Procedural modelling of tree growth using multi-temporal point clouds

Noortje van der Horst

1st supervisor: Liangliang Nan
2nd supervisor: Jantien Stoter
Co-reader: Sören Pirk

June 23 2022

Contents

- Introduction
- Method
- Results & Discussion
- Conclusion & Future work

Research goal

- How can multi-temporal point clouds be used to model tree growth?
- Sub-questions:
  - What digital **representation** of a tree is best suited to model growth?
  - To what extent can a procedural growth model accurately **reconstruct** the growth of a tree based on known ground-truth data models at different times?
  - How can a plausible reconstruction be made in areas for which **no** reliable ground-truth **data exists**?

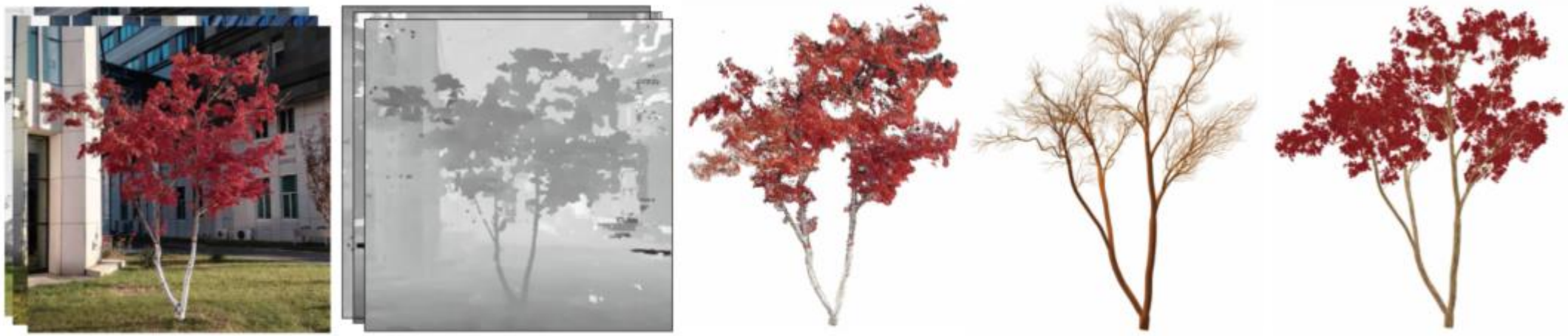
Digital plant reconstruction

- Plant reconstruction for:
  - Realistic looking virtual plants
  - Forestry management, environmental analysis, city planning, biology
- Reconstructing plants is still an open problem due to:
  - Inherent complex branching structure
  - Need for balancing model soundness with adherence to (often incomplete) data

Digital plant reconstruction

Livny et. al. (2010)

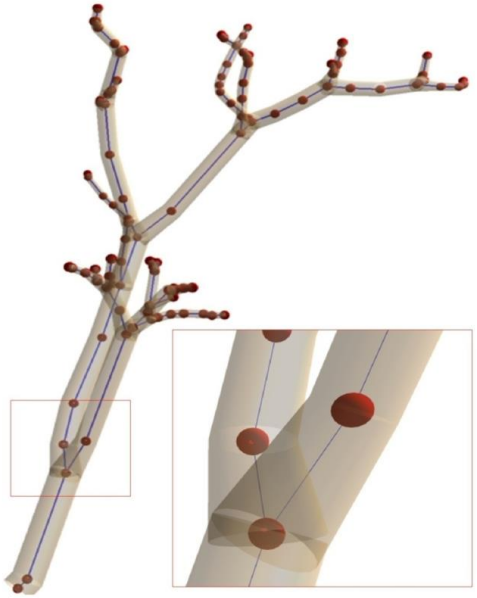
Digital plant reconstruction



(a) Input images (b) Depth maps (c) Point cloud (d) Skeletal structure (e) Textured plant

Guo et. al. (2020)

Digital plant reconstruction

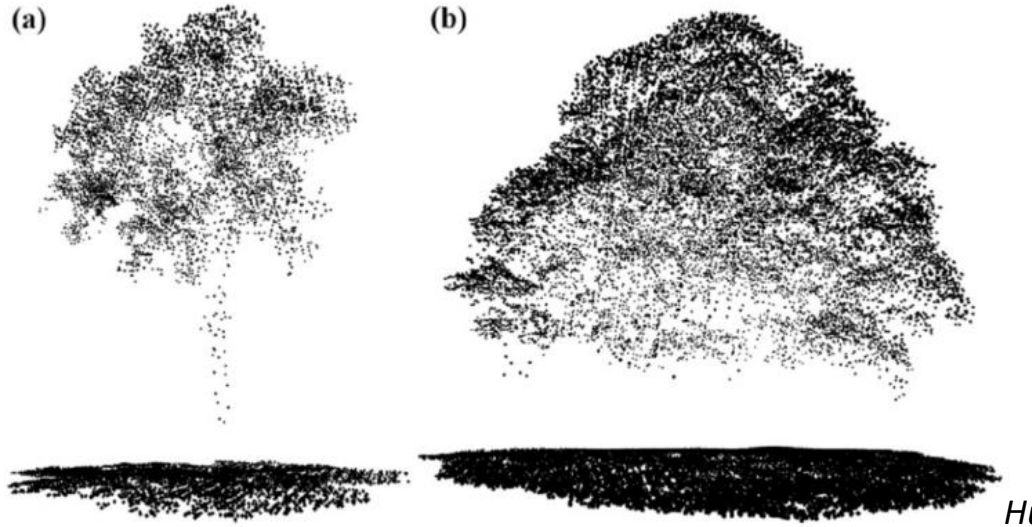


Livny et. al. (2010)

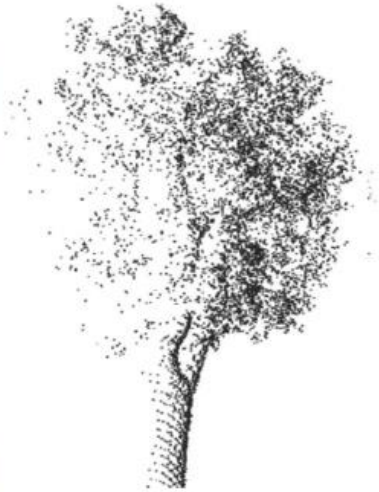
terrestial LiDAR

Du (2019)

aerial LiDAR



Hu et. al. (2017)

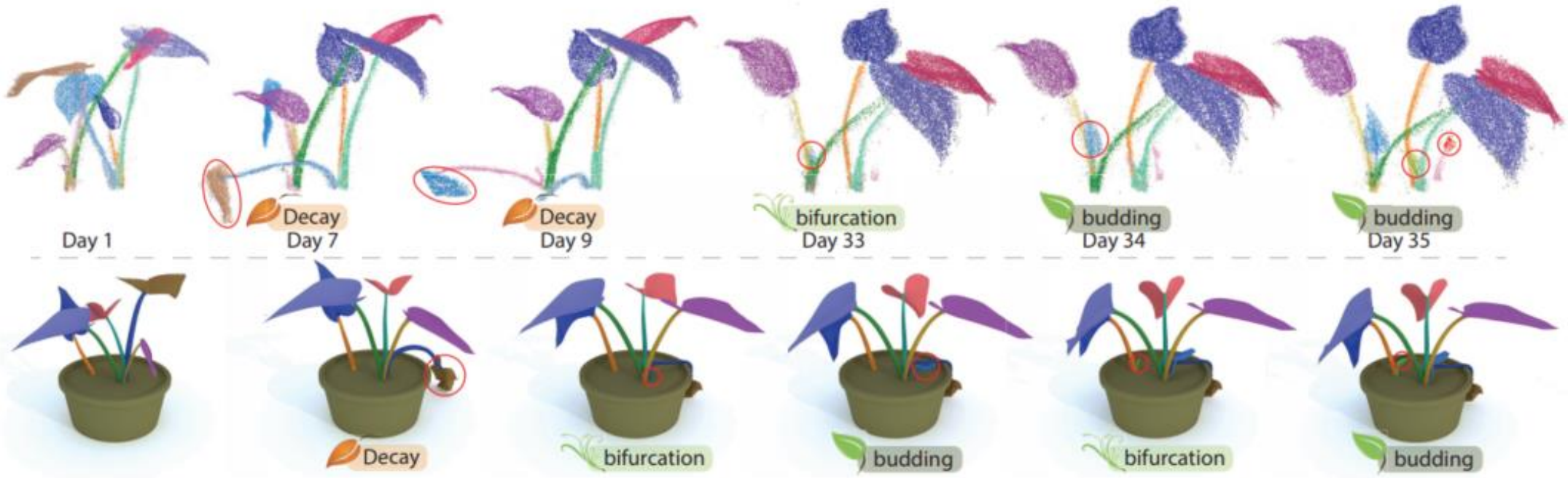


Livny et. al. (2010)

Digital plant growth reconstruction

- Growth model for:
  - Customizable, widely applicable and non-destructive study of plants
  - Animation
- Constructing plant growth models is still an open problem due to:
  - Large number of factors influencing the growth process
  - Inherent complexity of plant growth and architecture
  - Frequent need for prior botanical knowledge and/or accurate multi-temporal field data

