



Eindrapport – Puntenwolken verwerking

Gemeente Den Haag

2021-11-25

Versie: 1.0

Confidential © 2021 CGI Inc.

Inhoudsopgave

[1 Introductie 3](#_Toc88748383)

[1.1 Achtergrond 3](#_Toc88748384)

[1.2 Onderzoeksvraag 3](#_Toc88748385)

[1.3 Deep Dive 3](#_Toc88748386)

[1.4 Overzicht oplossing 4](#_Toc88748387)

[1.5 Gebruik applicatie 4](#_Toc88748388)

[2 Methoden 6](#_Toc88748389)

[2.1 Invoer 6](#_Toc88748390)

[2.2 Verwerkingsmodule 6](#_Toc88748391)

[2.3 Uitvoer 9](#_Toc88748392)

[3 Resultaten 10](#_Toc88748393)

[3.1 Pre-processor 10](#_Toc88748394)

[3.2 Splitsen verdiepingen 11](#_Toc88748395)

[3.3 Splitsen kamers 11](#_Toc88748396)

[3.4 Reconstructie 3D objecten 13](#_Toc88748397)

[3.5 Berekenen oppervlaktes en volumes 13](#_Toc88748398)

[3.6 Oppervlakte vergelijking met BIM 15](#_Toc88748399)

[3.7 Resultaten overzicht bestandsnamen 15](#_Toc88748400)

[3.8 Resultaten resterende Woonstad puntenwolken 16](#_Toc88748401)

[4 Conclusie 18](#_Toc88748402)

[5 Lessons Learned 19](#_Toc88748403)

[5.1 Inwinning puntenwolken 19](#_Toc88748404)

[5.2 Verwerking puntenwolken 20](#_Toc88748405)

[5.3 Afwijking van programma van eisen 21](#_Toc88748406)

[5.4 Aanpassingen op plan van aanpak 22](#_Toc88748407)

[6 Aanbevelingen 23](#_Toc88748408)

[6.1 Inwinning Puntenwolken 23](#_Toc88748409)

[6.2 Licenties 23](#_Toc88748410)

[6.3 Methode 23](#_Toc88748411)

[6.4 Implementatie 24](#_Toc88748412)

# Introductie

## Achtergrond

De gemeente Den Haag is één van de drie Nederlandse gemeenten die deelneemt aan het subsidieprogramma “Proefopstelling Totaal Driedimensionaal” (T3D). Samen met de Vereniging van Nederlandse Gemeenten (VNG) richt dit programma zich op de effectieve verwerking en het gebruik van verschillende 3D-gegevens van de stedelijke omgeving uit de Nederlandse vastgoedwereld.

Gemeente Den Haag focust zich op het eerste deel van de “data pijplijn”, namelijk de extractie en voorverwerking van de bestaande 3D gegevens. Om deze gegevens in te winnen en te verwerken heeft de gemeente Den Haag CGI gevraagd een Proof of Concept (PoC) uit te voeren. In het kader van deze PoC streeft de gemeente ernaar om drie soorten gegevens op te nemen: BIM-modellen, puntenwolken en (potentieel) digitale beelden. Deze gegevens moeten vervolgens worden voorbewerkt in de regiekamer waarna de data kan worden doorgegeven aan het registratie component en uiteindelijk gebruikt kan worden.

CGI heeft zich voor deze PoC gefocust op de het verwerken van puntenwolken.

## Onderzoeksvraag

De scope van deze PoC is het maken van een verwerkingsmodule voor puntenwolken die afkomstig zijn uit het vergunningsverleningsproces. Het doel van deze module is het effectief kunnen werken met de puntenwolken bestanden om op een automatische manier oppervlaktes en volumes van ruimtes te kunnen bepalen. De focus ligt hierbij op het verwerken van puntenwolken van bestaande gebouwen met bettrekking tot onroerende zaken en het beoordelen van een aanvraag voor een bouwvergunning. Dit leidt tot de volgende onderzoeksvraag:

*“Is het mogelijk om uit een ingewonnen puntenwolk automatisch de (object)afmetingen van gebouwen te extraheren ten behoeve van het taxatieproces van de WOZ en/of het beoordelen van een bouwvergunning?”*

## Deep Dive

In de beginfase van het onderzoek is een deep-dive georganiseerd met een aantal stakeholders. Deze stakeholders zijn eigenaar van (een deel van) de use-cases zoals gespecificeerd in de uitvraag. De stake-holders gaven een gedetailleerd inzicht in de relatie van de use-cases met dit onderzoek.

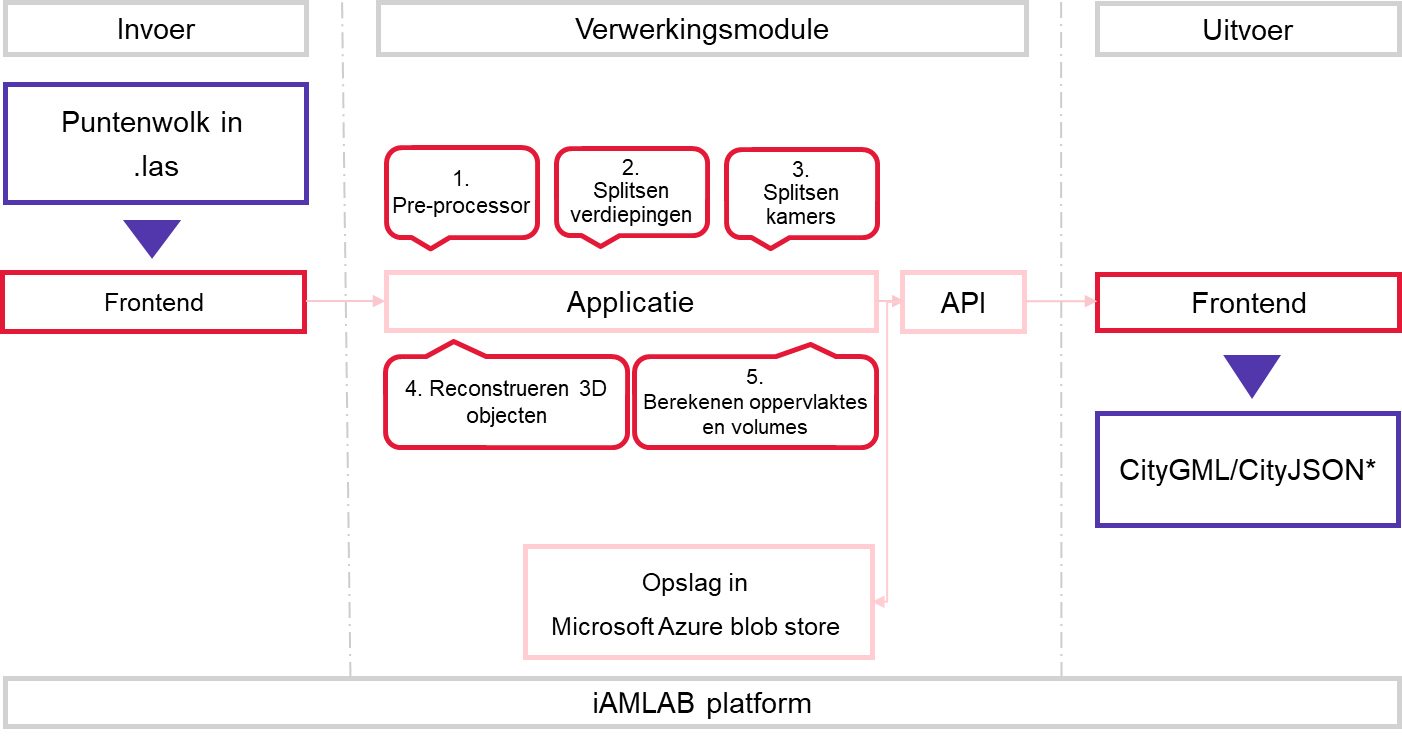
Vanuit de use-case WOZ kwam er veel detailinformatie over het inwinningsproces. De bestaande informatie vanuit de basisregistraties, bouwtekeningen en dossiers voeden het WOZ taxatie proces. Een puntenwolkscan is pas het laatste redmiddel. Die situatie komt niet veel voor want een WOZ ambtenaar mag niet zomaar een woning betreden om bijvoorbeeld een scan te maken, en als deze een woning betreedt neemt deze zelf de bestaande metingen mee en signaleert de veranderingen en meet deze handmatig bij. Een blinde vlek van de basisregistraties is de verandering die voortkomt uit vergunningvrij bouwen. Deze veranderingen worden niet gesignaleerd. Er werden veel creatieve ideeën besproken over mogelijke signalen voor vergunningvrije bouw zoals het kappen van een boom of de aanvraag van een afvalcontainer of zelfs het scrapen van Funda. Dit ligt buiten de scope van dit onderzoek.

Ook is er gesproken over vergunningen. Hierbij is er een duidelijk verschil tussen vergunningvrij en niet vergunningvrije wijzigingen. Het vergunningsproces begint bij de bouwaanvraag. Deze activiteit wordt dan via een lijst gecheckt en waar nodig verder doorgezet naar adviseurs zoals de brandweer. Hierop volgt dan een besluit tot vergunning. Het kan ook voorkomen dat er vergunningvrij kan worden gebouwd en dat de gemeente dus wel op de hoogte is gesteld, maar dat de wijziging niet wordt doorgevoerd.

Verder is er nog gesproken over de Samenhangende Objectenregistratie (SOR). Deze moet up-to-date worden gehouden met de laatste informatie en dan het liefst automatisch gevuld. Maar hoe bepaal je of deze automatisch doorvoert, en wie vertrouw je? Registratie op een volautomatische manier met feitelijke informatie kan hier helpen met een keuze te maken of om verschillen te signaleren.

## Overzicht oplossing

In de aanbieding heeft CGI een overzicht van de beoogde oplossing getoond. In Figuur 1 is de oplossing gevisualiseerd zoals deze daadwerkelijk is toegepast tijdens de PoC door het iAMLAB team. Hierin zijn de verschillende stappen van de applicatie zichtbaar. In hoofdstuk 2 worden de verschillende stappen in meer detail besproken. In hoofdstuk 5.4 wordt beschreven waarom bij bepaalde stappen afgeweken is van het originele plan van aanpak.

