Isolating a Tree's Skeleton using a 3-Dimensional Reconstruction

Using NeRF to find 1-dimensional topology of 3-dimensional trees

Shashwat Sahay[1]

Supervisor(s): Elmar Eisemann[1]**, Petr Kellnhofer**[1]**, Lukas Uzolas**[1]

[1]EEMCS, Delft University of Technology, The Netherlands

A Thesis Submitted to EEMCS Faculty Delft University of Technology,
In Partial Fulfilment of the Requirements
For the Bachelor of Computer Science and Engineering
January 28, 2024

Name of the student: Shashwat Sahay
Final project course: CSE3000 Research Project
Thesis committee: Elmar Eisemann, Petr Kellnhofer, Lukas Uzolas, Marcel Reinders

Abstract

L-Systems allow for the efficient procedeural generation of trees to be used for rendering in video games and simulations. Currently, however, it is difficult to engineer grammars that mimic the behaviours of real life trees in 3 dimensions. To be able to deduce them, the skeleton of a tree can be used to train a model and generate an L-system for a given tree in particular. The aim of this paper is to provide a pipeline to isolate these skeletons from images of a tree, using Neural Radiance Fields (NeRFs) to reconstruct the tree, and using Laplacian Based Contraction to retrieve the underlying skeleton. We find that this approach leads to 3-dimensional topologies that very closely resemble the given tree.

1 Introduction

This paper will discuss the use of NeRF methodologies to be able to isolate the skeleton of a tree, to eventually be able to derive an L-system that imitates said tree. The idea is that isolating a tree's skeleton, given only a 2D representation of what it looks like, is very difficult to do with great precision, because there could be a very large number of possible skeletons that would result in a similar looking tree. Having access to a 3-dimensional model, however, may lead to better results, as being able to glean information from multiple directions of the tree can give us more insights as to how the tree's skeleton looks. Being able to capture this information would allow us to render trees for video games or training simulations that are much more akin to what we see in the world around us.

There exists a great deal of literature about Neural Radiance Fields (or NeRFs) [10], which use a neural network to generate a 3-dimensional representation of a scene, given multiple 2-dimensional images of said scene. The same can be said about L-systems grammars, which are used to procedurally generate strings to represent fractals (a specific case being trees, for the purposes of this paper). What's missing is a link between the two – the idea of using a NeRF model to generate an L-system that imitates a tree (hopefully) better than can be done with only two dimensions of information.

The question this paper will focus on is then: "How can Neural Radiance Fields be used to isolate the skeleton of a tree?" This question tackles the largest challenge of using a NeRF to derive an L-system, because removing the tree's foliage (leaves, flowers, etc.) is integral to finding the underlying structure that we want the L-system to replicate. This is also the question that most lends itself to the idea of working with a 3-dimensional rendering, as the 2-dimensional information is, in theory, the bottleneck we hope to overcome with the methodology outlined in this paper.

To answer this question, this paper outlines a pipeline that uses an off-the-shelf solution for NeRF implementation, paired with filtering techniques and a skeletonization implementation to attempt to isolate the skeleton of the tree in 3 dimensions.

2 Background and Related Works

2.1 NeRF

NeRF is a method of reconstructing a scene using a neural network, trained on images of the scene, with known camera poses. In general, a trained NeRF model takes a 5-dimensional input, a spacial location in (x, y, z), as well as a viewing direction (θ, ϕ) and returns a 4-dimensional output, being the emitted color (r, g, b) and a volume density α. By then querying numerous inputs, it is able to put together a complete reconstruction of the scene. [9]

2.2 Skeletonization

There exists a great amount of research on methodologies for the skeletonization of a mesh, an overview of which can be found in [11]. The methodology in this paper will focus on using Laplacian Based Contraction, as seen in [3]. I first considered a medial axis approach which is outlined in [4], but as stated in the original paper on LBC [3], such methods lead to incorrect results when faced with regions that are less frequently sampled. That is, artifacts in the input image will lead to errors in the skeletonization. This is an issue that can arise from using NeRF to generate models, especially since it might be difficult to procure the required number and quality of images to be able to produce a model in great detail.

