UHASSELT
KNOWLEDGE IN ACTION

Maastricht University

Faculteit Wetenschappen
School voor Informatietechnologie

master in de informatica

Masterthesis

Puntenwolken van bomen parametiseren om nieuwe variaties te genereren

Sem Heleven

Scriptie ingediend tot het behalen van de graad van master in de informatica

PROMOTOR :

dr. Nick MICHIELS

BEGELEIDER :

De heer Steven MOONEN

2021
2022

De transnationale Universiteit Limburg is een uniek samenwerkingsverband van twee universiteiten in twee landen: de Universiteit Hasselt en Maastricht University.

UHASSELT
KNOWLEDGE IN ACTION

www.uhasselt.be
Universiteit Hasselt
Campus Hasselt:
Martelarenlaan 42 | 3500 Hasselt
Campus Diepenbeek:
Agoralaan Gebouw D | 3590 Diepenbeek

Maastricht University

Faculteit Wetenschappen
School voor Informatietechnologie

master in de informatica

Masterthesis

Puntenwolken van bomen parametiseren om nieuwe variaties te genereren

Sem Heleven

Scriptie ingediend tot het behalen van de graad van master in de informatica

PROMOTOR :
dr. Nick MICHIELS

BEGELEIDER :
De heer Steven MOONEN

Universiteit Hasselt

Masterproef voorgedragen tot het behalen van de graad van master in de informatica

Puntenwolken van bomen parametriseren om nieuwe variaties te genereren.

Auteur:

Heleven Sem

Promotor:

Dr. Nick Michiels

Begeleider(s):

De heer Steven Moonen

Academiejaar 2021-2022

Dankwoord

Graag wil ik Nick en Steven bedanken voor de meetings, de vele goede ideeën en het geduld met mij. Ook wil ik mijn medestudenten bedanken voor de nodige motivatie en ontspanning.

Samenvatting

In deze thesis wordt er gekeken als het mogelijk is om variaties te genereren van een boom. Dit wordt gedaan door een scan van een boom als puntenwolk om te zetten naar een verzameling van cilinders die de takken voorstellen. Met deze cilinders kunnen dan makkelijk verschillende parameters berekend worden, zoals de tak lengte, de hoek tussen gesplitste takken, de vorm van de kruin en nog enkele andere. Deze implementatie wordt in diepgang uitgelegd in Hoofdstuk: 3 Deze parameters worden dan gebruikt als input in een 3D L-systeem om zo variaties te maken. Eerst wordt dit analytisch getest door te beginnen met een gegenereerde boom van het L-systeem, deze wordt dan omgezet naar een puntenwolk. De puntenwolk wordt dan gebruikt als input voor deze thesis, de berekende parameters worden dan in Hoofdstuk: 4 vergeleken met de initiële parameters om de accuraatheid van de methode te testen. Vervolgens heb ik de technieken ook geëvalueerd met een praktische test op echte bomen. Hierbij heb ik gebruik gemaakt van photogrammetry om zelf een boom te scannen en een puntenwolk te bekomen. Ook wordt de methode toegepast op een lidar scan van een boom dat afkomstig is van en publieke dataset. De resultaten van analytische manier geven een hoge accuraatheid aan. De resultaten van de echte bomen werken minder goed, dit kan zijn door factoren zoals slechte scans en moeilijke bomen. Er zijn echter nog veel mogelijkheden voor uitbreiding en robuustheid.

Inhoudsopgave

1 Inleiding **5**
- 1.1 Situering . . . 5
- 1.2 Onderzoeksvraag . . . 7
- 1.3 Use cases . . . 7

2 Gerelateerd werk **8**
- 2.1 Puntenwolk . . . 8
- 2.2 Puntenwolk segmentatie . . . 9
- 2.3 Boom model opbouwen . . . 10
  - 2.3.1 Single Image Tree Modeling . . . 10
  - 2.3.2 3D Tree Reconstruction from Simulated Small Footprint Waveform Lidar . . . 10
  - 2.3.3 Automatic Reconstruction of Tree Skeletal Structures from Point Clouds . . . 11
  - 2.3.4 TreeQSM . . . 11
- 2.4 L-systeem . . . 13

3 Implementatie **14**
- 3.1 2D L-systeem . . . 14
- 3.2 3D Bomen: Analytisch . . . 15
  - 3.2.1 Boom generatie . . . 15
  - 3.2.2 Omzetten naar puntenwolk . . . 18
  - 3.2.3 Tree QSM . . . 19
  - 3.2.4 Parameter extractie . . . 19
- 3.3 Echte boom . . . 21

4 Resultaten **23**
- 4.1 Analytisch . . . 23
- 4.2 Echte bomen . . . 28

5 Uitbreidingen/verbeteringen **31**

6 Conclusies **32**

Hoofdstuk 1

Inleiding

1.1 Situering

Bomen komen voor in verschillende vormen, groottes en types. Deze zijn belangrijk voor de biodiversiteit en landbouw zoals appel- of peren- bomen. Hier kan het belangrijk zijn om bomen te classificeren of te digitaliseren. Een digitale boom kan op verschillende manieren gemaakt en opgeslagen worden. Het kan een volledige mesh zijn, opgebouwd uit punten en kleine vlakken of een andere modeleer structuur dat bestaat uit wiskundige formules om curves voorstellen die dan worden opgeblazen om takken te maken. Deze representaties kunnen enkel manueel worden gemaakt of gegenereerd door software. Om een echte boom te digitaliseren moet deze boom gescand worden om zo alle vormen en takken op te nemen in het model ofwel moet een designer de boom volledig nabouwen in een modelleer omgeving.

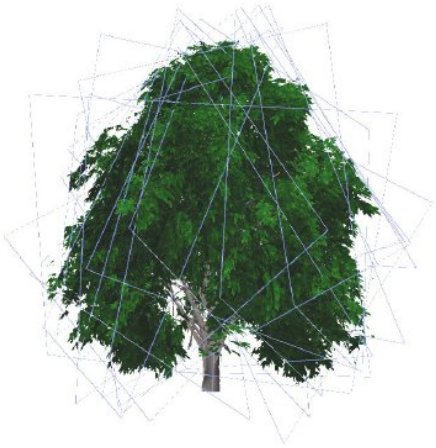
Een andere manier om een boom model op te slaan, is door het gebruik van een puntenwolk. Dit is een verzameling van punten dat op de buitenkanten van de boom liggen. Een puntenwolk kan makkelijk gemaakt worden door verschillende technieken, deze technieken worden verder aangehaald in Hoofdstuk: 2. Een puntenwolk kan zo de volledige vorm van een boom bevatten, maar het heeft ook een minpunt. Omdat dit gewoon losse 3D punten zijn met een kleur is het moeilijk om deze data te gebruiken. Zo kan het zeer reken intensief zijn om de boom te visualiseren in een 3D omgeving.

Er bestaan echter ook manieren om automatisch een boom om te zetten in een model, een voorbeeld is in de paper van Shlyakhter et al. [Shlyakhter et al., 2001]. Hier vertrekken zij van een tiental afbeeldingen van dezelfde boom en zet deze om naar een 3D model. Hier gebruiken ze de omtrek van de boom om de takken structuur te bepalen. Hierdoor krijgen ze een boom dat uiteindelijk hetzelfde silhouet heeft, maar een andere verdeling van de stam en aftakkingen.

Wanneer een echte boom omgezet is naar een digitale boom kan deze voor verschillende toepassingen gebruiken worden, zoals in een spel. Maar dan is er nog altijd enkel 1 boom, om een andere te krijgen kan moet er dan opnieuw een andere boom gescand worden of er moet manueel de eerste boom aangepast worden. Dit is veel werk en dus is het interessant om variaties van bomen te kunnen genereren.

Er zijn verschillende manieren om te interpreteren wat een variatie van een boom is. Dit kan gezien worden als letterlijk takken aanpassen van een boom om zo een andere boom te bekomen, maar dit kan er voor zorgen dat de variatie opgebouwd is uit een andere structuur en dus niet meer dezelfde soort is als de oorspronkelijke boom. Zo werkt de manier van Argudo et al. [Argudo et al., 2020]. Hier gebruiken zij een afbeelding van een boom als input en verdelen deze dan in stam en kruin. Deze stam en kruin worden dan vervormd om te passen op een silhouet van een andere boom. Hun techniek werkt voor 2D bomen, maar kan ook omgezet worden naar een 3D model, een bilboard cloud(Figuur: 1.1). Dit is een model dat bestaat uit een aantal 3D vlakken met een textuur. Dit zorgt voor een 3D effect, maar het is geen echt 3D model. Een betere manier om variaties te maken is om de groei structuur in rekening te brengen zodat de nieuwe boom nog steeds dezelfde type of soort boom is.

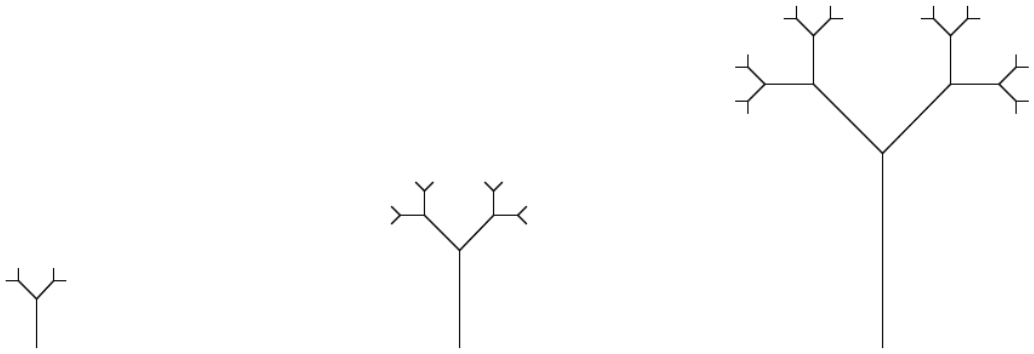
Om dus een variatie te maken moet de boom wiskundig worden benaderd, hier kan er gekeken worden naar hoe een boom groeit van een klein zaadje tot een hele boom. Gelukkig is dit proces niet willekeurig en zit er een patroon achter het groeien. De methode dat gebruikt wordt is L-systemen, ook wel Lindenmayer systemen genoemd. Bij een L-systeem vertrekt men van een axiom (dit kan gezien worden als het zaadje



Figuur 1.1: Voorbeeld billboard cloud bestaande uit 25 vlakken. [Fuhrmann et al., 2005]

van de boom) en een set van regels (hoe de boom groeit). Zo kan deze axiom elke iteratie groter worden totdat de uiteindelijk boom bereikt is.

De makkelijkste L-systemen zijn echter niet sterk genoeg om een boom onmiddellijk voor te stellen, deze zijn te geometrisch. Dit is te zien op Figuur: 1.2. Hier zijn er 2 regels, iedere tak word elke iteratie langer en het eindpunt splits in 2 nieuwe takken met elk een eindpunt. Er zijn echter ook meer geavanceerde L-systemen die wel een realistische boom kunnen voorstellen door gebruik te maken van kansen, meerdere regels en sub L-systemen.



(a) Iteratie 1 **(b)** Iteratie 2 **(c)** Iteratie 3

Figuur 1.2: Simpel L-systeem [Wikipedia contributors, 2022]

Om een boom te genereren met een geavanceerd L-systeem moet deze eerst ingevuld worden met eigenschappen van de boom. Deze parameters kunnen manueel opgemeten worden, maar dit is moeilijk, tijdrovend en kan ook veel fouten bevatten. Daarom is het beter om deze boom parameters automatisch te kunnen berekenen. Voor deze berekening moet dus de volledige takkenstructuur van een boom digitaal beschikbaar zijn. Eerder in deze sectie werd er aangehaald dat een puntenwolk goed de vorm van een boom kan opnemen, maar met een puntenwolk kunnen er moeilijk parameters berekend worden. Daarom moet deze wolk eerst nog omgezet worden naar een bruikbaar formaat, als een groep punten samengenomen wordt kunnen deze benaderd worden met een cilinder dat zo goed mogelijk deze punten op de rand heeft liggen. Zo kan heel de boom dan omgezet worden naar een verzameling van cilinders en is de boom niet meer een hoop losse punten, maar een volledig volume. Met deze cilinders is het dan veel makkelijker om parameters te berekenen en met deze parameters kan dan het L-systeem gevuld worden om zo variaties te genereren.

1.2 Onderzoeksvraag

In deze thesis wordt er gekeken als er genoeg parameters uit een puntenwolk berekend kunnen worden, om zo variaties van de boom te genereren. Deze vraag kan opgesplitst worden in enkele subdelen:

- **Hoe kan een puntenwolk omgezet worden naar een bruikbare datastructuur?**
- **Welke parameters kunnen uit een digitale boom gehaald worden?**
- **Kunnen variaties gemaakt worden door deze parameters?**

1.3 Use cases

Een systeem om bomen te parametriseren kan gebruikt worden voor verschillende doel einde. Ten eerste is zulk systeem handig voor boswachters of bomen liefhebbers. Zij kunnen dan bomen inscannen en mogelijks classificeren op basis van een bepaalde parameter of de bomen te groeperen via soort. Hiermee kunnen ze inschatten wat het gedrag van de boom zal zijn als er een nieuwe geplant wordt. De digitale boom kan ook gebruikt worden om eventuele simulaties op uit te voeren zoals waar het best gesnoeid word of wat het effect zou zijn als er een zware storm komt.

Een voor de hand liggende use case is natuurlijk het gebruik van het systeem voor de game-industrie. Zo kan men makkelijk een realistische boom van de echte wereld in een virtuele wereld krijgen voor betere inleving, door de oneindige variaties hoeft geen enkele boom hetzelfde te zijn. De boom kan dan rechtstreeks in de virtuele wereld worden gezet of kan gebruikt worden als startpunt voor een designer. Dit is natuurlijk veel goedkoper dan een designer te huren om alle bomen manueel te maken. Omdat er variaties gemaakt kunnen worden aan de hand van parameters kan de nodige hoeveelheid opslag van een spel verminderd worden door alleen deze parameters op te slaan en niet de volledige boom meshes.

Niet alleen voor spellen zijn variaties handig, maar ook in het machine learning domein. Hier is er vaak veel echte of synthetische data nodig. Om deze synthetische data te kunnen maken kunnen er willekeurige parameters gekozen worden, maar dat is vaak niet interessant. Met de methode van deze thesis wordt er vertrokken van 1 of meerdere bomen om hiervan bijna oneindige variaties te maken die dan gebruikt kunnen worden om machine learning op te laten trainen.

Generatie van vele bomen kan ook bruikbaar zijn om bv. appel- en peer- boomgaarden makkelijk te digitaliseren. Door enkele bomen in te scannen kan er makkelijk honderden variaties gegenereerd worden om zo de hele boomgaard te visualiseren. Deze kan dan ook weer gebruikt worden voor simulaties op uit te voeren. Of voor eventuele machine learning doeleinden.