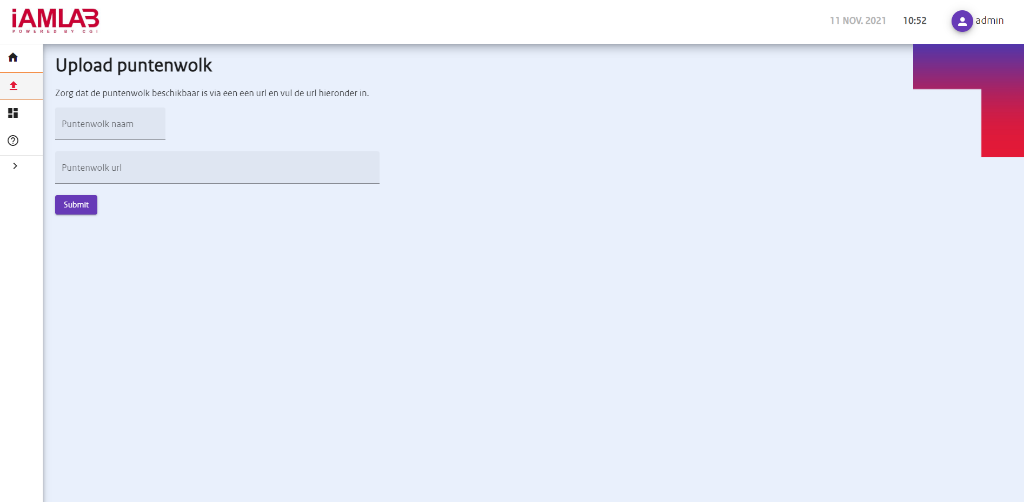
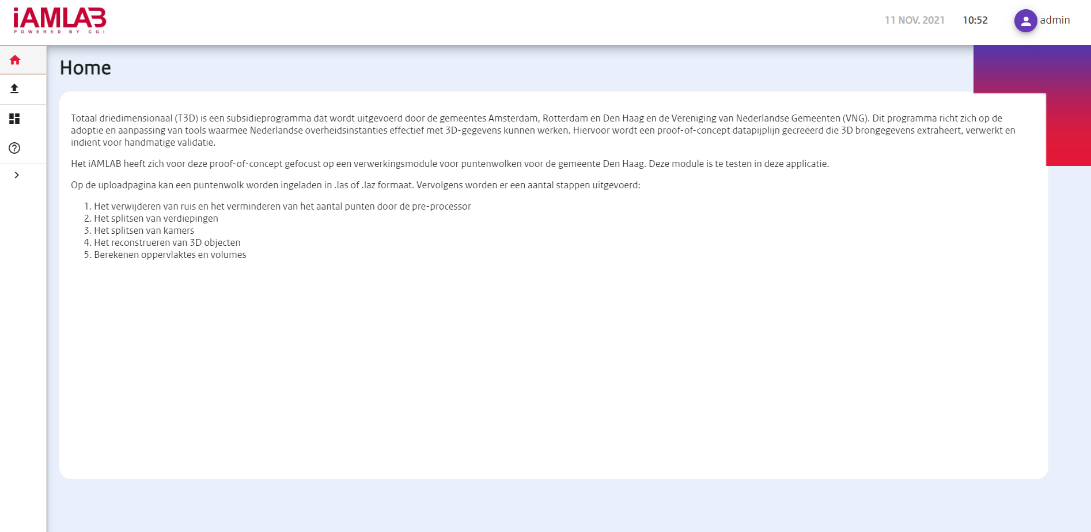
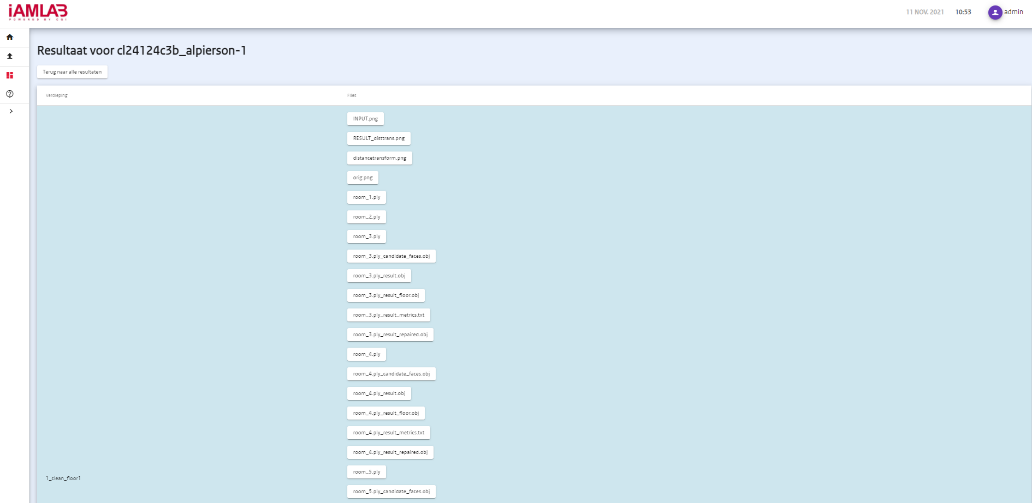
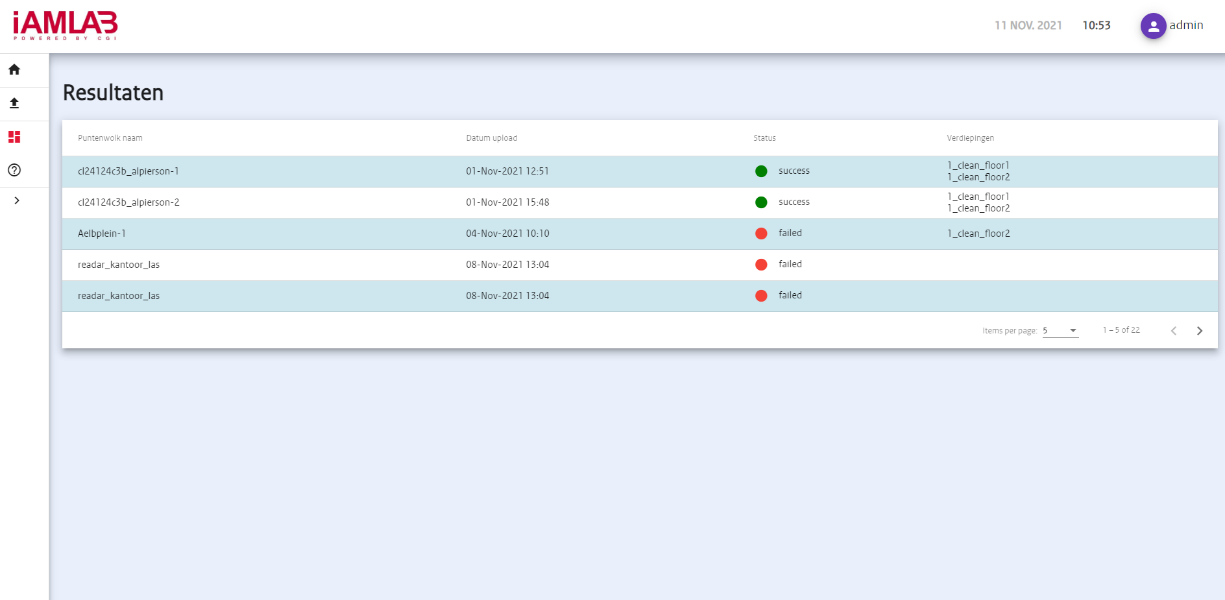


Figuur 1: Een overzicht van de oplossing.   
\*Op dit moment is de output nog niet in CityGML/CityJSON. Zie ook hoofdstuk 2.3.

## Gebruik applicatie

De ontwikkelde code is in een .zip bestand aan de gemeente Den Haag geleverd. Hier bevinden zich ook de Readme.md bestanden waarin wordt uitgelegd hoe de code moet worden gebruikt. Op deze manier kan de gemeente zelf de pipeline opzetten en aanpassingen doorvoeren aan de code.

CGI heeft daarnaast een frontend ontwikkeld waarmee op een laagdrempelige en gebruiksvriendelijke manier de pipeline automatisch wordt aangeroepen. Voorbeelden van deze frontend zijn weergegeven in Figuur 2. Het is in de frontend mogelijk om een URL in te geven die naar een puntenwolk bestand verwijst. De invoer eisen van een puntenwolk staan beschreven in hoofdstuk 2.1. Op de resultaten pagina worden verwerkte puntenwolken weergegeven en kunnen de tussen- en eindresultaten worden gedownload. In hoofdstuk 3.7 worden de output bestanden in detail beschreven.

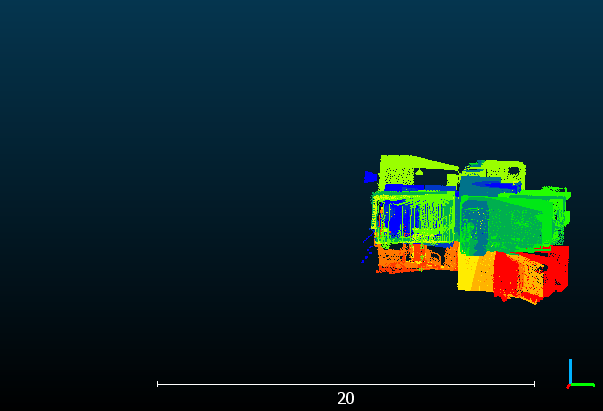
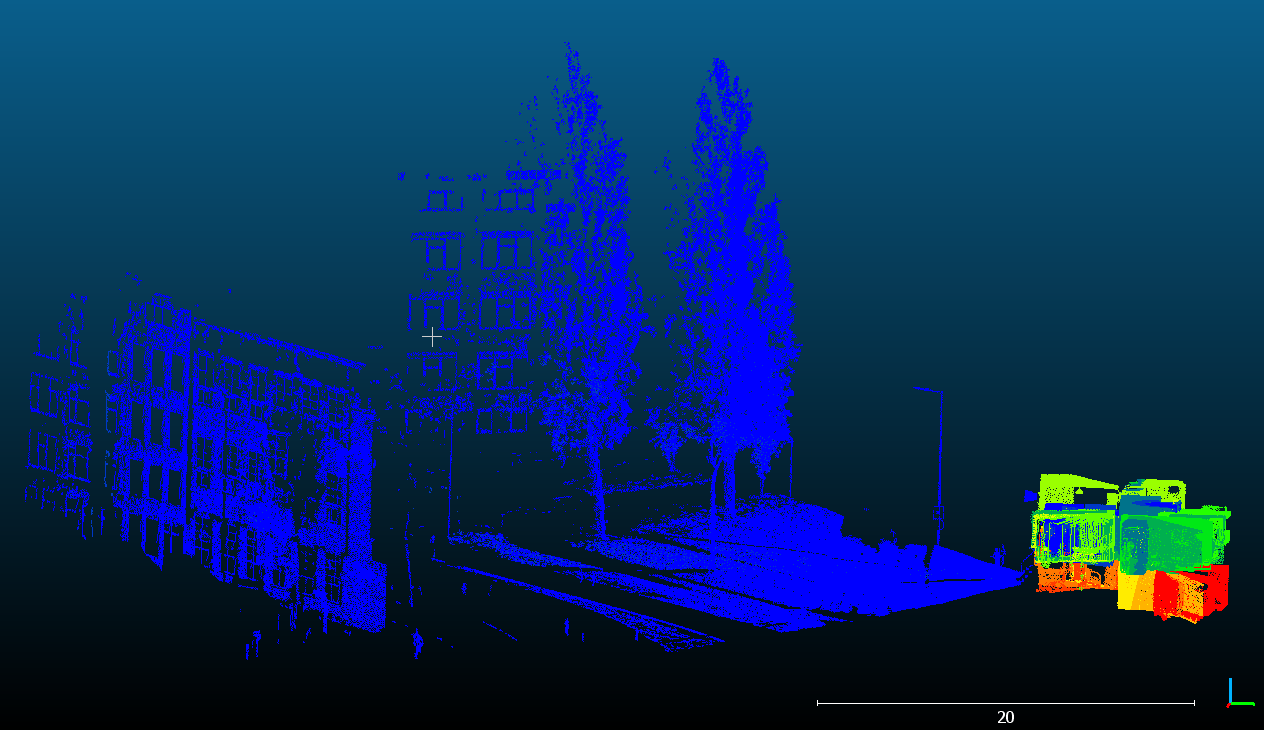


Figuur 2: Screen captures van de frontend met opeenvolgend het beginscherm, de upload-pagina, het resultaten overzicht en de gedetailleerde resultaten

# Methoden

## Invoer

De algoritmes zijn in eerste instantie ontwikkeld op een puntenwolk van een universiteit van Duitsland[[1]](#footnote-2) omdat er bij aanvang van het project nog geen puntenwolken beschikbaar waren vanuit de gemeente Den Haag. In een later stadium zijn puntenwolken van Woonstad aangeleverd en zijn de algoritmes doorontwikkeld en getest op deze puntenwolken. Zes van de acht puntenwolken van Woonstad bevatten punten van buiten, zoals bomen en andere gebouwen. Voor het ontwikkelen van de algoritmes is dit gebied er handmatig afgehaald. Dit geldt ook voor de resultaten van hoofdstuk 3.1 t/m 3.5. In Figuur 3 is gevisualiseerd welke punten van buiten er precies van een puntenwolk afgeknipt moeten worden zodat alleen de punten van de binnenruimtes overblijven. Een puntenwolk kan ook nog aan de binnenruimte zoveel mogelijk worden opgeschoond maar daar is tijdens de ontwikkeling geen handmatige actie voor ondernomen en dus niet strikt noodzakelijk.