LBC is based on the idea that given a point cloud, we can achieve skeletonization by drawing the points in the cloud closer to each other, based on distance, until the resulting point cloud has no volume, reducing the 3D model to a 1D skeleton. A minimal spanning tree algorithm can then be used on this skeleton to produce a tree topology that mimics the structure of the input point cloud.

This method is less prone to errors caused by an imperfect NeRF model. In addition, from my own experimentation, in cases where leaves of a tree are not substantially larger than its branches, LBC produces a near perfect topology, even without filtering out the leaves from the point cloud.

Regardless of the downsides of the medial axis approach, for the sake of comparison, scikit-image [13] has an implementation of a skeletonization algorithm similar to medial axis transformation that can be found in [7]. This algorithm clears a 3x3 neighborhood of each pixel (or 3x3x3 neighborhood per voxel in the case of 3 dimensions) until only a skeleton remains.

3 Method and Implementation

3.1 NeRF Implementation

The NeRF scene was generated with the Nerfacto approach, implemented using nerfstudio [12]. This method was "heavily influenced by the structure of MipNeRF-360" [12], with improved performance by combining a number of other methods. In addition, nerfstudio is also a plug-and-play framework that makes it easy to process several images taken on a phone, train the neural network, and then export the desired parts of the scene as either a mesh or a point cloud.

Figure 1: One of 76 images of a house plant used to train the NeRF model

3.2 Data

Because I lacked the means to take real-life pictures of a fully-grown tree from all the angles required to train a robust NeRF model, I used a house plant that has features reminiscent of a real tree. 76 pictures were taken of this plant from various angles, after which the images were aligned using Agisoft Metashape [1], processed using nerfstudio, and then the NeRF model was trained with the same. The house plant mentioned can be seen in Figure 1, and the exported point cloud can be seen in Figure 2.

Figure 2: Point cloud of the house plant exported from nerfstudio

3.3 Implementing Skeletonization

To implement skeletonization, I used a tool called CherryPicker [8], which implements LBC and Semantic-LBC. The tool was designed specifically for cherry trees, but I was able to change variables such as initial contraction and attraction as required in order to widen its use case.

As can be seen in Figure 3, the topology here is distorted, and it contains some disconnected parts. This is because the leaves of the plant used were significantly larger than the width of the branches, and so they have an effect of the computed topology. To deal with this, some preprocessing of the point cloud must be done to attempt to filter out the leaves.

For the sake of comparison, I also used the scikit-image implementation of $skeletonize - 3d$ [13]. This was done by voxelizing the point cloud using open3D [15], creating a 3-dimensional image using the locations of the resulting voxels, and then applying this function to find the skeleton, before returning the result to a point cloud for visualization.

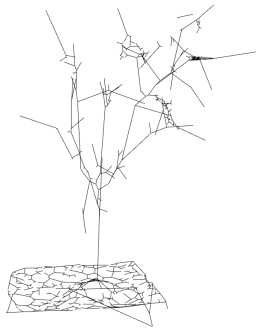

Figure 3: Initial skeleton using Laplacian-Based Contraction

3.4 Leaf filtering

To filter out the larger leaves, I leveraged the fact that Laplacian-Based contraction doesn't need a perfect point cloud to produce reliable results. The filtering was done by applying a color mask to the point cloud to remove green points from the input. This masking had to be done quite aggressively, as a large number of points throughout the point cloud, including in the leaves, are not colored at all, and are just black.

Therefore, a less aggressive color masking would leave enough points such that, combined with the black points that shouldn't be filtered, the skeletonization was still impacted by the leaves. Specifically, with r, g, b representing the red, green and blue color channels on a range of $[0, 1]$, the mask used was

$$(r > 0.9; g < 0.1; b > 0.9)$$

.

This means that any point with a value greater than 0.1 on the green channel is filtered out, as well as any point with less than 0.9 on either the red or blue channel.

