ONDERZOEKSDOCUMENT

Point Cloud Analysis

IC-INF-2B

Pascal Westerhof, Julian Woo, David Klein, Tobias Schipper, Joran Vos, Stefan Spitse

Inhoudsopgave

Versiebeheer 2
Inleiding 3
Hoofdstuk 1: Vissengraat diagram 4
Hoofdstuk 2: Onderzoeksvraag 5
Hoofdstuk 3: Opzet van het onderzoek 6
Hoofdstuk 4: Uitwerking van het onderzoek 8
4.1 Literatuur studie 8
4.2 Best good and bad practices 8
4.3 Requirements prioritization 8
4.4 Prototyping 8
4.5 Code review 10
4.6 Product review 10
Conclusie 11

Versiebeheer

	Bewerking
0.1	Bestand aangemaakt, voorblad en paginanummers toegevoegd
0.2	Hoofdstukken, inleiding en onderzoeksvraag toegevoegd

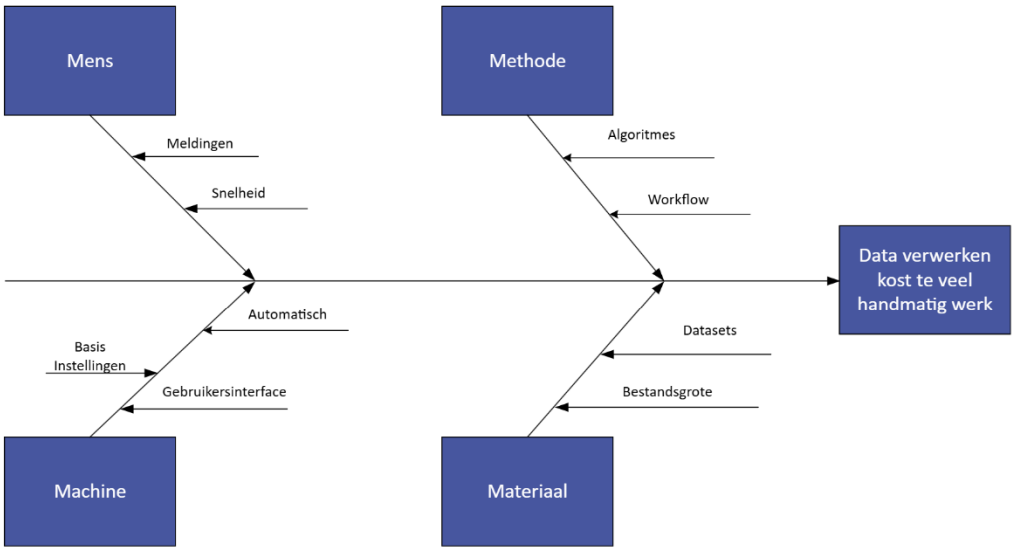
Inleiding

In dit project werken wij aan een oplossing voor Emit IT, een bedrijf dat zich bezighoudt met het verwerken van 3D-gegevens, onder andere in de vastgoed- en bouwsector. Emit IT wil het proces van oppervlakteberekening van appartementen automatiseren, zodat ze niet langer afhankelijk zijn van externe partijen.

Op dit moment wordt de oppervlakte van appartementen vaak handmatig berekend of uitbesteed aan derden die gespecialiseerd zijn in het verwerken van 3D-point clouds. Dit proces is tijdrovend, foutgevoelig en kostbaar. Emit IT wil dit proces in eigen beheer kunnen uitvoeren, door middel van een softwarematige oplossing die automatisch de oppervlakte van een appartement kan berekenen uit een **point cloud** (een puntenwolk die een 3D-scan van een ruimte vertegenwoordigt).

Ons onderzoek richt zich daarom op de vraag hoe we zo'n automatische methode kunnen ontwikkelen met behulp van bestaande technologieën en technieken voor 3D-gegevensverwerking. Hierbij ligt de nadruk op het verkennen van bestaande oplossingen, het ontwikkelen van een prototype en het valideren van de resultaten in samenwerking met experts.

Hoofdstuk 1: Vissengraat diagram



Hoofdstuk 2: Onderzoeksvraag

In dit onderzoeksdocument staat de volgende hoofdvraag centraal:

Hoe kunnen we, op basis van bestaande technieken voor point cloud-verwerking, een geautomatiseerde methode ontwikkelen die de vloeroppervlakte van appartementen betrouwbaar berekent uit 3D-point clouddata?

Hierbij horen onderstaande deelvragen:

- Welke bestaande algoritmen en segmentatietechnieken zijn het meest geschikt om bouwkundige vlakken (zoals vloeren, muren en plafonds) automatisch te herkennen en te onderscheiden in 3D-point clouds van appartementen?

- Welke bestaande softwarebibliotheken en frameworks ondersteunen deze technieken het beste op het gebied van nauwkeurigheid, prestaties, gebruiksgemak en integratiemogelijkheden binnen Python-omgevingen?