Figuur 3: Voorbeeld van het verwijderen van punten van buiten

De aangeleverde puntenwolken van Woonstad zijn aangeleverd in .rcp files. Deze zijn eenmalig geconverteerd naar .e57 files via een proeflicentie van AutoDesk Recap (meer hierover in paragraaf 6.2.1). Vervolgens zijn deze files omgezet naar .las files met behulp van de open source tool CloudCompare. De aangeleverde en omgezette puntenwolken zijn in de blobstore in Microsoft Azure te vinden van de gemeente Den Haag.

In de oorspronkelijke uitvraag was het invoer bestandsformaat gedefinieerd als .las en/of .laz bestand. Om .laz bestanden te kunnen omzetten naar .las bestanden was een converter opgenomen in de aanbieding. Deze converter is niet meer nodig en daarom ook niet meer getoond in Figuur 1. Het is ook mogelijk om puntenwolken met andere extensies in te laden, zoals bijvoorbeeld een .ply bestand. Voor de volledige lijst van mogelijke invoermogelijkheden kan de CloudCompareWiki[[2]](#footnote-3) bekeken worden.

## Verwerkingsmodule

De verwerkingsmodule voert vijf stappen uit om de puntenwolk te verwerken tot een bestand dat de oppervlakte en de volumes van de verschillende ruimtes berekent.

### Pre-processor

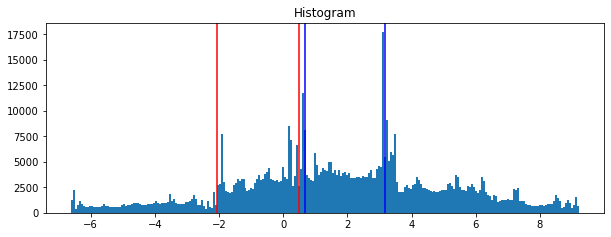
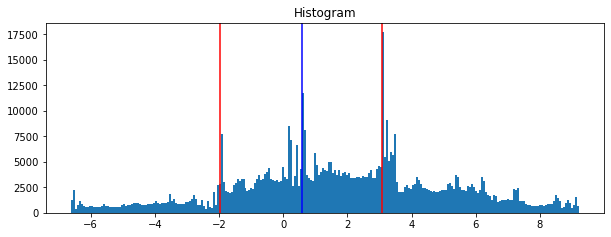
Om de gebruikte algoritmes te laten werken, is het nodig om het totaal aan punten in een puntenwolk te beperken tot de orde van grote rond de 100.000 punten. Dit wordt gedaan door in CloudCompare een subsample[[3]](#footnote-4) te nemen door de afstand tussen twee punten een minimale waarde mee te geven. Dit resulteert in een drastische reductie van het puntenaantal. Gemiddeld bevat een puntenwolk uit de set van Woonstad puntenwolken ongeveer 20 miljoen punten. De minimale afstand die gebruikt is in dit project is 0.05 meter. Hiermee komen de meeste puntenwolken van Woonstad op een aantal punten rond de 200.000. Dit is nog aan de ruime kant van wat er minimaal nodig is, maar de performance blijft hierdoor wel optimaal. Deze waarde van 0.05 meter is in de broncode aan te passen.

Het Statistical Outlier Removal (SOR)[[4]](#footnote-5) filter van CloudCompare wordt gebruikt om ruis te verwijderen uit de puntenwolk. Hierbij wordt voor elk punt uit de puntenwolk bekeken wat de gemiddelde afstand is tot een aantal dichtstbijzijnde andere punten. Punten waarbij de gemiddelde waarde groter is dan het gemiddelde plus een aantal keer de standaarddeviaties worden vervolgens verwijderd uit de puntenwolk. Het aantal punten dat gebruikt wordt om de gemiddelde afstand te berekenen tot een punt en het aantal keer standaarddeviatie dat wordt gebruikt om de maximale gemiddelde afstand voor een punt te bepalen zijn parameters die kunnen worden ingesteld. De parameters die in dit project gebruikt zijn, zijn de default waardes van de CloudCompare software en dat is 6 voor het aantal naburige punten en een standaarddeviatie van 1. Deze parameters zijn in de source code aan te passen.

### Splitsen verdiepingen

Voor het herkennen van kamers en het construeren van 3D objecten van een gebouw is het handig om deze per verdieping te kunnen verwerken. Ook is het daarna gemakkelijk om te tellen hoeveel verdiepingen een gebouw bevat. Om de verdiepingen van een puntenwolk te kunnen splitsen, is er gebruik gemaakt van density histograms (zie Figuur 4). De puntenwolk wordt als het ware platgeslagen tot alleen de hoogtewaarde van de punten in de ruimte. Door te tellen hoe vaak een punt zich op een bepaalde hoogte bevindt kan de relatieve dichtheid van punten worden bepaald in de hoogte as. Door de waarden te groeperen in een histogram worden duidelijk de pieken zichtbaar waar zich horizontale vlakken (dus vloeren en plafonds, maar ook tafels) zich bevinden. De pieken uit het histogram geven hierbij suggesties voor de waarden waar vloeren en plafonds kunnen zijn. Als er een oneven aantal pieken is worden deze bewerkt zodat er een even aantal verdiepingen is. Bij drie pieken wordt de middelste bij beide verdiepingen meegenomen en bij vijf pieken worden de tweede en vierde vervangen door de middelste. Hierdoor gaat er niks van de puntenwolk onnodig verloren. Als er meer dan 3 verdiepingen zijn die een oneven aantal pieken veroorzaakt (bijvoorbeeld door schuine daken of splitlevels) dan kan het algoritme dit (nog) niet verwerken.

Vervolgens wordt een buffer van 10 centimeter genomen boven de plafondpiek en 10 centimeter onder de vloerpiek. Ook wordt er gecheckt of tussen het plafond en vloer minimaal 2.1 meter zit. Tenslotte wordt de puntenwolk opgeknipt aan de hand van deze plafond en vloer hoogtes. Door de buffer kunnen er punten dubbel in de gesplitte puntenwolken voorkomen maar deze worden in de volgende stappen weer ongedaan gemaakt.



Figuur 4: Density histograms gebruikt voor het splitsen van de verdiepingen. Boven de originele piek detectie, onder de splitsing in verdieping (rood is vloer, blauw is plafond)

### Splitsen kamers

Om de verschillende kamers van één verdieping te onderscheiden wordt gebruikt van een distance transform algoritme[[5]](#footnote-6). Dit algoritme wordt uitgevoerd op een 2D doorsnede van de puntenwolk. Deze 2D doorsnede wordt gemaakt door een plakje uit de puntenwolk te knippen die 10 centimeter onder het plafond begint (om zoveel mogelijk de muur boven deuren mee te nemen) en 20 centimeter dik is. Daarna worden de punten platgeslagen om tot de 2D weergave te komen. Hiermee worden verschillende morfologische operaties[[6]](#footnote-7) gedaan om de punten te verdikken en aan te laten sluiten om vervolgens lijnen te herkennen. Met deze 2D zwart/wit lijnenplaat wordt vervolgens het distance transform algoritme uitgevoerd. Dit algoritme herkent voor ieder punt in de ruimte wat de afstand is tot een dichtstbijzijnde muur. Als deze afstand groter is, krijgt dat punt een hogere waarde, waardoor middelpunten van ruimten hoge waarden toegewezen krijgen en muren lage waarden. Tenslotte worden door een watershed algoritme[[7]](#footnote-8) de ruimtes vanuit de middelpunten opgevuld tot de muren om de ruimtes te onderscheiden. De verschillende tussenstappen worden ook uitgeschreven in de pipeline om tussentijds te kunnen analyseren hoe de resultaten zijn. Deze zijn beschreven in paragraaf 3.3. Het resultaat is een 2D kleurenplaat van de verschillende kamers waarmee, door de x,y coördinaten terug te geven aan de puntenwolk, de puntenwolk in kamers verdeeld wordt en als input gebruikt wordt voor de volgende stap.

### Reconstrueren 3D objecten

Voor het reconstrueren van 3D ruimtes uit de puntenwolk wordt gebruik gemaakt van de Computational Geometry Algorithms Library (CGAL). Dit is een open source softwarepakket geschreven in C++ dat verschillende geometrische algoritmes bevat. Voor de 3D reconstructie van de puntenwolk is gebruik gemaakt van het Polygonal Surface Reconstruction (PSR) algoritme[[8]](#footnote-9).

Het PSR algoritme is oorspronkelijk ontwikkeld voor puntenwolken van de buitenkant van gebouwen. Echter, omdat de verwerking per individuele kamer van de binnenkant van het gebouw wordt gedaan, zijn de resultaten van het algoritme goed te gebruiken. Het PSR-algoritme detecteert eerst de vlakken in een puntenwolk. Daarna worden de vlakken doorgetrokken om zo mogelijke vlakken van het object te vinden. Hierna wordt een berekening toegepast om de meest waarschijnlijke vlakken over te houden door rekening te houden met de vlakken in relatie tot de punten die in dit vlak liggen. Hierbij moeten objecten helemaal omringt worden door vlakken en een “waterdicht” geheel vormen. Een bijkomend voordeel van dit algoritme is dat vlakken die geen deel uitmaken van de structuur van het huis (zoals meubilair) wegvallen uit het resultaat omdat deze niet een sluitend geheel kunnen vormen met de rest van de ruimte. Wel kan het zijn dat bedden en banken het zicht tot muren en vloeren belemmeren en daarmee een deel van de inhoud van de puntenwolk wegneemt.

Er kunnen verschillende methoden toegepast worden om de berekeningen uit te voeren; dit zijn zogenaamde solvers. Het PSR algoritme heeft twee solvers geïmplementeerd, namelijk SCIP[[9]](#footnote-10) en GLPK[[10]](#footnote-11). Deze zijn beide getest voor de ontwikkelde applicatie en hebben verschillende performance maar ook restricties in licenties. Meer hierover in hoofdstuk 5.2.4.

### Berekenen oppervlaktes en volumes

Het resultaat het PSR algoritme bestaat uit verschillende meshes als .obj bestand. Van elke losse mesh, dus een losse kamer, wordt het volume en het vloeroppervlak berekend met behulp van meshlab[[11]](#footnote-12). Om dit te doen wordt eerst de mesh sluitend gemaakt om te zorgen dat het volume bepaald kan worden. Vervolgens wordt een her-calculatie gedaan van alle vlakken van de mesh om te zorgen dat deze naar buiten georiënteerd zijn. Hiermee kunnen de ‘normaalvectoren’, een vector loodrecht op een vlak, bepaald worden. Alle vlakken met normalen die naar beneden wijzen worden geselecteerd om alleen de vlakken van de vloer te krijgen, waarmee tenslotte het oppervlak wordt berekend.

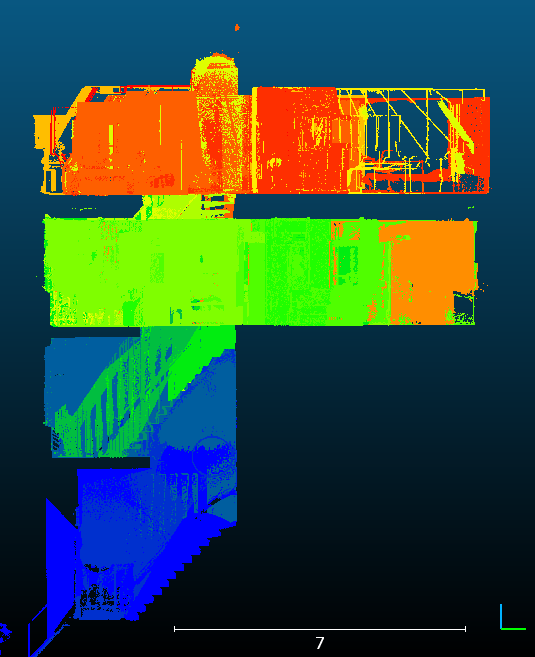
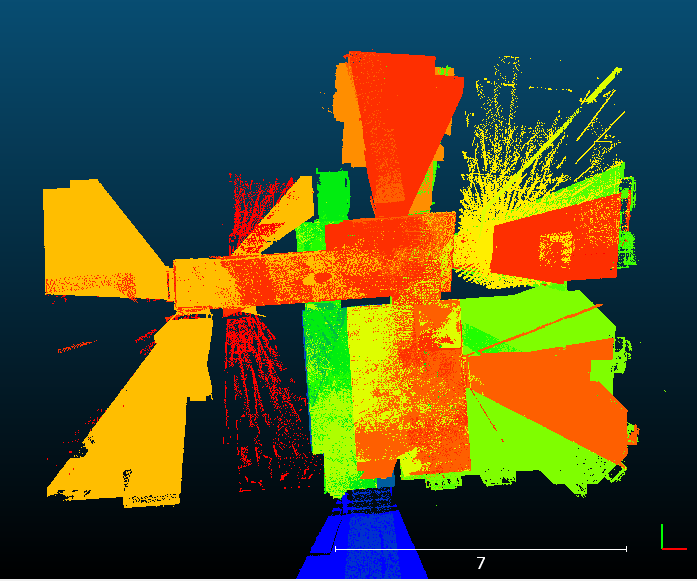
## Uitvoer

In de oorspronkelijke uitvraag werd als output file gevraagd om een CityGML of een CityJSON bestand.