4 Responsible Research

This pipeline makes use of multiple pre-existing tools, being nerfstudio [12] and CherryPicker [8]. CherryPicker is under an MIT license which allows the use of the software, without limitation. Similarly, nerfstudio is and under an Apache-2.0 license, a clause of which allows the "perpetual, worldwide, non-exclusive, no-charge, royalty-free, irrevocable patent license to make, have made, use, offer to sell, sell, import, and otherwise transfer the Work".

Additionally, both of these tools are open-source, which means I was able to read the underlying code in order to use utility scrips outside of their intended use, as well as make modifications to suit the needs of this research. These modifications are used solely for the purposes of this paper, with no intention to redistribute the changes.

(a) Laplacian-Based Contraction [8]

(b) Scikit-image skeletonize-3d [13]

Figure 4: Topology of tree with different skeletonization algorithms, compared to original point cloud and ground trust in red

5 Results and Discussion

As can be seen in Figure 4, both skeletonization methods were able to somewhat mimic the topology of the tree. However, it's also apparent that the lack of a robust method for filtering leaves causes a lot of distortion, as well as parts of the computed topology that mimic only the leaves themselves. It is worth noting, however, that these results may be heavily influenced by the specific tree that the pipeline was tested on, as Laplacian-Based Contraction struggles with extremely large leaves such as this one. Therefore, while these results look qualitatively similar to the hand-annotated ground truth if we disregard the part of the topology that follows the leaves, the robustness of the pipeline could definitely be improved.

5.1 Alternate Leaf Filtering Methods

For the purposes of this research, multiple leaf filtering methods were tested. Here we outline methods that either I was unable to properly implement, or that worked worse than expected

Clustering and Classification

The first method I tried to employ was DBScan [5], a density-based clustering algorithm implemented in the open3D library [15]. This did not work at all, because the point cloud exported by nerfstudio seems to be uniformly sampled, and so clustering by density did not make sense. Clustering on color did not work as well as expected either, because the exported point cloud is generated by integrating over multiple viewpoints in the NeRF model. This means that the view-dependent coloring of the model integrated into a large majority of the points being colored black, and so color-based clustering wasn't able to segment the leaves.

Another method that attempted to distinctly classify points in the point cloud as either leaf or wood, found in [14], was based on the idea of using multiple filters, as well as a path detection algorithm with the assumption that a tree can be viewed as a network. For whatever reason, this approach, using a fork of the implementation referenced in the text, failed to correctly classify about half of the leaf points.

Planar Detection

Another method implemented in the open3D library is planar patch detection, the methodology of which can be found in [2]. This actually worked very well, as can be seen in Figure 6. However, I was unable to develop a method to differentiate which of these planes corresponded to leaves. With such a method, this would be a great way to filter out leaves, though it's possible that such a method would work less well on trees that have much smaller, more plentiful leaves.

Figure 5: Topology of tree filtered using wood and leaf classification, compared to original point cloud and ground truth in red

Figure 6: Planes detected in the house plant point cloud

Filtering on Manifold Distance and Normal Estimation

A method that I tried and failed to properly implement can be found in [6], which is a method that is again based on clustering, but also utilizes an estimate of each point's normal, as well as manifold distance to possibly cluster the leaves more reliably. This method is built to be used to segment

and analyze leaves (though given a working implementation, it should be trivial to simply remove the leaves instead), but due to the fact that no implementation is provided, and I was unable to write an implementation myself, I'm unsure how well it would work in this specific context.

6 Conclusions and Future Work

6.1 Conclusion

In conclusion, the pipeline outlined in this paper is able to extract a tree skeleton from multiple images, using NeRF to reconstruct the scene in 3 dimensions, and then filtering the leaves and applying skeletonization algorithms to find a close resemblance to the tree's topology. The ability to do this should allow us to construct more realistic L-Systems, and as such, generate and render trees more efficiently, for use in video games and simulations.

6.2 Future Work

This pipeline could definitely be improved in the future by implementing more robust methods of leaf filtration, as this is the area that I think would have the greatest impact on the final result. Some of the methods outlined in Section 5.1 could possibly be great improvements to this pipeline, however I believe that, more than anything, it should be taken into consideration that the best method depends on the input tree. While it looks like the planar detection method [2] would work very well for the tree used in this paper, it could possibly not be the best method for trees with smaller leaves, in which case the path detection algorithm [14] might be a better fit. I believe that being able to differentiate which algorithm should be used depending on the input would result in much better outputs.