Hoofdstuk 2

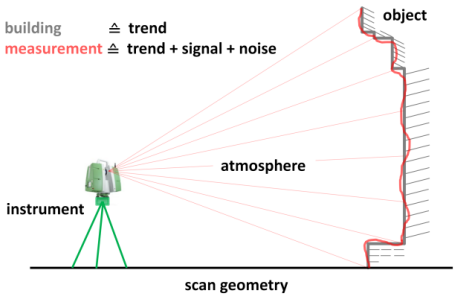
Gerelateerd werk

In dit hoofdstuk komt een samenvatting van reeds bestaande en relevante werken.

2.1 Puntenwolk

Puntenwolken zijn een belangrijk aspect van deze thesis, daarom is het ook belangrijk dat deze zo goed mogelijk is. Zo zijn er verschillende manieren om tot een puntenwolk te bekomen, in volgende paragrafen licht ik de belangrijkste technieken even toe.

De eerste techniek is **terrestrial laser scanning** [Vosselman and Maas, 2010] (Figuur: 2.1) ook wel TLS genoemd. Dit is de meest industrieel gebruikte manier van werken door de hoge punt dichtheid, lage error en groot gebied van werking. Deze scanners sturen onzichtbaar licht uit en meten hoelang het duurt voordat het licht terug is. Zo weet men de afstand tussen dat punt en de scanner. Industriële scanners staan meestal op een driepikkel en kunnen zo 360 graden horizontaal en 310 graden verticaal waarnemen [Chris, 2020]. Er kunnen ook meerdere scans samengevoegd worden tot 1 grote puntenwolk, zo kan er een scan opgebouwd worden dat volledig rond een boom gaat.



Figuur 2.1: Terrestial Laser Scanning[1]

Een tweede techniek is "structured light". Hier worden meerdere patronen geprojecteerd op het object dat gescand moet worden, in dit geval een boom. Dan worden er meerdere camera's gebruikt om tegelijk een foto te maken. Met de foto's van elke camera te vergelijken kan er dan een puntenwolk worden berekend [Chang, 2003, Pribanić et al., 2010]. Omdat deze techniek een projector en minstens 2 camera's nodig heeft, is dit niet zo gebruiksvriendelijk. Ook door de extra tijd dat nodig is om elk patroon te projecteren kan het zijn dat de takken van de boom wat zijn bewogen, dit kan er voor zorgen dat de uiteindelijke puntenwolk veel errors bevat of dat er veel detail verloren gaat. Omdat de projector en camera een bepaalde resolutie hebben en een maximale kijkhoek kan het zijn dat een grote boom meerdere keren gescand moet worden. Deze techniek is dus niet zo ideaal om makkelijk 1 of meerdere bomen in te scannen.

[1] https://www.gib.uni-bonn.de/research/correlations-at-terrestrial-laser-scanning/correlations-at-terrestrial-laser-scanning

De derde techniek is "photogrammetry" [Wang, 1990]. Dit is de goedkoopste en toegankelijkste manier van 3D scannen, het enige wat nodig is is 1 camera/smartphone. Bij deze techniek moeten er vele foto's genomen worden van het te scannen object vanuit telkens een andere positie. Als er genoeg overlap is tussen de foto's kan er een puntenwolk berekend worden. Dit wordt gedaan door op elke afbeelding eerst kenmerken te detecteren, een bekende manier is SIFT (Scale-invariant feature transform) [Sun et al., 2014]. Hiermee worden punten op een afbeelding gevonden en geclassificeerd zodat deze punten ook op andere foto's gevonden kunnen worden. Dit is echter niet de enige manier, zo zijn er meerdere die hetzelfde doel hebben met een iets andere werking, bv. SURF, KAZE, AKAZE, ORB en BRISK [Tareen and Saleem, 2018]. Eens alle kenmerken uit de foto's zijn gehaald kunnen deze met elkaar vergeleken worden om zo de 3D posities van de camera's te krijgen. Met alle gevonden camera posities kan een puntenwolk gemaakt worden. Hier kan makkelijk de resolutie en accuraatheid van de puntenwolk verhoogd worden door meer foto's te nemen. Er is ook bijna geen limiet voor de grootte van het object. Het nadeel is wel dat er veel foto's moeten worden genomen om een goede puntenwolk te kunnen krijgen, dit betekend ook dat het lang duurt voordat de berekeningen van de puntenwolk klaar zijn. Het resultaat van photgrammetry is vaak ook niet op de juiste schaal en zal manueel nog gecorrigeerd moeten worden.

Bij het scannen van planten kunnen er nog extra moeilijkheden voorkomen, de paper van Aguilar er al. [Aguilar et al., 2008] geeft hier een mooi voorbeeld van. Hier proberen ze met photogrammetry het omhullend volume van een tomaten plant te krijgen om te berekenen hoeveel deze plant besproeid moet worden. Hiervoor gebruiken ze een plastieken net om over de plant te leggen. Maar omdat alles groen is van de plant en van het net moeten ze extra bolletjes om het net hangen in een andere kleur om zo een betere matching te krijgen. In de paper van Andujar et al. [Andujar et al., 2018] gaan ze kijken of het mogelijk is om aan de hand van photogrammetry kleine onkruid plantjes te kunnen scannen. Hier wordt aangetoond dat een goede scan gereconstrueerd kan worden, maar er zijn nog altijd wel wat errors zo wordt vaak de stengels niet goed opgebouwd. Maar dit komt omdat deze stengels vaak dun zijn, om deze op te kunnen nemen moet er van dichterbij foto's genomen worden. Volgens Martinez-Guanter et al. [Martinez-Guanter et al., 2019] is photogrammetry wel 1 van de beter manieren om gedetailleerd een scan te maken van planten. In zijn paper wordt photogrammetry vergeleken met een dieptecamera (specifiek de Kinect v2). Hier zien ze dat er bij beide manieren nog steeds een error is. Bij de Kinect manier is de structuur beter, maar bij de photogrammetry is er meer detail. Zij concluderen dat er een afweging gemaakt moet worden en er per object beslist moet worden welke techniek toegepast moet worden.

Bij de vorige voorbeelden werd er gekeken naar kleine objecten, maar dit is mogelijks niet hetzelfde voor grotere objecten. Op grote schaal heeft Baltsavias E. [Baltsavias, 1999] een vergelijking gemaakt tussen actieve laser scanning(ALS) en photogrammetry. In deze vergelijking worden beide technieken vanuit de lucht uitgevoerd. Ookal is dit een oude paper worden er nog steeds relevanten voor en nadelen uit gehaald. Zo wordt aangehaald dat ALS beter kan werken voor bomen omdat er lasers zijn dat door takken of bladeren kunnen meten, zo kan de laser elke terugkomend signaal oppakken om een punt op een hoge tak en een punt op de grond terug te krijgen van 1 laserstraal uit te sturen. De laser was toen en is nog altijd veer preciezer en heeft een hoge resolutie vanop een langere afstand. Een ALS laser is ook veel sneller omdat deze weinig berekeningen moet doen.

2.2 Puntenwolk segmentatie

Zoals vermeld in de vorige sectie kan het zijn dat een puntenwolk zeer groot is en een heel gebied kan omvatten. Dit is natuurlijk niet ideaal om een specifieke boom te parametriseren. Daardoor zijn er al een paar manieren ontwikkeld die proberen om een puntenwolk te segmenteren per individuele boom. Dit is ook nuttig als voorafgaande stap voor de toepassing van deze thesis die in Hoofdstuk: 3 aan bod zal komen. Hier wordt er verwacht dat de puntenwolk alleen bestaat uit een individuele boom. De volgende papers maken gebruik van TLS om hun puntenwolk te bekomen.

In de paper van [Burt et al., 2018] willen zij een pointcloud van een heel bos of gebied van dicht bebouwde bomen splitsen per boom. Hun software genaamd 'treeseg' werkt bijna volledig automatisch. Eerst wordt de pointcloud gegroepeerd door euclidische clustering, hierdoor kan er al veel van een specifieke boom afgezonderd worden. In elke cluster wordt er dan gezocht naar de hoofdstam van de boom, als deze gevonden is wordt de grond en andere losstaande vegetatie verwijderd. Met de overblijvende stam worden de takken en kruin gezocht, hier moet de gebruiker nog een afstand instellen om zo heel de kruin te omvatten. Als resultaat is er een aparte pointcloud voor elke boom.

Een andere manier is uitgelijnd in de paper van [Liu et al., 2021]. Hier wordt een nieuwe methode ontwikkeld wat ze de 'Trunk Growth' methode noemen. Hier wordt ook als eerste stap de hoofdstammen gedetecteerd. Deze worden dan als startpunt gebruikt samen met de normaal vector om de volledige hoofdstam te vinden. Hieruit 'groeien' dan de takken en als laatste worden de bladeren nog erbij genomen door te zoeken naar de dichtst aanliggende punten.

2.3 Boom model opbouwen

Een belangrijk aspect van deze thesis is om berekening uit te voeren op een 3D boom model en om dit te doen moet er natuurlijk eerst een model zijn. Een model van een boom opbouwen kan echter op verschillende manieren, in de volgende sectie worden enkele manieren aangehaald.

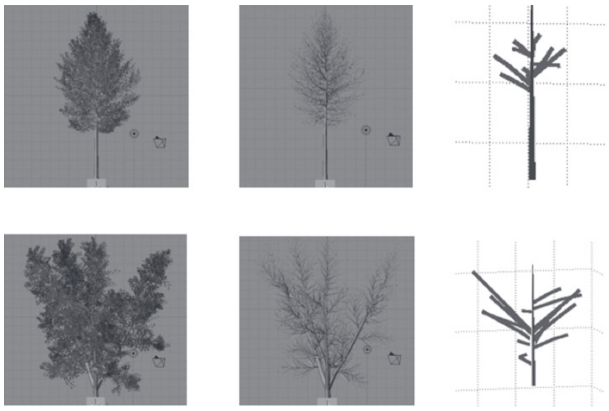
2.3.1 Single Image Tree Modeling

De eerste manier is van [Tan et al., 2008]. Hier bouwen ze een 3D model van een boom op op basis van 1 foto en wat gebruiker input. Zo moet de gebruiker op de afbeeldingen de kruin van de boom omcirkelen, de hoofdstam aanduiden met een lijn en eventueel nog enkele extra zichtbare takken aanduiden. Dan wordt nog automatisch verdere zichtbare takken gezocht en aangeduid. Deze aanduidingen worden dan gebruikt als patroon voor de takken van de rest van de boom. Als er niet genoeg zichtbare takken zijn dan wordt er een voorgedefinieerd patroon gebruikt. Daarna wordt het skelet van de boom gegenereerd door een groei algoritme te gebruiken op het patroon. Hier wordt de boom niet omgezet in 3D, maar groeit de boom door de afbeelding als een gids tot hoe ver deze mag groeien. Als dit skelet dan klaar is kunnen er bladeren gemaakt worden door de textuur uit de afbeelding te halen. Uit hun resultaten kan vastgesteld worden dat de gereconstrueerde bomen heel goed de look en feel kunnen overbrengen, zoals te zien is op Figuur: 2.2, maar als er meer in detail gekeken wordt, zijn er nog verbeteringen mogelijk. Zo hebben de 3D modellen in de diepte een plattere kruin en zijn de takken volledig anders dan de realiteit. Deze manier kan dus niet gebruikt worden om relevante parameters uit te berekenen.

Figuur 2.2: (a) Input Afbeelding, (b) Door gebruiker aangeduide kruin en stam, (c) Skelet structuur, (d) Finaal resultaat. Bron: [Tan et al., 2008]

2.3.2 3D Tree Reconstruction from Simulated Small Footprint Waveform Lidar

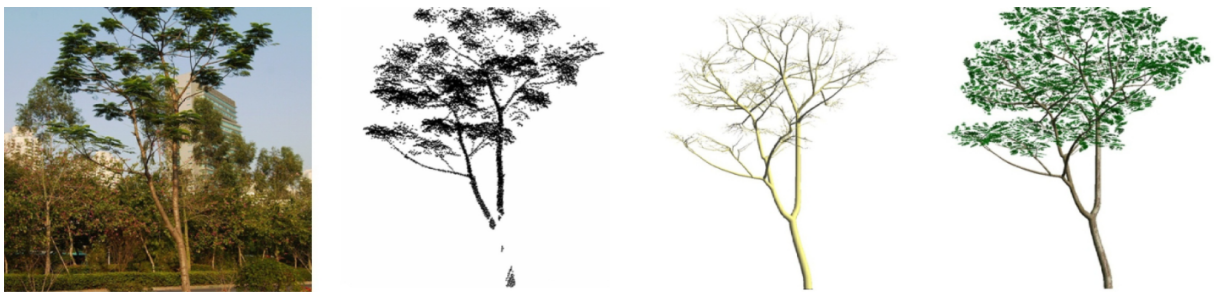
Een andere manier is die van [Wu et al., 2013]. In deze paper gebruiken ze golfvorm Lidar, deze Lidar kan door objecten gaan en kan zo meerdere dieptes per laser straal terug krijgen, zoals ook aangehaald in de paper van [Baltsavias, 1999]. Om hun werking te evalueren zijn ze vertrokken van een reeds bekend boom model. Dit model krijgt materialen toegekend aan de takken en bladeren om dan een gesimuleerde Lidar scan op te sturen. Hierdoor krijgen ze per laser straal pieken te zien op de plekken waar meerdere fotonen terug komen. Door een goede threshold te kiezen kunnen er meerdere correcte punten gevonden worden. Met de puntenwolk word er dan een dichtheid gebaseerde clustering algoritme toegepast om punten van een tak te groeperen. Dan wordt de stam gemapt door er vanuit te gaan dat deze kegelvormig is met het midden van de puntenwolk lagen als positie. Voor de takken wordt er vanuit gegaan dat 1 cluster gemodelleerd kan worden door 1 cilinder. Dan worden deze takken nog verbonden met elkaar en de stam. Zo bekomen ze het uiteindelijke 3D model. Op Figuur: 2.3 staat het resultaat van hun methode. Het is hier duidelijk dat de reconstructie niet accuraat is of niet volledig is. Anders bekeken is dit resultaat vrij indrukwekkend voor de schaarse puntenwolk. Deze manier is dus ook niet geschikt om correcte parameters mee te berekenen.



Figuur 2.3: Links: input boom, Midden: input zonder bladeren, Rechts: gereconstrueerd model. Bron: [Wu et al., 2013]

2.3.3 Automatic Reconstruction of Tree Skeletal Structures from Point Clouds

Een manier dat wel goed de volledige skelet structuur van een boom kan construeren staat beschreven in de paper van [Livny et al., 2010]. Hier beginnen ze van een puntenwolk gemaakt door een laser met 1 of meerdere bomen en zet deze om naar 3D modellen. Deze manier werkt volledig automatisch en heeft geen gebruikers input nodig. Eerst gaan ze de stammen proberen te vinden in de puntenwolk van de bomen. Hier worden de punten geprojecteerd op een horizontaal vlak. Hierdoor komen verticale stammen samen te liggen en zal hier een hoge hoeveelheid punten op dezelfde plaats liggen. Zo weten ze de positie van de stam, hierna wordt dan de grond verwijderd. Vanuit dit startpunt wordt er een BSG opgesteld, een branch-structure graph. Hierdoor word een boom voorgesteld als een aantal nodes dat met elkaar verbonden zijn. Om te beginnen word er een BSG opgesteld met een zo min mogelijke afstand tussen de nodes. Dan wordt er iteratief een nieuw BSG opgesteld op basis van de vorige om zo telkens een beter resultaat te bekomen. Als dan de BSG optimaal is kan deze omgezet worden naar 3D cilinder, hier wordt er nog wat gefilterd om zo nutteloze cilinders te verwijderen. Op Figuur: 2.4 is het resultaat van hun methode te zien. Hieruit blijkt dat hun methode zeer goed werkt en zelfs nog goed kan omgaan als de scan niet compleet is (het gat in pointcloud). De methode zorgt voor accurate hoeken, lengtes en posities van takken, wat ideaal is voor verdere berekeningen. Hun methode is ook zeer efficiënt en kan een enkele boom reconstrueren in enkele seconden of een grote scan met meerdere bomen in enkele minuten. Jammer genoeg is hun methode niet publiek beschikbaar en kan dus niet gebruikt worden in deze thesis.