De resultaten komen nu voort uit het PSR-algoritme en zijn .obj bestanden. Dit zijn losse meshes per kamer en worden gebruikt om de oppervlakte en het volume te berekenen. In de huidige status van dit rapport (v1.0) worden deze losse kamers samengevoegd in één mesh en opgeleverd als obj bestand. Daarnaast komt er een json bestand uit die de oppervlaktes en volumes per kamer omschrijft.

# Resultaten

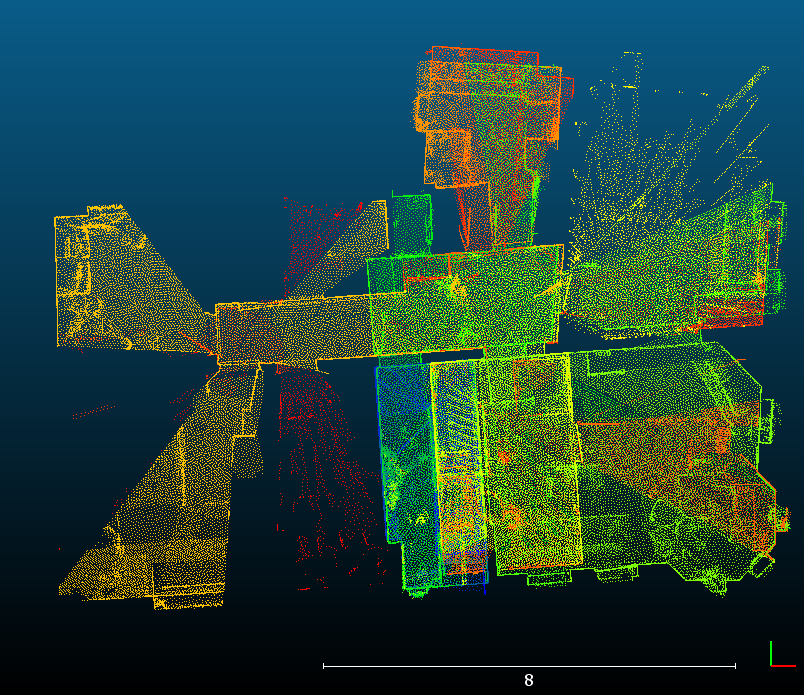
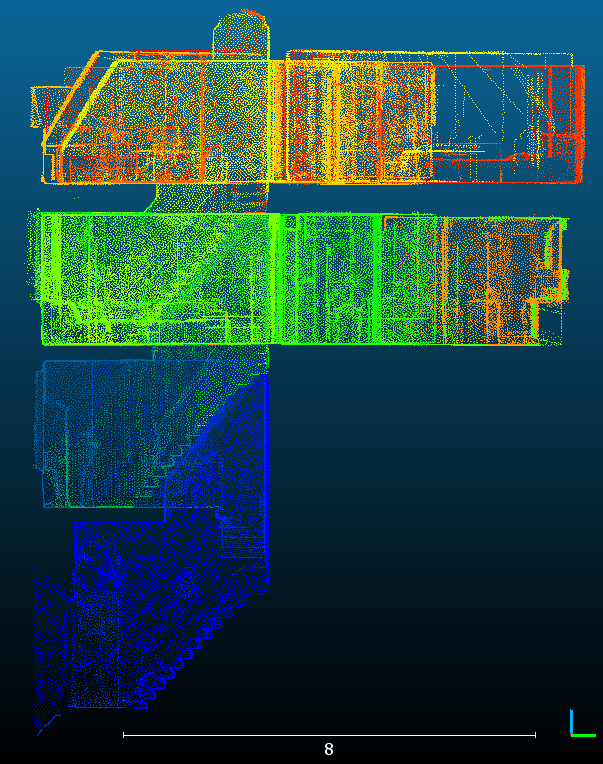
In dit hoofdstuk worden de resultaten van de verschillende stappen getoond voor een aangeleverde puntenwolk van Woonstad, namelijk *cl24124f1a\_alpierson-1.las*. Voor de ontwikkeling van het algoritme en het tonen van de resultaten zijn de punten van buiten de woning *handmatig* verwijderd. In hoofdstuk 3.8 worden de resultaten besproken van de aangeleverde puntenwolken waarbij er geen handmatige stappen vooraf zijn gedaan. In Figuur 5 is de oorspronkelijke puntenwolk zonder de ruis van buiten het pand weergegeven.



Figuur 5: De originele puntenwolk cl24124f1a\_alpierson-1.las zoals aangeleverd door Woonstad zonder het buitengebied. Links is een zijaanzicht te zien, rechts een bovenaanzicht.

## Pre-processor

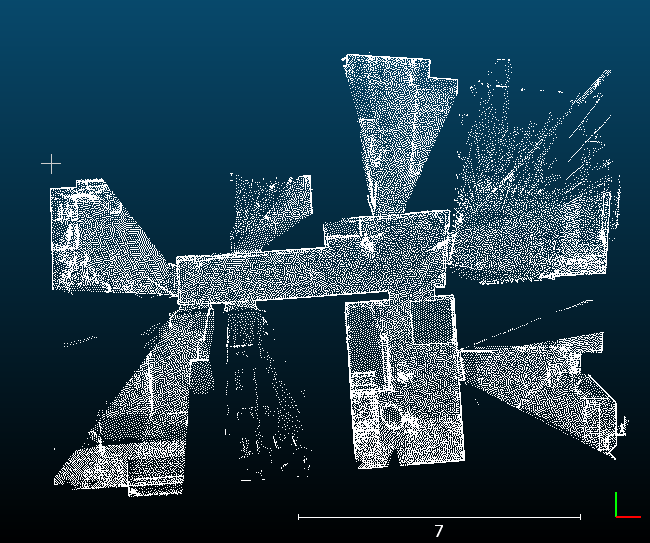
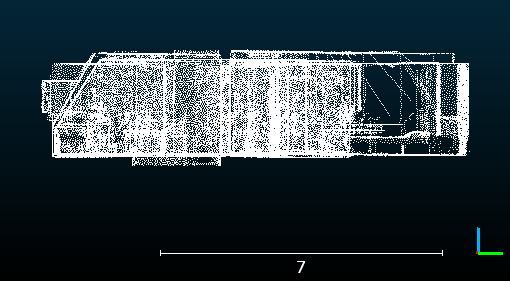
In Figuur 6 wordt het resultaat weergegeven van de pre-processor. Er is duidelijk te zien dat de puntenwolk veel meer minder punten bevat. Daarnaast is er vooral in het bovenaanzicht te zien dat er ook punten zijn weg gefilterd in gebieden waar weinig punten bij elkaar liggen.



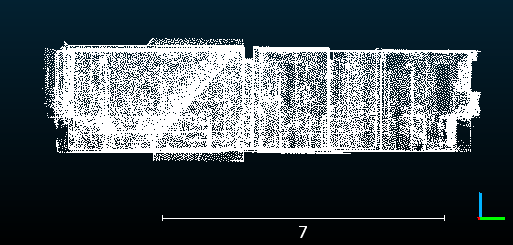
Figuur 6: Resultaat van de pre-processor voor puntenwolk cl24124f1a\_alpierson-1.las. Links is een zijaanzicht te zien, rechts een bovenaanzicht.

## Splitsen verdiepingen

De resultaten van het splitsen van de puntenwolk in verschillende verdiepingen zijn gevisualiseerd in Figuur 7 en Figuur 8.



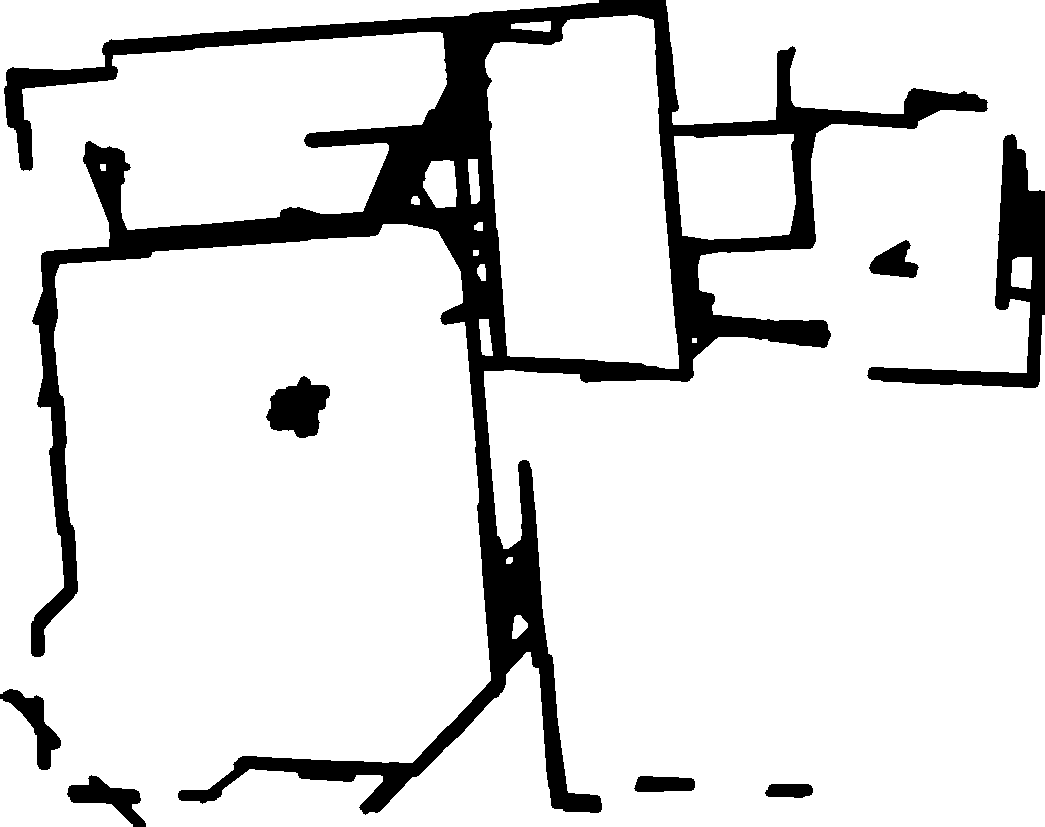
Figuur 7: Resultaat van het splitsen van de verdiepingen voor puntenwolk cl24124f1a\_alpierson-1.las. Links is een zijaanzicht te zien, rechts een bovenaanzicht.