- Hoe kunnen de geselecteerde algoritmen en bibliotheken worden gecombineerd tot een werkende prototype-oplossing die automatisch oppervlakten berekent uit 3D-point clouds?

Hoofdstuk 3: Opzet van het onderzoek

Gekozen onderzoekspatroon: Realise as an Expert

Voor dit project maken we gebruik van het onderzoekspatroon “Realise as an Expert”. Dit patroon past goed bij onze onderzoeksvraag, omdat we werken met technologieën en technieken waar we nog niet volledig vertrouwd mee zijn (zoals point cloud-verwerking, 3D-geometrische analyse en automatische oppervlaktebepaling).

Waarom dit patroon?

Het patroon “Realise as an Expert” richt zich op het ontwikkelen van innovatieve oplossingen op basis van bestaande kennis en technieken. Het helpt ons om:

- Nieuwe technologieën te leren toepassen;
- Bestaande oplossingen en methoden te onderzoeken en verbeteren;
- Een generieke oplossing te ontwikkelen die door Emit IT hergebruikt kan worden.

Daarnaast stimuleert dit patroon een iteratief proces van **onderzoek – ontwikkeling – validatie**, waarbij we actief feedback verzamelen van experts en stakeholders.

Fase	Beschrijving	Toepassing in dit project
Library	Onderzoek naar bestaande kennis, technieken en libraries.	We doen literatuuronderzoek naar point cloud-verwerking, oppervlakteberekening en vergelijkbare softwaretoepassingen.
Workshop	Toepassen van kennis en bouwen van een werkend prototype.	We ontwikkelen en testen een prototype dat de oppervlakte automatisch kan berekenen.
Showroom	Valideren van de oplossing met experts en opdrachtgever.	productreviews door experts en opdrachtgever (Emit IT).

Onderzoekskaarten

Om de structuur van ons onderzoek te waarborgen, hebben we zes onderzoekskaarten gekozen die passen binnen het patroon. Deze kaarten helpen ons om onze aanpak systematisch en doelgericht uit te voeren.

1. Literatuurstudie
Omdat we werken met technologie waarin veel bestaande kennis beschikbaar is, is het belangrijk om eerst te begrijpen wat er al bestaat en wat de huidige best practices zijn voordat we zelf gaan ontwikkelen.

2. Best, Good and Bad Practices
Dit helpt ons om onze eigen aanpak te verbeteren en valkuilen te vermijden, waardoor we sneller tot een kwalitatieve oplossing kunnen komen.

3. Requirements Prioritization
Dit zorgt ervoor dat we eerst werken aan de belangrijkste en meest risicovolle onderdelen, en dat het eindproduct aansluit op de verwachtingen van de opdrachtgever.

4. Prototyping
Prototyping maakt het mogelijk om vroeg in het proces te leren wat wel en niet werkt, waardoor we gerichter kunnen ontwikkelen en de risico's verminderen.

5. Code Review
Omdat we werken aan software die later door anderen onderhouden zal worden, is het belangrijk dat de code duidelijk, betrouwbaar en onderhoudbaar is.

6. Product Review
Deze stap waarborgt dat het eindresultaat niet alleen technisch werkt, maar ook voldoet aan de verwachtingen en behoeften van de opdrachtgever.

Hoofdstuk 4: Uitwerking van het onderzoek

In dit hoofdstuk wordt de uitvoering van het onderzoek beschreven aan de hand van de gekozen onderzoekskaarten.
Elke kaart vertegenwoordigt een onderdeel van onze onderzoeksaanpak en laat zien hoe wij te werk zijn gegaan, welke keuzes we hebben gemaakt en wat de belangrijkste resultaten waren.

4.1 Literatuur studie

Wat houdt de kaart in?

Een literatuurstudie is een methode om nieuwe informatie en nieuwe inzichten op te doen over jouw gekozen onderwerp of probleemstelling. Deze informatie vind jij via verschillende bronnen, zoals wetenschappelijke artikelen, boeken, papers of archiefmateriaal.

Hoe heb je onderzoek gedaan?

Online heb ik verschillende bronnen geraadpleegd. Als eerst heb ik met Google en Google Scholar gezocht naar informatie over point clouds en algoritmes zoals de Ball Pivot Algorithm (BPA), Region Growing en de Poisson Surface Reconstruction (PSR).

Hierna heb ik over elk onderwerp een aantal publicaties doorgenomen, zowel wetenschappenlijk als reguliere artikelen. Daarbij heb ik gelet op de uitleg van de werking van de algoritmes, hun toepassingen in 3D-modellering en de voor- en nadelen die in verschillende studies werden beschreven. Ook heb ik gekeken naar recente ontwikkelingen en vergelijkingen tussen deze methodes, om een beter beeld te krijgen van wanneer elk algoritme het meest geschikt is.

Wat is het resultaat?