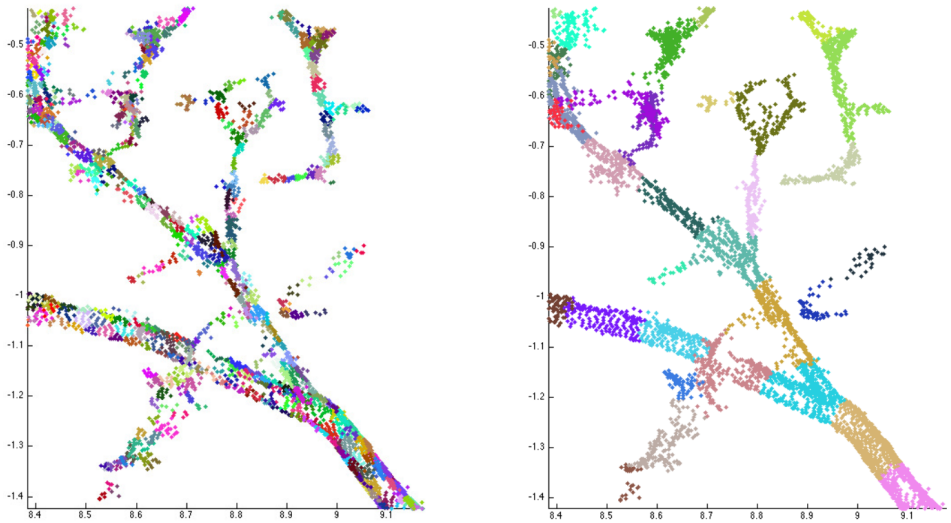
Figuur 2.4: Van links naar rechts: aflbeelding boom, puntenwolk van laser, skelet reconstructie, volledige reconstructie. Bron: [Livny et al., 2010]

2.3.4 TreeQSM

De laatste mogelijkheid dat bekeken wordt in deze thesis is de software TreeQSM (V2.4.0). Deze is opensource en is beschikbaar als een Matlab library. Deze software gemaakt door [Åkerblom, 2020, Raumonen et al., 2013] is in staat om een puntenwolk van 1 boom om te zetten naar een verzameling van cilinders. Deze cilinders volgen zo goed mogelijk de vorm van de boom. Dit zorgt er voor dat er gemakkelijker berekeningen kunnen uitgevoerd worden om verschillende data te bekomen. Hier staat QSM voor "quantitate structure model". Dit wil zeggen dat er geprobeerd word om met geometrische

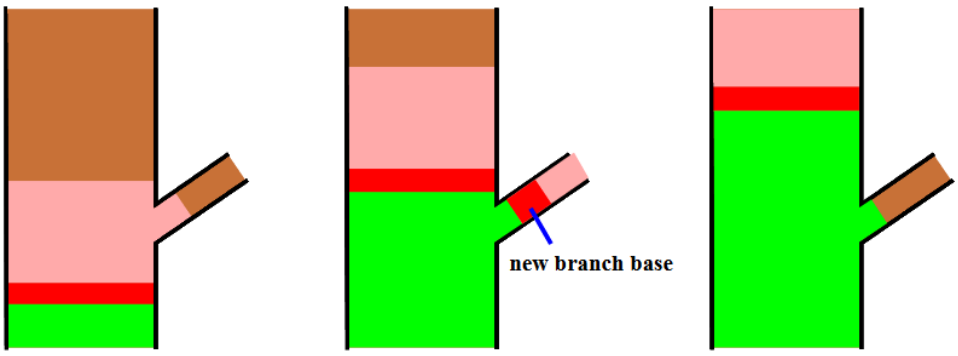
primitieven, in dit geval cilinders, de echte vorm te benaderen.

In deze software wordt eerst de puntenwolk verdeeld in kleine groepen/patches van punten. Hier moet eerst de minimumdiameter van de patches ingesteld worden, op Figuur: 2.5 wordt het effect van deze parameter getoond. Een kleinere waarde zorgt voor meer detail, maar kan ook zorgen dat er meer noise geïntroduceerd wordt en zorgt voor een slechtere reconstructie. Deze parameter moet manueel ingesteld worden en moet ook veranderen als de scan van de boom veranderd. Een te grote waarde zorgt er voor dat kleinere takken niet meer herkend worden.



Figuur 2.5: Links: Patches van 2cm (0.02), Rechts Patches van 10cm (0.1). Bron: [Raumonen et al., 2013]

Deze patches worden dan gebruikt om de puntenwolk te segmenteren om te bepalen welke punten op welke stam/aftakking liggen. Dit werkt door te beginnen bij de hoofdstam en zo in richting van de stam omhoog te gaan, deze richting wordt berekend door de normaal vectors van de patches samen te nemen, dit geeft een indicatie van de richting. Dan worden de eerste patches geclassificeerd en gaat de software verder omhoog werken per kleine slice. Als de volgende slice niet meer 1 geheel vormt, is dit een teken van een splitsing en wordt deze gezien als het startpunt van een nieuwe tak. Zo gaat deze verder tot alle punten van de boom gesegmenteerd zijn per tak.



Figuur 2.6: Segmentatie van patches, groen is het gevonden segment en rood zijn de patches dat bekeken worden. Bron: [Raumonen et al., 2013]

Na de segmentatie kunnen er cilinders gepast worden. Ze beginnen telkens bij het startpunt van de stam of tak. Hier wordt dan een bepaalde hoogte aan punten genomen voor de eerste cilinder te gaan passen. Dan worden er enkele cilinders met verschillende groottes en locaties op de puntenwolk gezet. Dan wordt er getest welke cilinder het beste past, deze is dan de beste benadering van de echte puntenwolk. Zo gaat het telkens verder totdat de volledige stam gevuld is, daarna worden alle aftakkingen op dezelfde manier verwerkt.

Omdat de aftakkingen vaak veel dunner zijn dan de stam kan het zijn dat er soms geen cilinder geplaatst kan worden. Als dit gebeurt, gaat de software verder met de volgende segmenten in de tak en zal op het einde het gemiste segment bijgewerkt worden.

Na afloop van de software is de puntenwolk volledig bedekt met cilinders, gegroepeerd per diepte van aftakking. Het resultaat van deze software word later getoond bij Hoofdstuk: 3. De software geeft niet enkel de cilinders terug, maar geeft ook al een aantal parameters terug. Deze parameters zijn vooral gefocust op het volume van de boom en de verdeling van takken. Deze zijn niet nodig voor variaties te kunnen maken.

De paper van [Lau et al., 2018] test de accuraatheid van TreeQSM en hoe goed de diameter en tak lengtes overeen komen met de realiteit. Uit hun werk kan geconcludeerd worden dat TreeQSM accuraat is voor vooral grote takken en minder voor kleinere takken. Deze testen werden wel uitgevoerd op tropische bomen met lianen, dit kan een van de redenen zijn waarom kleinere takken minder accuraat zijn.

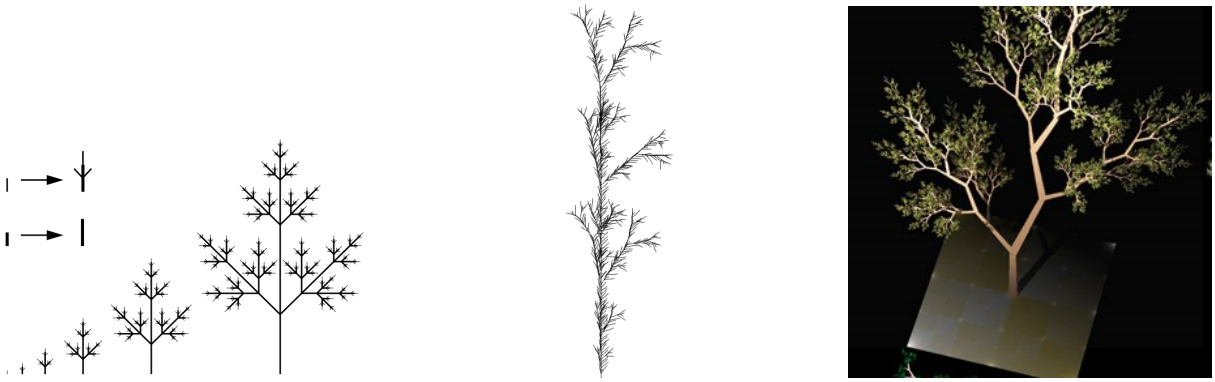
2.4 L-systeem

In een vorige sectie is aangehaald hoe men bomen kan digitaliseren door een mesh of puntenwolk. Dit is echter niet de enige manier. Een boom groeit vrij in de natuur, maar toch kunnen er patronen gevonden worden in vele bomen. Zo kan een tak van een boom lijken op de boom zelf, maar dan kleiner. Dit patroon kan beschreven worden aan de hand van een L-systeem, ook wel een Lindenmayer systeem genoemd [Prusinkiewicz et al.,]. Een L-systeem is een verzameling van regels van de vorm: $A- > AB, B- > B$. Waar A een tak kan zijn en B een blad. Met deze regels kan men verschillende geometrische vormen bekomen, maar dit is niet zo interessant voor bomen en planten. Daarom zijn er nog uitbreidingen van een L-systeem dat beter werken voor planten en bomen.

Een voorbeeld is de paper van [Samal et al., 1994]. Hier tracht de auteur om planten te classificeren aan de hand van L-systeem regels. Om natuurlijke planten te bekomen wordt er gebruik gemaakt van een stochastische L-systeem hier krijgt elke regel een kans om te gebeuren. Zo kan de plant realistisch gemodelleerd worden.

Echter stochastisch alleen is ook nog niet voldoende. In de paper van Ochoa [Ochoa, 1998] kan het L-systeem willekeurig aangepast worden door een symbool of groep symbolen te vervangen door een andere groep karakters. Hier worden vele heuristieken toegepast over hoe een plant groeit. Zo wordt er rekening gehouden met het aantal aftakkingen, minder aftakking betekend een stabielere hoofdtak. Ook wilt men dat een plant naar het licht groeit en nog enkele andere heuristieken.

Om een nog sterker L-systeem te bekomen kan men een extra dimensie toevoegen, in plaats van een 2D systeem naar een 3D systeem te gaan. Hier kan de software van [Hewitt, 2022], [Hewitt,] bij helpen. Dit is een implementatie van een 3D L-systeem gebaseerd op het werk van [Weber and Penn, 1995]. In de software van Hewitt kan een ruim assortiment van parameters aangepast worden om heel veel verschillende bomen te modelleren. Deze software wordt gebruikt in Hoofdstuk: 3 en hier zal ook veel van de instelbare parameters aan bod komen.



(a) Geometrisch **(b)** Stochastisch **(c)** 3D

Figuur 2.7: Voorbeelden verschillende L-systemen

Hoofdstuk 3

Implementatie

Eerst wordt er gekeken als er parameters kunnen berekend worden uit een 2D L-systeem met kansen en willekeurigheid. Zo wordt er getoetst als berekende parameters overeen komen met de input parameters. Als dit positief is dan begint men met een 3D L-systeem. Bij het 3D systeem wordt er eerst circulair en analytisch gewerkt. Namelijk er wordt eerst een boom gemaakt met het L-systeem, dan word deze omgezet naar een puntenwolk. Deze puntenwolk kan dan gebruikt worden voor de implementatie en zal als resultaat een aantal parameters teruggeven. Deze parameters worden dan vergeleken met de originele parameters. Hiermee kan de accuraatheid getoetst worden. Omdat dit een ideaal scenario zal er ook een empirisch deel zijn waar men begint van een echte boom en kijkt hoe goed de resultaten zijn.

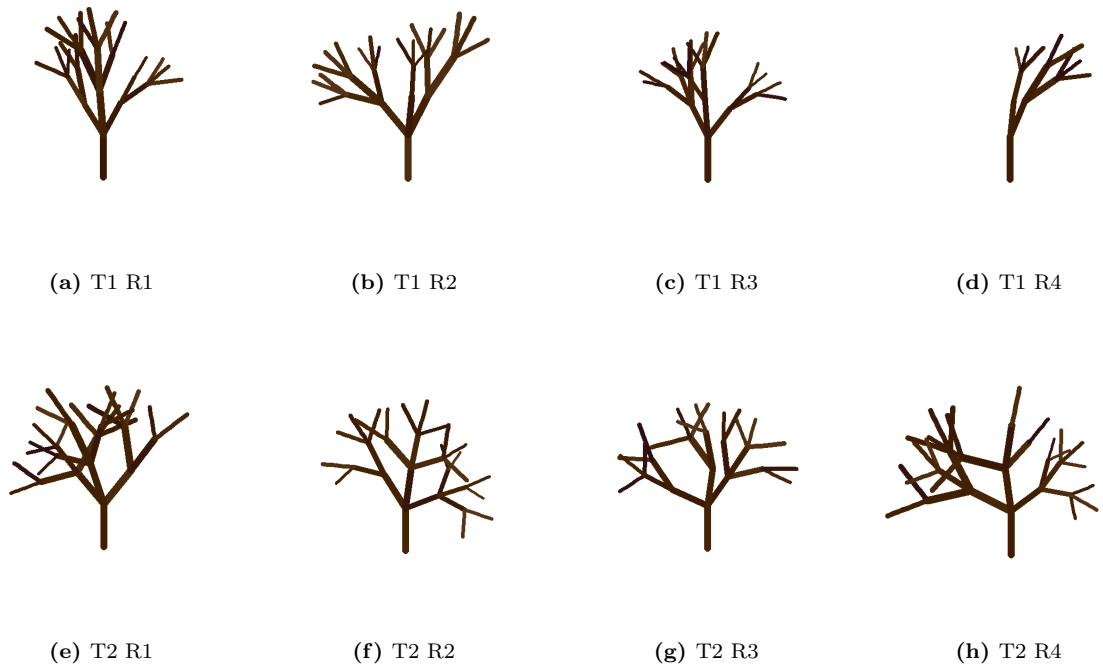
3.1 2D L-systeem

Voor 2D bomen te genereren heb ik gebruik gemaakt van "2D-Tree-Generator"[1], deze software geschreven in Python laat toe om fractalen bomen te genereren. Hier kunnen ook parameters op ingesteld worden zoals kleuren, bladeren, hoeken tussen aftakkingen, kansen van de aftakkingen, hoeveel een tak verkleint en nog enkele andere. Hier heb ik enkel gebruik gemaakt van de hoek, hoek variatie en tak verkleining. Hier heb ik 2 testen gedaan met verschillende parameters en deze ieder 5 bomen laten genereren, op Figuur: 3.1 zijn enkele getoond. De resultaten van deze berekeningen zijn te zien in Tabel: 3.1. Hier komt in sommige gevallen 1 boom niet goed overeen met de ingestelde waarde voor een goed resultaat. Maar door meerdere bomen samen te nemen, in dit geval 5, kan er wel goed gemiddelde berekend worden. Zo is de gemiddelde hoek van test 1 (30,26°) bijna exact de ingestelde waarde (30°). Hier valt wel op dat de hoek variatie niet zo correct is, omdat deze berekend is op de gemiddelde hoek per boom. Het zou beter zijn om per boom de variatie van de hoeken te berekenen. De hoek variatie geeft wel al een aanwijzing naar de ingestelde waarde, zo is de berekende variatie groter als de ingestelde waarde ook groter is.