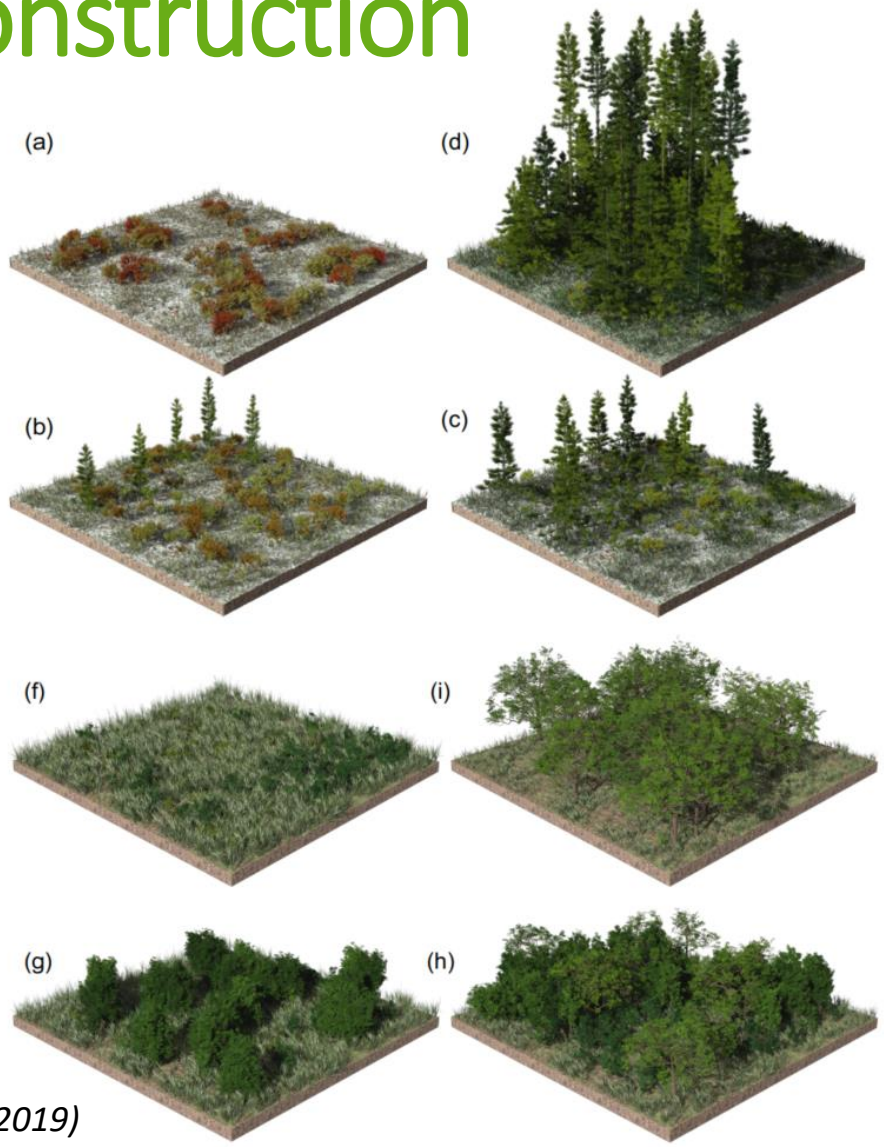
Digital plant growth reconstruction



Li et. al. (2013)

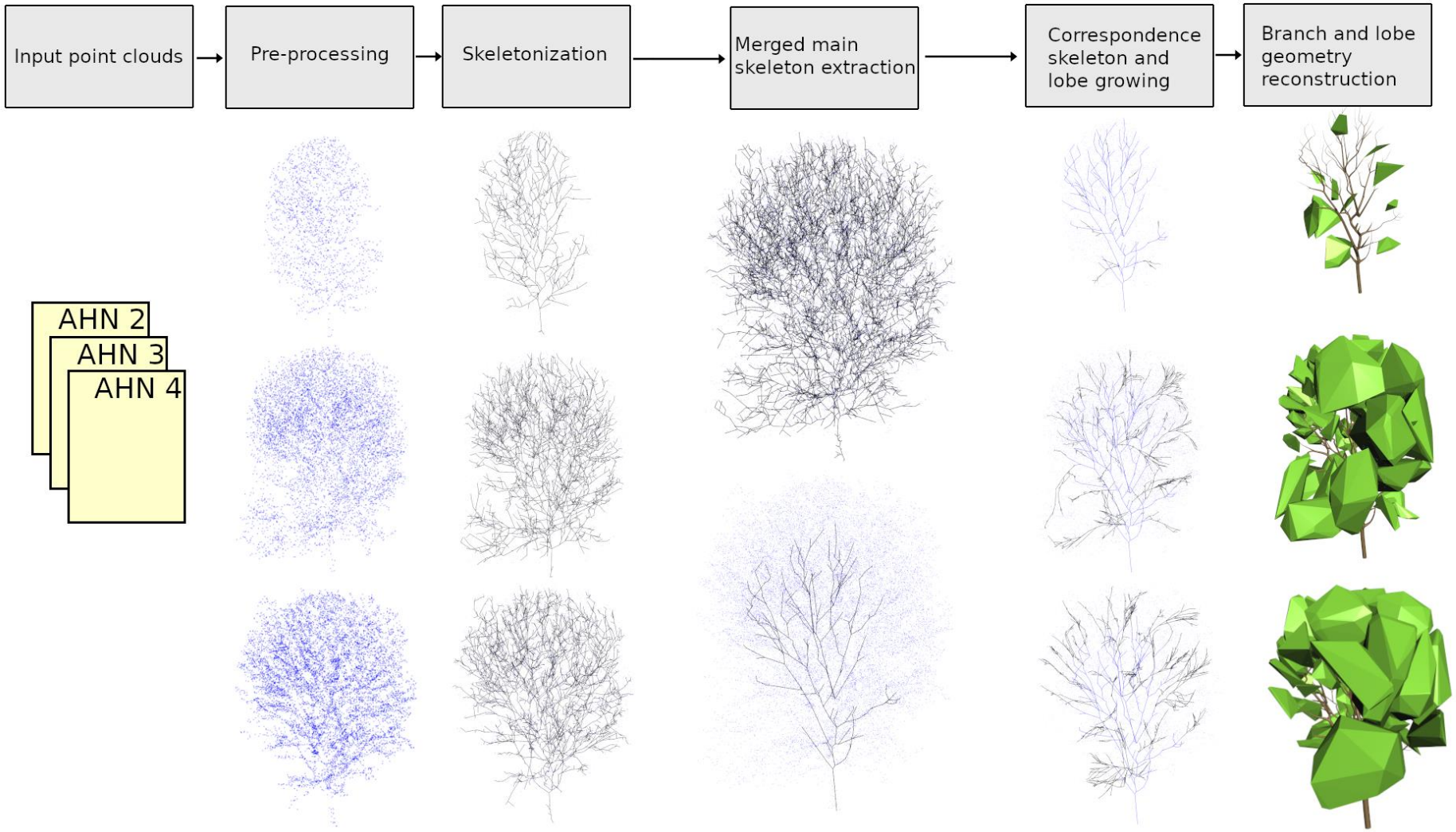
Digital plant growth reconstruction

Pirk et. al. (2012)



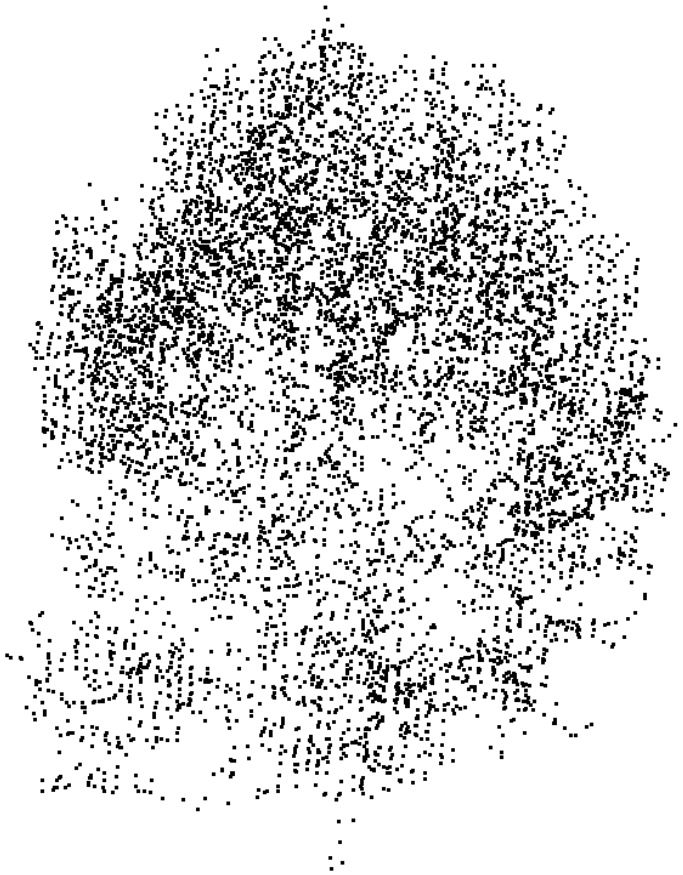
Makowski et. al. (2019)

Method overview

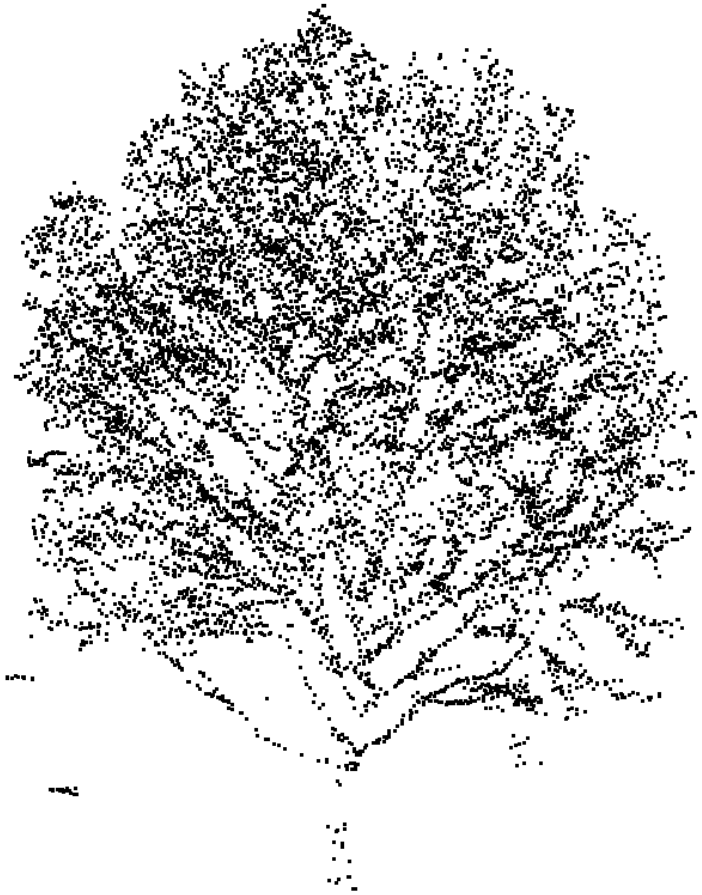


Input data

AHN2 (2009)



AHN3 (2016)