- **Point clouds**
    - Point clouds zijn driedimensionale representaties van een object of omgeving, opgebouwd uit een groot aantal punten die elk een specifieke positie in de ruimte hebben. Deze punten bevatten coördinaten op de x-, y- en z-as, waardoor de vorm, structuur en afmetingen van het gescande object nauwkeurig kunnen worden vastgelegd. Point clouds worden doorgaans gegenereerd met behulp van 3D-scanners, LiDAR-systemen of fotogrammetrische software die beelden vanuit verschillende hoeken combineert. Elk punt in de point cloud vertegenwoordigt een gemeten positie op het oppervlak van het object, wat samen resulteert in een gedetailleerde digitale reconstructie die gebruikt kan worden voor toepassingen zoals 3D-modellering, inspectie, cartografie en bouwkundige analyses. (JOUAV, 2025)
- **Ball Pivot Algorithm**

    - Het Ball-Pivoting Algorithm (BPA) is een methode om van een puntenwolk een 3D-oppervlak te maken, opgebouwd uit driehoeken. De punten in zo'n puntenwolk komen meestal van 3D-scans die het oppervlak van een object vanuit meerdere hoeken vastleggen. Het idee achter het algoritme is vrij eenvoudig: drie punten vormen samen een driehoek als er een denkbeeldige bol van een bepaalde straal precies langs die drie punten kan worden geplaatst, zonder dat er andere punten binnen de bol liggen. Het algoritme start met één driehoek en laat de bol vervolgens "draaien" rond de randen van die driehoek. Telkens wanneer de bol een nieuw punt raakt, wordt er een extra driehoek toegevoegd. Dit proces gaat net zolang door tot alle punten zijn verwerkt. Als de puntenwolk ongelijk verdeeld is (bijvoorbeeld met dichte en minder dichte gebieden), kan het algoritme opnieuw worden uitgevoerd met een grotere bol om ook die delen goed te reconstrueren. Het BPA werkt efficiënt, gebruikt weinig geheugen en levert nauwkeurige 3D-modellen, zelfs bij grote datasets met miljoenen punten. (Bernardini, Mittleman, Rushmeier, Silva, & Taubin, 1999)
- **Region Growing**
    - Het Region Growing-algoritme is een methode om een puntenwolk op te delen in verschillende clusters van punten die samen een relatief vlak oppervlak vormen. De punten in de puntenwolk komen meestal van 3D-scans die het oppervlak van een object vanuit meerdere hoeken vastleggen. Het idee achter het algoritme is eenvoudig: punten die dicht bij elkaar liggen en een vergelijkbare oriëntatie van hun normale vector hebben, worden samengevoegd in dezelfde regio. Het algoritme begint met een zaadpunt in een vlak gebied en voegt stapsgewijs naburige punten toe die aan de voorwaarden voldoen, waardoor de regio als het ware "groeit". Telkens wanneer een nieuw punt wordt toegevoegd, wordt dit punt ook gebruikt om verder te groeien naar andere naburige punten. Dit proces gaat door totdat er geen punten meer overblijven die aan de criteria voldoen voor de huidige regio. Daarna wordt een nieuw zaadpunt gekozen en begint een nieuwe regio. Door te werken met drempels voor vlakheid (hoek tussen normale vectoren) en kromming (curvatuur) kan het algoritme zorgen dat de clusters stabiel en logisch blijven. Het Region Growing-algoritme is efficiënt, eenvoudig toe te passen en levert duidelijke segmentaties van 3D-objecten, zelfs bij grote datasets met miljoenen punten. (PCL, z.d.)
- **Poisson Surface Reconstruction**
    - Het Poisson Surface Reconstruction-algoritme is een methode om van een puntenwolk een aaneengesloten 3D-oppervlak te reconstrueren, waarbij gebruik wordt gemaakt van de oriëntaties van de punten (de oppervlakte-naturen). De punten in de puntenwolk komen meestal van 3D-scans waarbij voor elk punt ook de normale (richting) bekend is. Het idee achter het algoritme is als volgt: je creëert een impliciete functie (de indicatorfunctie) die intern waarden 1 heeft binnen het object en 0 buiten,

en je zoekt een functie waarvan het gradient-veld overeenkomt met de normalevectoren van de gescande punten. Door de divergente operator te gebruiken, wordt dit probleem omgezet in een Poisson-vergelijking: de Laplaciaan van de indicatorfunctie moet gelijk zijn aan de divergentie van het geobserveerde vectorveld. Dit Poisson-formaat heeft als voordeel dat je alle punten tegelijk in beschouwing neemt, in plaats van heuristisch delen van de ruimte op te delen of oppervlakken lokaal te combineren. Het resultaat is dat je een glad oppervlak krijgt dat robuuster is tegen ruis en dat minder afhankelijk is van lokale beslissingen. In de uitvoering wordt vaak gewerkt met een hiërarchische datastructuur zoals een adaptieve oktree (octree). Hoe dichter je bij het oppervlak bent, hoe fijnmaziger de resolutie; verder weg kun je een grovere resolutie gebruiken. Vervolgens los je de Poisson-vergelijking op in dat adaptieve domein en extraheer je uit de verkregen indicatorfunctie een isosurface (via technieken zoals Marching Cubes) als de reconstructie van het oppervlak. Het resultaat is dat het Poisson-algoritme in staat is fijne details te reconstrueren en goed om te gaan met ruis en ongelijke puntverdelingen. (Kazhdan, Bolitho, & Hoppe, 2006)

4.2 Best good and bad practices

Wat houdt de kaart in?