[1] `https://github.com/TimoFlesch/2D-Tree-Generator`

Test 1	Ingestelde Waardes	Run 1	Run 2	Run 3	Run 4	Run 5	Avg alle runs	Variatie
Tak hoek	30°	29,54°	27,93°	26,6°	32,2°	35,04°	30,26°	3,39°
Tak hoek variatie	10°							
Tak verkleining	6	4,11	10,19	7,9	10,79	4,97	7,592	
Test 2								
Tak hoek	50°	40,23°	58,94°	51,23°	44,07°	57,2°	50,34°	8,12°
Tak hoek variatie	20°							
Tak verkleining	6	2,6	5,35	7,48	3,17	8,9	5,5	

Tabel 3.1: 2D bomen berekende parameters



(a) T1 R1 **(b)** T1 R2 **(c)** T1 R3 **(d)** T1 R4

(e) T2 R1 **(f)** T2 R2 **(g)** T2 R3 **(h)** T2 R4

Figuur 3.1: 2D bomen gegenereerd met de initiële parameters van Tabel: 3.1. T staat voor test en R voor run.

3.2 3D Bomen: Analytisch

3.2.1 Boom generatie

De eerste stap bij de analytische werkwijze is de boom te generen die gebruikt wordt in plaats van een scan. Dit wordt gedaan met het programma Blender[2] en de plugin van [Hewitt, 2022]. Deze plugin heeft een uitgebreide lijst van instelbare parameters (Figuur: 3.2). En ook enkele vooraf ingestelde waardes, bv. apple tree, Douglas fir, palm tree. Hierdoor kan de gebruiker beginnen met een bestaande boom en deze aanpassen. De plugin gebruikt ook een willekeurige of ingegeven waarde als seed om zo de willekeurigheid van de boom te beïnvloeden, zo kunnen bomen met dezelfde parameters lichtjes verschillen van elkaar om zo natuurlijke variaties te genereren.

In de plugin zijn sommige parameters belangrijker dan andere, hier volgt een opsomming van de betekenis en relevantie van de parameters. De parameters zijn opgesplitst in 2 delen, parameters voor de vorm van de boom (Figuur: 3.2a) en parameters voor het gedrag van elk niveau van takken (Figuur: 3.2b).

Boom parameters:

- **Tree Shape:** in dit menu kan men kieze wat voor vorm de kruin van de boom moet hebben. Dit kan bolvormig, halve bol, cilinder, kegel, geïnverteerde kegel en nog enkele andere vormen aannemen. Het is ook mogelijk om zelf de vorm te definiëren.
- **Level Count:** dit is het niveau van de vertakkingen. Waarde 1 betekend enkel de stam, 2 is de stam en op deze stam komen aftakkingen. Bij waarde 3 komen er aftakkingen op de takken van niveau 2. De waarde kan tot 4 ingesteld worden, maar 3 is de waarde dat men best gebruikt voor een dichte boom.
- **Prune:** alle parameters van Prune ratio, width, enz. Zijn nuttig om het effect van snoeien te simuleren. Dit is echter niet belangrijk omdat deze thesis voornamelijk gaat over natuurlijke bomen. Daarom staat de parameter Prune Ratio altijd op 0, zo wordt snoeien niet gebruikt.
- **Trunk Splits:** hoe vaak de hoofdstam splitst.

[2] `www.blender.org`

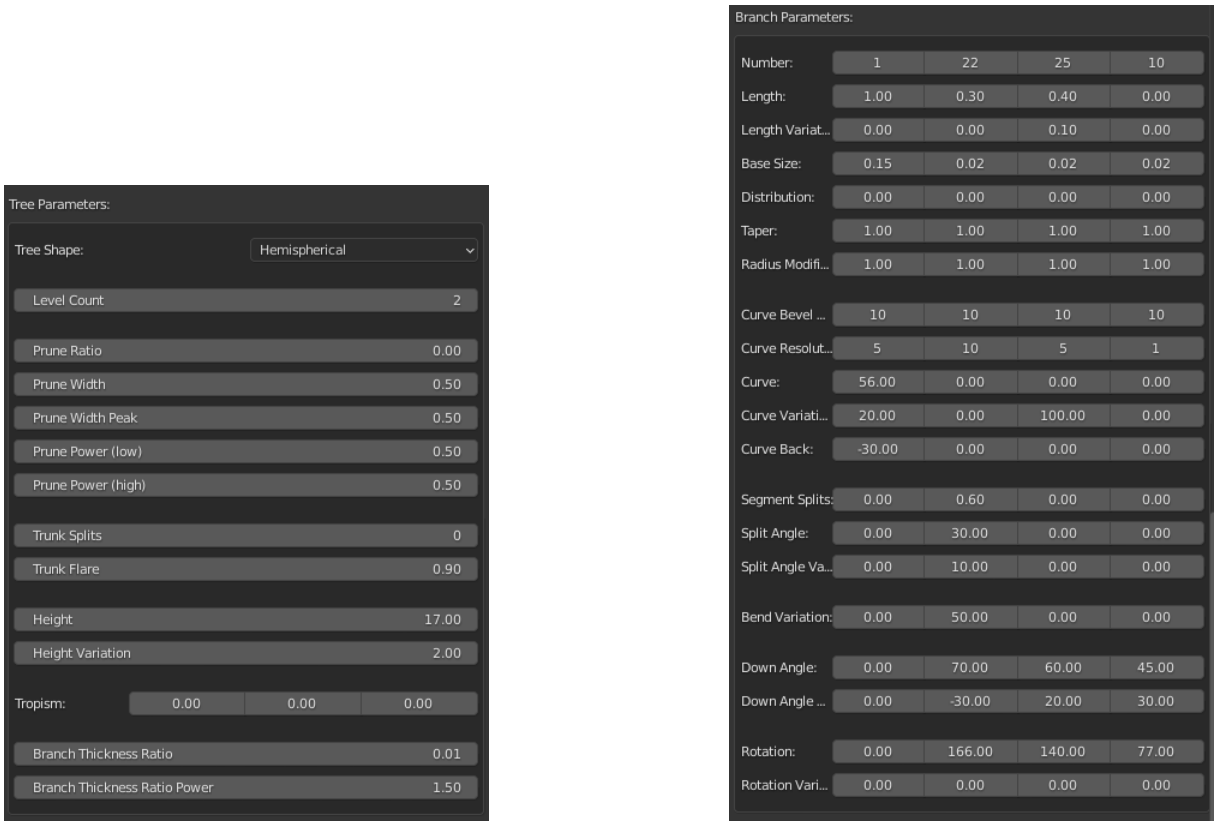
- **Trunk Flare:** hoeveel de diameter aan de voet van de boom toeneemt.
- **Height, Height Variation:** de gemiddelde hoogte van de boom en de maximale afwijking als een willekeurige seed wordt gekozen.
- **Tropism:** Dit kan gebruikt worden om effecten zoals wind en zwaartekracht te simuleren, maar word constant gehouden in deze thesis.
- **Branch Thickness Ratio:** de verhouding tussen de straal en de lengte van de takken. Ratio Power is hoe sterk de diameter verander tussen elk niveau van aftakkingen.

Tak parameters: Elke kolom stelt het niveau van aftakkingen voor, bv. als 'Level Count' 3 is dan zullen de eerste 3 kolommen gebruikt worden.

- **Number:** is het aantal aftakkingen op het vorige niveau. Zo kan men in de eerste kolom kiezen als er 1 of meerdere stammen van de grond vertrekken. De 2e kolom is dan hoeveel takken er op de stammen komen enzovoort.
- **Length:** de lengte van tak als fractie van de tak van het vorige niveau, hier kan ook variatie op ingesteld worden.
- **Base Size:** de verhouding op de lengte van de tak waar geen aftakkingen gebeuren. Zo kan ingesteld worden dat bijvoorbeeld de takken pas boven de helft van de stam beginnen.
- **Distribution:** de verdeling van de takken. Deze is nuttig als de takken samen gegroepeerd moeten worden in bepaalde knooppunten. Dit is een randgeval en zal verder niet gebruikt worden.
- **Taper:** beïnvloed hoe een tak van grote diameter naar een punt gaat, deze is niet belangrijk en zal in het verder verloop op 1 staan. Dit wil zeggen de diameter wordt lineair kleiner tot een punt.
- **Curve Bevel Resolution:** beïnvloed de resolutie van het uiteindelijk model, maar veranderd niets aan de structuur van de boom. Deze waarde wordt niet gebruikt.
- **Curve Resolution:** deze waarde bepaald uit hoeveel segmenten een tak bestaat.
- **Curve:** curve is de hoek dat de tak veranderd van start tot eind. Dit kan ook gevarieerd worden. Curve Back is hoeveel de tak terug buigt.
- **Segment Splits:** dit is het aantal splitsingen per segment in de tak. Zo kan men kieze als men 1, 2, ... splitsingen wil hebben, maar dit gebeurt niet vaak. Een belangrijker aspect is waardes tussen de 0 en 1. Dit zorgt dat de splitsingen willekeurig verdeeld worden over de segmenten. Deze waarde samen met de "Curve resolution"kan gebruikt worden om de kans van splitsen te benaderen. Met andere woorden een laag aantal segmenten met een hoge splits waarde bv 0.9 is equivalent aan een hoog aantal segmenten met een lagere splits waarde.
- **Split Angle:** de hoek tussen dichotome takken. Dit zijn takken dat op een splitsing alle 2 naar buiten toe gaan en niet dat de hoofdtak rechtdoor gaat en de andere een hoek maakt. Hier kan ook variatie op toegepast worden.
- **Bend Variation:** hiermee kunnen de takken wat gedraaid worden, maar is niet zo belangrijk.
- **Down Angle:** dit is de hoek dat de aftakking maakt op de tak van het hogere niveau. Hier is ook variatie mogelijk.
- **Rotation:** dit is de hoek dat een aftakking roteert rond de tak van het hogere niveau.

Om nu een boom te generen kan men de instelleningen van een appel boom laden en deze aanpassen. Om een simpel resultaat te krijgen kan de 'Level Count' op 2 worden gezet. Ook kan er gekozen worden om geen bladeren te maken. Zo is het model zeer proper en is er minder filtering nodig voor de verdere stappen. Dan kan men nog kiezen voor een eigen seed in te geven of niet. Als resultaat krijgt men een realistisch model van een boom, te zien in Figuur: 3.3.

Uit Blender kan de boom geëxporteerd worden naar een .obj bestand voor verder gebruik. Hier moet men wel opletten dat de Z-waarde de hoogte is.



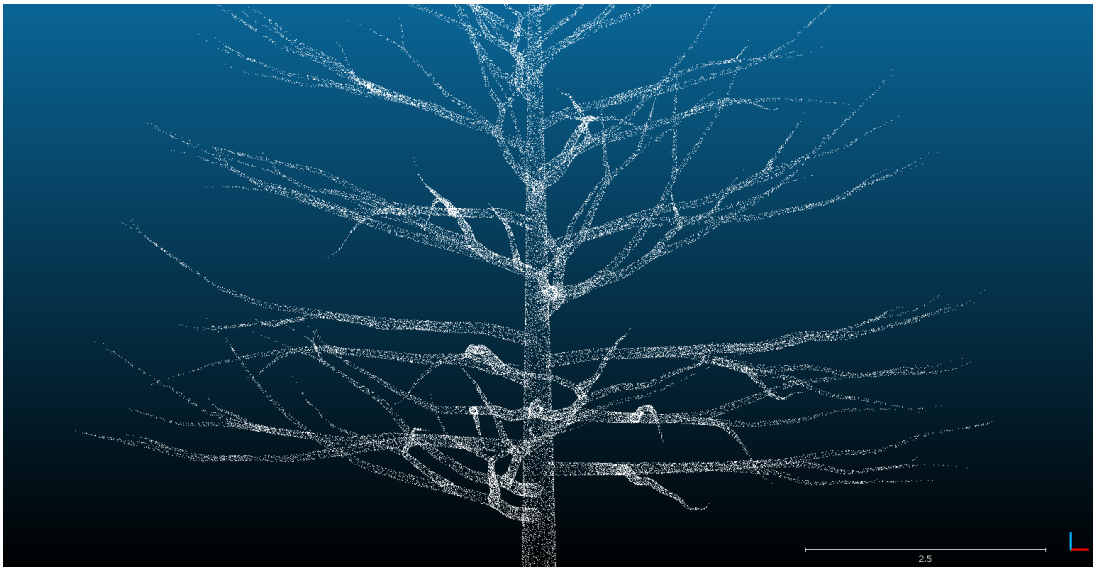
(a) Boom parameters **(b)** Tak parameters

Figuur 3.2: Instelbare parameters in Treegen in Blender

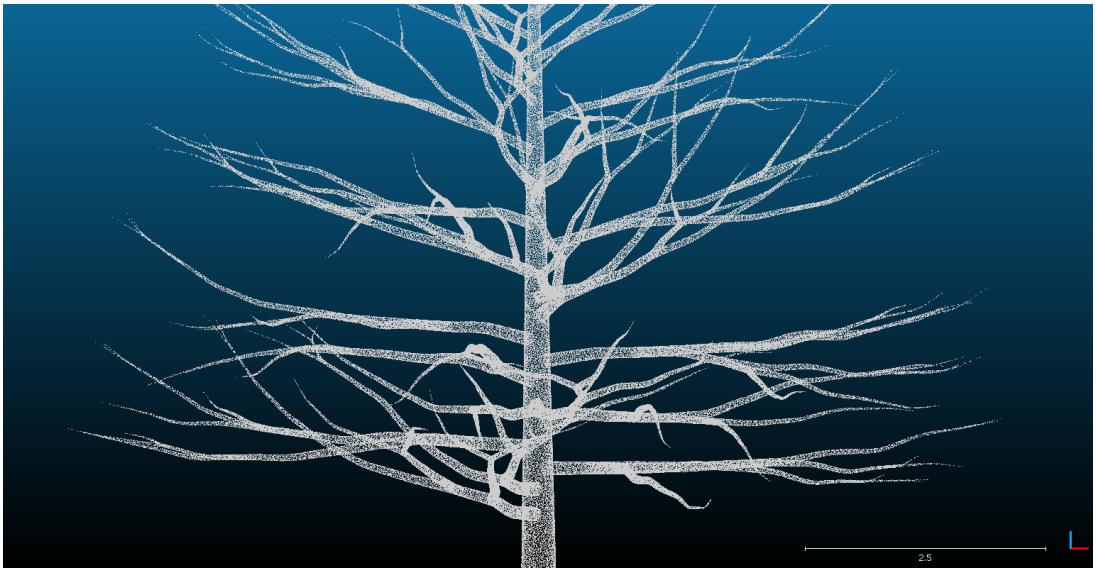
Figuur 3.3: Appel boom gegeneerd in Blender

3.2.2 Omzetten naar puntenwolk

Eens ik het boom model heb in .obj of een ander formaat, kan deze omgezet worden naar een puntenwolk. Dit doe ik door gebruik te maken van CloudCompare(CC) [3]. Deze open source software heeft veel functies om met puntenwolken overweg te kunnen. Zo kan men verschillende puntenwolken samenvoegen, filteren en vele andere. Deze zijn echter niet nodig. CC kan ook de mesh omzetten naar een puntenwolk op 2 manieren. Er kan een x aantal punten gekozen worden die verdeeld worden over de mesh of er kan een dichtheid doorgegeven worden als aantal punten per vierkante meter. Hoe hoger de dichtheid hoe beter het resultaat. In Figuur: 3.4 kan men het verschil zien van een dichtheid van 1000 en 6000. Hier is 6000 een goede middenweg tussen dichtheid en snelheid van berekeningen. Deze resultaten moeten dan worden opgeslagen als text bestand met op elke lijn de x, y, z coördinaat van een punt.