Similarly, another way this pipeline can be extended upon in the future is by implementing Semantic Laplacian-Based Contraction as seen in [8]. The pipeline, as it is, would perhaps not be suitable for trees in which the width of the trunk is significantly larger than the width of the branches. Being able to segment the trunk of the tree from the branches, in a similar fashion to the attempts to segment the leaves, would allow us to extend the skeletonization algorithm to result in an output with less irregularities.

In addition, given more time, I would've liked to improve on the evaluation of the pipeline as a whole. I wasn't able to train multiple significantly different NeRF models to experiment on different kinds of trees. I also would've liked to implement some kind of quantitative analysis on how well the pipeline was performing, with perhaps computing Chamfer distance between the computed topology and some hand-annotated ground truth.

References

[1] Agisoft. Metashape.

[2] Abner M. C. Araújo and Manuel M. Oliveira. A robust statistics approach for plane detection in unorganized point clouds. *Pattern Recognition*, 100:107115, 2020.

[3] Junjie Cao, Andrea Tagliasacchi, Matt Olson, Hao Zhang, and Zhinxun Su. Point cloud skeletons via laplacian based contraction. In *2010 Shape Modeling International Conference*, pages 187–197, 2010.

[4] Tamal K Dey and Jian Sun. Defining and computing curve-skeletons with medial geodesic function. In *Symposium on geometry processing*, volume 6, pages 143–152, 2006.

[5] Martin Ester, Hans-Peter Kriegel, Jörg Sander, and Xiaowei Xu. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, KDD'96, page 226–231. AAAI Press, 1996.

[6] Chunhua Hu, Zhou Pan, and Pingping Li. A 3d point cloud filtering method for leaves based on manifold distance and normal estimation. *Remote Sensing*, 11:198, 01 2019.

[7] T.C. Lee, R.L. Kashyap, and C.N. Chu. Building skeleton models via 3-d medial surface axis thinning algorithms. *CVGIP: Graphical Models and Image Processing*, 56(6):462–478, 1994.

[8] Lukas Meyer, Andreas Gilson, Oliver Scholz, and Marc Stamminger. Cherrypicker: Semantic skeletonization and topological reconstruction of cherry trees, 2023.

[9] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. Nerf: representing scenes as neural radiance fields for view synthesis. *Commun. ACM*, 65(1):99–106, dec 2021.

[10] Fabio Remondino, Ali Karami, Ziyang Yan, Gabriele Mazzacca, Simone Rigon, and Rongjun Qin. A critical analysis of nerf-based 3d reconstruction. *Remote Sensing*, 15(14), 2023.

[11] Andrea Tagliasacchi, Thomas Delame, Michela Spagnuolo, Nina Amenta, and Alexandru Telea. 3d skeletons: A state-of-the-art report. In *Computer Graphics Forum*, volume 35, pages 573–597. Wiley Online Library, 2016.

[12] Matthew Tancik, Ethan Weber, Evonne Ng, Ruilong Li, Brent Yi, Justin Kerr, Terrance Wang, Alexander Kristoffersen, Jake Austin, Kamyar Salahi, Abhik Ahuja, David McAllister, and Angjoo Kanazawa. Nerfstudio: A modular framework for neural radiance field development. In *ACM SIGGRAPH 2023 Conference Proceedings*, SIGGRAPH '23, 2023.

[13] Stefan Van der Walt, Johannes L Schönberger, Juan Nunez-Iglesias, François Boulogne, Joshua D Warner, Neil Yager, Emmanuelle Gouillart, and Tony Yu. scikit-image: image processing in python. *PeerJ*, 2:e453, 2014.

[14] Matheus B. Vicari, Mathias Disney, Phil Wilkes, Andrew Burt, Kim Calders, and William Woodgate. Leaf and wood classification framework for terrestrial lidar point clouds. *Methods in Ecology and Evolution*, 10(5):680–694, 2019.

[15] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Open3d: A modern library for 3d data processing, 2018.