2020•2021
Faculteit Industriële Ingenieurswetenschappen
master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Snelle en nauwkeurige posebepaling van infrarood- en dieptecamera t.o.v. een werkstation: experimentele vergelijking van registratiealgoritmes

PROMOTOR :
Prof. dr. ir. Eric DEMEESTER

PROMOTOR :
ir. Kim RUTTEN

BEGELEIDER :
ing. Peter AERTS

COPROMOTOR :
ing. Maarten VERHEYEN

COPROMOTOR :
Dhr. Stijn DEBRUYCKERE

Kyle Severi, Jan Van den Berckt
Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

Gezamenlijke opleiding UHasselt en KU Leuven

UHASSELT KU LEUVEN

UHASSELT KU LEUVEN

2020•2021

Faculteit Industriële Ingenieurswetenschappen

master in de industriële wetenschappen: elektronica-ICT

Masterthesis

Snelle en nauwkeurige posebepaling van infrarood- en dieptecamera t.o.v. een werkstation: experimentele vergelijking van registratiealgoritmes

PROMOTOR :
Prof. dr. ir. Eric DEMEESTER

PROMOTOR :
ir. Kim RUTTEN

BEGELEIDER :
ing. Peter AERTS

COPROMOTOR :
ing. Maarten VERHEYEN

COPROMOTOR :
Dhr. Stijn DEBRUYCKERE

Kyle Severi, Jan Van den Berckt

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: elektronica-ICT

UHASSELT KU LEUVEN

Voorwoord

Deze masterpoef was een mooie opportuniteit om binnen een professionele omgeving aan een project te kunnen werken. Hierbij konden we onze communicatie vaardigheden verder ontplooien en kregen we de kans om te werken binnen een domein waarin we weinig tot geen voorkennis over hadden. Om dit project tot een goed einde te kunnen brengen konden we rekenen op een uitgebreid team bestaande uit interne en externe begeleiders.

Op extern vlak willen we Arkite bedanken voor hun warme verwelkoming in hun team. Specifiek willen we Stijn Debruyckere en ir. Kim Rutten bedanken voor hun snelle en duidelijke communicatie, alsook de vele tips die we van hen gekregen hebben.

Ook willen we graag onze interne begeleiders Prof. dr. ir. Eric Demeester, ing. Peter Aerts en ing. Maarten Verheyen bedanken voor hun technische inbreng tijdens de vele Google Meets. Met hierbij extra aandacht voor Peter Aerts die ons van nabij geholpen heeft op theoretisch vlak en bij het schrijven van de thesis.

Als laatste gaat er een grote appreciatie uit naar onze ouders, die ons op persoonlijk vlak ondersteund hebben om dit alles tot een goed einde te brengen tijdens een globale pandemie.

Jan Van den Berckt

Kyle Severi

Inhoudstafel

Lijst van tabellen

Lijst van figuren

Abstract Nederlands

Deze masterproef werd uitgevoerd in samenwerking met het bedrijf Arkite gelegen te Genk. Dit bedrijf ontwikkelde de HIM (Human Interface Mate), een product dat operatoren begeleidt bij montageprocessen binnen een werkcel. Hiertoe worden handelingen waargenomen met een diepte- en infraroodcamera en worden werkinstructies geprojecteerd op de werkomgeving. Dit vereist een nauwkeurige positionering of extrinsieke kalibratie van deze camera's t.o.v. de werkomgeving. In de praktijk gebeurt het echter dat deze positionering wijzigt, wat een negatieve impact heeft op het projecteren van de werkinstructies en het interpreteren van de menselijke handelingen.

De doelstelling van deze masterproef is om de extrinsieke kalibratie snel en nauwkeurig te herberekenen. Hiervoor is er een vergelijkende studie gemaakt van drie verschillende registratiealgoritmes: Iterative Closest Point (ICP), Iterative Closest Point with Normals (NICP) en Generalized Iterative Closest Point (GICP).

In een eerste fase zijn deze algoritmes geëvalueerd voor scenario's gaande van een volledige puntenwolk tot een puntenwolk bestaande uit de tafel zonder ruis. Hierbij scoorde GICP het slechtst op vlak van nauwkeurigheid.

