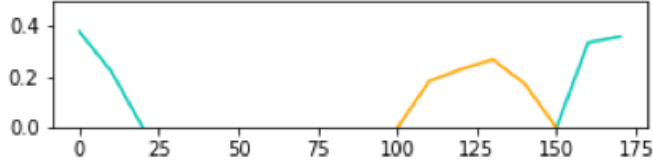
(a) An image crop (left), its corresponding ground truth segmentation mask with a pixel of interest colored in red (middle), and the two directions detected at the pixel of interest during flow field generation (right).



(b) Plot of filter activation vs filter angle (in degrees) for the pixel of interest in (a). Each colored line corresponds to the activations at that pixel's location for a different image scale.



(c) We discard activations from (b) that are below a user-specified threshold.



(d) We reduce the activations from (c) to a single activation map by taking the maximum value over all scales. These values are grouped into clusters bounded by local minima (here blue and orange represent two clusters). Note that a cluster may wrap around from  $170^\circ$  to  $0^\circ$  since the filters are bidirectional.



(e) Kernels corresponding to nonzero values in the final activations from (d).

Figure 11: An illustration of how filter activations are grouped into clusters as part of the flow field generation process.

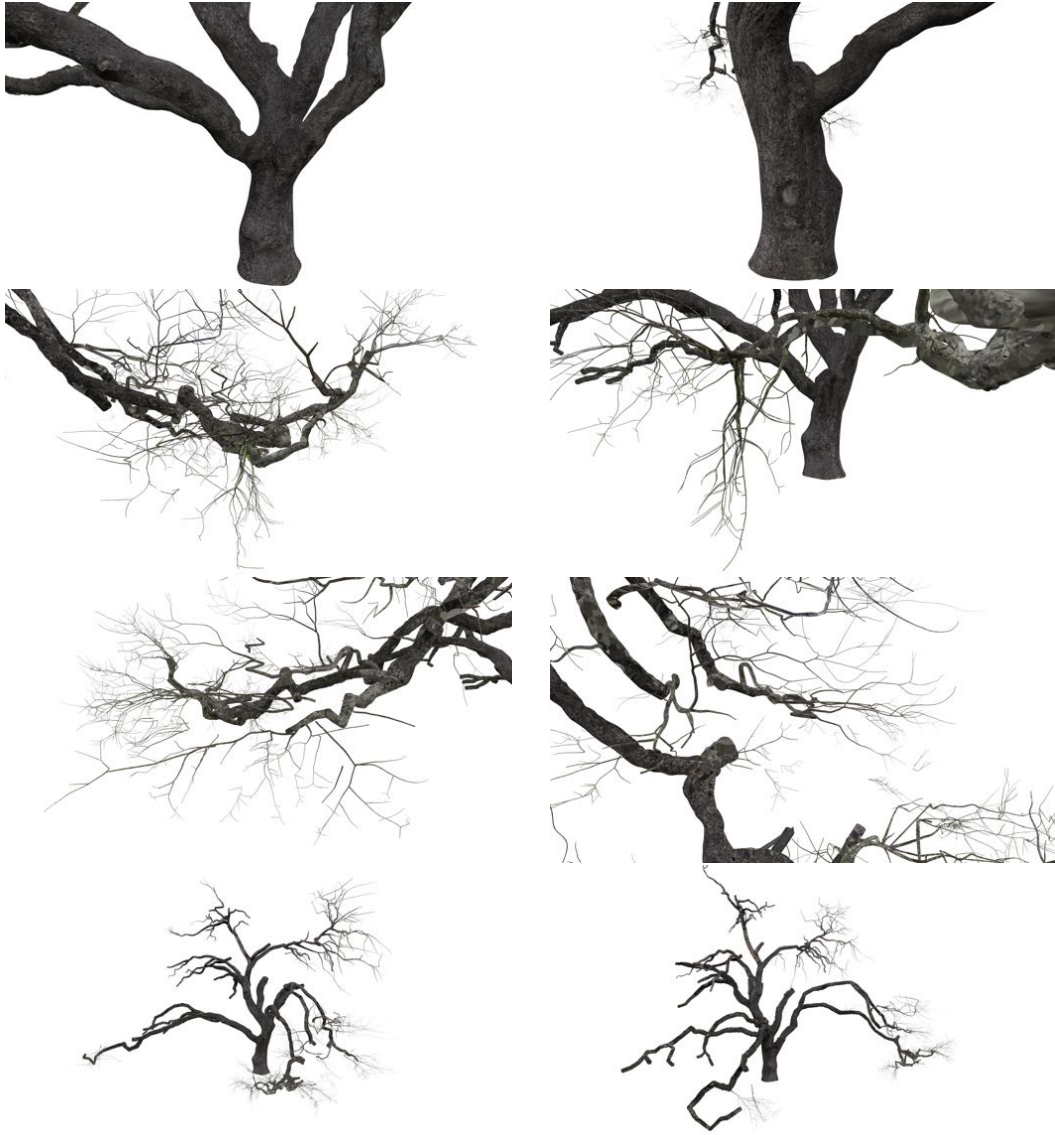


Figure 12: (Top row) Triangle mesh geometry perturbed using point cloud data. (Middle two rows) Medium scale branches generated from image annotations. (Bottom row) Zoomed out views of the final tree model (so far).

## REFERENCES

- Marcelo Bertalmio, Guillermo Sapiro, Vincent Caselles, and Coloma Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pp. 417–424. ACM Press/Addison-Wesley Publishing Co., 2000.
- Robert L Cook. Shade trees. *ACM Siggraph Comput. Graph.*, 18(3):223–231, 1984.
- M. Desbrun, M. Meyer, P. Schröder, and A. H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. *Comput. Graph. (SIGGRAPH Proc.)*, pp. 317–324, 1999.
- Michael Kazhdan, Matthew Bolitho, and Hugues Hoppe. Poisson surface reconstruction. In *Proc. of the Fourth Eurographics Symp. on Geom. Processing, SGP '06*, pp. 61–70, Aire-la-Ville, Switzerland, Switzerland, 2006. Eurographics Association. ISBN 3-905673-36-3. URL <http://dl.acm.org/citation.cfm?id=1281957.1281965>.
- C. Loop. Triangle mesh subdivision with bounded curvature and the convex hull property. Technical Report MSR-TR-2001-24, Microsoft Research, 2001.
- Ed Quigley, Yue Yu, Jingwei Huang, Winnie Lin, and Ronald Fedkiw. Real-time interactive tree animation. *IEEE transactions on visualization and computer graphics*, 24(5):1717–1727, 2018.
- Gabriel Taubin. A signal processing approach to fair surface design. In *Proc. of the 22nd annual conf. on Comput. graphics and interactive techniques*, pp. 351–358. ACM, 1995.
- Zhengyou Zhang. A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22, 2000.