Best good and bad practices richt zich op het analyseren van bestaande werkwijzen. Er wordt gekeken naar verschillende methodes en tools om te achterhalen wat wel goed werkt en wat juist niet. Hierdoor worden werk fouten voorkomen en optimaliseren we het project.

Hoe heb je onderzoek gedaan?

Om de beste good and bad pratices van Point Cloud te identificeren hebben we meerdere bronnen onderzocht.

Literatuuronderzoek en documentatie
We hebben meerdere publicaties over point cloud segmentatie doorgenomen. Met focus op algoritmes zoals, Region Growing en Ball Pivot algoritme, zoals Region Growing en het Ball Pivoting Algorithm (Lotemn, z.d.-) (PCL, z.d.).

Vergelijking van softwaretools
We hebben verschillende libraries getest, waaronder Open3D, PCL (Poin**t** Cloud Library) en CloudCompare. We keken naar gebruiksvriendelijkheid, prestaties, documentatie en mogelijkheden voot automatisering. Vooral de documentatie van PCL over Region Growing was nuttig om parameters en segmentatiegedrag beter te begrijpen(PCL, z.d.).

Praktijkvoorbeelden

We hebben gekeken naar ervaringen van ontwikkelaars via GitHub, zoals bij de repository *learn_region (Jingdao, z.d.-b).*, Dit gaf inzicht in veelvoorkomende uitdagingen bij het werken met point clouds zoals ruis, datavolume en onnauwkeurige segmentatie(Jingdao, z.d.-b)

Eigen experimenten
Door zelf kleine experimenten uit te voeren met verschillende instellingen. Methoden en algoritmes konden we direct ervaren welke aanpakken effectief waren en welke beperkingen hadden. Dit werd gedaan doormiddel van code met de verschillende algortimes te runnen in Vscode en laten visualiseren doormiddel van print lines, open3d en csv files.

Wat is het resultaat?

Best Practices

- Gebruik van region growing voor vlakdetectie:
  Deze methode presteert goed bij het herkennen van grote, vlakke oppervlakken en is robuust tegen kleine hoeveelheden ruis (PCL, z.d.).
- Combinatie van filtering en segmentatie:
  Voorafgaand filteren van de point cloud (bijv. met voxel downsampling of statistical outlier removal) verhoogt de nauwkeurigheid van vlakdetectie aanzienlijk.
- Visualisatie in elke stap:
  Het visueel controleren van resultaten na elke bewerkingsfase helpt fouten vroegtijdig te identificeren en beter te begrijpen hoe het algoritme reageert op verschillende instellingen.
- Iteratief afstellen van parameters:
  Kleine aanpassingen aan parameters zoals afstands- en hoektolerantie leveren vaak grote verbeteringen in nauwkeurigheid op(Bernardini et al., 1999).

Good Practices

- Gebruik maken van meerdere libraries parallel:
  Open3D is geschikt voor prototyping vanwege de eenvoud, terwijl PCL meer controle biedt voor optimalisatie. Het combineren van inzichten uit beide werelden levert betere resultaten.
- Documenteren van testresultaten:
  Door elke test en parameterwijziging vast te leggen, konden we beter onderbouwen waarom bepaalde keuzes zijn gemaakt.
- Automatische drempelbepaling:
  Experimenteel bleek dat automatische parameterafstemming (bijv. met adaptieve toleranties) de robuustheid verhoogt bij verschillende datasets.

Bad Practices

- Geen filtering uitvoeren voor segmentatie:
  Wanneer ruwe data direct werd gebruikt, leidde dit vaak tot onnauwkeurige of incomplete vlakken.
- Te hoge resolutie behouden:
  Onnodig grote point clouds vertraagden de berekening en bemoeilijkten interpretatie zonder merkbare kwaliteitswinst.
- Gebrek aan visuele controle:
  Enkel vertrouwen op numerieke output zonder visuele validatie resulteerde soms in foutieve interpretaties van de resultaten.

4.3 Requirements prioritization

Wat houdt de kaart in?

De *requirements prioritization* kaart wordt gebruikt om alle eisen of wensen van de opleveringen overzichtelijk te maken en tegelijkertijd te wegen. Het bijbehorende doel is om inzicht te krijgen in welke requirements van groot belang zijn binnen het project en welke juist eerder optioneel. Op die manier kunnen de beschikbare tijd en middelen op een efficiëntere wijze worden ingezet.