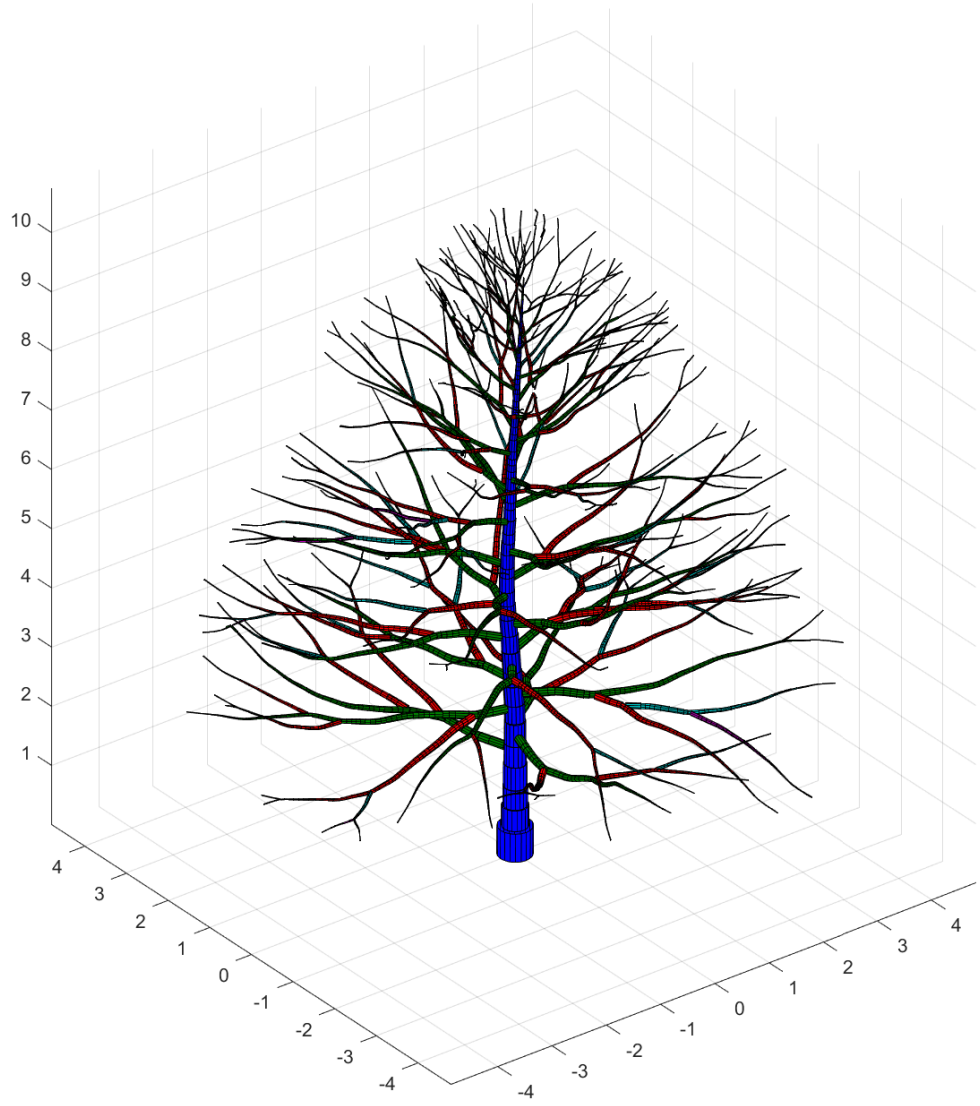
(a) Dichtheid 1000



(b) Dichtheid 6000

Figuur 3.4: Puntenwolk gemaakt door CloudCompare met verschillende dichtheid

[3] `https://www.danielgm.net/cc/`



Figuur 3.5: Appel boom opgebouwd door cilinders van TreeQSM

3.2.3 Tree QSM

Nu dat ik een puntenwolk heb in goed formaat kan ik verder met TreeQSM. Deze software is geschreven in Matlab[4]. Met TreeQSM kan ik de functie treeqsm(P, inputs) op roepen met P de puntenwolk en inputs als input parameters voor de qsm. Inputs word gemaakt door de create_inputs() functie, hier kan men aanpassen wat er berekend en gevisualiseerd moet worden. Maar ook enkele parameters die de werking sterk kunnen beïnvloeden, deze moeten meestal ook aangepast worden om tot een goed resultaat te komen. Deze zijn **PatchDiam1:** de diameter voor de eerste groepering/patches in meter, **PatchDiam2Min en PatchDiam2Max:** de minimum en maximum diameter van de 2e iteratie van patches in meter. Na verschillende waardes uit te proberen blijken de waardes 0.09m 0.02m 0.07m in volgorde. Omdat de input puntenwolk bijna perfect is is er geen nood aan extra filtering. Als we dan de puntenwolk van vorige sectie in TreeQSM gebruiken krijgen we Figuur: 3.5 als resultaat. Deze cilinders staan heel dicht bij het werkelijk resultaat. Hierop kan men ook de tak en aftakking relatie zien. Zo is de hoofdstam blauw, de eerste aftakkingen groen, de tweede aftakkingen rood, en zo verder.

3.2.4 Parameter extractie

De QSM software geeft een cilinder model terug en ook data over welke cilinder bij welke tak hoort en nog andere representaties, bv. de takken hiërarchisch gesorteerd. Ook geeft de software data over de boom terug zoals het totale volume, oppervlakte en nog enkele andere grafieken. Deze zijn niet bruikbaar

[4] `https://nl.mathworks.com/products/matlab.html`

in het L-systeem, daarom moet ik enkele eigen parameters gaan berekenen. Om de data in Matlab is ga ik de berekeningen ook in Matlab uitvoeren, dit is echter niet vereist.

De TreeQSM software werkt wel niet hetzelfde als het L-systeem. In de vorige sectie werd uitgelegd dat het L-systeem een 'Level Count' parameter verwacht. TreeQSM werkt echter anders, deze software ziet een splitsing in de tak als een aftakking. Dit betekend dat de diepte van aftakking in TreeQSM altijd hoger is dan de 'Level Count' van het L-systeem. Maar omdat 'Level Count' met 3 of hoger toch kleine takken genereerd. Ga ik er vanuit dat deze kleine takken toch niet gescand worden of wegvallen met filtering. Daardoor kan de 'Level Count' op 2 worden gezet, dit wil zeggen 1 niveau van aftakkingen.

De eerste parameter is de **Tree Height**, dit is de enige parameter dat gegeven word door TreeQSM en kan makkelijk opgevraagd worden. Omdat er maar 1 boom als input is, is het dus niet mogelijk om hiermee de variatie te berekenen. Als er meerdere bomen worden gescand kan de variatie wel benaderd worden.

De volgende parameter is **Branch Thickness**, deze kan makkelijk worden berekend door de straal van de eerste cilinder te delen door de hoogte.

Na dit worden de **Down Angle** en **Split Angle** berekend. Door de werking van QSM kunnen we alle hoeken vinden tussen de eerste aftakkingen (groene takken op Figuur: 3.5) en de hoofdstam, dit zijn alle "Down Angles". Voor de rest van de takken is het moeilijk om te weten wat een aftakking is en wat een splitsing is door de mismatch tussen TreeQSM en het L-systeem, Er vanuit gaande dat de boom grotendeels bestaat uit grotere takken van het 2e niveau kunnen de splitsingen van de rest van de takken samengenomen worden, bv. de hoek tussen de groene en rode takken, rode en lichtblauwe takken (Figuur: 3.5. Met deze 2 lijsten van hoeken kunnen we nu het gemiddelde of de mediaan van nemen. Uit testing blijkt het gemiddelde hier beter te werken. Omdat dit ook een lijst is kunnen we de standaard afwijking berekenen, hiermee weten we ook de variatie. Dit werkt enkel goed als er voldoende takken zijn. Een boom met 4 takken zal niet goed werken. Nu hebben we de **Down Angle** en **Down Angle Variation**. Omdat het L-systeem dichotome splitsingen gebruikt en TreeQSM niet, moeten de splits hoeken nog gedeeld worden door 2, zo krijgen we ook de **Split Angle** en **Split Angle Variation**.

De volgende parameter is de **Rotation**, er van uit gaande dat de boom enkel bestaat uit grote takken en dus de 'level count' als waarde 2 heeft, moet de rotatie enkel van de eerste aftakkingen ten opzichte van de hoofdstam worden berekend. Eerst wordt van alle eerste aftakkingen de horizontale hoek berekend door gebruik te maken van de boogtangens. Dan wordt er van beneden naar boven de horizontale hoek berekend tussen 2 opvolgende takken. Met deze lijst van hoeken kan het gemiddelde of mediaan genomen worden. Na testen blijkt de mediaan hier een beter resultaat te geven. Hierdoor hebben we nu 1 parameter, namelijk de hoek t.o.v. de vorige tak rondom de stam, specifiek de **Rotation** van de 2e kolom in het L-systeem. Deze parameter heeft geen variatie.

Om de **Base Size** van de eerste kolom (hoofdstam) te berekenen, wordt de lengte van de grond tot de eerste aftakking gedeeld door de totale lengte.

De **Segment Splits** kunnen niet rechtstreeks berekend worden omdat dit af hangt van de **Curve Resolution**. Maar deze 2 zijn omgekeerd evenredig. Daarom kunnen we de resolutie constant houden, een goede waarde hiervoor is 10. Nu moet voor elke eerste aftakking het aantal splitsingen gevonden worden. Hier kan men dan ook de mediaan van nemen en deze delen door ingestelde resolutie. Zo bekomen we dan de **Segment Splits**.

Een volgende parameter is **Length** van de 2e kolom in Figuur: 3.2b, de fractie lengte van de eerste aftakkingen ten opzichte van de stam. Hier wordt het gemiddelde genomen van alle lengtes van de eerste aftakkingen, behalve de 5 kleinste waardes, dit zijn vaak errors. Dan wordt dit gemiddelde gedeeld door de lengte van de stam.

Om de vorm te bepalen van de boom (**Tree Shape** parameter), wordt eerst een verticaal profiel opgesteld. De gemiddelde straal per hoogte laag. Hier word dan de stam zonder takken weggegooid. Wat overblijft kan geplot worden op een grafiek, hierop is dan 1 helft van het silhouet van de kruin te zien. Dan word deze data gemapt naar interval 0-1, zo hebben we de vorm van de boom. Dan word een vorm profiel opgesteld voor halve bol, bol, cilinder en kegel. Dan word de vorm van de boom vergeleken met elk profiel. Het profiel met de minste error wordt genomen als uiteindelijke vorm.

Voor de **Curve, Curve Variation en Back Curve** van de hoofdstam (kolom 1) te berekenen word eerst de hoek van alle cilinders t.o.v. de grond berekend. Dan word hier het maximum van genomen, zo

weet men de Curve. Om de Back Curve te weten wordt de hoek van de hoogste stam cilinder min de Curve gedaan. Omdat er maar 1 stam kan zijn is het niet mogelijk om de variatie te benaderen.

De berekeningen voor **Curve, Curve Variation en Back Curve** van de aftakkingen (kolom 2) zijn wat ingewikkelder. Eerst word voor elke tak apart de verticale hoek berekend (hoek t.o.v. de grond). Dan wordt elke tak uitgemiddeld door het gemiddelde van 4 cilinders te nemen, om zo beter om te kunnen met errors. Daarna wordt elke tak doorlopen van begin tot einde, hier wordt er gekeken als er een wissel van richting plaatsvind. Op dit punt is dan de hoek de Curve, en het verschil met het eind punt is dan de Back Curve. Dan worden de uitschieters verwijderd. Hierna wordt de mediaan genomen voor de **Curve** en **Curve Back**. Voor **Curve Variation** wordt de standaard afwijking genomen.

Nu dat de parameters berekend zijn moeten deze nog opnieuw ingevuld worden in het L-systeem en kunnen er variaties gegeneerd worden. Voor enkele parameters is wel nog nodig om meerdere bomen te scannen zodat bijvoorbeeld de hoogte gevarieerd kan worden. In Hoofdstuk: 4 zijn de resultaten zichtbaar van deze analytische methode.

3.3 Echte boom

Vertrekken van een 3D-model is natuurlijk niet nuttig. Daarom is het beter om de methode toe te passen op echte bomen. Het eerste wat men moet doen is om een puntenwolk van een boom te bemachtigen. Zoals aangehaald in Hoofdstuk: 2 zijn er verschillende manieren. Een TLS laser is niet ter beschikking en structured light is ook niet gebruiksvriendelijk, dat laat nog 1 optie over photogrammetry. Dit kan makkelijk en vooral economisch uitgevoerd worden. Om photogrammetry te doen is er veel software beschikbaar. Als eerste heb ik 3DF Zephyr [5] geprobeerd. De gratis versie werkt goed voor een 40-tal foto's, jammergenoeg kan de gratis versie geen puntenwolk exporteren. Dan heb ik Meshroom[6] van AliceVision geprobeerd. Dit is open-source software en hier kan men wel de puntenwolk exporteren. Hier zijn wel meerdere foto's nodig om tot een goed resultaat te bekomen.