In de tweede fase zijn enkel keypoints, geëxtraheerd uit de 2D-infraroodbeelden en gealloceerd aan de puntenwolk, gebruikt voor registratie. Hierbij werden enkel NICP en ICP getest. Hieruit bleek dat ICP in combinatie met keypoints de beste transformatie berekende in een gemiddelde tijd van 21.25 ms t.o.v. 2236 ms bij ICP in de eerste fase.

Abstract Engels

This master's thesis was carried out in collaboration with the company Arkite, located in Genk. This company developed the HIM (Human Interface Mate), a product that guides operators through assembly processes within a work cell. To this end, actions are observed through a depth and infrared camera and instructions are projected onto the working environment. This requires precise positioning or extrinsic calibration of these cameras relative to the work environment. In practice however, it happens that this positioning changes, which has a negative impact on the projection of the instructions and the interpretation of the human interactions.
The objective of this master's thesis is to recalculate the extrinsic calibration quickly and accurately. Therefore, a comparative study was made consisting of three different registration algorithms: Iterative Closest Point (ICP), Iterative Closest Point with Normals (NICP) and Generalized Iterative Closest Point (GICP).
During the first phase, these algorithms were evaluated for scenarios ranging from a complete point cloud to a point cloud consisting of just the table without noise. GICP scored the worst in terms of accuracy.
In the second phase, only keypoints, extracted from the 2D infrared images and allocated within the point cloud, were used for registration. Only NICP and ICP were tested. This showed that ICP in combination with keypoints calculated the best transformation in an average time of 21.25 ms compared to 2236 ms for ICP in the first phase.

1 Inleiding

1.1 Situering

Deze masterproef is uitgevoerd bij het bedrijf Arkite met ondersteuning van de onderzoeksgroep ARCO. Arkite is een bedrijf gelegen te Genk. Het product dat zij aanbieden is de Human Interface Mate (HIM) (figuur 2). Dit systeem begeleidt operatoren bij het uitvoeren van hun taken m.b.v gevisualiseerde instructies. De instructies bestaan uit tekst, foto's en video's die rechtstreeks in het gezichtsveld van de operator worden geprojecteerd (augmented reality). Hiervoor worden één of meerdere projectoren aangesloten op de HIM. Bij iedere stap van het assemblageproces wordt geprojecteerd op de werkcel welke handeling uitgevoerd moet worden en welke onderdelen hiervoor vereist zijn. Hiervan is een voorbeeld zichtbaar in figuur 1. Om waar te nemen welke handelingen allemaal plaatsvinden, maakt de HIM gebruik van: een RGB camera, een infraroodcamera en een dieptesensor. De sensor heeft een beperkte openingshoek, waardoor de positionering van de HIM verschillend is voor iedere werkcel. Bovendien kan deze positie of de werkplek zelf doorheen de tijd worden aangepast en is er dus nood aan een kalibratie. Tijdens deze kalibratie worden de locaties van materialen en onderdelen binnen de werkcel gedefinieerd. Als laatste kunnen elektrische apparaten ook verbonden worden met de HIM. Dit zal ervoor zorgen dat de correcte gereedschappen enkel tijdens de correcte instructie gebruikt zal worden.