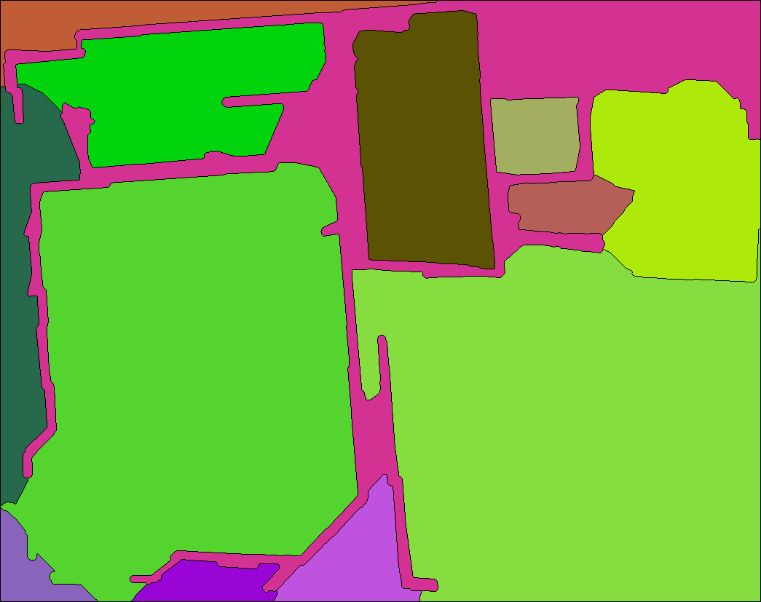
Figuur 8: Resultaat van het splitsen van de verdiepingen voor puntenwolk cl24124f1a\_alpierson-1.las. Links is een zijaanzicht te zien, rechts een bovenaanzicht

## Splitsen kamers

In Figuur 9 en Figuur 10 zijn stapsgewijs de resultaten van de kamerdetectie zichtbaar. Allereerst de doorsnede van de puntenwolk in ruwe punten. Deze doorsnede is van 10 centimeter onder het plafond met 20 centimeter dikte. Daarna het resultaat van de morfologische operaties om de punten sluitend te maken en tot lijnen te detecteren. Ook zijn de kleuren omgekeerd als juiste input voor het distance transform algoritme. Het distance transform resultaat is gevisualiseerd waarin duidelijk de pieken van plekken te zien zijn die verder van een muur vandaan zijn. Tenslotte is het laatste beeld van de gekleurde 2D visualisatie van de kamers.

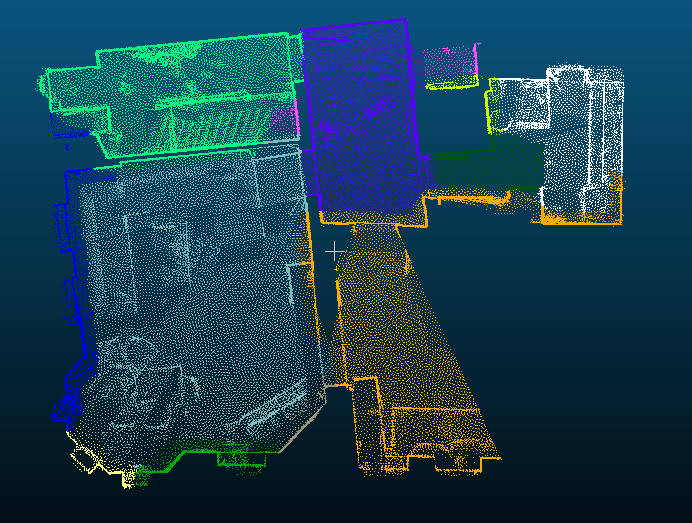


Figuur 9: De eerste twee stappen in het splitsen van de kamers voor puntenwolk cl24124f1a\_alpierson-1.las.   
Links: Doorsnede van de puntenwolk. Rechts: Resultaat na morfologische operaties.



Figuur 10: Stap 3 en 4 voor het splitsen van de kamers voor puntenwolk cl24124f1a\_alpierson-1.las.   
Links: Resultaat van het distance transform algoritme. Rechts: Eindresultaat van de gesplitste kamers per kleur.

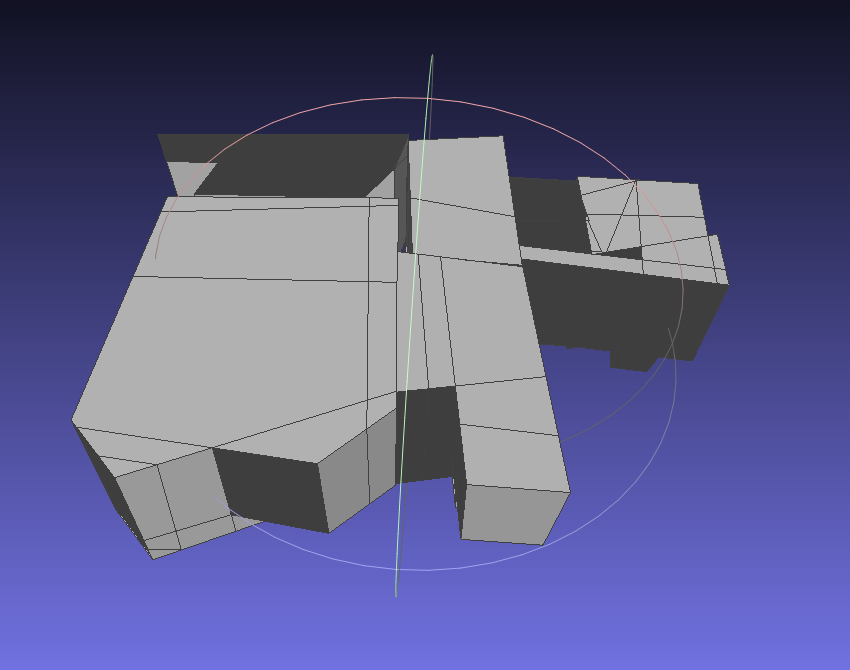
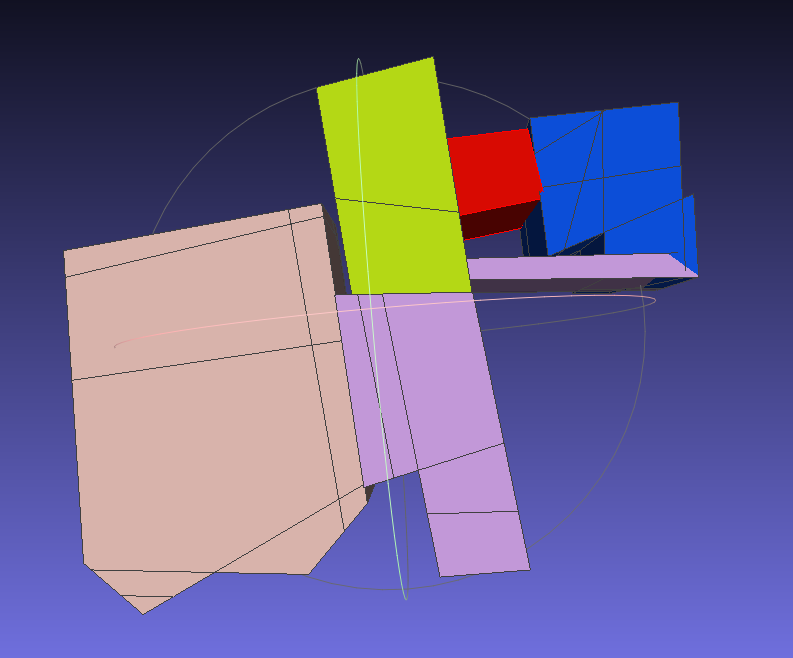
Als resultaat van deze stap horen ook de puntenwolken die zijn opgesplitst in de verschillende kamers. Elke kamer uit het 2D krijgt een extra buffer om te zorgen dat de muren volledig meegenomen worden. Met deze x,y coördinaten wordt de puntenwolk vervolgens opgeknipt. Dit betekent dat een punt uit de puntenwolk in meerdere kamers kan zitten, omdat het bijvoorbeeld de tussenmuur is. De visualisatie als opgesplitste puntenwolken is te vinden in Figuur 11, waarin dus sommige punten één kleur hebben gekregen terwijl ze er eigenlijk meerdere moeten hebben.



Figuur 11: Het resultaat van het splitsen van de kamers voor puntenwolk cl24124f1a\_alpierson-1.las.

## Reconstructie 3D objecten

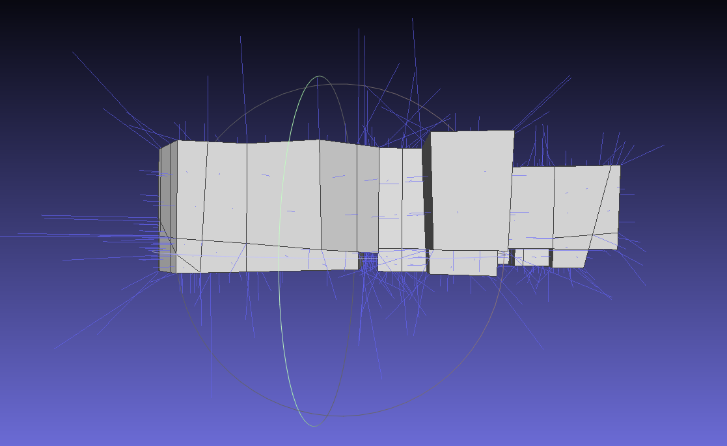
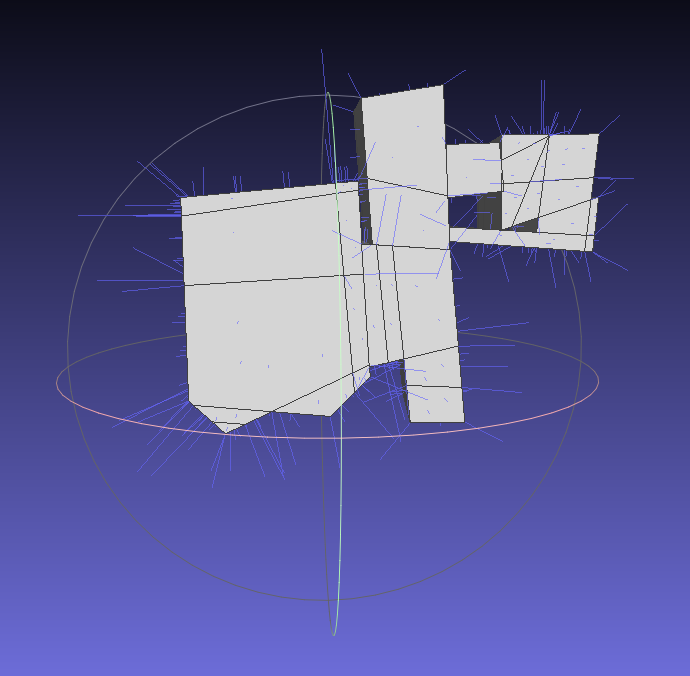
De resultaten van het PSR-algoritme zijn weergegeven in Figuur 12. Hierin is zichtbaar dat niet elk stuk van de puntenwolk leidt tot een 3D object als er geen sluitend geheel kan worden gevonden. Dit klopt in het geval van de stukken aan de randen maar daardoor wordt ook niet het trappenhuis gevonden.



Figuur 12: Resultaat voor puntenwolk cl24124f1a\_alpierson-1.las van het CGAL PSR algoritme voor alle kamers. Rechts zijn handmatig de kleuren gevisualiseerd.

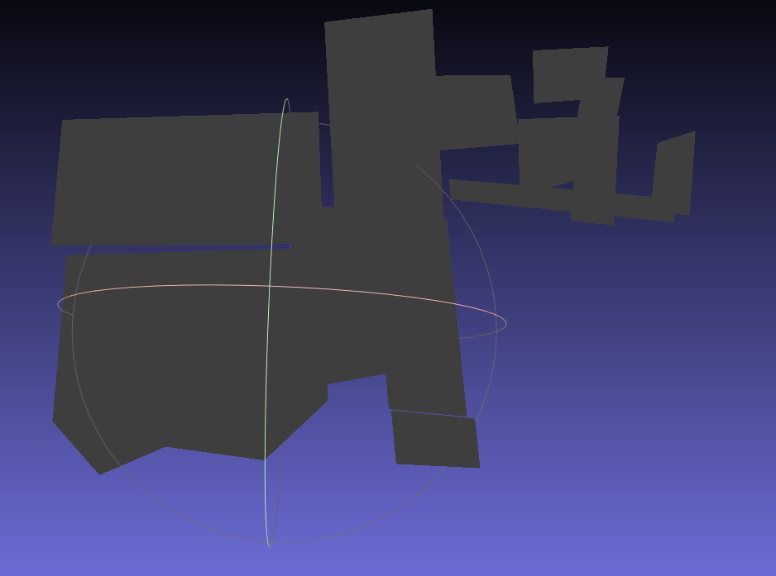
## Berekenen oppervlaktes en volumes

In Figuur 13 is weergegeven hoe de normaal vectoren eruit zien van de 3D objecten als deze gerepareerd en opnieuw berekend zijn. De normalen zijn gevisualiseerd als blauwe lijnen en geven de richting aan van de vlakken.