Als eerste test heb ik geprobeerd een perenboom te scannen in een boomgaard. Hier heb ik 50 foto's genomen en deze met Meshroom verwerkt tot een puntenwolk. Het resultaat is te zien op Figuur: 3.6. Deze leidt tot een mooie puntenwolk, het resultaat van deze puntenwolk is te zien in Hoofdstuk: 4, maar eerst moeten de inputs van TreeQSM nog aangepast worden voor een betere reconstructie.

Omdat een perenboom niet representatief is van een volwaardige boom heb ik nog een andere soort geprobeerd te scannen. Dit is een kleine, nieuw geplante boom in een park. Deze is klein en heeft een ideale takken structuur. Voor deze heb ik 71 foto's genomen. Deze boom is te klein om een goed resultaat te bekomen. Zelfs door kwaliteitsinstellingen te veranderen in meshroom is het niet mogelijk om een puntenwolk te berekenen. Dit resultaat kan men zien op Figuur: 3.7. Deze puntenwolk mislukt waarschijnlijk omdat de takken veel te dun zijn en er teveel textuur in de achtergrond zit waardoor de weinige punten van de boom vergeten worden.

[5] `https://www.3dflow.net/3df-zephyr-photogrammetry-software/`
[6] `https://alicevision.org`

(a) Foto perenboom

(b) Puntenwolk perenboom

Figuur 3.6: Photogrammetry op perenboom

(a) Foto kleine boom

(b) Puntenwolk kleine boom

Figuur 3.7: Photogrammetry op kleine boom

Hoofdstuk 4

Resultaten

4.1 Analytisch

Voor de analytische test begin ik met het invullen van parameters in het L-systeem. Hiermee worden enkele bomen gegenereerd met een verschillende seed. Deze worden dan omgezet naar puntenwolken en worden dan gebruikt als input voor de TreeQSM software. Met de QSM's worden dan de parameters per boom berekend. In Tabel: 4.1 wordt het L-systeem ingesteld op de appelboom preset, dit zijn de initiële parameters. Dan worden er 5 bomen gegenereerd en berekend, de QSM's zijn zichtbaar op Figuur: 4.1. In deze tabel is te zien dat sommige parameters goed overeen komen en sommige iets minder. Bv. Een goede parameter is de hoogte, deze is maar een halve meter verschillend met de realiteit. Met genoeg bomen kan de variantie ook geschat worden, deze geeft 1,12 meter variatie wat toch wel dichtbij 2 ligt omdat deze maar berekend is om 5 waardes. Wat ook opvalt is dat de vorm van de boom (halve bol) vaak als kegel word gezien, dit is normaal gedrag omdat een halve bol en kegel niet veel van elkaar verschillen. Zo zijn beide breed vanonder en smal vanboven. Wat ook opmerkelijk is, is dat de 'Trunk curve' is ingesteld op 0, maar toch geven de berekende parameters een waarde rond 35 voor 'Curve' en 'Curve back'. Dit komt omdat de "Curve variation"niet alleen werkt op de totale afbuiging, maar dit werkt op elk segment apart. Daardoor kan de stam afbuigen en daarna terug recht gaan. De volgende parameters komen wel goed overeen met de initiële parameters tot de level 2 "Curve". Deze komt niet goed overeen, dit kan zijn door 2 factoren: aan de ene kant is de variatie zeer hoog en dit kan leiden tot slechte resultaten, aan de andere kant kan er nog een probleem zijn met de berekening van deze parameter.

Als men de parameters neemt van de 5 bomen en deze uit middelt, dan kunnen deze terug in het L-systeem gestoken worden. Zo kan er dan een variatie van gegenereerd worden. Zie Figuur: 4.3a voor het resultaat van deze uitgemiddelde parameters.

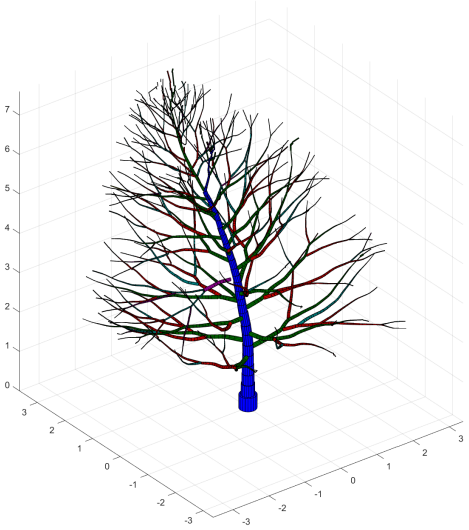
Een andere test gedaan met zelf gekozen parameters kan men zien op Figuur: 4.2 en in Tabel: 4.2. Dit zijn smallere bomen waar de takken pas hoog beginnen. Hetgeen hier uitschiet is de "Trunk Curve", deze is ingesteld op 0, maar heeft toch een hoge waarde (50 graden). Dit komt omdat de toppen van de boom zeer klein zijn en hier worden de cilinder niet goed geplaatst. Het kan goed zijn dat 1 kleine cilinder fout staat, dit is genoeg om de berekening te beïnvloeden. Bij deze is de level 2 curve ook niet ideaal en het lijkt alsof de waardes maal 2 moeten. Dit lijkt me eerder toeval. De nieuwe variatie is te zien op Figuur: 4.3b.

Tree params	Initial params	Run 1	Run 2	Run 3	Run 4	Run 5	Avg	verschil
Tree shape	Hemispherical	Conical	Conical	Conical	Conical	Hemi	Conical	
Height (meter)	9	7,59	10,89	9,93	10,21	9,29	9,582	0,582
Height variance (meter)	2						1,120	0,879
Branch thickness ratio (fractie)	0,02	0,027	0,025	0,027	0,025	0,026	0,026	0,006
Level 1								
Trunk height (fractie)	0,15	0,17	0,16	0,17	0,14	0,16	0,16	0,01
Trunk cruve (graden)	0	36	35	39	25	29,6	32,92	32,92
Trunk curve variation (graden)	70						4,990	65,009
Trunk curve back (graden)	0	-26	-27	-21	-11	-22	-21,4	21,4
Level 2								
Length fraction	0,5	0,485	0,464	0,485	0,459	0,476	0,473	0,026
Amount branches	35	26	33	26	39	36	32	3
Segmentsplits (fractie)	0,6	0,5	0,5	0,55	0,5	0,4	0,49	0,11
Split angle (graden)	20	21,89	21,2	22,26	21,31	21,74	21,68	1,68
Split angle variation (graden)	10	6,6	6,2	4,62	6,42	5,8	5,928	4,072
Down angle (graden)	60	62	62,6	69,7	65	65,3	64,92	4,92
Down angle variation (graden)	30	19,78	21	15,84	17,18	14	17,56	12,44
Rotation (graden)	140	147	140	140,08	140,13	139,8	141,402	1,402
Curve (graden)	-20	-5	-1	0,3	1,3	-2,7	-1,42	18,58
Curve back (graden)	0	-7	-8	-5,4	-4	-5	-5,88	5,88
Curve variation (graden)	140	10,9	12,45	13	9,6	14	11,99	128,01

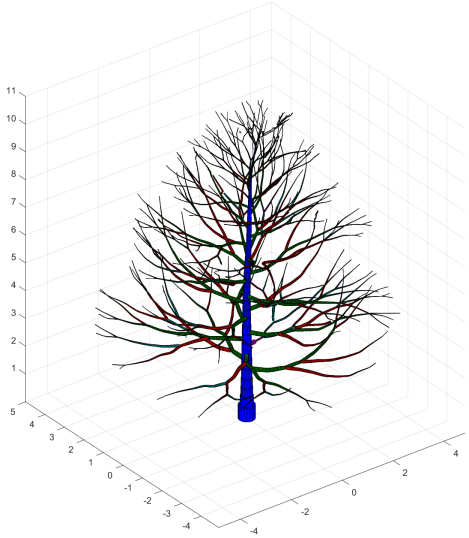
Tabel 4.1: Vergelijking van berekende parameters van 5 gegenereerde appelbomen.

Tree params	Initial params	Run 1	Run 2	Run 3	Run 4	Run 5	Avg	verschil
Tree shape	Conical	Conical	Conical	Conical	Conical	Conical	Conical	
Height (meter)	15	17,8	14,4	15,08	15,97	16,47	15,944	0,944
Height variance (meter)	2						1,170	0,829
Branch thickness ratio (fractie)	0,01	0,018	0,018	0,018	0,018	0,018	0,018	0,008
Level 1								
Trunk height (fractie)	0,5	0,48	0,5	0,49	0,5	0,5	0,494	0,006
Trunk cruve (graden)	0	55,6	47	63,4	54,95	45,8	53,35	53,35
Trunk curve variation (graden)	10						6,417	3,582
Trunk curve back (graden)	0	-6	-6	-16	-8	-8	-8,8	8,8
Level 2								
Length fraction	0,25	0,21	0,22	0,22	0,22	0,22	0,218	0,032
Amount branches	50	49	47	47	50	50	48,6	1,4
Segmentsplits (fractie)	0,2	0,1	0,1	0,1	0,1	0,1	0,1	0,1
Split angle (graden)	30	18,45	19,4	19,6	19,84	18,63	19,184	10,816
Split angle variation (graden)	10	3,8	4,7	3,27	3,9	3,14	3,762	6,238
Down angle (graden)	45	49	50,25	48,6	49	49,3	49,23	4,23
Down angle variation (graden)	-20	9	9,9	9,3	9,6	9,18	9,396	29,396
Rotation (graden)	100	101	102,6	100,1	100,8	103	101,5	1,5
Curve (graden)	30	15,5	13,98	14,6	13,8	14,78	14,532	15,468
Curve back (graden)	0	0	0	0	0	0	0	0
Curve variation (graden)	10	5,3	4	4,2	4,1	4,6	4,44	5,56