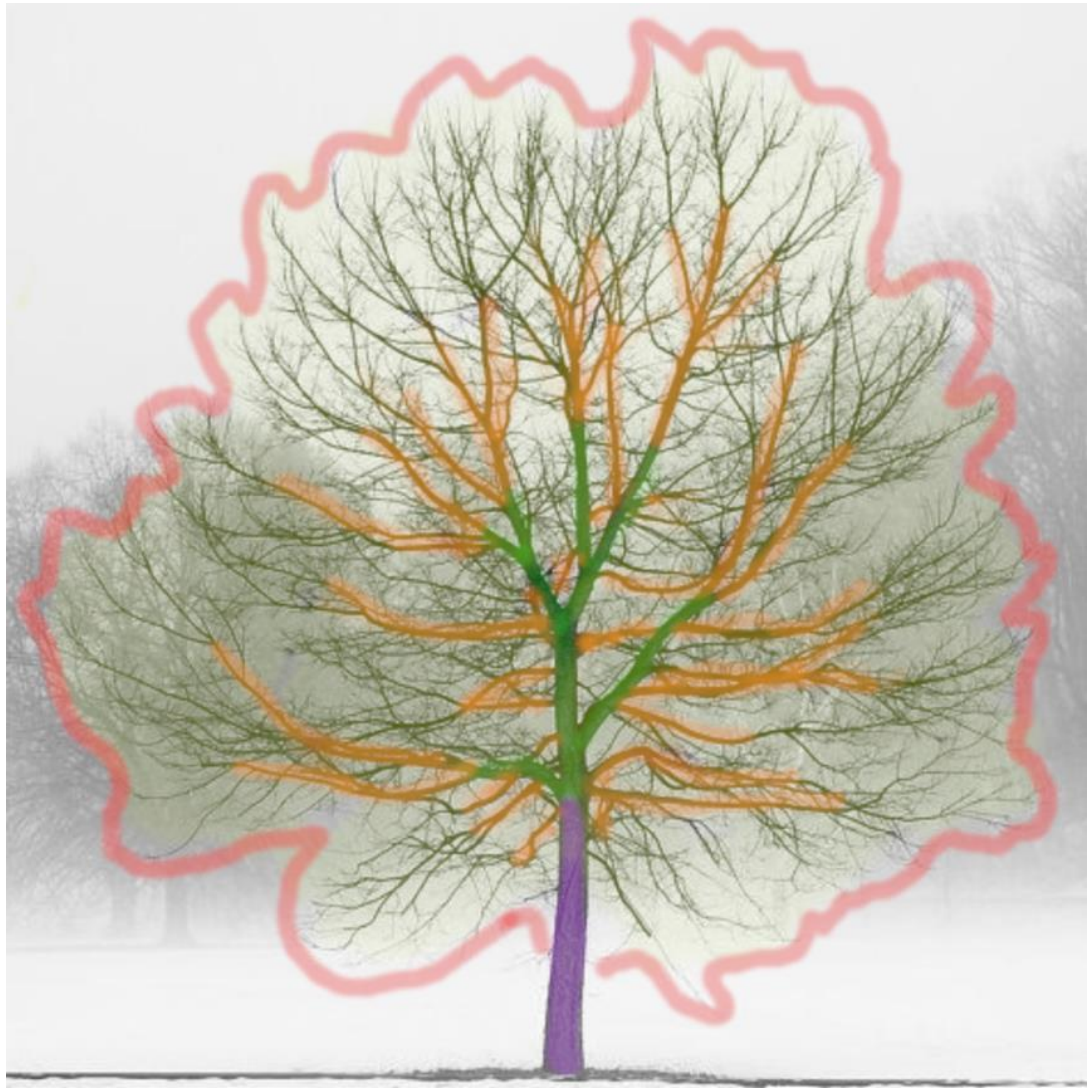


AHN4 (2020)

Point cloud cleaning and segmentation

- AHN point cloud segmentation and cleaning
  - Filter on class (AHN3 & AHN4)
  - Project points with 3+ returns to 2d grid, keep filled cells
  - Merge & process sets of cells into footprint polygons for AHN4
  - Use footprints to clip all point clouds
- Individual tree point cloud segmentation and cleaning
  - 2D clustering with convex hulls in lower region
  - Clip lower region with clusters
  - Keep 1 viable cluster center per tree footprint
  - Insert center for polygons without viable clusters
  - Draw radius around centers and filter trunk points
  - Insert trunk bases

Growth-based representation



- Stem:
  - Skeleton representation
  - AHN4 captures incompletely
- Main branches:
  - Skeleton representation
  - AHN4 captures completely
- Secondary branches:
  - Skeleton representation
  - AHN4 captures incompletely
- Canopy:
  - Lobe representation
  - AHN4 captures incompletely

Skeletonization

- Delaunay triangulation → Minimum Spanning Tree → Simplification

(a) Input LiDAR point cloud.

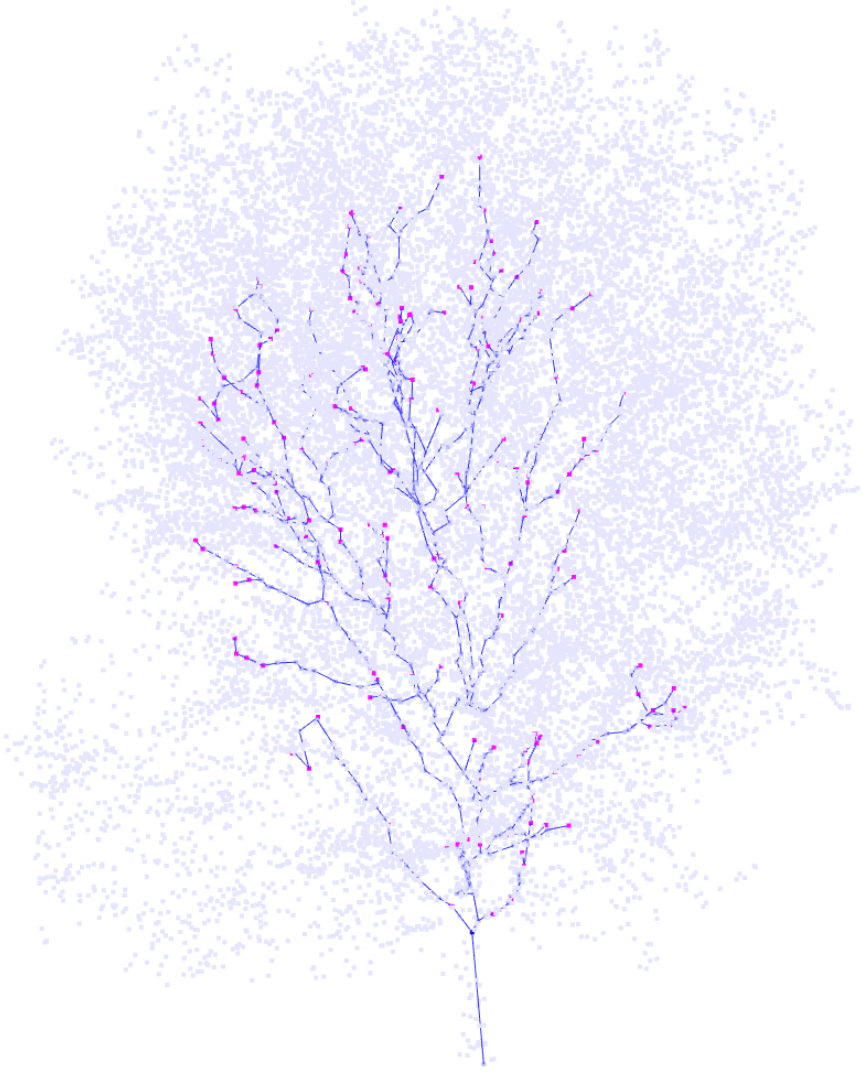
(b) DT of all the points.

(c) Minimum spanning tree.

(d) Simplified skeleton graph.

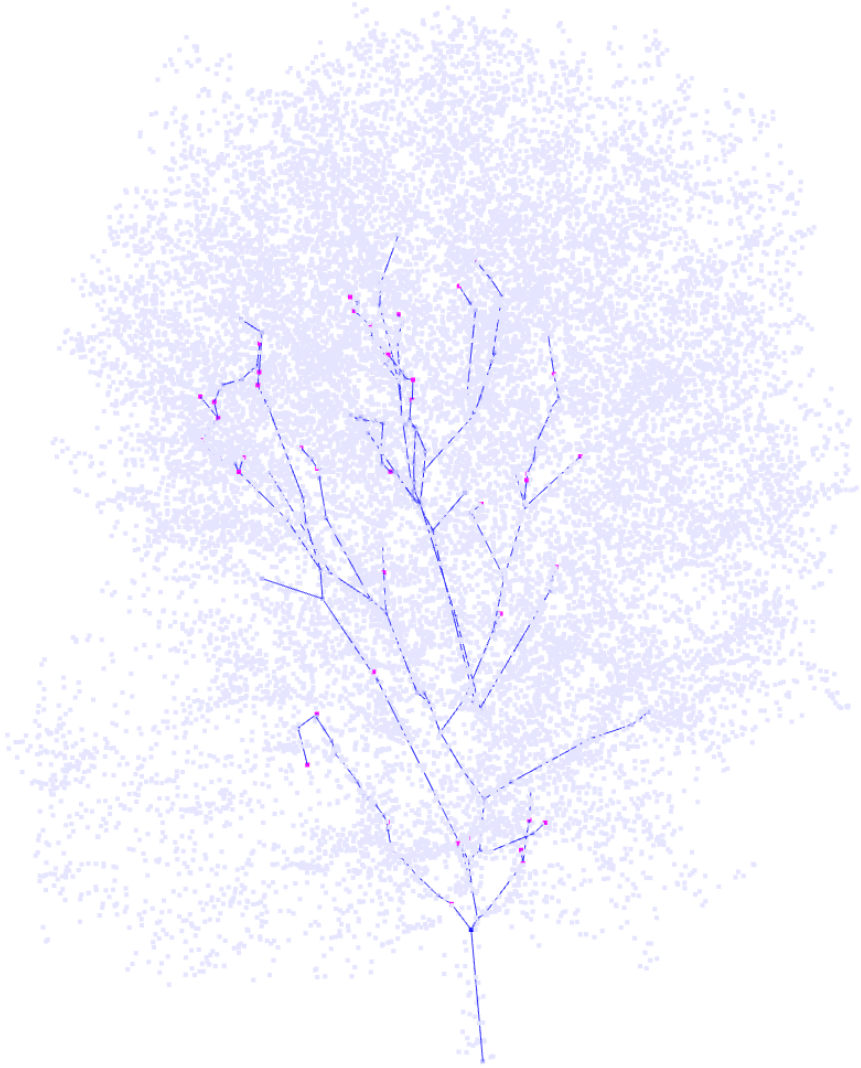
Merged main skeleton extraction

- Step 1: find corresponding main edges, connect them
  - Based on distance of endpoints to edges in all timestamps
  - If there is an edge within tolerance for all timestamps, it corresponds