Bij deze onderzoeksmethode wordt er rekening gehouden met de belangen van alle betrokkenen, waaronder eindgebruikers en ontwikkelaars. De eisen kunnen betrekking hebben op functionaliteit, ontwerp, technische aspecten of prestaties. De prioritering van de requirements kan bijvoorbeeld worden uitgevoerd door middel van de MoSCow-methode of een backlog. Aan de hand van de prioritering wordt duidelijk welke eisen als eerst opgepakt moeten worden.

Hoe heb je onderzoek gedaan?

Het opstellen van de initïele requirements is vooral gedaan op basis van het uitgevoerde literatuuronderzoek. Uiteraard volgden er meer functionele en onderzoeksgerichte eisen die werden afgeleid uit (wekelijkse) gesprekken met de opdrachtgever en eventueel feedback van de begeleiders. Om de prioriteiten vervolgens te kunnen vaststellen, is in dit geval gebruikgemaakt van de MoSCow-methode. De requirements worden verdeeld over vier verschillende categorieën: Must have, Should have, Could have en Won’t have.

- **Must have:** essentieel voor het behalen van de projectdoelen, zonder deze vereisten voldoet het product of onderzoek niet aan zijn minimale functie of doel. In dit project betreft het allereerst de onderdelen die noodzakelijk zijn om bruikbare resultaten uit puntenwolken te kunnen krijgen.
- **Should have:** belangrijk, maar niet per se van dermate belang voor het behalen van de projectdoelen. Bij tijdgebrek kunnen ze nog later worden toegevoegd. De verwerking van puntenwolken kan in principe zonder.
- **Could have:** wenselijk, de vereisten verhogen namelijk het gebruiksgemak of de kwaliteit van het product of onderzoek. Ze worden enkel opgepakt wanneer er tijd over is.
- **Won't have:** niet relevant of haalbaar, dit zijn eisen die bewust buiten de scope van het project vallen. Zo richt het onderzoek betreffende het berekenen van oppervlaktes uit puntenwolken zich alleen op klassieke algoritmes, gezien de beschikbare tijd.

Aan het begin van de analyse is een overzicht gecreëerd van alle functies, mogelijkheden en randvoorwaarden die relevant zijn voor het project. Dit overzicht bevatte functionele eisen zoals het automatisch herkennen van vloeren/muren en niet-functionele eisen zoals bestandsgrootte en performance. Alle eisen zijn samen met de projectgroep en alle betrokken stakeholders beoordeeld.

Hierbij is telkens gekeken naar:

- Het belang van een eis voor het einddoel van de tool.
- De complexiteit of moeite die nodig is om de eis te realiseren.
- Mogelijke afhankelijkheden tussen eisen.

Uitgaande van deze drie punten, zijn alle user stories opgesteld en hebben de requirements een prioriteitslabel gekregen. Om meer duidelijkheid te scheppen, is besloten om te werken met Epics. Zo zijn de requirements opgedeeld in die verschillende hoofdonderdelen: het onttrekken van oppervlaktes, de CLI-tool en het onderzoek naar de algoritmes. Deze komen ook terug in de MoSCow-analyse.

Neem als voorbeeld de requirement over het herkennen van vloervlakken vanuit een puntenwolk. Deze valt onder het onttrekken van oppervlaktes en is uiteindelijk geclassificeerd als een Must have. Zonder vloerdetectie is namelijk geen betrouwbare oppervlakteberekening mogelijk.

Onder andere door resultaten vanuit de prototyping zijn de prioriteiten echter niet definitief. Zo is gebleken dat het gebruik van PointNet, een deep learning model, voor het bepalen van oppervlaktes niet realistisch is. De bijbehorende requirement is dan ook veranderd naar een Won't have. Ook is de indeling continu gevalideerd door gesprekken met de opdrachtgever en controle op inconsistenties en afhankelijkheden. Hierdoor blijft de analyse representatief.

Alles bij elkaar heeft de volgende tabellen, gebruikmakend van de MoSCow-methode, opgeleverd:

Onttrekken van oppervlaktes

Must have	• Meubels en obstakels negeren zodat alleen gebouwstructuur overblijft. • Afmetingen (lengte, breedte, oppervlakte) van elke kamer tonen. • Kamers automatisch groeperen en scheiden. • Muren identificeren. • Vloeren herkennen in de puntenwolk.
Should have	• Deel selecteren om oppervlakte te meten. • Nauwkeurigheid van meting tonen. • Kamers nummeren. • Rapportage van mogelijke fouten na calculatie.
Could have	• Ondersteuning voor zowel .las als .laz bestanden. • Waarschuwing tonen bij grote bestandsgrootte.
Won't have	• Exporteren of visualiseren in externe tools