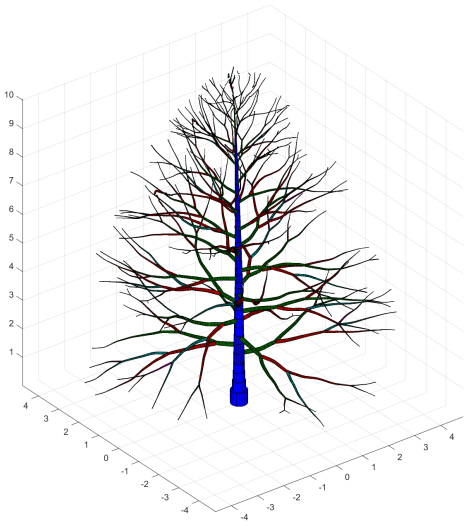
Tabel 4.2: Custom parameters



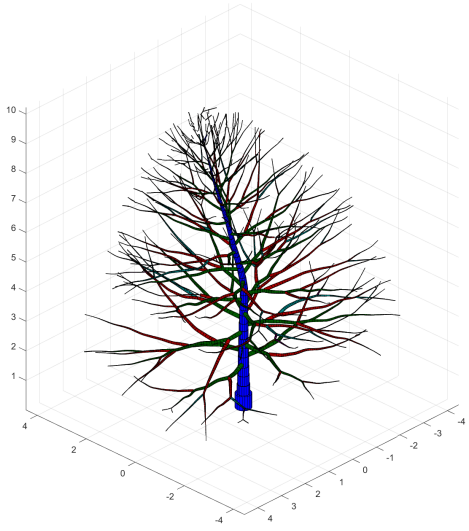
(a) Run 1



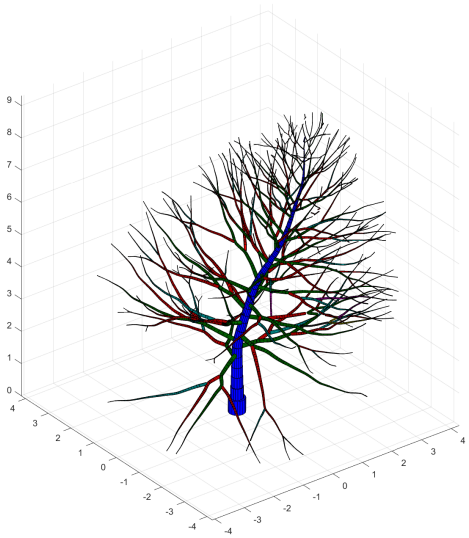
(b) Run 2



(c) Run 3

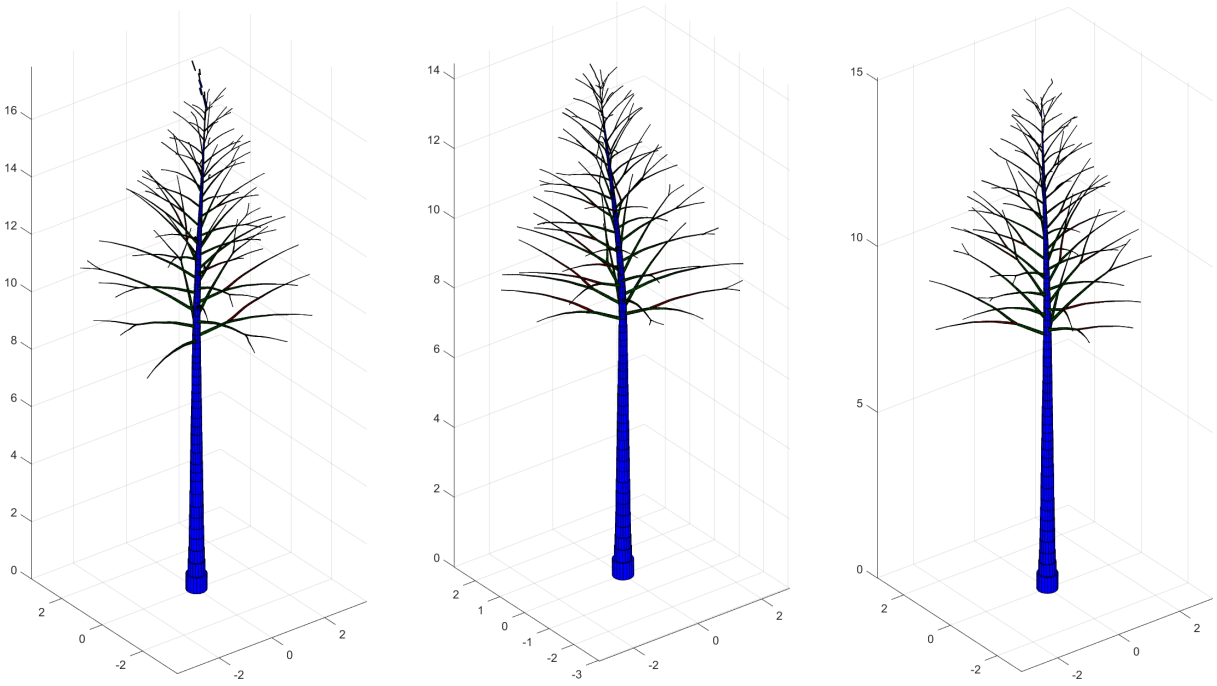


(d) Run 4

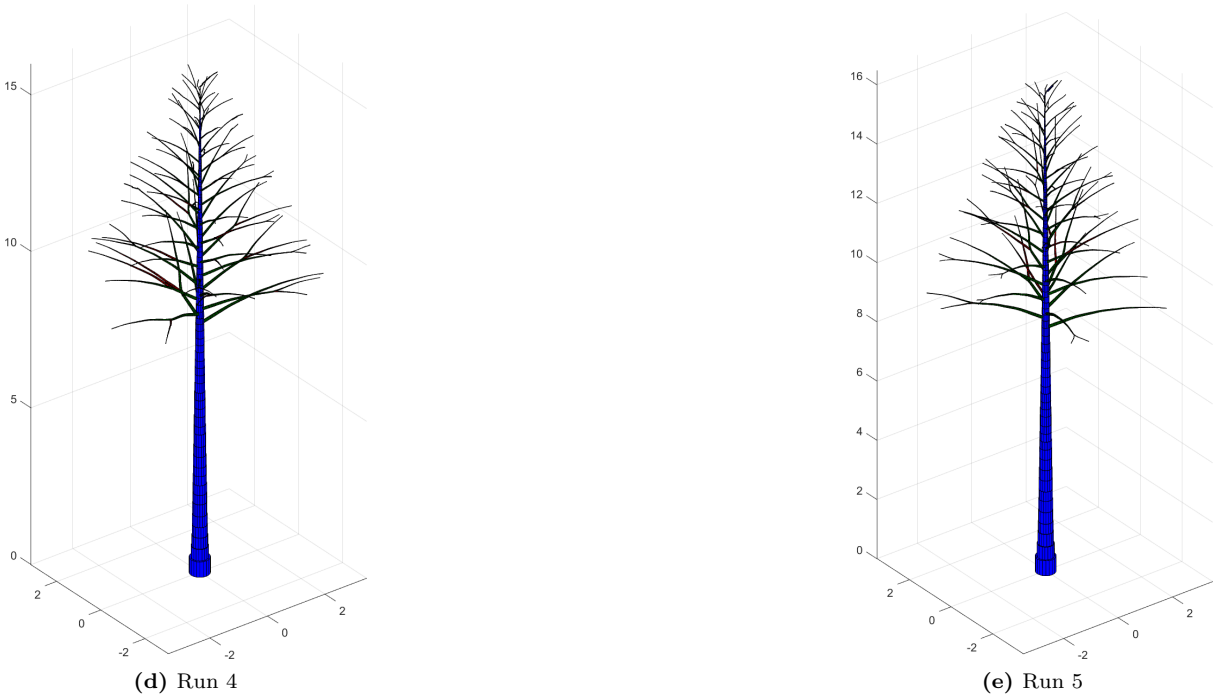


(e) Run 5

Figuur 4.1: QSM's van 5 gegenereerde appelbomen

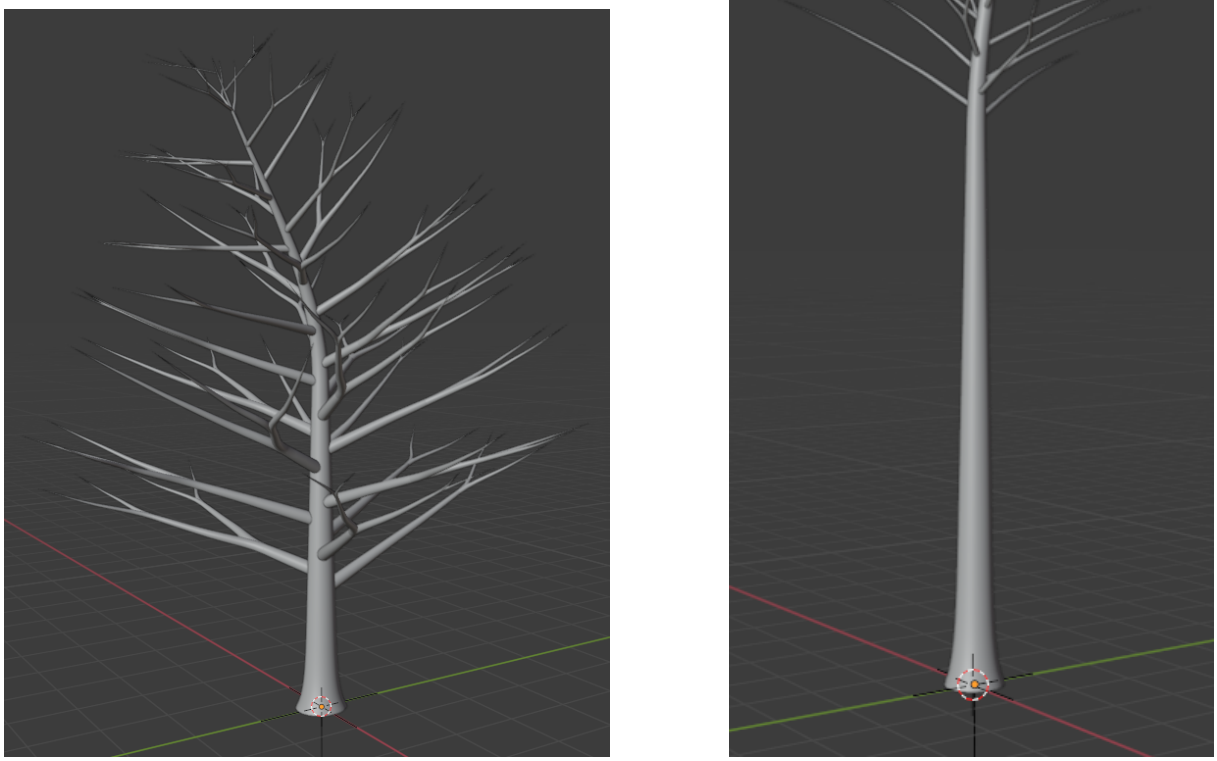


(a) Run 1 **(b)** Run 2 **(c)** Run 3



(d) Run 4 **(e)** Run 5

Figuur 4.2: QSM's van 5 gegenereerde bomen



(a) Variatie van de appelboom parameters

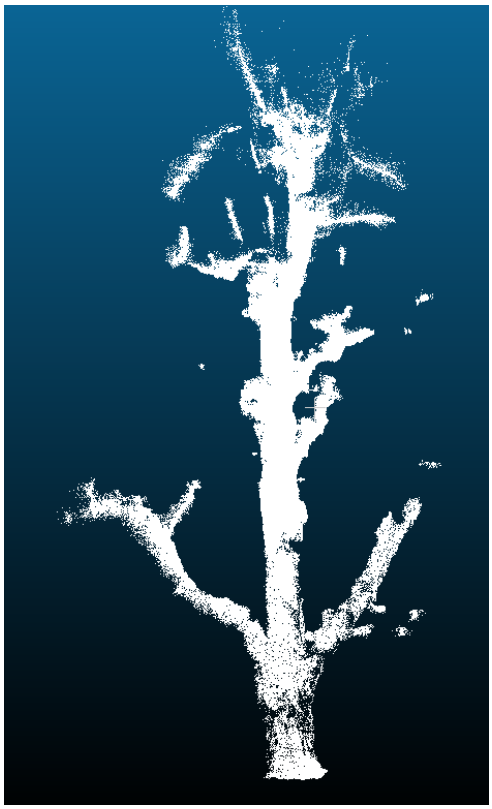
(b) Variatie van de "dennenboom"parameters

Figuur 4.3: Nieuwe variaties van analytische methode

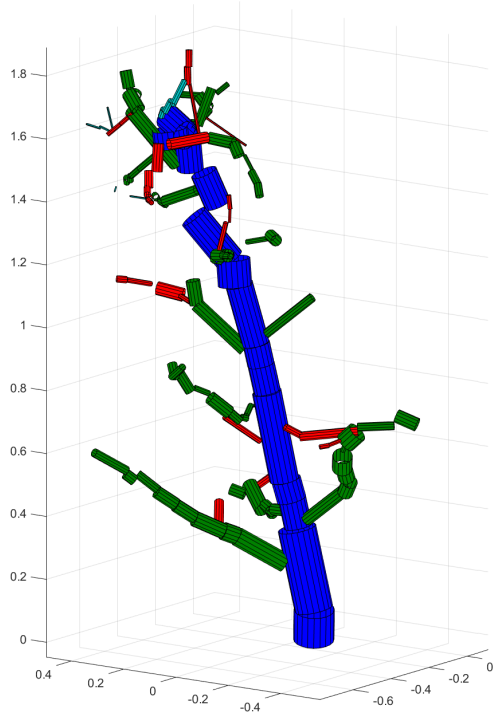
4.2 Echte bomen

In het vorig hoofdstuk werd een scan van een echte boom voorgesteld, de puntenwolk van de perenboom. Deze gaan we nu ook proberen om deze als QSM te krijgen. Omdat deze puntenwolk een andere grootte heeft en in vergelijking ook dikkere takken, kan het zijn dat de QSM input parameters aangepast moeten worden. Voor een goede reconstructie is de PatchDiam1 = 0.015, PatchDiam2Min = 0.0078 en PatchDiam2Max = 0.18. Het resultaat is te zien op Figuur: 4.4. De berekende parameters staan in Tabel: 4.3. Hier zijn enkele parameters mislukt omdat de boom te weinig takken heeft om de berekeningen goed af te werken. Als we deze parameters op 0 zetten en dan de rest invullen in het L-systeem krijgen we Figuur: 4.6a als resultaat. Zoals men kan zien is deze variatie niet goed gelukt. Dit komt door de weinige takken, maar het kan ook zijn dat de boom te vaak is gesnoeid en dus niet meer representatief is.

Om dan toch een grotere boom te vinden heb ik online gezocht naar puntenwolken van bomen. Er zijn maar enkele datasets beschikbaar, sommige zijn niet bruikbaar door lage resolutie/kwaliteit of hadden maar weinig takken, waardoor deze niet interessant zijn. De dataset dat ik heb gebruikt is de dataset van 3Dforest[1][2]. Uit deze dataset heb ik boom 17 geselecteerd. Deze heeft voldoende takken en heeft weinig extra bladeren en noise. Hier is wel geen foto van, enkel de puntenwolk. Het resultaat van QSM en de puntenwolk zelf is te zien in Figuur: 4.5. De berekende parameters zijn te zien in Tabel:4.3. Hier valt weinig speciaal op te zien, enkel word de vorm gezien als bolvormig, terwijl het beter een cilindrische vorm zou moeten hebben. Na de parameters in het L-systeem te gebruiken kunnen variaties zoal Figuur: 4.6b gegenereerd worden. Bij deze boom is de variatie goed gelukt en lijkt de boom veel op het origineel. Wat hier wel opmerkelijk is dat de buiging van de boom wel goed lijkt te zijn, maar niet de plaats van de buiging. Het L-systeem laat niet toe om dit aan te passen en berekend zijn eigen plaats van afbuigen.