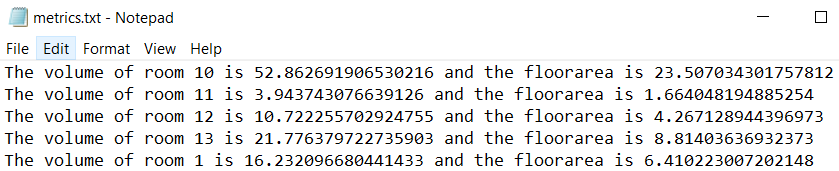


Figuur 13: De normaalvectoren (in blauw) zijn zichtbaar op de gerepareerde meshes van puntenwolk cl24124f1a\_alpierson-1.las.

De vloervlakken worden bepaald door de vlakken te selecteren die voldoen aan een normaalvector die naar beneden wijst. Deze vlakken zijn namelijk aan de onderkant van het 3D object. Het resultaat daarvan is weergegeven in Figuur 14. De oppervlakte van deze vlakken is daarna met behulp van MeshLab berekend en wordt uitgeschreven een tekst bestand zoals te zien in Figuur 15.



Figuur 14: Vloeroppervlak selectie van vlakken met normaal naar beneden van puntenwolk cl24124f1a\_alpierson-1.las.

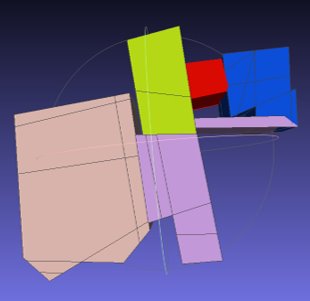
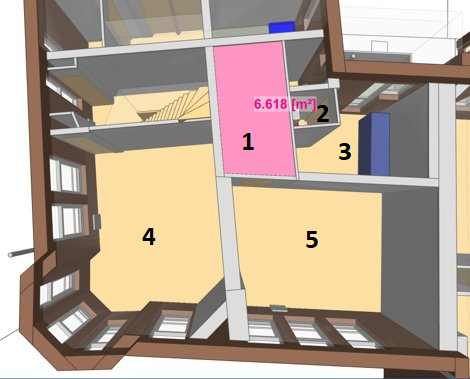


Figuur 15: Resultaten meetwaardes in tekstbestand voor puntenwolk cl24124f1a\_alpierson-1.las.

Tenslotte worden de verschillende obj files als één bestand aan elkaar geplakt zodat deze in één keer gevisualiseerd kan worden.

## Oppervlakte vergelijking met BIM

Om enig inzicht te krijgen in de kwaliteit van het resultaat van de voorbeeld puntenwolk, worden de oppervlakte waardes vergeleken met die vanuit de BIM.



Figuur 16: Oppervlakte vergelijking van gereconstrueerde kamers uit de puntenwolk met BIM

In onderstaande tabel is dit kwantitatief weergeven. Hieraan moet opgemerkt worden dat kamer 4 en 5 in de BIM één kamer is van 44.62 m2. Daarom is handmatig opgemeten dat kamer 5 circa 4 bij 5 meter groot is en dus 20 m2. Deze wordt flink onderschat uit de puntenwolk omdat deze kamer niet compleet was ingemeten. Kamer 4 komt daardoor uit op circa 24.62m2, wat vergeleken wordt met 23.51m2 uit de puntenwolk analyse.

|  |  |  |
| --- | --- | --- |
| **Kamer​** | **Oppervlakte​**  **Puntenwolk (m2) ​** | **Oppervlakte BIM (m2)​** |
| **1**​ | 6.41 ​ | 6.62​ |
| **2**​ | 1.66​ | 1.34​ |
| **3**​ | 6.41​ | 7.39​ |
| **4**​ | 23.51​ | ~ 44.62 – (5 \* 4) =​ ~ 24.62​ |
| **5**​ | 8.81​ | ~ (5 \* 4) ​ |

## Resultaten overzicht bestandsnamen

In onderstaande tabel is een overzicht weergegeven van de (tussen)resultaten die uit de verwerkingsmodule komen.

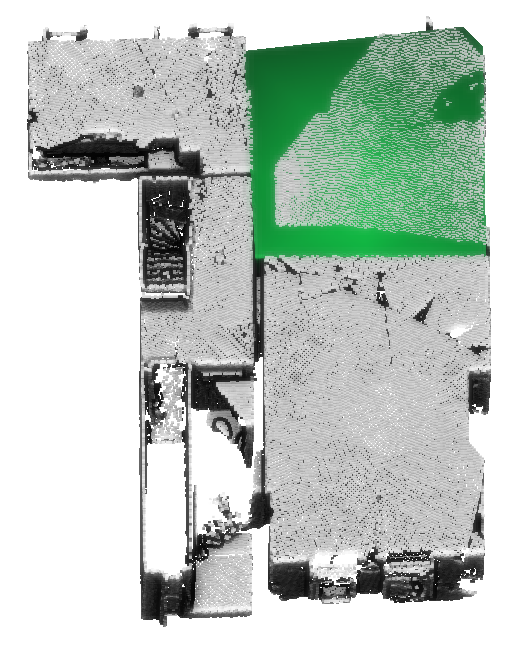
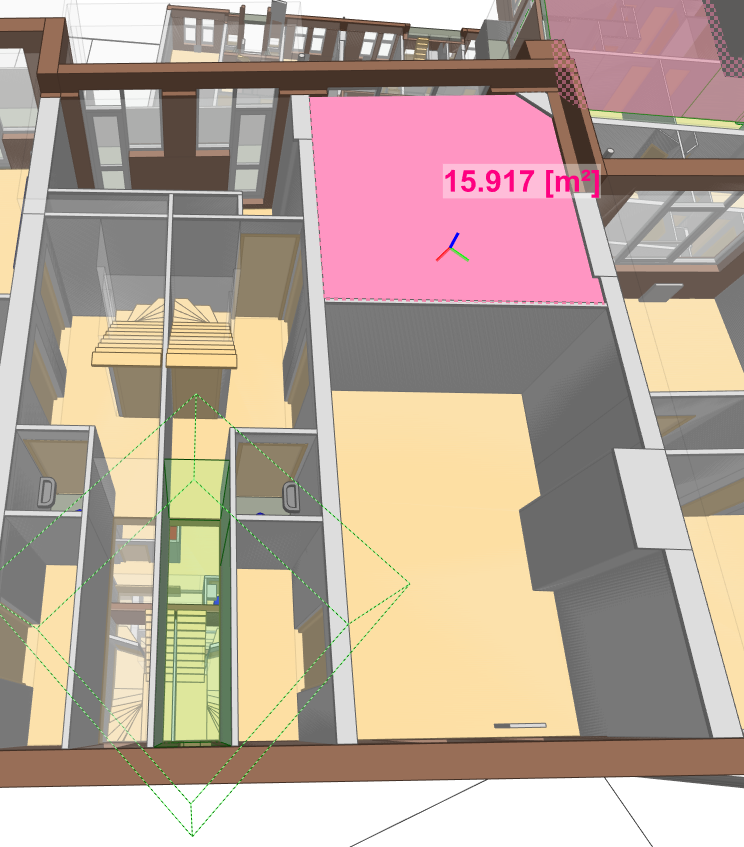
|  |  |  |
| --- | --- | --- |
| Stap | Outputbestandsnaam | Omschrijving |
| Input | Inputbestandsnaam |  |
| Preprocessor | Inputbestand+output.ply | Resultaat van CloudCompare preprocessing |
| Verdiepingen splitsen | Inputbestand+output\_floor\*.ply | Puntenwolk opgesplitst in meerdere verdiepingen. |
| Kamer splitsen | Elke verdieping krijgt een apart map met output: |  |
| orig.png | Slice van puntenwolk naar 2D weergave |
| INPUT.png | 2D weergave na morfologische operaties |
| distancetransform.png | 2D weergave distance transform |
| RESULT\_disttrans.png | 2D weergave kamer splitsing |
| slice.ply | Slice van puntenwolk |
| room\*.ply | Puntenwolk gesplitst in kamers |
| Kamerreconstructie | room\_\*\_out\_result.obj | 3D reconstructie mesh per kamer |
| Oppervlakte berekening | room\_\*\_out\_result\_repaired.obj | 3D reconstructie mesh per kamer na reparatie en her-calculatie van oriëntatie |
| room\_\*\_out\_result\_floor.obj | 3D reconstructie mesh van vlakken in de vloer per kamer |
| metrics.txt | Volumes en oppervlaktes per kamer uitgeschreven in een .txt bestand |
| Samenvoegen kamer objecten | roomsmerge.obj | 3D reconstructie mesh van alle kamers gecombineerd in één bestand |

## Resultaten resterende Woonstad puntenwolken

De twee puntenwolken van Woonstad zonder punten van buiten het pand zijn door de pipeline gehaald. De resultaten staan in tabel 1. Hiervan zijn handmatig een aantal gegevens uit de puntenwolk en het BIM-model gehaald om te vergelijken. Dit is ook weergegeven in Figuur 17.

Tabel 1: De resultaten van de Woonstad puntenwolken zonder buiten punten vergeleken met het BIM-model.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Puntenwolknaam | Aantal ruimten in puntenwolk (Handmatig bekeken) | Aantal succesvol gevonden ruimten | Oppervlakte en inhoud | BIM-model |
| cl24124c3b\_alpierson-1.las | Verdieping1: 4  Verdieping2: 4 | Verdieping1: 4  Verdieping2: 0 | Kamer1: 7.95m2  en 22.45 m3  Kamer2: 16.62 m2 en 36.7 m3  Kamer3: 0.58 m2 en 1.03 m3  Kamer4: 2.85 m2 en 6.18 m3 | Kamer1: 7.86 m2 en 20.82 m3  Kamer2: 15.9 m2 en 41 m3  Kamer3: 0.38 m2 en 1.02 m3  Kamer4: 2.6 m2 en 6.81 m3 |
| cl24124i2c\_albplein-1.las | Verdieping1: 3  Verdieping2: 5 | Verdieping1: 1  Verdieping2: 1 | Kamer 1: 4.09 m2 en 11.88 m3  Kamer 1 (2e verdieping): 18.89m2 en 46.12m3 | Kamer 1: 3.14m2 en 8.49m3  Kamer 1: (2e verdieping): 19.31m2 en 39.24 m3 |



Figuur 17: Vergelijking van oppervlakte van “verdieping 1: kamer 2” uit puntenwolk “cl24124c3b\_alpierson-1.las met de oppervlakte van het BIM-model.   
Links oppervlakte uit het BIM-model, rechts oppervlakte vanuit de puntenwolk. De waarde van het vloeroppervlak van de groene kamer is 16.62m2.

# Conclusie

De onderzoeksvraag van deze PoC was:

*“Is het mogelijk om uit een ingewonnen puntenwolk automatisch de (object)afmetingen van gebouwen te extraheren ten behoeve van het taxatieproces van de WOZ en/of het beoordelen van een bouwvergunning?”*

Om deze onderzoeksvraag te beantwoorden is een pipeline ontwikkeld die een puntenwolk in ongeveer 20 minuten automatisch verwerkt tot ruimten waar oppervlaktes en volumes van worden bepaald. De applicatie is modulair opgebouwd waardoor deze relatief eenvoudig uitgebreid kan worden met nieuwe functionaliteiten. In hoofdstuk 3.8 is te zien dat de algoritmes van de applicatie de oppervlaktes vrij nauwkeurig bepalen. Echter, de algoritmes detecteren niet altijd alle ruimtes.