CLI-tool

Must have	• Filters kunnen toepassen (RGB, S.O.R., noisefilter). • Meerdere pointclouds kunnen inladen via CLI. • Keuzevrijheid in welke filters worden toegepast.
Should have	• Voortgang (progressie) van filters kunnen zien. • Logging en errorhandling.
Could have	• Verbeterde CLI-ervaring (kleur, interactieve feedback, etc.).
Won't have	• Grafische interface

Research onttrekken van oppervlaktes

Must have	• Onderzoek naar Ball Pivot Algorithm (BPA), Region Growing, Poisson Reconstruction en PointNet (werking, voor-/nadelen, documentatie). • Adviesdocument met bevindingen en aanbevelingen. • Inzicht in accuratie per techniek.
Should have	• Performancevergelijking van technieken (tijd, stabiliteit, bestandsgrootte). • Documentatie van limieten per algoritme.
Could have	• Visualisatie of benchmarkingtool van prestaties. • Automatische vergelijking van resultaten.
Won't have	• Implementatie van deep learning-modellen zoals PointNet.

Wat is het resultaat?

Het toepassen van de requirements prioritization heeft geleid tot een duidelijke en onderbouwde rangschikking van eisen, wat op diens beurt richting heeft gegeven aan het verdere onderzoek en de proof of concepts. Door goed gebruik te maken van de MoSCoW-methode konden de verschillende wensen en eisen worden ingedeeld in vier categorieën: Must have, Should have, Could have en Won't have.

Deze indeling zorgde voor een duidelijk overzicht van alles wat nodig is om het project te kunnen laten slagen. Het maakte ook inzichtelijk welke eisen eventueel later nog konden worden uitgewerkt, zolang de tijd en middelen nog beschikbaar waren. Daardoor kon de focus blijven op de kern van al het onderzoek: het onttrekken van oppervlaktes uit puntenwolken.

Het onderzoek heeft tot slot ook bijgedragen aan het leren van nieuwe lessen. Deze specifieke onderzoeksmethode benadrukte het belang van duidelijke definities en consistentie bij het formuleren van eisen, liet zien dat er afhankelijkheden kunnen zijn tussen de verschillende eisen, en maakte het makkelijker om stakeholders op de hoogte te kunnen houden van de progressie en prioriteiten. Dit alles zorgde uiteindelijk voor een beter eindresultaat.

4.4 Prototyping

Wat houdt de kaart in?
De Prototyping-kaart richt zich op het ontwikkelen en testen van concepten of technische benaderingen om inzicht te krijgen in de haalbaarheid van een oplossing. In dit project gebruiken we prototyping om te ontdekken welke methode het meest geschikt is om automatisch oppervlakten te onttrekken uit point clouds.

Doel van de prototypes is om te onderzoeken hoe verschillende algoritmes zich gedragen bij het segmenteren en classificeren van vlakken, en om te bepalen welke aanpak het meest nauwkeurig, efficiënt en robuust is voor appartementenscans.

Hoe heb je onderzoek gedaan?
We hebben meerdere prototypes ontwikkeld, elk met een ander doel of variant van hetzelfde algoritme. De prototypes maakten gebruik van de region growing-methode, een techniek waarmee punten in een 3D-scan worden samengevoegd tot vlakken op basis van hoe dicht ze bij elkaar liggen en hoe gelijk hun oriëntatie is.

Door verschillende versies te maken konden we beter begrijpen:

- Hoe gevoelig de methode is voor instellingen (zoals afstand en hoek tussen punten);
- Hoe goed de techniek werkt op verschillende point clouds (met meer of minder detail);
- En wat er nodig is om vloeren, muren en plafonds automatisch te herkennen.

Voor elk prototype hebben we een klein plan opgesteld:

- We bepaalden wat we wilden leren (bijvoorbeeld: “werkt de detectie van vloeren goed met deze instellingen?”).
- We pasten parameters aan en vergeleken de uitkomsten.
- We visualiseerden de resultaten om te zien of de vlakken goed waren herkend.

Daarnaast hebben we nog enkele kortere experimenten gedaan, bijvoorbeeld met andere instellingen en filters, om te onderzoeken hoe we kleine objecten (zoals meubels) konden negeren en hoe we de berekening sneller konden maken.

Wat is het resultaat?
De prototypingfase heeft ons veel inzicht gegeven in welke aanpak het beste werkt voor ons doel.
We ontdekten dat region growing goed in staat is om grote, vlakke oppervlakken te

herkennen, zoals vloeren en muren. Door zorgvuldig afstellen van de instellingen konden we een goede balans vinden tussen snelheid en nauwkeurigheid.

Belangrijke bevindingen:

- Region growing is betrouwbaar voor het segmenteren van vloeren, muren en plafonds.
- Parameterinstellingen (zoals afstand en oriëntatie) hebben een grote invloed op de kwaliteit van de resultaten.
- Visualisatie van de point clouds is essentieel om te beoordelen of het algoritme goed werkt.
- Kleine objecten, zoals meubels, kunnen de resultaten beïnvloeden, maar zijn te filteren met de juiste instellingen.