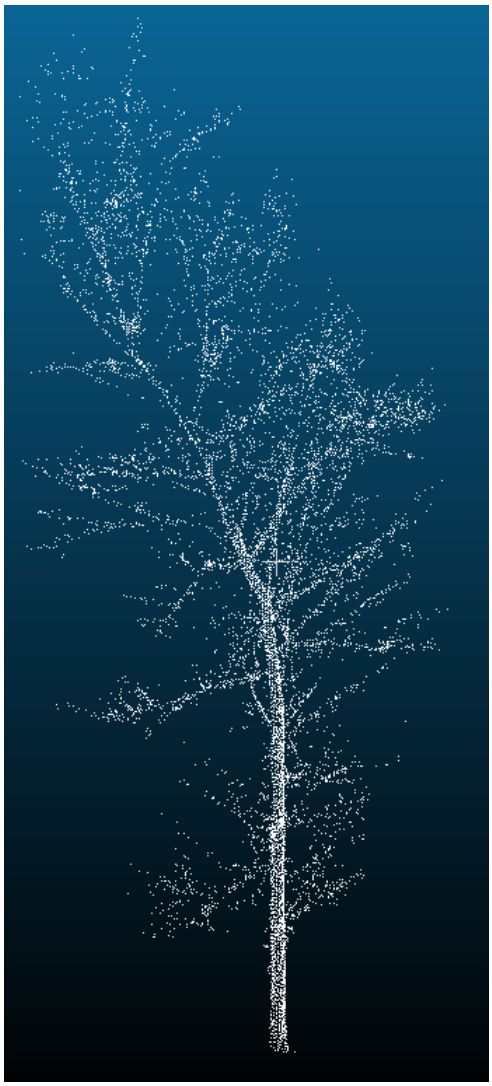
(a) Puntenwolk van perenboom



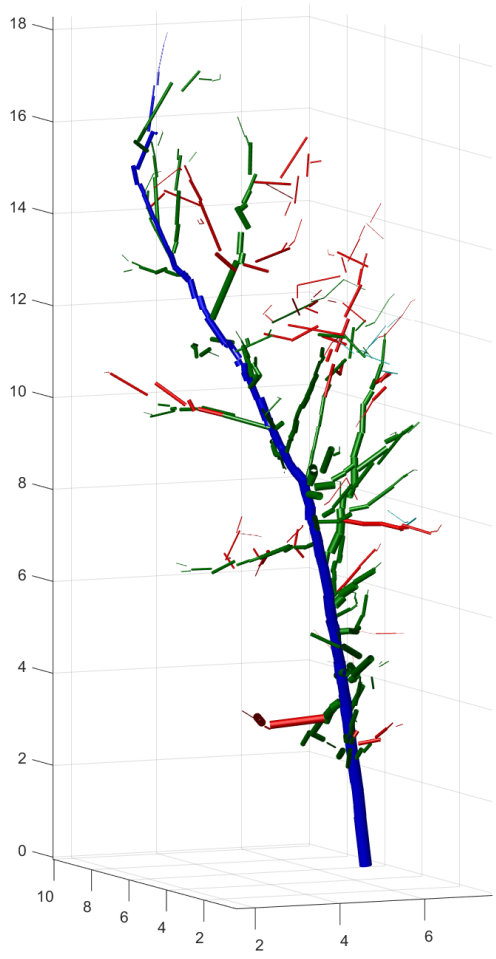
(b) QSM van perenboom

Figuur 4.4: QSM van perenboom en puntenwolk

[1] https://www.3dforest.eu/,
[2] https://github.com/VUKOZ-OEL/3dforest-data



(a) Puntenwolk van de dataset boom

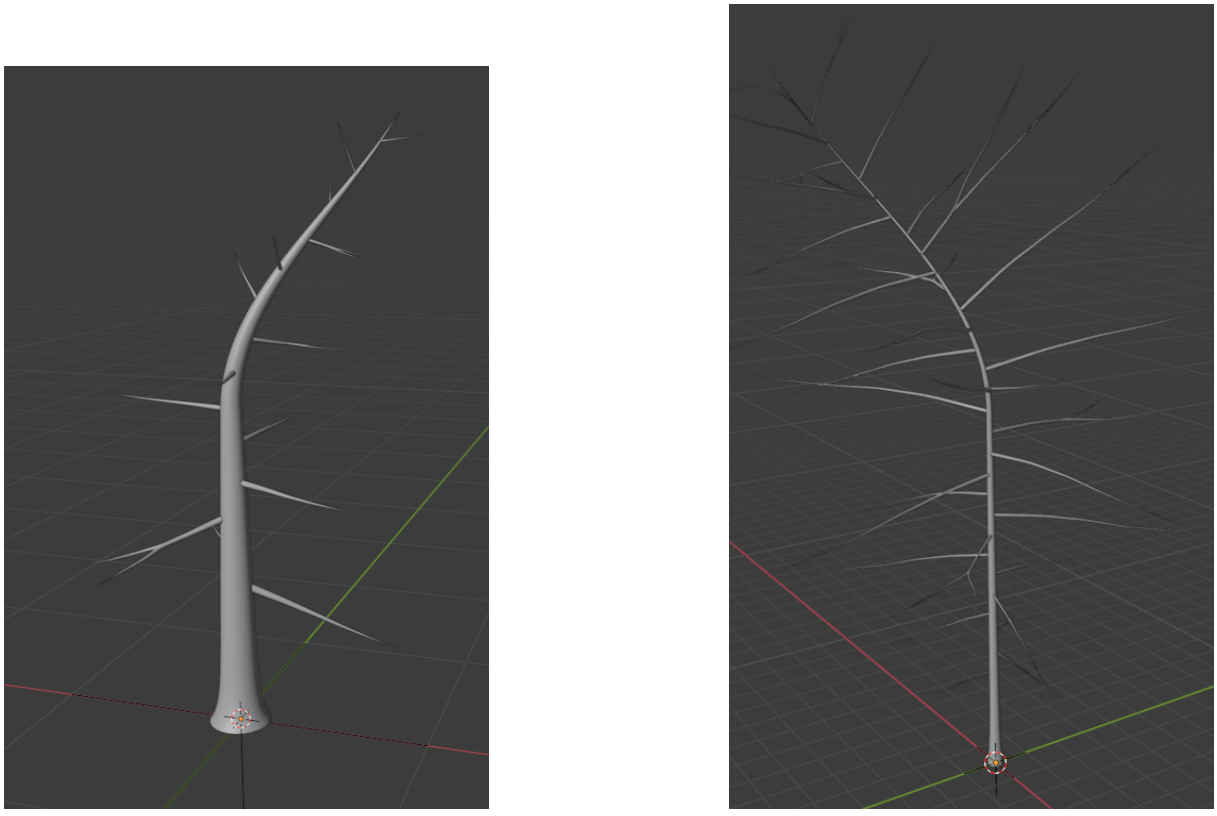


(b) QSM van de dataset boom

Figuur 4.5: QSM van de dataset boom en puntenwolk

Tree params	Perenboom	Dataset boom
Tree shape	Conical	Sphere
Height	1,92	18,56
Height variance		
Branch thickness ratio	0,032	0,0074
Level 1		
Trunk height	0,19	0,125
Trunk cruve	67,1	55,6
Trunk curve variation		
Trunk curve back	-32	-30
Level 2		
Length fraction	0,26	0,351
Amount branches	14	31
Segmentsplits	0,1	0,1
Split angle	31	5,7
Split angle variation	14	18,4
Down angle	75,3	74,53
Down angle variation	35	35,3
Rotation	111	139,4
Curve	error	-4
Curve back	error	6,3
Curve variation	error	9,84

Tabel 4.3: Parameters van echte bomen



(a) Variatie perenboom **(b)** Variatie dataset boom

Figuur 4.6: Nieuwe gegenereerde variaties van de berekende parameters(Tabel: 4.3

Hoofdstuk 5

Uitbreidingen/verbeteringen

Uit de resultaten kunnen we concluderen dat het mogelijk is om variaties te maken van een gescande boom, maar dit kan uitgebreider en robuuster.

Een mogelijke verbetering is om bomen te kunnen ondersteunen die splitsingen heeft in de hoofdstam. De huidige code zal een splitsing zien als een aftakking, hierdoor worden alle takken op deze aftakking verkeerd geïnterpreteerd. Dit verpest parameters zoals de Down Angle, Curve en mogelijks de Height. Om dit wel te laten werken moet men zelf nog deze splitsingen kunnen detecteren, om dan de datastructuur aan te passen zodat elk niveau nog klopt.

Een andere grote verbetering is om te zorgen dat er diepere niveaus worden berekend, dit zijn dan aftakkingen (Level 3) op de eerste aftakking (Level 2). Dit is moeilijk te maken omdat TreeQSM elke splitsing ook ziet als een aftakking. Om dit werkend te krijgen zal de datastructuur strek aangepast moeten worden. Niet alleen de berekeningen zijn moeilijk, maar om dit succesvol te doen moet de puntenwolk van de boom ook zeer precies zijn omdat de takken op dit niveau zeer klein zijn.

De vorm detectie van de boom kruin moet ook nog uitgebreid en verbeterd worden. Momenteel worden er 4 vormen ondersteund: halve bol, bol, cilinder en kegel. Er zijn echter meerdere vormen in L-systeem dat ik gebruik beschikbaar, namelijk: Flame, Tend Flame en Tapered Cylindrical. Ook is er nog ruimte voor een beter detectie systeem dat accurater is.

In de methode kan men enkel overweg met splitsingen waar er 2 takken uitkomen, het L-systeem kan echter meerdere aan, 3 of 4. Maar bomen dat zulke takken hebben zijn niet zo algemeen.

Sommige bomen zoals dennen bomen kunnen takken hebben dat niet gelijkmatig verdeeld zijn over de stam, deze takken zitten soms samen in groepen op verschillende hoogte niveaus. Het L-systeem kan deze bomen maken, maar de parameter berekening houd hier geen rekening mee.

Als men een scan neemt van een boom dan is deze beïnvloed door de zwaartekracht, hierdoor gaan de takken feller hangen. Als deze dan gescand zijn gaat de parameter berekening een beetje naast de correcte waardes liggen. Het L-systeem kan de zwaartekracht simuleren, maar om dit te gebruiken moeten de parameters gecorrigeerd worden.

Een goede filtering van bladeren ontbreekt nog, om een zo goed mogelijke QSM reconstructie te hebben moet de boom enkel uit hout bestaan. Er zit een filtering functie in TreeQSM, maar deze verwijderd voornamelijk noise en outliers en doet weinig tegen bladeren.

Hoofdstuk 6

Conclusies

Na afloop van deze thesis kan ik concluderen dat de onderzoeksvragen aangehaald in Hoofdstuk: 1 beantwoord zijn en het dus mogelijk is om een boom in te scannen en hiervan variaties te genereren. Maar deze is natuurlijk nog niet perfect. De methode werkt, maar er zijn vereisten om het te laten werken zo moet de boom niet te veel bladeren hebben, er moet een goede dichte scan zijn en door de limitatie van TreeQSM moet de boom enkel bestaan uit 1 stam. Dit maakt het moeilijk om met zekerheid te kunnen zeggen dat de methode nuttig is voor verder gebruik.

Wat wel jammer is dat de resultaten van de echte bomen ondermaats is. Hier heeft de perenboom een slechte vorm en een ondermaatse scan, bij de dataset boom is de scan niet duidelijk genoeg en is er geen afbeelding beschikbaar om te kunnen vergelijken met de echte wereld. Hierdoor lijkt de methode enkel te werken in theorie.

De manier van werking, beginnen met een puntenwolk en deze omzetten naar cilinders, lijkt wel de juiste manier te zijn om variaties te kunnen genereren. Uit het gerelateerd werk is dit de enige manier om een correct mogelijke boom te verkrijgen om zo ook juiste parameters te berekenen. Wat hier nog interessant is, is om de TreeQSM manier te vergelijken met het werk van [Livny et al., 2010] dat gebruik maakt van een graaf structuur om een puntenwolk om te zetten in een model. Deze manier kan mogelijks beter werken om nog accurater parameters te kunnen berekenen.

Omdat ik ben begonnen met eerst de analytische methode en daarna pas met echte bomen, had ik maar een korte tijd dat er geen bladeren op de bomen stonden. Als ik eerder was begonnen met bomen te scannen kan het zijn dat ik wel een goede scan van een boom kon maken met photogrammetry. Achteraf bekeken had ik beter ook meer moeten doen in het eerste semester, op dat moment lijkt alles nog ver weg te zien, maar het einde komt toch snel dichterbij.

In de methode word een bestaand L-systeem gebruikt om de variaties te genereren. Dit is mogelijks een te beperkte manier om variaties te maken. Het is misschien beter om dynamisch een eigen L-systeem op te bouwen of door geen L-systeem te gebruiken en eventueel een manier van grafen en andere mogelijkheden te gebruiken.

Bibliografie

[Aguilar et al., 2008] Aguilar, M. A., Pozo, J. L., Aguilar, F. J., Sanchez-Hermosilla, J., Páez, F. C., and Negreiros, J. (2008). 3D SURFACE MODELLING OF TOMATO PLANTS USING CLOSE-RANGE PHOTOGRAMMETRY. page 6.

[Andujar et al., 2018] Andujar, D., Calle, M., Fernandez-Quintanilla, C., Ribeiro, , and Dorado, J. (2018). Three-Dimensional Modeling of Weed Plants Using Low-Cost Photogrammetry. *Sensors*, 18(4):1077. Number: 4 Publisher: Multidisciplinary Digital Publishing Institute.

[Argudo et al., 2020] Argudo, O., Andújar, C., and Chica, A. (2020). Image-Based Tree Variations. *Computer Graphics Forum*, 39(1):174–184. _eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13752.

[Baltsavias, 1999] Baltsavias, E. P. (1999). A comparison between photogrammetry and laser scanning. *ISPRS Journal of Photogrammetry and Remote Sensing*, 54(2-3):83–94.

[Burt et al., 2018] Burt, A., Disney, M., and Calders, K. (2018). Extracting individual trees from lidar point clouds using treeseg. *Methods in Ecology and Evolution*.

[Chang, 2003] Chang, N. L. (2003). Efficient dense correspondences using temporally encoded light patterns. In *Proc. IEEE International Workshop on Projector-Camera Systems (ProCams)*.

[Chris, 2020] Chris (2020). What is Terrestrial Laser Scanning (TLS)? `https://www.pmcorp.com/what-is-terrestrial-laser-scanning/`.

[Fuhrmann et al., 2005] Fuhrmann, A., Umlauf, E., and Mantler, S. (2005). Extreme model simplification for forest rendering. pages 57–66.

[Hewitt,] Hewitt, C. Procedural generation of tree models for use in computer graphics. page 76.

[Hewitt, 2022] Hewitt, C. (2022). tree-gen. `https://github.com/friggog/tree-gen`.

[Lau et al., 2018] Lau, A., Bentley, L. P., Martius, C., Shenkin, A., Bartholomeus, H., Raumonen, P., Malhi, Y., Jackson, T., and Herold, M. (2018). Quantifying branch architecture of tropical trees using terrestrial LiDAR and 3D modelling. *Trees*, 32(5):1219–1231.

[Liu et al., 2021] Liu, Q., Ma, W., Zhang, J., Liu, Y., Xu, D., and Wang, J. (2021). Point-cloud segmentation of individual trees in complex natural forest scenes based on a trunk-growth method. *Journal of Forestry Research*, 32(6):2403–2414.

[Livny et al., 2010] Livny, Y., Yan, F., Olson, M., Chen, B., Zhang, H., and El-Sana, J. (2010). Automatic reconstruction of tree skeletal structures from point clouds. In *ACM SIGGRAPH Asia 2010 papers*, SIGGRAPH ASIA '10, pages 1–8, New York, NY, USA. Association for Computing Machinery.

[Martinez-Guanter et al., 2019] Martinez-Guanter, J., Ribeiro, , Peteinatos, G. G., Pérez-Ruiz, M., Gerhards, R., Bengochea-Guevara, J. M., Machleb, J., and Andújar, D. (2019). Low-Cost Three-Dimensional Modeling of Crop Plants. *Sensors*, 19(13):2883. Number: 13 Publisher: Multidisciplinary Digital Publishing Institute.

[Ochoa, 1998] Ochoa, G. (1998). On genetic algorithms and lindenmayer systems. In Eiben, A. E., Bäck, T., Schoenauer, M., and Schwefel, H.-P., editors, *Parallel Problem Solving from Nature — PPSN V*, Lecture Notes in Computer Science, pages 335–344, Berlin, Heidelberg. Springer.

[Pribanić et al., 2010] Pribanić, T., Mrvoš, S., and Salvi, J. (2010). Efficient multiple phase shift patterns for dense 3d acquisition in structured light scanning. *Image and Vision Computing*, 28(8):1255–1266.

[Prusinkiewicz et al.,] Prusinkiewicz, P., Hanan, J., Hammel, M., and Mech, R. L-systems: from the Theory to Visual Models of Plants.

[Raumonen et al., 2013] Raumonen, P., Kaasalainen, M., Åkerblom, M., Kaasalainen, S., Kaartinen, H., Vastaranta, M., Holopainen, M., Disney, M., and Lewis, P. (2013). Fast Automatic Precision Tree Models from Terrestrial Laser Scanner Data. *Remote Sensing*, 5(2):491–520. Number: 2 Publisher: Multidisciplinary Digital Publishing Institute.

[Samal et al., 1994] Samal, A., Peterson, B., and Holliday, D. (1994). Recognizing plants using stochastic L-systems. In *Proceedings of 1st International Conference on Image Processing*, volume 1, pages 183–187 vol.1.

[Shlyakhter et al., 2001] Shlyakhter, I., Rozenoer, M., Dorsey, J., and Teller, S. (2001). Reconstructing 3D tree models from instrumented photographs. *IEEE Computer Graphics and Applications*, 21(3):53–61. Conference Name: IEEE Computer Graphics and Applications.

[Sun et al., 2014] Sun, Y., Zhao, L., Huang, S., Yan, L., and Dissanayake, G. (2014). L2-SIFT: SIFT feature extraction and matching for large images in large-scale aerial photogrammetry. *ISPRS Journal of Photogrammetry and Remote Sensing*, 91:1–16.

[Tan et al., 2008] Tan, P., Fang, T., Xiao, J., Zhao, P., and Quan, L. (2008). Single image tree modeling. *ACM Transactions on Graphics*, 27(5):108:1–108:7.

[Tareen and Saleem, 2018] Tareen, S. A. K. and Saleem, Z. (2018). A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK.

[Vosselman and Maas, 2010] Vosselman, G. and Maas, H.-G. (2010). *Airborne and Terrestrial Laser Scanning.*

[Wang, 1990] Wang, Z. (1990). *Principle of Photogrammetry: With Remote Sensing.* Press of Wuhan Technical University of Surveying and Mapping.

[Weber and Penn, 1995] Weber, J. and Penn, J. (1995). Creation and rendering of realistic trees. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*, SIGGRAPH '95, pages 119–128, New York, NY, USA. Association for Computing Machinery.

[Wikipedia contributors, 2022] Wikipedia contributors (2022). L-system. `https://en.wikipedia.org/w/index.php?title=L-system&oldid=1081285581`.

[Wu et al., 2013] Wu, J., Cawse-Nicholson, K., and van Aardt, J. (2013). 3D Tree Reconstruction from Simulated Small Footprint Waveform Lidar. *Photogrammetric Engineering & Remote Sensing*, 79(12):1147–1157.

[Åkerblom, 2020] Åkerblom, M. (2020). InverseTampere/TreeQSM: Version 2.4.0.