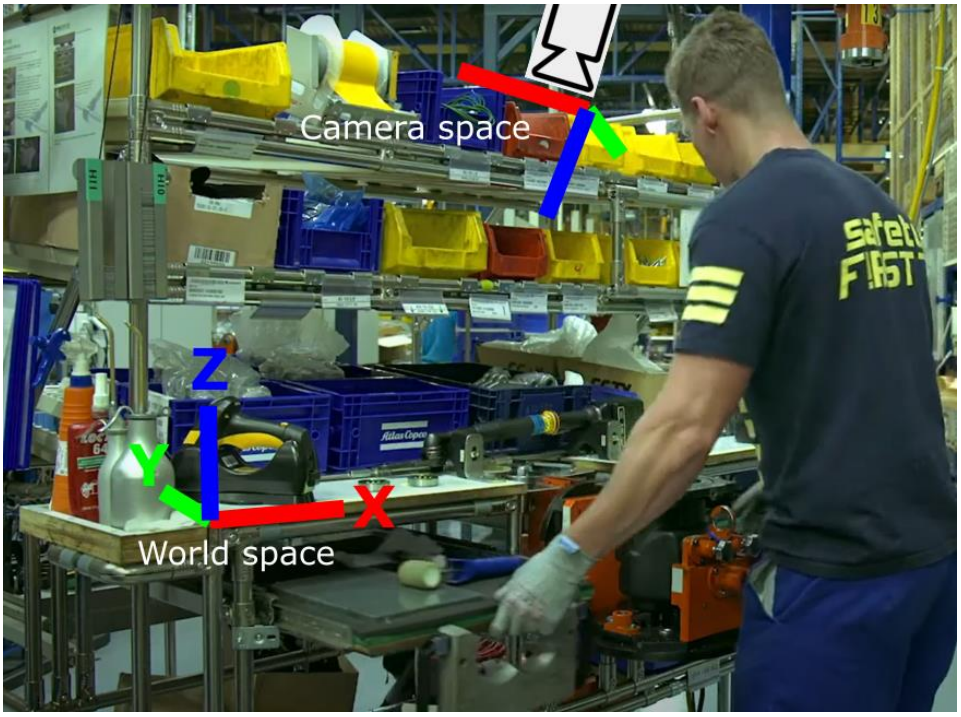
Figuur 1: Voorbeeld instructie door HIM

Figuur 2: HIM

Het gebruiken van de HIM brengt grote voordelen met zich mee op vlak van kwaliteit, efficiëntie, data en vermogen van de arbeider. Ten eerste verbetert de kwaliteit van het product, doordat de kans op fouten zo goed als volledig gereduceerd wordt. Ten tweede wordt de efficiëntie van het proces verhoogd, aangezien er duidelijk wordt aangegeven hoe en wanneer welke stap moet worden uitgevoerd. Ook kan de HIM gekoppeld worden aan de server van het bedrijf. Zo kan automatisch aangegeven worden wat wanneer geproduceerd is geweest. Hierdoor moet de arbeider dit niet meer handmatig doen. Een ander voordeel is dat de kennis over het productieproces niet verloren gaat wanneer een werknemer vervangen wordt, aangezien de processen opgeslagen zijn door de HIM. Een gevolg hiervan is dat de werkgever gemakkelijk werknemers op andere posities kan inzetten, zonder dat de persoon in kwestie nood heeft aan voorkennis van dat bepaalde takenpakket. Als laatste zijn er ook enkele voordelen voor de arbeider zelf. Zo is het gemakkelijker om vertrouwd te geraken met het productieproces en biedt het ook een geruststelling dat ze niets fout kunnen doen. Ook is het mogelijk om mensen in te zetten die voordien niet de capaciteiten hadden om bepaalde taken uit te voeren, omdat ze bijvoorbeeld te ingewikkeld zouden zijn.

1.2 Probleemstelling

Momenteel neemt de HIM beelden waar van 512 bij 424 pixels. De waarde van elke pixel is de afstand in millimeter tussen de sensor en het waargenomen punt samen met de intensiteit van de reflectie in de infrarode band. De parameters van de sensor lens zijn op voorhand gekend en zijn gedefinieerd door de intrinsieke camera parameters. Dit laat toe om elke diepte waarde om te zetten in een punt in een cartesiaans assenstelsel in de reële wereld. Dit coördinatensysteem is bevestigd aan de sensor zelf en is gekend als de camera ruimte. Buiten de camera ruimte is er ook nog de wereld ruimte. Deze is bevestigd aan voorwerpen in de werkelijke wereld zoals de tafel in figuur 3.



Figuur 3: Camera ruimte (camera space) en wereld ruimte (world space) in een werkomgeving

Het overgaan van de camera ruimte naar de wereld ruimte is gedefinieerd door de extrinsieke matrix. Deze matrix definieert de transformatie tussen de camera ruimte en de wereld ruimte en heeft volgende vorm:

$$[R\,|\,\boldsymbol{t}] = \left[\begin{array}{ccc|c} r_{1,1} & r_{1,2} & r_{1,3} & t_1 \\ r_{2,1} & r_{2,2} & r_{2,3} & t_2 \\ r_{3,1} & r_{3,2} & r_{3,3} & t_3 \end{array}\right] \quad (1)$$

Waarbij de vector t kan geïnterpreteerd worden als de positie van de oorsprong van de wereldruimte in de cameraruimte en de matrix r de richtingen van de assen in de wereldruimte aangeven in de camera ruimte.