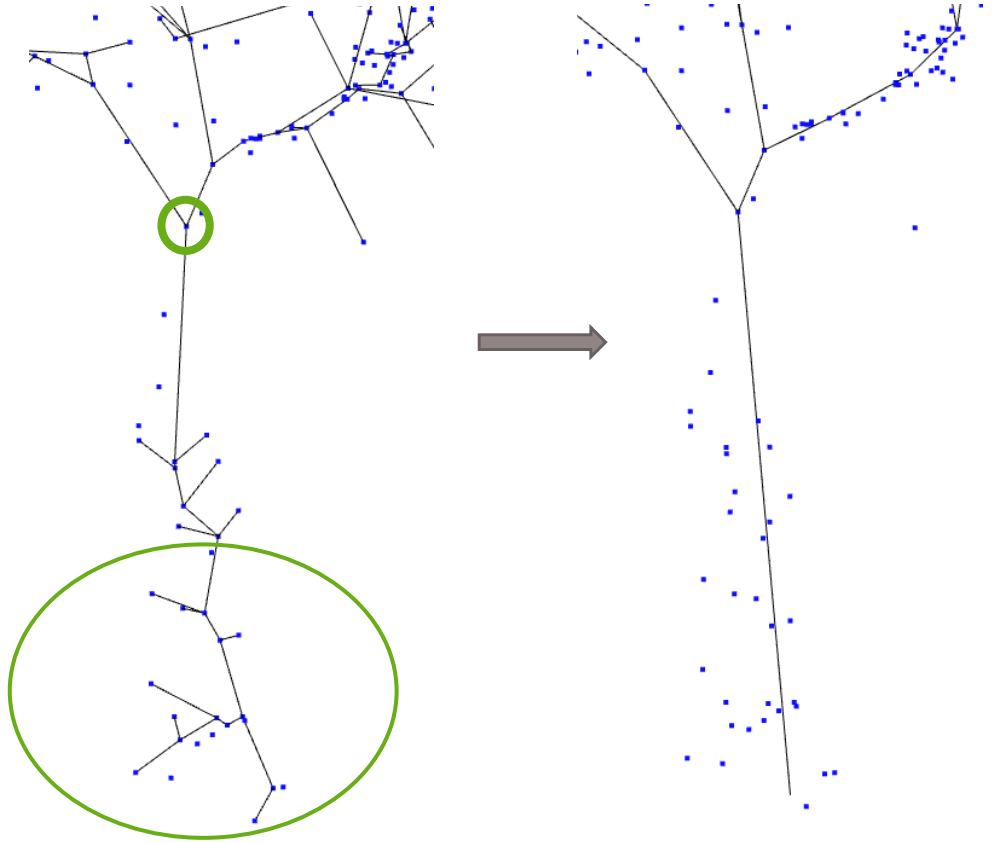
Merged main skeleton extraction

- Step 1: find corresponding main edges, connect them
- Step 2: simplify
  - Same as for skeletonization:
    - Douglas-Peucker line simplification for all straight segments
    - Remove very short segments



Merged main skeleton extraction

- Step 1: find corresponding main edges, connect them
- Step 2: simplify
- Step 3: improve
  - Remove all non-main edges
  - Find the first main bifurcation point
    - 2+ main children
    - At least 2 main children lead to a chain of 5 main edges
    - Maximum height threshold
    - If no point is found at the threshold, the most promising candidate is used in stead (most main children, longest main child chains)
  - Find the trunk base center (XY)
  - Replace trunk line with the line:
    trunk base center <> main bifurcation point

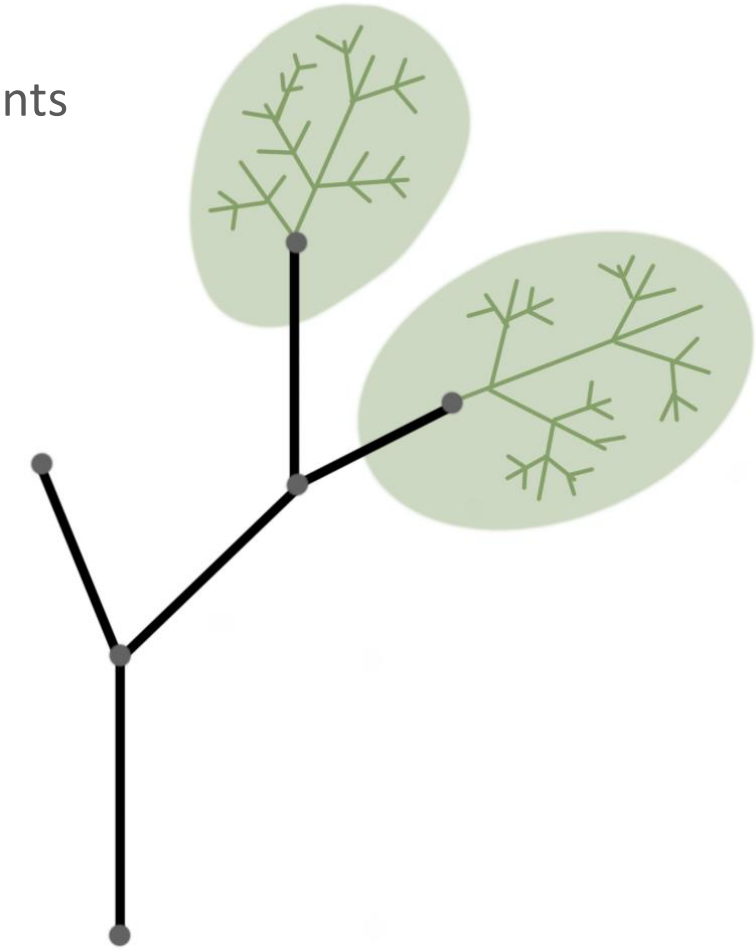


Correspondence

- Simplified timestamp skeleton → shared main skeleton + timestamp specific continuations
- Step 1: find all timestamp vertices that correspond to a main edge from the merged main skeleton, based on:
  - (Euclidean 3D) distance <= 20cm
  - OR distance <= 50cm and angle between edges <= 30°
- Step 2: make a new Delaunay triangulation of the timestamp points
  - Minus the corresponding points
  - Constrain it with the merged main edges
- Step 3 Continue skeletonization MST → simplification
- Step 4: detect lobe connector points
  - All main vertices with non-main child chains of at least 5 steps

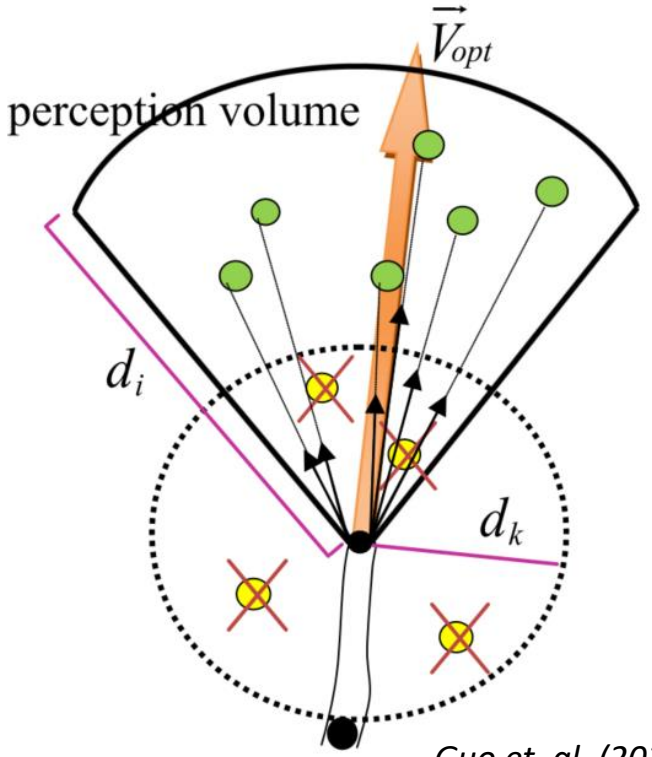
Lobe reconstruction

- A lobe can be described by the set of points spanned by the non-main edges connected to a (main) connector point
- For each lobe: compute the convex hull of these points
  - Filter out lobes with volume < $1m^3$
  - No other filters needed because lobes will not be hard constraint



Lobe growth

- Procedural region growing/space colonization algorithm
- Using an L-system notation
- Using original point cloud points as attractor points
  - d_i = perception cone depth = 4 * length parent edge
  - d_k = kill radius = 30 cm
  - Perception cone angle = 40°
  - For each new branch tip p_i find neighborhood points (n) within the perception cone
  - $V_{opt} = \sum_{j=0}^{n} V_j$
- For convex hull: add weighted inverse attraction point
- For lateral buds:
  - Enforce angle constraint so they don't both grow to the same attraction points
- Enforce a minimum and maximum internode length, as well as minimum distance between inserted points

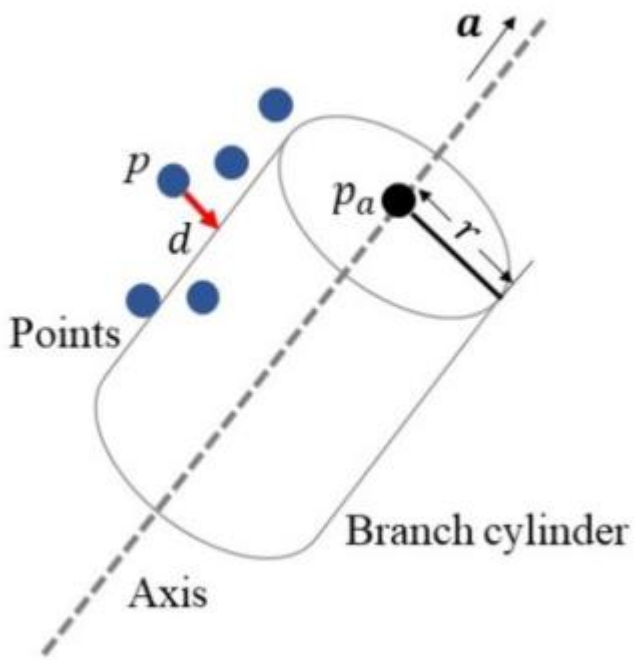


Guo et. al. (2020)

```
A = N internode T OR D
T = M internode A OR D
N = 2 * [internode A] OR [D]
M = 2 * [internode T] OR [D]
```

Geometry reconstruction

- For lobes: simply convex hull
- For branches: cylinder fitting
  - Constrain width of trunk with allometric rule based on crown measurements



Du (2019)