Uiteindelijk konden de prototypes duidelijk de verschillende vlakken, zoals: muren, vloeren en plafonds onderscheiden. Ook konden de prototypes verschillende kamers detecteren en segmenteren. Deze resultaten kunnen gebruikt worden om nauwkeurig de oppervlakte van iedere kamer te berekenen.

4.5 Code review

Wat houdt de kaart in?

De Code review kaart is mensen elkaars broncode kritisch nakijken om de kwaliteit, leesbaarheid en betrouwbaarheid van de software te verbeteren. Het doel is niet alleen om fouten of inefficiënties op te sporen, maar ook om te controleren of de code voldoet aan afgesproken standaarden, goed gedocumenteerd is en aansluit bij de onderzoeksdoelen of prototypes die binnen het project worden ontwikkeld. Door middel van code reviews leren studenten bovendien van elkaars programmeerstijl en keuzes, wat bijdraagt aan kennisdeling, teamontwikkeling en het verhogen van de algehele kwaliteit van het onderzoeksproject.

Hoe heb je onderzoek gedaan?

Het projectteam heeft gedurende het ontwikkeltraject wekelijks code reviews uitgevoerd om de kwaliteit, betrouwbaarheid en onderhoudbaarheid van de software te verbeteren. Tijdens deze sessies werd de code gezamenlijk besproken door het team: één of meerdere teamleden presenteerden hun geschreven code op een groot scherm, waarna de rest van de groep actief meedacht, vragen stelde en suggesties deed voor verbeteringen. Deze werkwijze zorgde voor open communicatie en stimuleerde een leercultuur waarin iedereen betrokken was bij elkaars werk. Er werd niet alleen gekeken naar fouten of inefficiënties, maar ook naar de algemene structuur van de code, de

consistentie van naamgevingen, de toepassing van programmeerprincipes en de naleving van best practices.

Omdat het projectteam relatief nieuw was met de gebruikte technologie en programmeertaal, dienden deze code reviews ook als een belangrijk leer- en kennisdelingsmoment. Door samen te analyseren waarom bepaalde oplossingen wel of niet optimaal waren, kregen teamleden beter inzicht in de werking van de taal en de gebruikte frameworks. De focus lag niet alleen op het corrigeren van code, maar juist op het begrijpen van achterliggende keuzes en het verbeteren van de algehele werkwijze.

Daarnaast droegen de code reviews bij aan een professionelere aanpak binnen het ontwikkelproces. Door regelmatig feedback te geven en te ontvangen, ontstond een beter begrip van gezamenlijke standaarden en werd de codebase consistenter. Ook werden potentiële problemen vroegtijdig gesignaleerd, waardoor technische schulden en fouten in latere fases konden worden voorkomen. De combinatie van samenwerking, kennisdeling en kwaliteitsbewaking maakte de code reviews tot een waardevol onderdeel van het project, zowel voor het eindresultaat als voor de persoonlijke groei van de teamleden als beginnende ICT-professionals.

Wat is het resultaat?

De code reviews hebben uiteindelijk gezorgd voor een hogere kwaliteit en consistentie van de code, doordat deze beter voldeed aan de afgesproken best practices. Problemen werden sneller opgemerkt en opgelost, wat de stabiliteit van het project ten goede kwam. Daarnaast zorgden de sessies ervoor dat teamleden meer inzicht kregen in elkaars werk en beter konden samenwerken, ondanks dat ieder aan verschillende onderdelen werkte. Hierdoor droegen de code reviews niet alleen bij aan betere software, maar ook aan een sterker projectteam.

4.6 Product review

Wat houdt de kaart in?
We hebben een tool ontwikkeld die automatisch afstanden berekent binnen puntenwolken (.las en .laz-bestanden).
De tool is ontworpen om nauwkeurige én snelle berekeningen uit te voeren, zodat deze geschikt is voor toepassingen waarbij zowel precisie als efficiëntie belangrijk zijn.
Deze review richt zich op het toetsen van de werking, prestaties en gebruiksgemak van de tool aan de hand van vooraf opgestelde producteisen.

Producteisen:

- De tool moet zowel .las als .laz bestanden kunnen inlezen.
- De berekeningen moeten nauwkeurig zijn (minimale afwijking t.o.v. referentiedata).
- De uitvoering moet snel verlopen (efficiënt met grote datasets).
- Resultaten moeten reproduceerbaar en inzichtelijk zijn voor de gebruiker.

Hoe heb je onderzoek gedaan?
Samen met de projectgroep hebben we onderzoek gedaan naar de werking van **point clouds** en verschillende **afstandsbepalingsalgoritmes**.
We hebben meerdere algoritmes geïmplementeerd, getest en geëvalueerd op prestaties en nauwkeurigheid.
Daarnaast hebben we de resultaten besproken met experts om feedback te verzamelen over de bruikbaarheid van de tool.