Voor het extraheren van (object)afmetingen van gebouwen ten behoeve van het taxatieproces van de WOZ dient de gemeente Den Haag op basis van de huidige puntenwolkkwaliteit een paar optimalisaties aan te brengen voor een verbetering van het resultaat. Dit zijn het gebruik van een mobiele LiDAR scanner waardoor looproutes beschikbaar komen en het opstellen van een inmeetinstructie met aandacht voor het inscannen van hele kamers (zie ook hoofdstuk 5.1).

Tijdens de Deep Dive is naar voren gekomen dat de puntenwolk verwerking in eerste instantie niet direct bruikbaar zal zijn voor het beoordelen van een bouwvergunning.

# Lessons Learned

## Inwinning puntenwolken

Er is een aantal bevindingen gedaan gerelateerd aan de inwinning van puntenwolken die voor betere kwaliteit van verwerking zorgen. Als gevolg hiervan zal de ruimtedetectie en oppervlakte bepaling nauwkeuriger en met meer zekerheid werken.

* Scannen van hele kamers

De puntenwolken van Woonstad Rotterdam zijn vanaf vaste plekken ingewonnen. Hierbij zijn af en toe delen van kamers niet gescand. Hierdoor is er veel moeite gestoken om te zorgen dat het algoritme in staat is om de afstand van bijvoorbeeld een muur te overbruggen. Dit zorgt echter voor veel aannames en alsnog zal het algoritme niet altijd slagen volledige output te genereren. Dit compliceert het reconstrueren van het 3d object zodat van deze ruimtes geen betrouwbare oppervlaktes kunnen worden bepaald.

* Scannen met zo weinig mogelijk ruis

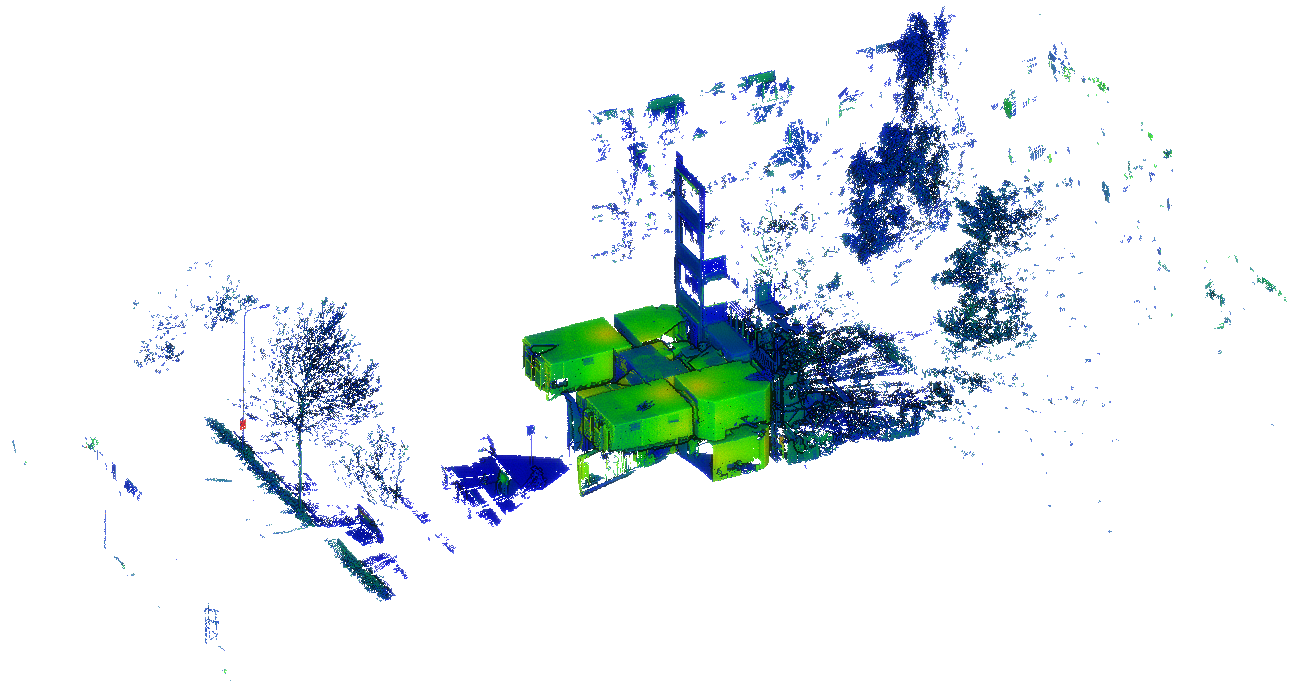
De puntenwolken van Woonstad bevatten ook punten van een deel dat zich buiten de woning bevindt. Het is lastig om deze punten er automatisch uit te filteren en dat wordt in de huidige applicatie dan ook met de hand gedaan. Idealiter zou je deze punten dus automatisch detecteren en wegfilteren. In Figuur 18 staat een voorbeeld van een puntenwolk met veel buitengebied. Deze puntenwolk is nu moeilijk automatisch te verwerken. Daarnaast is ook ruis in de binnenruimte beperkend, bijvoorbeeld dat een lamp in de 2D doorsnede wordt meegenomen of omdat de schaduw van een bed of bureau zorgt dat een deel van de vloer mist.

* Scannen met een mobiele scanner

De puntenwolken van Woonstad zijn ingewonnen met behulp van statische scanners waardoor er geen looproute bekend is. Wanneer er wel looproutes beschikbaar zijn, kunnen er algoritmes ontwikkeld worden die het detecteren van ruimtes en het herkennen van deuren kunnen verbeteren. CGI heeft hier ervaring mee opgedaan in eerdere onderzoeken.

* Georeferentie van puntenwolk

De puntenwolken van woonstad waren niet gegeorefereerd. Dit maakt de vergelijking van de oppervlaktes uit de puntenwolk met de BIM erg moeilijk omdat ze niet aan elkaar te relateren vallen. In dit project is dat opgelost door een handmatige check maar zeker in het vervolg is dit wel noodzakelijk. Ook leidt een puntenwolk zonder georeferentie tot objecten die lokaal georiënteerd zijn. Dit maakt ook de omzetting naar CityGML/CityJSON onmogelijk.



Figuur 18: Voorbeeld van een puntenwolk met veel buitengebied

## Verwerking puntenwolken

### Preprocessing

Het filteren van ruis en het verminderen van het aantal punten in de puntenwolk zorgt ervoor dat er betere resultaten kunnen worden behaald en dat dit ook sneller gebeurt. De filtering zou nog wel verbeterd kunnen worden. Zo wordt de ruis die ontstaat naar buiten toe door ramen niet altijd helemaal weg gefilterd en zou het filteren van objecten (zoals een bed en een kast) nog van meerwaarde kunnen zijn.

### Splitsen verdiepingen

Voor het splitsen van de verdiepingen wordt een methode gebruikt om de pieken van het 2D density histogram te bepalen. Verschillende parameters bepalen o.a. vanaf wanneer een piek gedetecteerd wordt. De waarden hiervan zijn op dit moment aangepast aan de Woonstad puntenwolken. Deze verschillen heel erg aan de hand van de puntendichtheid van de puntenwolk en de gekozen hoeveelheid bins van het density histogram.

Daarnaast zijn er verschillende scenario’s gevonden die het splitsen van verdiepingen bemoeilijken. Als niet alle plafonds of alle vloeren op dezelfde hoogte liggen, kunnen er te weinig pieken of te veel pieken ontstaan. Ook door schuine daken wordt het moeilijker om de exacte juiste pieken te vinden omdat het signaal van het plafond niet dominant genoeg is.

### Splitsing kamers

#### Slice hoogte en dikte

Voor de kamer splitsing is van groot belang op welke hoogte de slice wordt gemaakt om de 2D weergave te krijgen. Deze hoogte wordt nu 10 centimeter onder het plafond met een dikte van 20 centimeter genomen. Dit is geconcludeerd na vele tests op verschillende hoogtes en met verschillende diktes. Een grotere dikte kan er namelijk voor zorgen dat er over de hele hoogte van een kamer punten worden meegenomen, maar dit zorgt voor teveel ruis van de objecten in de kamer waardoor kamers teveel opsplitsen. De hoogte is ook getest op borst- en vloerhoogte, maar hier zorgen objecten ook voor teveel schaduw waardoor muren missen.   
  
Met schuine muren wordt een ‘smaller’ stuk kamer genomen. Daarom wordt van elke kamer nog een buffer meegegeven in het opknippen. Met een verdieping met meerdere hoogtes van plafond kan het voorkomen dat de slice deels een kamer mist of dat de slice ‘in’ de plafond punten valt.

Idealiter wordt de slice hoogte en dikte uit de specificaties van de kamer bepaald. Bijvoorbeeld door de hoogte uit te snijden op de plek waar de hoogste dichtheid aan punten zit en de dikte af te laten hangen aan de hoeveelheid ruis ten opzicht van muur.

#### Distance transform

In de eerste fase van de kamer detectie werden hoekpunten van de 2D weergave gedetecteerd om tussen deze hoekpunten missende lijnen op te kunnen vullen. Dit bleek slecht te werken als er teveel van een ruimte niet gescand is. Dit kwam omdat er ook hoekpunten misten en omdat een grote opvullengte ook zorgde voor extra opsplitsing tussen kamers. Er is daarom voor een andere oplossing gekozen.

De distance transform werkt veel beter omdat deze de pieken detecteert die het midden van de kamer representeert omdat hier de afstand tot de muren het grootst is. Ook hier worden kamers af en toe nog te vaak gesplitst maar al veel beter. Vanaf de piekwaardes wordt de kamer vervolgens naar de muur toe opgevuld. Deze piekwaardes worden bepaald door de mediaan van de distance transform. Omdat de distance transform het formaat heeft van de originele extent van de puntenwolk geeft dit enorme pieken als de buitenruimte meegenomen wordt, waardoor de pieken in de binnenruimte onder deze mediaan vallen. Hierdoor worden kamers vaak niet of niet volledige gesplitst als er een grote buitenruimte in de puntenwolk zit.

### Reconstrueren 3D objecten

Het Polgyonal Surface Reconstruction algoritme van CGAL kan gebruik maken van twee methoden die de berekening kunnen doen voor de reconstructie, GLPK en SCIP. GLPK is meer geschikt voor het oplossen van kleine problemen met een redelijk simpele structuur. SCIP kan ook meer complexere problemen aan.

De software van SCIP valt echter onder een ZIB Academic License. Dit betekent dat de library alleen gratis te gebruiken is voor onderzoeksdoeleinden voor niet-commerciële organisaties en academische instellingen. Voor onderzoeksdoeleinden heeft CGI gebruik gemaakt van beide libraries. Uiteindelijk is in de applicatie gebruik gemaakt van de GLPK software zodat er geen licentie hoeft te worden aangeschaft.

Er zijn drie parameters die voor Polygonal Surface Reconstruction (PSR) belangrijk zijn. Deze staan beschreven in de code repository. Voor dit project staan de parameters goed voor puntenwolken van Woonstad. Het kan voorkomen dat deze voor andere puntenwolken getuned moeten worden.

#### RANSAC

Random sample consensus (RANSAC) is gebruikt om vlakken te detecteren uit de puntenwolk. Dit algoritme wordt toegepast tijdens stap 4 (Polygonal Surface Reconstruction). Voor de uitvoering van de reconstructie zijn vlakken nodig vanuit de puntenwolk. Er kunnen verschillende parameters worden meegegeven aan RANSAC. Afhankelijk van de parameters kunnen de resultaten verschillen. In dit onderzoek zijn de parameters van RANSAC afgestemd op het aantal punten die ontstaan na de preprocessing stap. Het is mogelijk deze parameters automatisch te kiezen aan de hand van de dichtheid van de input. Dit is in de huidige fase niet uitgevoerd.