Growth between timestamps

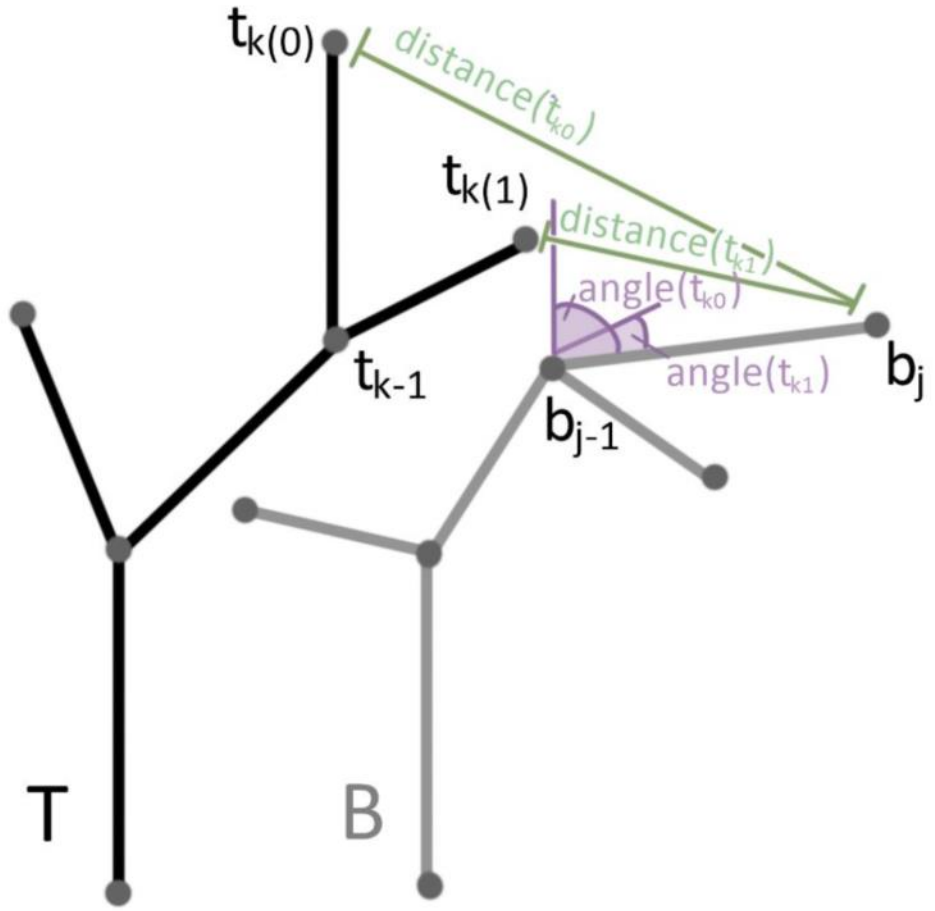
- Have: shared, corresponding main structure & growth model in lobes
- Interesting: what happens in between the known timestamps?
- How do the original timestamp structures relate to each other over time?

- → establish growth interpolation between the known timestamps

Correspondence

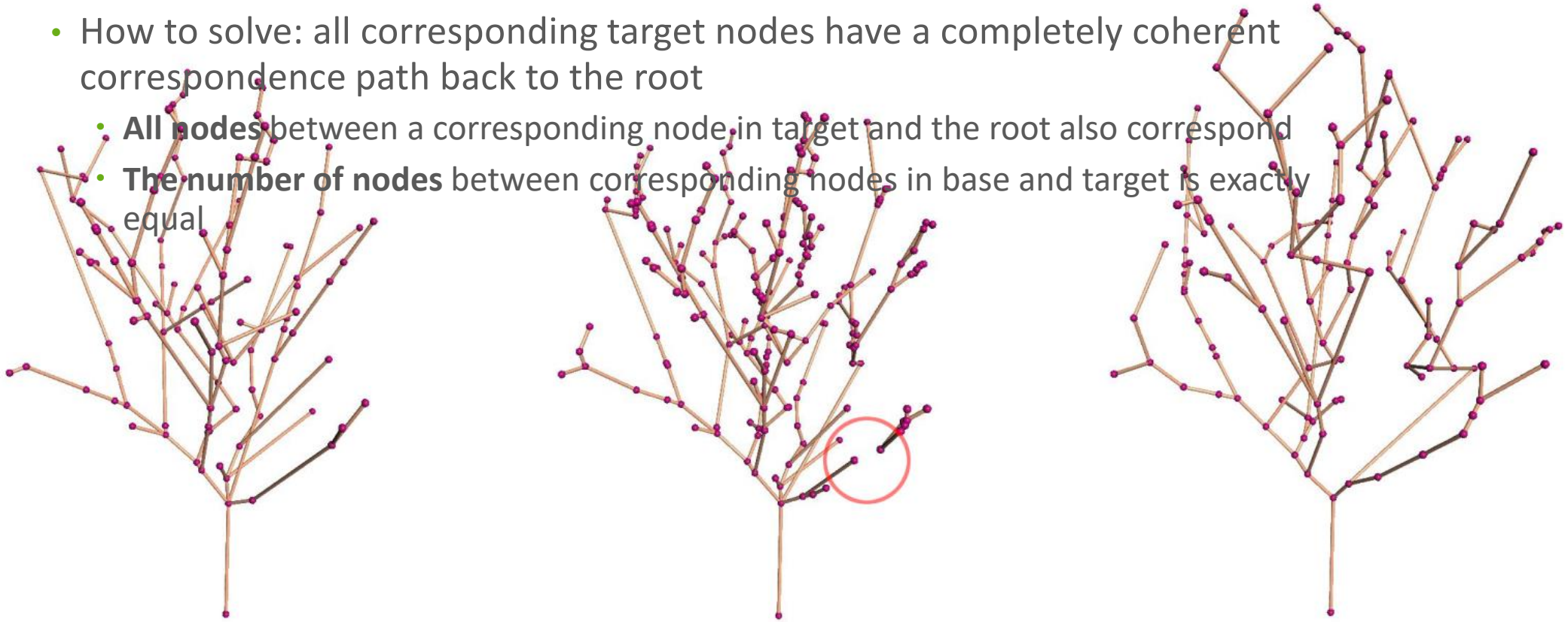
- Relate the nodes/edges of two different timestamp-specific skeleton graphs
- Base graph: younger timestamp
- Target graph: next timestamp (more recent)
- → transform base graph into target graph
  - Add nodes/edges
  - Remove nodes/edges
  - Transform nodes/edges
- Based on distance & angle:

1. distance $(b_j, t_k) <= \delta_{lower}$
2. distance $(b_j, p_k) <= \delta_{upper}$ AND angle $(b_j, t_k) <= \alpha$



Correspondence consistency

- Issue: gaps
- How to solve: all corresponding target nodes have a completely coherent correspondence path back to the root
  - **All nodes** between a corresponding node in target and the root also correspond
  - **The number of nodes** between corresponding nodes in base and target is exactly equal



(a) Timestamp 0 (b) Intermediate time (c) Timestamp 1

Correspondence consistency

- Step 1: detect all tip correspondences in target
  - Tip = the last node on a sub-branch for which correspondence was found

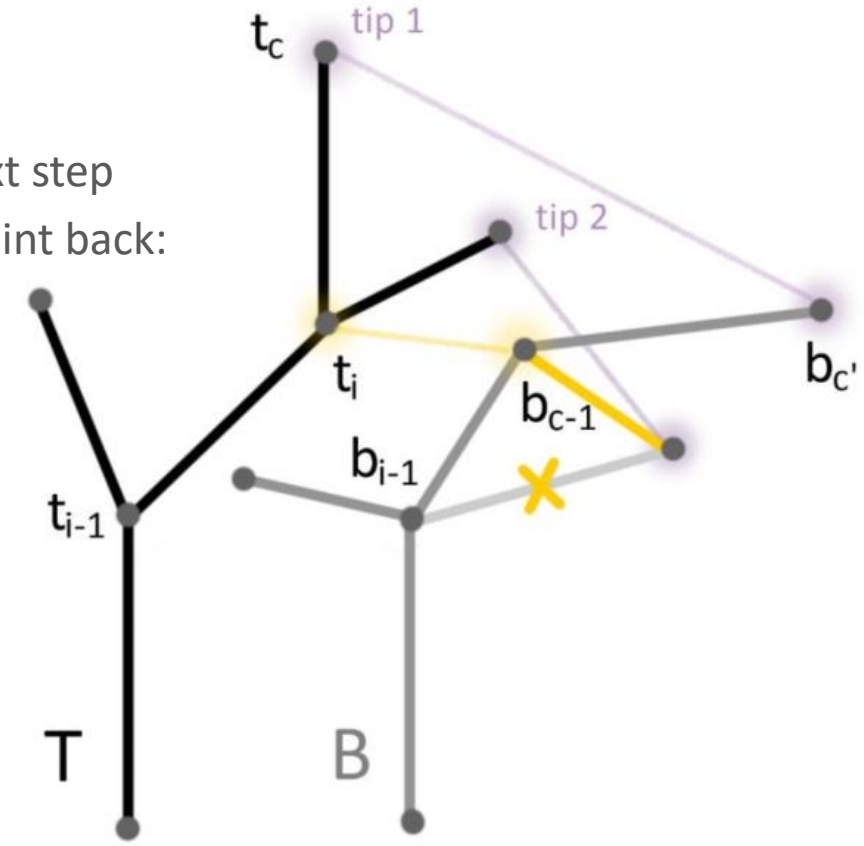
Correspondence consistency

- Step 1: detect all tip correspondences in target
- Step 2: walk back to the root from each tip, fix correspondences
  - For each step back in target, find the next correspondences (towards root) in both base and target
  - If they are not the same, remove edge between base node and parent
  - Establish new edge between next corresponding base node and base twin of next corresponding target node

Correspondence consistency

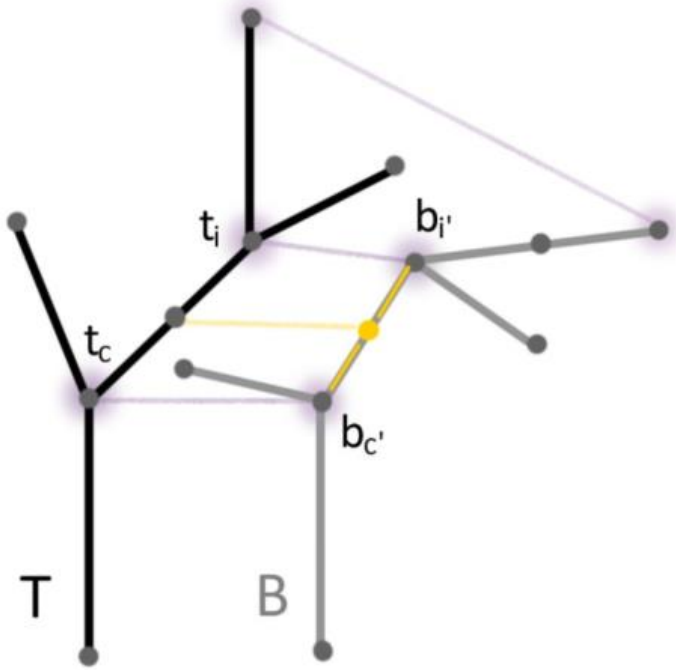
- Step 1: detect all tip correspondences in target
- Step 2: walk back to the root from each tip, fix correspondences
- Step 3: fix all bifurcation points
  - Corresponding bifur points were fixed in step 2, but empty ones are not guaranteed to be correct in next step
  - Depth-first traversal: if tip is found*, check first bifur point back:
    - If empty & base has bifur point there as well: assign
    - If empty & no empty base at that point: insert base vertex and assign
  - Find the rest of the tips leading to this bifur point
    - Remove old base edges & insert new ones towards the bifur point