Tijdens het productieproces kan het voorkomen dat de positie van de werkplaats verandert. Dit kan voorkomen wanneer bijvoorbeeld de werkcel vernieuwd wordt, er nieuw gereedschap wordt toegevoegd of het proces wordt aangepast. In de aangehaalde situaties worden er bewust aanpassingen aangebracht binnen de werkcel. Het kan ook echter voorkomen dat de werkbank of de sensor onbedoeld verschoven wordt door bijvoorbeeld een onverwachte botsing. Dit heeft als gevolg dat de originele kalibratie van de extrinsieke matrix niet meer overeenkomt met de nieuwe situatie. Dit is ongunstig aangezien de detecties en de bijbehorende projecties niet meer in de correcte ruimtelijke positie plaatsvinden. De software van de HIM kan zelf niet detecteren wanneer er zich een verplaatsing voordoet en is evenals niet in staat om de plaatsgevonden verschuiving te berekenen. Er dus nood aan een nieuwe kalibratie wanneer een dergelijke situatie zich voordoet. Daarvoor komt een ingenieur ter plaatse, die manueel al deze detecties en projecties terug juist afstemt. Dit is een tijdrovende methode en zorgt ervoor dat de productie geruime tijd stil kan liggen.

1.3 Doelstellingen

Het doel van deze masterproef is om een methode te onderzoeken die de extrinsieke kalibratie snel en nauwkeurig kan herberekenen. Om dit te kunnen bereiken is het nodig om de matrix te vinden, die de relatieve verplaatsing tussen de sensor en de werkcel beschrijft. De nieuwe extrinsieke matrix wordt dus gevonden door de oude te transleren en roteren volgens de gevonden verplaatsing tussen de oude en nieuwe positie. In formulevorm ziet dit er als volgt uit:

$$E' = R.E + T \quad (2)$$

Waarbij E' de nieuwe extrinsieke matrix is, E de oude extrinsieke matrix, R de gevonden rotatie en T de gevonden translatie.

Een hoge nauwkeurigheid is de eerste vereiste. Dit omdat de nauwkeurigheid een grote impact heeft op de locatie van de projectie en van het detecteren van handelingen. Een foutenmarge van enkele millimeters is hierbij aanvaardbaar. De tweede vereiste is performantie. Hierbij is het belangrijk dat de andere processen van de HIM niet gehinderd worden tijdens de herberekening van de extrinsieke kalibratie. Ook is een zo performant mogelijke methode gewenst, zodat deze ook in de toekomst gebruikt kan worden in scenario's waarbij de HIM bevestigd is aan een bewegend object.

Een courante methode om dit probleem op te lossen is door gebruik te maken van fysieke markeringen in het zichtveld van de sensor. Door de aard van de omgeving waar de HIM gebruikt wordt, is dit echter vaak geen mogelijkheid. Daarom moet er ten derde een methode ontwikkeld worden zonder gebruik van zulke markeringen. De ontwikkeling van dit algoritme wordt in een Windows omgeving uitgevoerd, aangezien de HIM op Windows opereert. Als programmeertaal wordt geopteerd voor c++, aangezien dit tot een betere performantie en een eenvoudigere implementatie leidt.

1.4 Methode

De ontwikkeling van een geschikte methode voor het herbereken van de extrinsieke matrix werd uitgevoerd volgens volgende stappen:

- In de eerste fase van de thesis wordt onze studie besproken naar de eigenschappen van de HIM, verschillende camera modellen en bestaande technieken om verplaatsingen te berekenen op basis van 2D en/of 3D beelden. Naast de theoretische achtergrond van deze algoritme wordt ook gezocht naar software bibliotheken, die implementaties voorzien in c++.

- In de volgende fase worden de gevonden technieken en de bijhorende bibliotheken getest. Specifiek gebeurt dit aan de hand van een ideale puntenwolk voor de registratiealgoritmes. Voor de feature detectie algoritmes wordt gebruik gemaakt van afbeeldingen, die willekeurig gevonden werden op het internet.

- Na het testen van de functionaliteiten van de methodes en het verifiëren van een correcte implementatie, wordt er overgegaan naar reële data. Deze data wordt gecapteerd met behulp van de HIM. De opgenomen scène bestaat uit het transleren van een lege werktafel. De performantie en nauwkeurigheid van de verschillende registratiealgoritmes worden op basis van deze scène met elkaar vergeleken.