Wat is het resultaat?
De geselecteerde algoritmes zijn succesvol toegepast in een **Proof of Concept**.
De tool kon automatisch afstanden berekenen in zowel .las als .laz-bestanden en voldeed aan de gestelde eisen voor snelheid en nauwkeurigheid.
De review bevestigde dat het product technisch goed functioneert en bruikbaar is voor verdere ontwikkeling

5. Conclusie

Tijdens dit onderzoek hebben wij onderzocht hoe een geautomatiseerde methode kan worden ontwikkeld die betrouwbaar de vloeroppervlakte van appartementen berekent uit 3D-point clouddata. Dit onderzoek is uitgevoerd in opdracht van Emit IT, met als doel het handmatige en foutgevoelige proces van oppervlakteberekening te automatiseren door middel van software die binnen het bedrijf kan worden ingezet.

Door het gebruik van het onderzoekspatroon Realise as an Expert hebben wij kennis opgedaan van bestaande technieken en deze toegepast om verschillende technieken te onderzoeken. De verschillende onderzoekskaarten waaronder literatuurstudie, best good and bad practices, requirements prioritization, prototyping, code review en product review gaven structuur en hielpen om zowel theoretische kennis als praktische resultaten te combineren.

Uit de literatuurstudie en de analyse van bestaande good en bad practices bleek dat technieken zoals Region Growing en Ball Pivot veelbelovend zijn voor het segmenteren van vlakke oppervlakken in point clouds. Vooral Region Growing leverde in de prototypes betrouwbare resultaten op bij het herkennen van vloeren, muren en plafonds, mits de juiste parameters werden gekozen en filtering vooraf plaatsvond.

De requirements prioritization met behulp van de MoSCoW-methode zorgde voor duidelijke focus tijdens de ontwikkeling. Hierdoor konden de belangrijkste functionaliteiten, zoals vloerdetectie en oppervlakteberekening, met prioriteit worden uitgewerkt. Minder essentiële onderdelen, zoals deep learning-methoden, zijn bewust buiten de scope gehouden.

De fase van prototyping leverde waardevolle inzichten op in de praktische toepasbaarheid van de onderzochte algoritmes. De prototypes toonden aan dat automatische vlakdetectie goed mogelijk is en dat nauwkeurige oppervlaktes kunnen worden berekend uit 3D-scans. De code reviews droegen daarbij bij aan een hogere kwaliteit van de software, doordat fouten vroegtijdig werden opgespoord en kennis werd gedeeld binnen het team.

Het onderzoek geleid tot een onderbouwde aanpak voor het automatisch berekenen van oppervlakten uit point clouds. De resultaten tonen aan dat een combinatie van filtering, segmentatie en parametrische afstemming een effectieve basis vormt voor verdere ontwikkeling van een geautomatiseerde tool binnen Emit IT.

Hoewel de prototypes nog niet volledig compleet zijn, bieden ze een solide basis voor vervolgstappen, zoals het optimaliseren van prestaties, uitbreiden van functionaliteiten en integratie met bestaande workflows van het bedrijf.

Samenvattend kan worden gesteld dat het onderzoek zijn doel heeft bereikt: er is een werkbare methode ontworpen, getest en gevalideerd waarmee automatisch oppervlakten uit 3D-point clouds kunnen worden berekend. Deze methode vormt een

belangrijke stap richting de automatisering van oppervlakteberekeningen binnen Emit IT.

Literatuurlijst

Bernardini, F., Mittleman, J., Rushmeier, H., Silva, C., & Taubin, G. (1999). The ball-pivoting algorithm for surface reconstruction. *IEEE Transactions On Visualization And Computer Graphics*, *5*(4), 349–359. https://doi.org/10.1109/2945.817351

Jingdao. (z.d.-b). *GitHub - jingdao/learn_region_grow: LRGNet: Learnable Region Growing for Point Cloud Segmentation*. GitHub. https://github.com/jingdao/learn_region_grow

Lotemn. (z.d.-c). *GitHub - Lotemn102/Ball-Pivoting-Algorithm: Python implementation of the Ball-Pivoting algorithm (BPA)*. GitHub. https://github.com/Lotemn102/Ball-Pivoting-Algorithm

Point Cloud Library (PCL): pcl::RegionGrowing< PointT, NormalT > Class Template Reference. (z.d.). https://pointclouds.org/documentation/classpcl_1_1_region_growing.html

JOUAV. (2025, 7 mei). *What is Point Cloud and What is it Used for? (A Beginner's Comprehensive Guide)*. https://www.jouav.com/blog/point-cloud.html

Kazhdan, M., Bolitho, M., & Hoppe, H. (2006). *Poisson surface reconstruction*. In *Proceedings of the 4th Eurographics Symposium on Geometry Processing* (pp. 61–70). https://hhoppe.com/poissonrecon.pdf

Point Cloud Library. (z.d.). *Region growing segmentation*. https://pcl.readthedocs.io/projects/tutorials/en/master/region_growing_segmentation.html