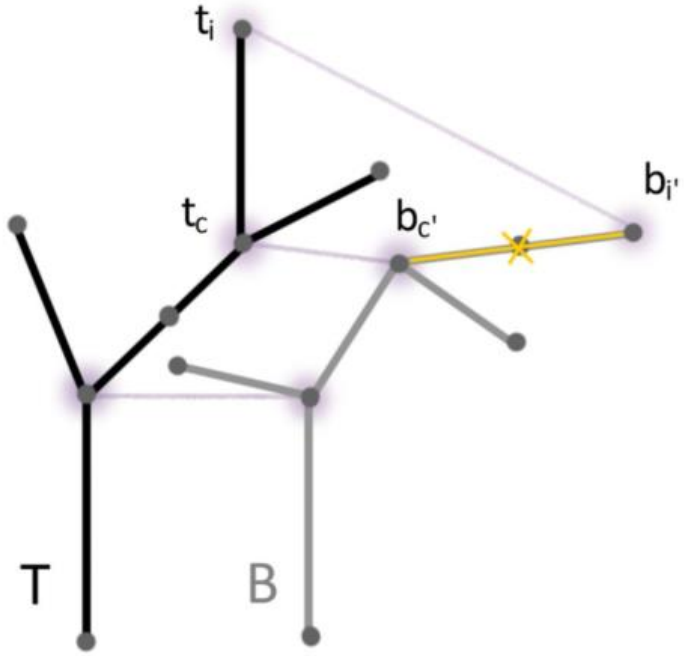
** note: edges are drawn between the nearest corresponding vertex of the sub-branch, not necessarily the tip vertex*



Correspondence consistency

- Step 1: detect all tip correspondences in target
- Step 2: walk back to the root from each tip, fix correspond
- Step 3: fix all bifurcation points
- Step 4: make number of nodes in correspondence path equal in base and target graphs
  - Find paths of non-corresponding vertices between two pairs of corresponding ones, in both base and target
  - Firstly, if base path > target path & target path > 0:
    - assign all available base vertices to the best target vertex (closest with guaranteed enough remaining for rest of target path)
  - Then, if base path > target path & target path == 0:
    - Collapse base vertices
  - If target path > base path:
    - Insert base vertices





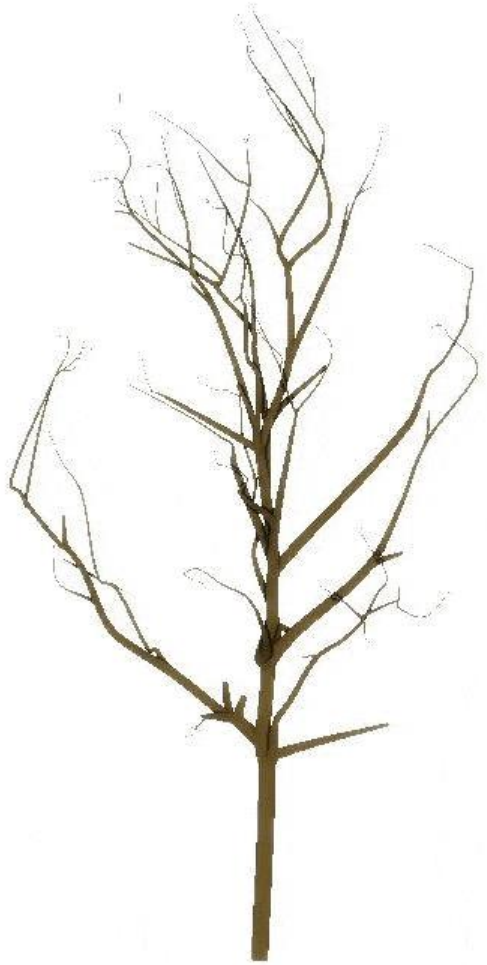
Interpolation

- Number of intermediary steps is user defined
- Interpolation = cubic spline curve interpolation between base and target coordinates of a node (from Easy3d, Nan (2021))
  - Deleted nodes will move towards the first existing target parent node
  - Added nodes will grow from first existing base parent node
- Base timestamp graph → target timestamp graph: by storing correspondence operations on all nodes and edges
  - Addition, deletion and transformation operations for each vertex and edge of both graphs
- Intermediary graph between each correspondence pair
  - Using stored correspondence operations, an intermediary graph can be constructed
  - Intermediary graph can have the shape of both the base and target graph, dependent on its node positions (2 sets of positions for each node)
  - Addition & deletion with placeholder vertices and extra edges

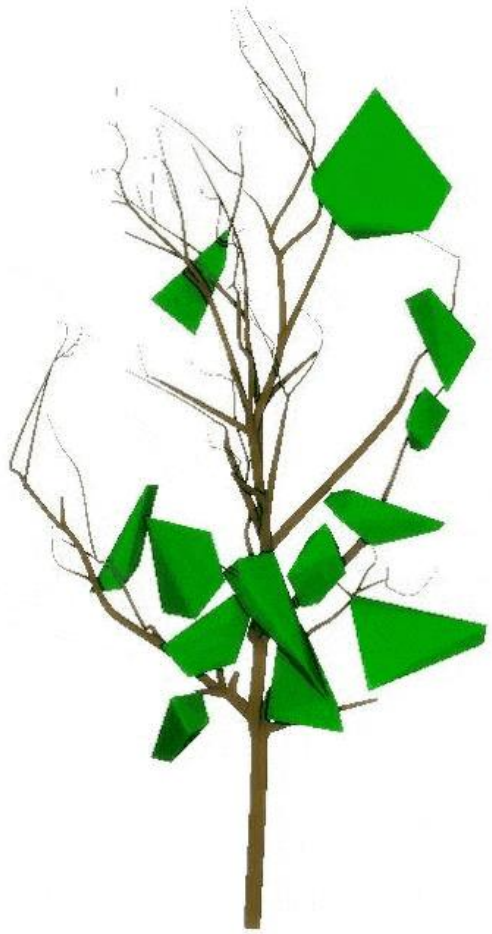
Visual results

(a) Timestamp 0 (AHN2). (b) Timestamp 1 (AHN3). (c) Timestamp 2 (AHN4).

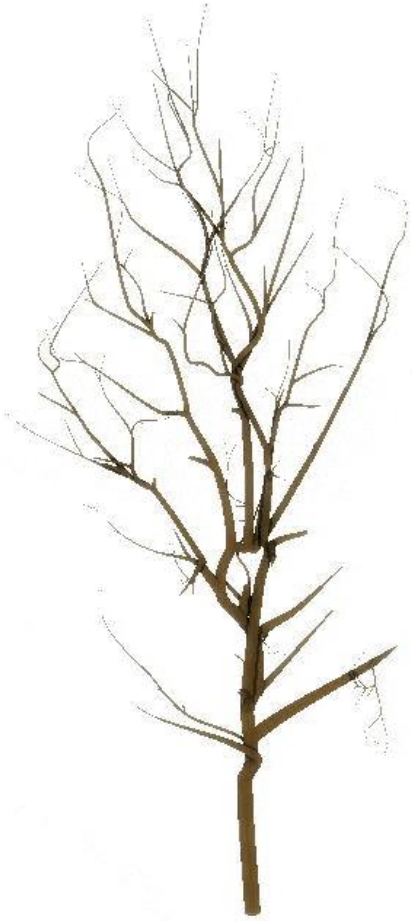
Corresponding reconstruction



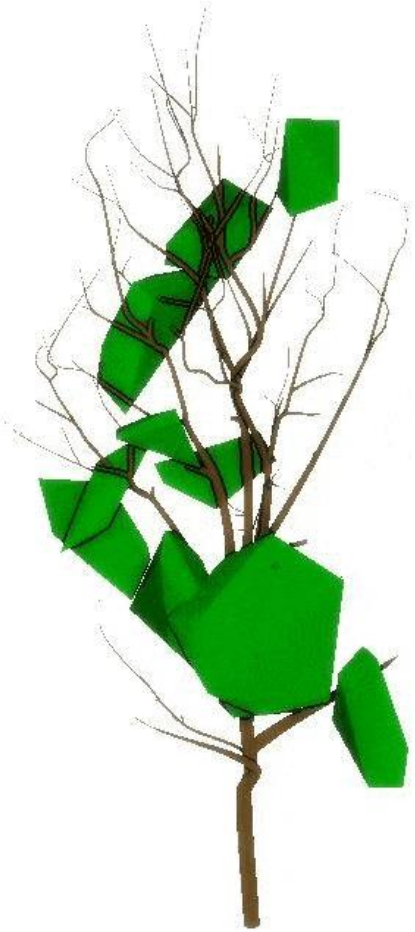
Corresponding reconstruction



Corresponding reconstruction

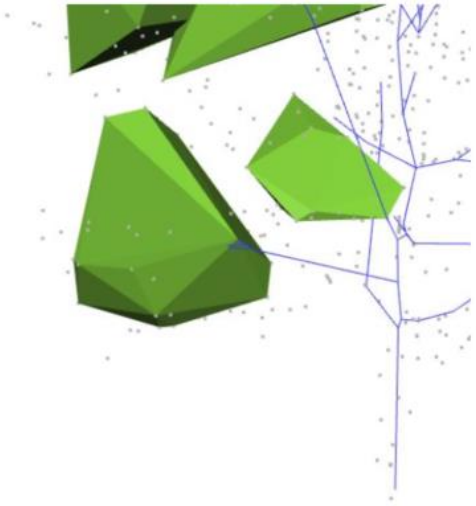


Corresponding reconstruction

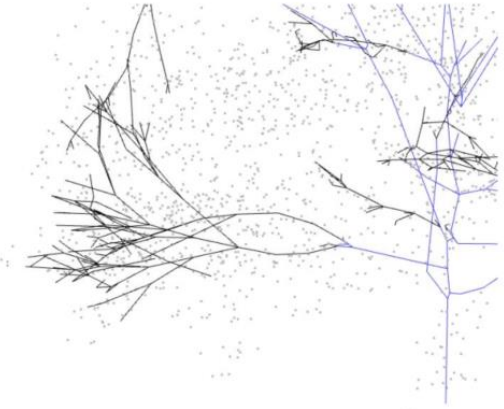


Lobe reconstruction

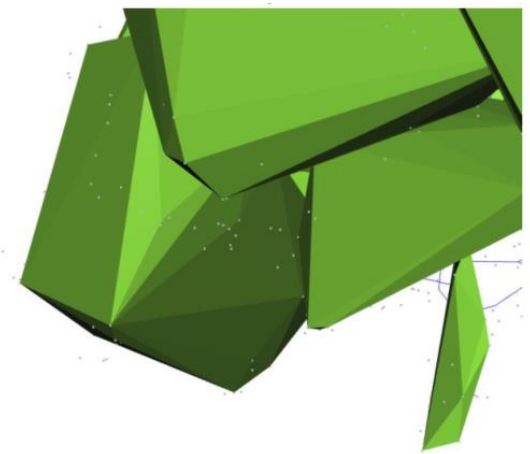
(a) Timestamp 0.