### Oppervlakte en volume berekening

Tijdens de berekening van de oppervlaktes en volumes zijn er meerdere problemen van meshes ondervangen. De meshes moeten nog sluitend gemaakt worden en opnieuw georiënteerd. Niet altijd lukt dit sluitend maken van de mesh waardoor kamers afvallen die wel succesvol uit de PSR reconstructie zijn gekomen.

Daarnaast komen uit de PSR meshes die gedeeltelijk overlap kunnen hebben. Het kan hierdoor voorkomen dat de meetwaardes dubbel worden opgemeten en dus een overschatting wordt gemaakt van de daadwerkelijke meetwaardes. In de vergelijking met BIM lijken de meetwaardes goed overeen te komen maar toch is dit een punt van aandacht. Het is wenselijk om eerst nog een combinatie te maken van alle kamers waarin dubbelen voorkomen worden. Ook voor visualisatie is dit gewenst om direct de verschillende kamers als objecten inzichtelijk te krijgen.

## Afwijking van programma van eisen

In bijlage A van het programma van eisen staan de te extraheren objecten uit de puntenwolken ten behoeve van de SOR. Deze lijst is erg uitgebreid en is op sommige punten lastig te bepalen vanuit de puntenwolk. De componenten die niet zijn gerealiseerd omdat ze slecht te bepalen zijn vanuit een puntenwolk zijn: *Keukenvoorzieningen, Badvoorzieningen, Meterkasten, Gebouwcomponenten, Gebouwzones, Verblijfsobjecten, Dragende en niet-dragende muren, Openbare Ruimte, Open Bouwwerk.* De data die voor dit onderzoek beschikbaar is gesteld bevat niet alle informatie om deze componenten te identificeren. Sommige componenten missen in de invoer data. Bovendien is voor veel van deze componenten semantische informatie nodig die simpelweg niet af te leiden is uit alleen puntenwolken, bijvoorbeeld het verschil tussen een dragende of niet-dragende muur.

De componenten die wel uit een puntenwolk zijn te detecteren maar niet gerealiseerd in dit project zijn: *Gebouw/pand, Voetafdruk pand, Toegangsdeur gebouw, Toegangsdeur verblijfsobject, Scheidingswanden.* Door verder te ontwikkelen in de richting van dit project zijn deuren te identificeren door nieuwe algoritmiek toe te passen. Ook kan, mits de input data het toelaat, nauwkeuriger worden bepaald wat de verdiepingsvloeren hoogte is en wat de voetafdruk en afmetingen van het pand zijn. Hiervoor moet dan wel de puntenwolk ingewonnen worden met ook de buitenkant van het pand voor zover mogelijk. Voor nauwkeurige hoogtebepaling moet de puntenwolk gegeorefereerd zijn. In de huidige implementatie worden ruimten los geïdentificeerd en als vervolg hierop kunnen aangrenzende wanden worden geannoteerd als scheidingswanden.

## Aanpassingen op plan van aanpak

Er zijn kleine aanpassingen geweest in de uitvoer ten opzichte van de origineel beoogde aanpak. Deze keuzes zijn voornamelijk gemaakt aan de hand van voortschrijdend inzicht gedurende het project.

### LASTools

In het originele plan van aanpak staat beschreven dat de conversie van de inputbestanden (.las/.laz) nodig is via een Laz-converter (Rapidlasso LASTools). In de uiteindelijke aanpak is CloudCompare gebruikt die zelf in staat is om zonder conversie las te lezen en te verwerken. Dit maakt de Laz-converter overbodig.

### Machine Learning

De beoogde applicatie voor het verwerken van puntenwolken staat beschreven als een “Machine Learning model”. In de uitvoering is geen machine learning gebruikt. Voor de scope van het project is die oplossing niet geschikt omdat dat veel tijd in beslag zou nemen. Ook was er bij de aanvang van het project geen data beschikbaar waarop modellen zouden kunnen worden getraind en getest. Daarom is er gekozen voor een oplossing zonder machine learning componenten.

### Opslag

In het plan van aanpak staat opslag beschreven met de componenten SQL server en/of PostGISSQL. Deze componenten zijn niet nodig geweest. Het alternatief dat is gekozen voor opslag is de Microsoft Azure Blob Storage. Gemeente Den Haag gebruikt ook hetzelfde platform voor data opslag en zorgt hiermee voor een gemakkelijke verbinding.

# Aanbevelingen

## Inwinning Puntenwolken

Zoals aangemerkt in paragraaf 5.1 zijn er veel lessen geleerd van de puntenwolken die te maken hebben met de inwinning. Daarom zijn de aanbevelingen ook om deze lessen te gebruiken om in een vervolg hogere kwaliteit puntenwolken in te winnen. Dit houdt in dat het gebruik van mobiele laserscanners aangeraden wordt en dat deze werken met een georeferentie punt of punten waardoor de puntenwolk gegeorefereerd is. Tijdens het scannen is het belangrijk om de gehele ruimtes in te winnen met zo min mogelijk ruis ín het gebouw en ook buiten het gebouw. Het scannen door ramen en door naar buiten moet geminimaliseerd worden. Ook binnenshuis dient alles rondom objecten ingemeten te worden wat al een stuk beter gaat als een mobiele laserscanner gebruikt wordt. Anders moet de vaste laser scanner op een zo strategisch mogelijke plek in de kamer geplaatst worden.

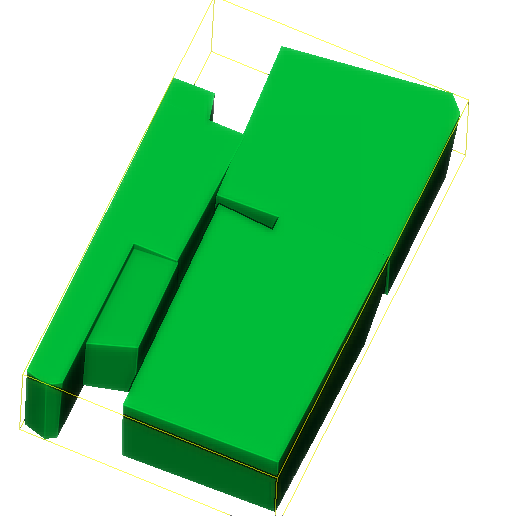
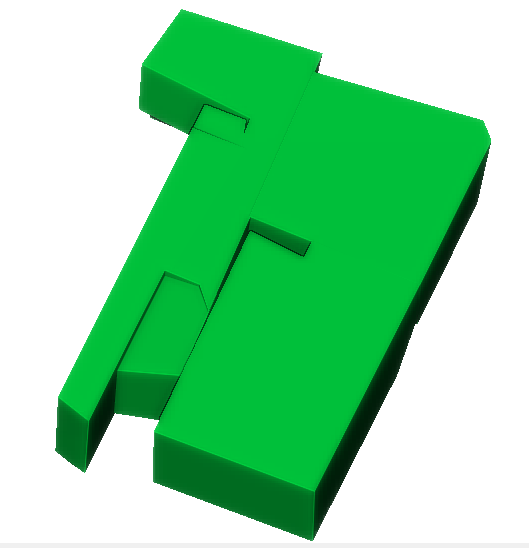
## Licenties

### AutoDesk Recap

De aangeleverde puntenwolken van Woonstad zijn aangeleverd in .rcp files. Deze zijn eenmalig geconverteerd naar .e57 files via een proeflicentie van AutoDesk Recap. Het is noodzakelijk om in de toekomst deze licentie aan te schaffen als de puntenwolken in .rcp bestandsformaat worden aangeleverd.

### SCIP

Om de Polygonal Surface Reconstruction (PSR) op te lossen is eerst gebruik gemaakt van de SCIP solver. Dit is echter alleen te gebruiken voor academisch gebruik of als non-commerciële organisatie. Voor commercieel gebruik is een licentie nodig. Daarom is over gestapt naar GLPK wat wel open-source beschikbaar is. Er is echter een groot verschil in performance tussen de twee solvers. GLPK heeft vaker moeite om de oplossing sluitend op te lossen en is veel langzamer dan SCIP (tientallen minuten versus tientallen secondes). Als beide tot een oplossing komen binnen afzienbare tijd is de kwaliteit redelijk vergelijkbaar dus daarom is voorlopig voor GLPK gekozen. Voor een geschikte performance wordt aanbevolen om een SCIP licentie aan te vragen.



Figuur 19: Links puntenwolk. Midden PSR oplossing met SCIP. Rechts PSR oplossing met GLPK

## Methode

Voor verbeteringen in de puntenwolken verwerking en de kwaliteit ervan kunnen de huidige methoden worden uitgebreid en getoetst met verschillende waarden voor parameters. Nu wordt een groot deel van de puntenwolk gefilterd in de preprocessing stap wat zorgt voor snelle verwerking in de vervolgstappen maar ook voor een mogelijk informatie verlies. Ook kan de precisie van de vlakkendetectie worden aangepast om kleinere vlakken te detecteren. Hier moet dan wel worden afgewogen hoeveel vlakken worden gedetecteerd die niet tot de vaste structuur van een gebouw behoren. Deze zorgen dan namelijk voor ruis in de 3D reconstructie.

Op het moment gaat de geïmplementeerde methode nog niet goed om met schuine vlakken (van bijvoorbeeld schuine daken). Door de manier hoe kamers worden gedetecteerd worden lage stukken van het schuine dak niet volledig mee genomen. Hiervoor zal nog een aanvullende algoritmiek moeten worden toegevoegd, bijvoorbeeld door altijd schuine vlakken los te detecteren en mee te nemen in de 3D reconstructie.

Voordat de voorgestelde methode verder kan worden ingezet is zorgvuldig testen met een verschillend soort puntenwolk aan te raden. Hierdoor komen nieuwe en bijzondere gevallen aan het licht. Verder is het verstandig om de effecten van de parameters door te lichten door een analyse te doen van de effecten op het eindresultaat.

Bij doorontwikkeling op deze code kunnen componenten worden toegevoegd voor verdere specificatie van gebieden in de puntenwolk. Zo kan er bijvoorbeeld worden gezocht naar objecten of groepen van objecten in de puntenwolk die de functie van een ruimte bepalen. Hiervoor kunnen specifieke detectoren worden ontwikkeld bijvoorbeeld met behulp van machine learning.

## Implementatie

Den Haag (of een derde partij) kan de componenten gebruiken in de aangeleverde code om een pipeline te ontwikkelen. De code is zo opgezet dat ieder component een klein doel heeft om te vervullen en deze zijn geplaatst in Docker images. De images kunnen via een externe tool worden afgetrapt om zo een pipeline te bouwen. Voor opschaling kan een orchestratietool zoals Kubernetes worden gebruikt om de individuele componenten horizontaal te schalen waar nodig.

Het plaatsen van de code in containers is geen harde eis voor de analyse. Er is ook de mogelijkheid om de componenten zonder container uit te voeren en deze zelf te implementeren in de gewenste infrastructuur.



[**cgi.com**](https://www.cgi.com)

1. <https://www2.isprs.org/commissions/comm4/wg5/dataset/> [↑](#footnote-ref-2)
2. <https://www.cloudcompare.org/doc/wiki/index.php?title=FILE_I/O> [↑](#footnote-ref-3)
3. <https://www.cloudcompare.org/doc/wiki/index.php?title=Edit%5CSubsample> [↑](#footnote-ref-4)
4. <https://www.cloudcompare.org/doc/wiki/index.php?title=SOR_filter> [↑](#footnote-ref-5)
5. <https://www.tutorialspoint.com/opencv/opencv_distance_transformation.htm> [↑](#footnote-ref-6)
6. <https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html> [↑](#footnote-ref-7)
7. <https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_watershed/py_watershed.html> [↑](#footnote-ref-8)
8. <https://doc.cgal.org/latest/Polygonal_surface_reconstruction/index.html> [↑](#footnote-ref-9)
9. <https://www.scipopt.org/> [↑](#footnote-ref-10)
10. <https://www.gnu.org/software/glpk/> [↑](#footnote-ref-11)
11. <https://pymeshlab.readthedocs.io/en/0.1.8/filter_list.html> [↑](#footnote-ref-12)