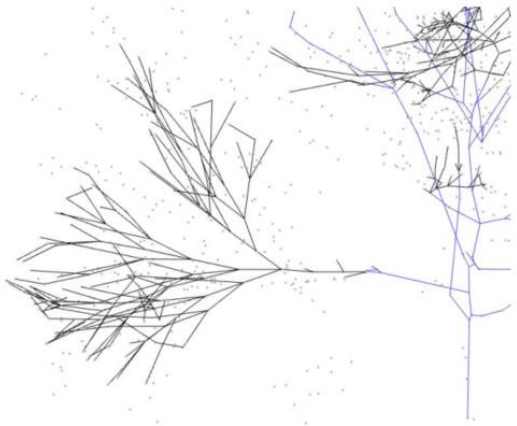
(b) Timestamp 0 mesh.



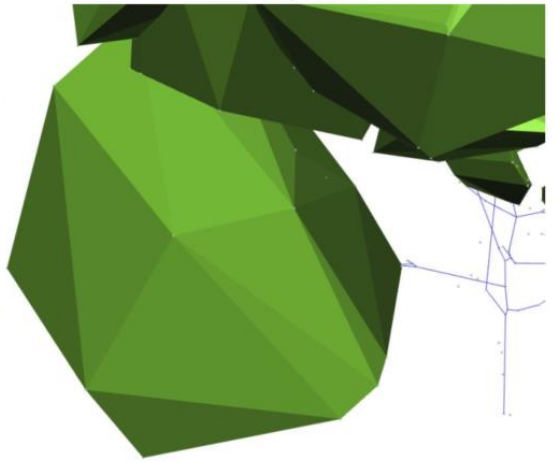
(c) Timestamp 1.



(d) Timestamp 1 mesh.



(e) Timestamp 2.



(f) Timestamp 2 mesh.

Timestamp reconstruction

(a) Timestamp 0: time specific.

(b) Timestamp 1: time specific.

(c) Timestamp 2: time specific.

(d) Timestamp 0: corresponding.

(e) Timestamp 1: corresponding.

(f) Timestamp 2: corresponding.

Timestamp reconstruction

(a) Timestamp 0: time specific.

(b) Timestamp 1: time specific.

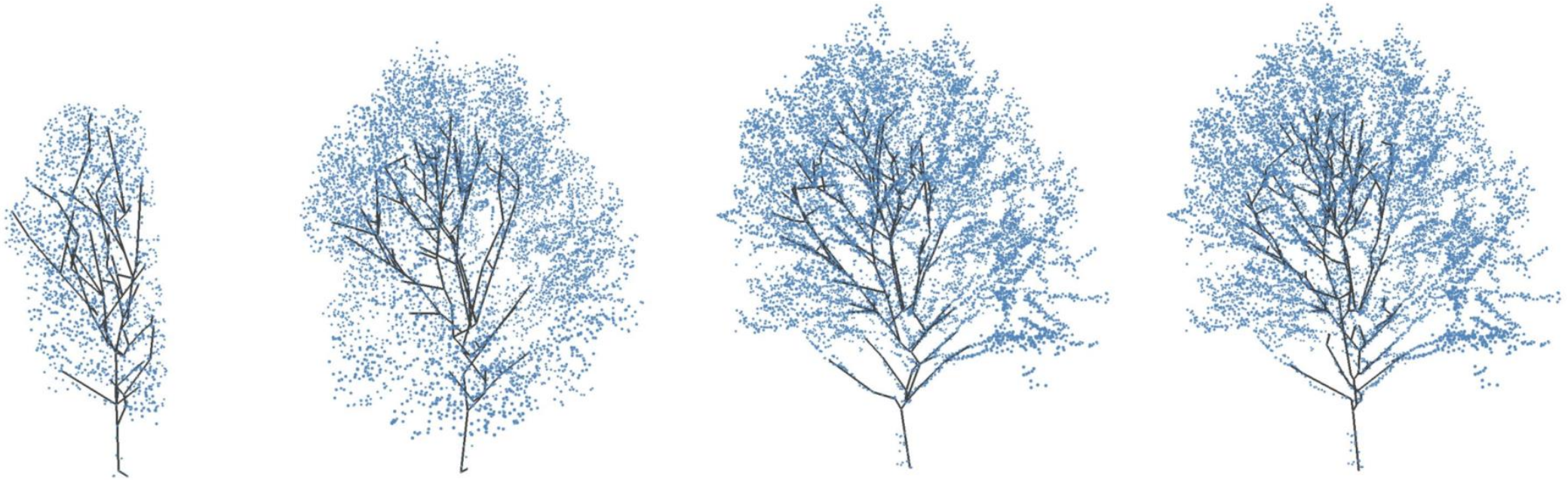
(c) Timestamp 2: time specific.

(d) Timestamp 0: corresponding.

(e) Timestamp 1: corresponding.

(f) Timestamp 2: corresponding.

Reconstruction and input point clouds



(a) Timestamp 0

(b) Timestamp 1.

(c) Timestamp 2.

(d) Merged main.

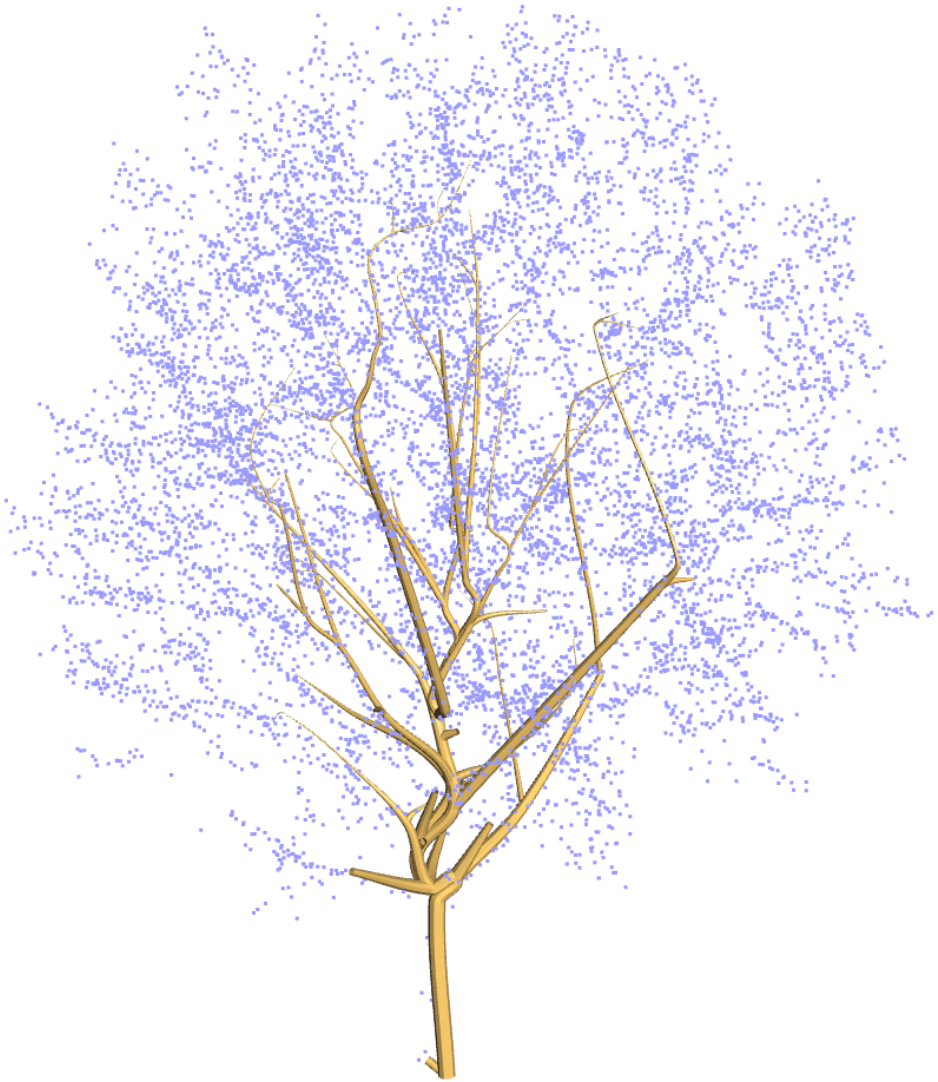
Merged and timestamp, with input clouds

(a) Timestamp-specific reconstruction

(b) Merged main reconstruction

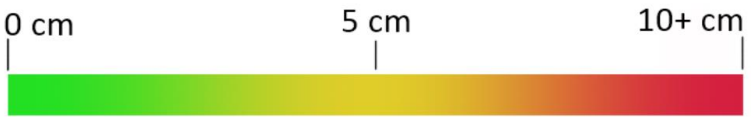
Growth interpolation

Growth interpolation

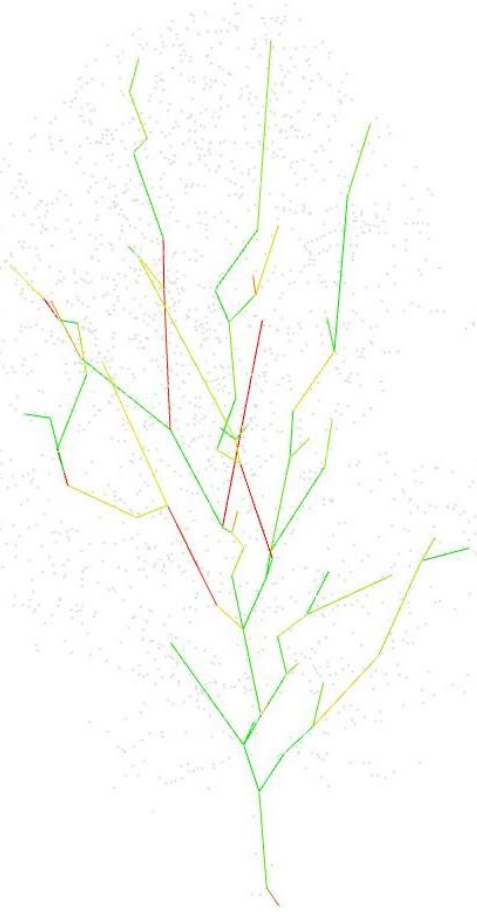


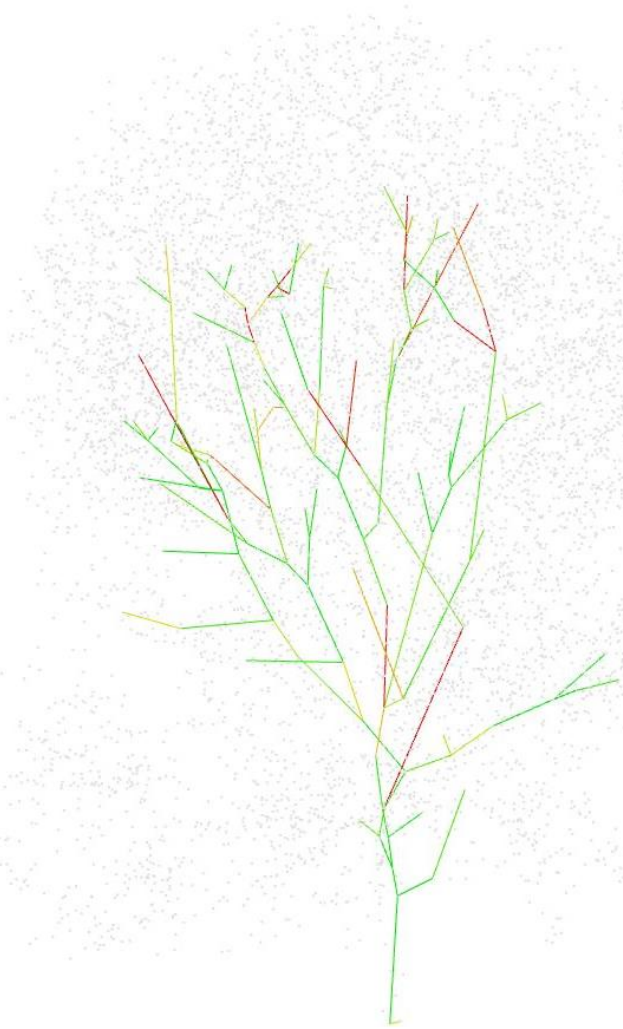
Quantitative analysis

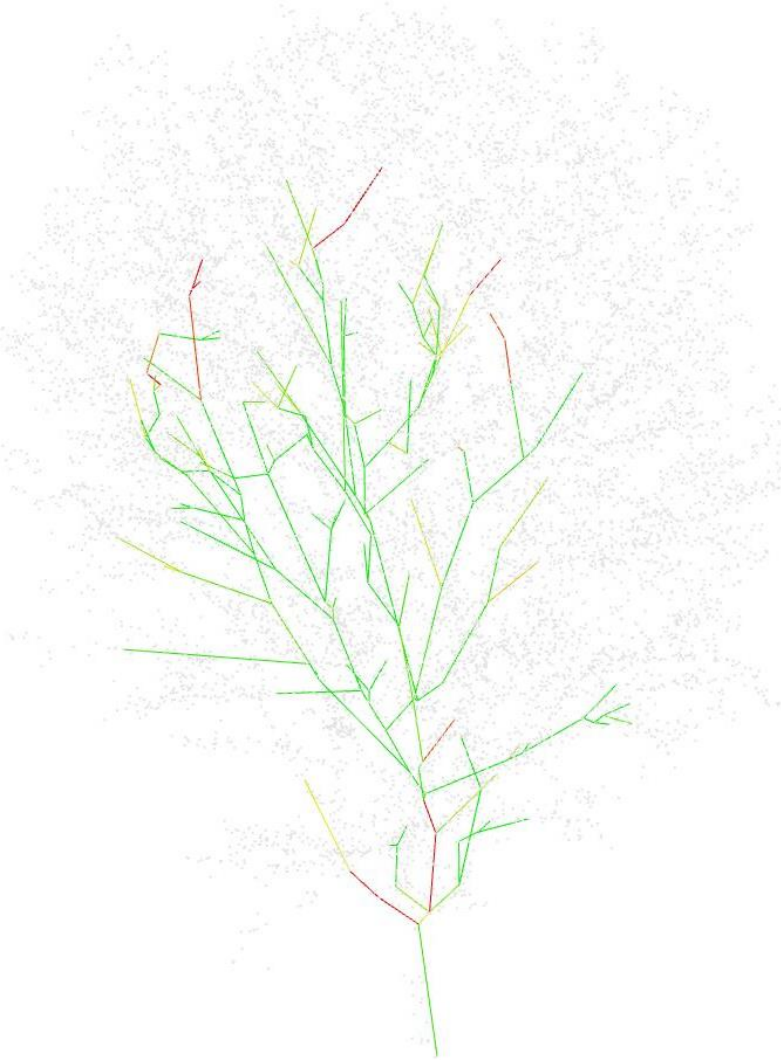
- Computing distance between skeleton graphs is an open problem
- Measures used in this work:
  - Geometric distance
  - Topological distance



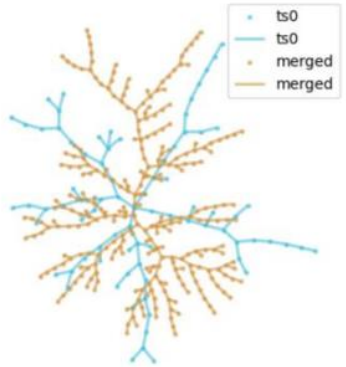
Geometric distance



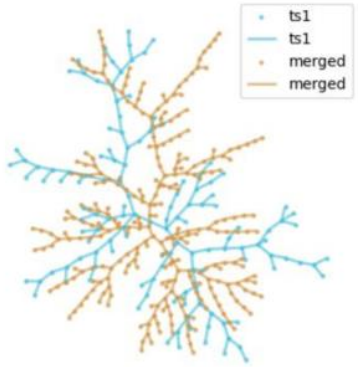




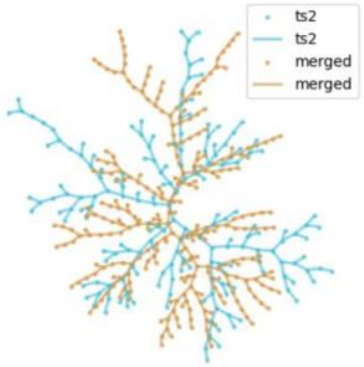
Topological distance



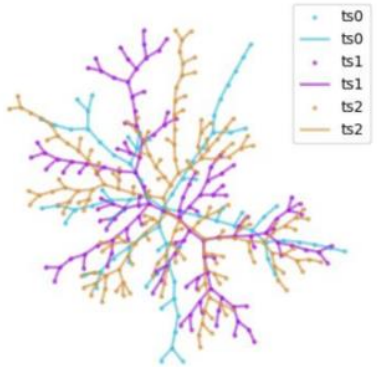
(a) Timestamp 0 and merged main.



(b) Timestamp 1 and merged main.



(c) Timestamp 2 and merged main.



(d) All timestamps together

Visualized using Kamada-Kawaii path-length cost function

Topological distance

Tree A	time 0	time 1	time 2	merged	# nodes	# edges
time 0	-	218	338	450	59	58
time 1	216	-	326	440	122	121
time 2	336	326	-	414	180	179
merged	448	434	412	-	235	234

Tree B	time 0	time 1	time 2	merged	# nodes	# edges
time 0	-	100	128	236	60	59
time 1	102	-	132	246	39	38
time 2	126	132	-	230	76	75
merged	236	246	230	-	134	133

Tree C	time 0	time 1	time 2	merged	# nodes	# edges
time 0	-	192	324	442	88	87
time 1	190	-	322	444	103	102
time 2	324	230	-	438	177	176
merged	442	442	434	-	233	232

Tree D	time 0	time 1	time 2	merged	# nodes	# edges
time 0	-	244	346	520	80	79
time 1	244	-	332	500	132	131
time 2	338	330	-	494	185	184
merged	518	500	494	-	275	274

- Distance = Graph Edit Distance
- Measures the cost of the minimum topological steps needed to transform a graph into one isomorphic to another
  - Insertion node/edge
  - Deletion node/edge
  - Substitution node/edge
- In this work, the cost of every step is 1

Limitations

- By establishing a main structure, the detail of the original timestamp data is lost
- Clean, segmented data is required
- There is no clean-cut way to validate the timestamp reconstructions
- There is no clean-cut way to validate the accuracy of the tree growth
- There is no clean-cut way to interpolate between trees

Conclusion

- Goal: research how multi-temporal LiDAR point clouds of trees could be used to model tree growth
- Results:
  - Establishing correspondence between main branches of different timestamps makes the reconstruction more robust
  - Information from other timestamps can be used to fill in gaps
  - The lobe/main skeleton representation was suitable for growth modelling
  - Growth could be modelled with multi-temporal timestamp data, also in between the known times

Future work

- Leaf geometry/added realism
- More involved strategy for establishing correspondences
  - Will make growth interpolation more biologically faithful
- Use additional structural and biological data to inform the reconstruction and growth model
- Lobe/tree crown interpolation
- Machine learning
- Species profile
- Incorporating the environment into the reconstruction/growth model
- Improved segmentation & processing of tree clusters

Acknowledgements

- Liangliang Nan
- Sören Pirk
- Jantien Stoter
- Dirk Voets & Cobra Groeninzicht

Thank you!

- Questions?

References

- Du, S., Lindenbergh, R., Ledoux, H., Stoter, J., and Nan, L. (2019). Adtree: Accurate, detailed, and automatic modelling of laser-scanned trees. *Remote Sensing*, 11(18).
- Guo, J., Xu, S., Yan, D.-M., Cheng, Z., Jaeger, M., and Zhang, X. (2020). Realistic procedural plant modeling from multiple view images. *IEEE Transactions on Visualization and Computer Graphics*, 26(2):1372–1384.
- Hu, S., Li, Z., Zhang, Z., He, D., and Wimmer, M. (2017). Effcient tree modeling from airborne lidar point clouds. *Computers & Graphics*, 67:1–13.
- Li, Y., Fan, X., Mitra, N. J., Chamovitz, D., Cohen-Or, D., and Chen, B. (2013). Analyzing growing plants from 4d point cloud data. *ACM Transactions on Graphics (TOG)*, 32(6):1–10.
- Livny, Y., Pirk, S., Cheng, Z., Yan, F., Deussen, O., Cohen-Or, D., and Chen, B. (2011). Texture-lobes for tree modelling. *ACM Transactions on Graphics (TOG)*, 30(4):1–10.
- Makowski, M., Hädrich, T., Scheffczyk, J., Michels, D. L., Pirk, S., and Pałubicki, W. (2019). Synthetic silviculture: multi-scale modeling of plant ecosystems. *ACM Transactions on Graphics (TOG)*, 38(4):1–14.
- Nan, L. (2021). Easy3D: a lightweight, easy-to-use, and effcient C++ library for processing and rendering 3D data. *Journal of Open Source Software*, 6(64):3255.
- Pirk, S., Stava, O., Kratt, J., Said, M. A. M., Neubert, B., Mˇech, R., Benes, B., and Deussen, O. (2012). Plastic trees: interactive self-adapting botanical tree models. *ACM Transactions on Graphics (TOG)*, 31(4):1–10.