- Tijdens de laatste fase wordt een andere aanpak getest waarbij een feature detectie algoritme gebruikt zal worden als filter voor het registratiealgoritme. Om dit te testen is er nood aan nieuwe opnames, aangezien er op een lege tafel weinig features gedetecteerd kunnen worden. Tijdens de opnames vinden verschillende rotaties en translaties plaats in de scène.

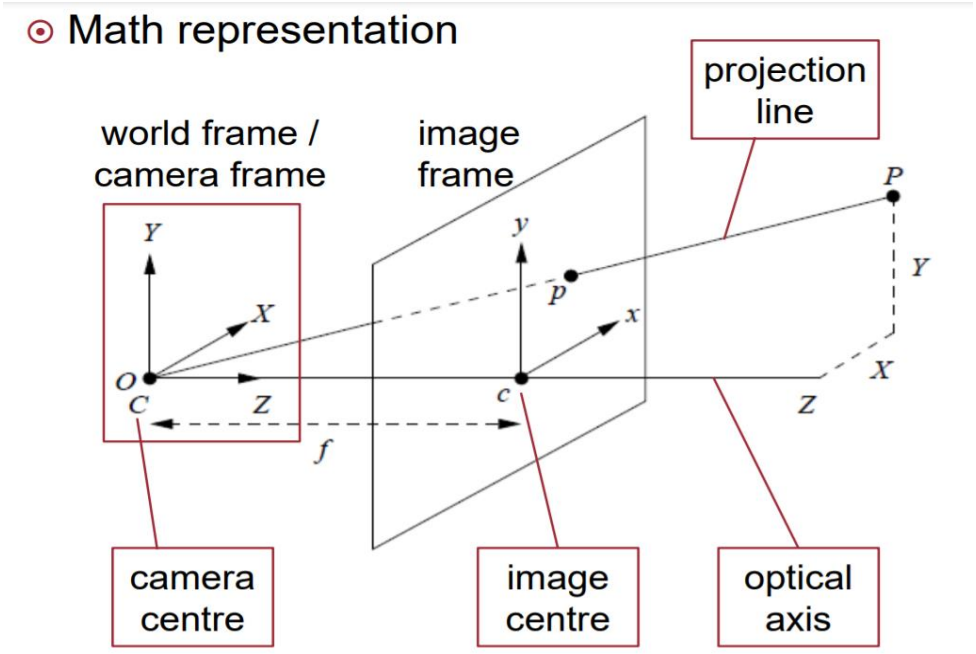
2 Literatuurstudie

2.1 Puntenwolk

Een puntenwolk is een datastructuur waarin 3D punten worden opgeslagen. Dit betekent dat ieder punt in een puntenwolk minstens een x-, y- en z-coördinaat bevat. Puntenwolken kunnen echter ook nog andere informatie bevatten, zoals onder andere een RGB-waarde van het punt of de normaalrichting van het punt. Puntenwolken zijn typisch een voorstelling van een object of scene in de echte wereld. Om deze scene of dit voorwerp vast te leggen wordt een diepte camera gebruikt. Hierbij worden diepte waardes toegewezen aan iedere pixel van de camera. Om van deze data de puntenwolk te reconstrueren wordt in deze thesis gebruik gemaakt van het pinhole camera model. De reconstructie van de puntenwolk is nodig om de transformatie tussen twee puntenwolken te berekenen.

2.2 Reconstructie puntenwolk

Bij het capteren van de dieptewaardes worden deze vastgelegd in het pixel coördinatenstelsel van de camera. Dit is een 2D coördinatenstelsel. Deze projectie gebeurt volgens het pinhole camera model. Dit model wordt visueel weergegeven in figuur 4. Het punt P in de echte wereld wordt geprojecteerd in het pixel coördinatenstelsel door het snijpunt van de lijn van dit punt naar het centrum van de camera met het pixel vlak. Op deze manier wordt een 3D punt geprojecteerd in een 2D vlak.



Figuur 4: Pinhole camera model [1]

In formulevorm ziet het pinhole camera model er als volgt uit:

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} R & T \end{bmatrix} \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} = M \begin{bmatrix} x_w \\ y_w \\ z_w \\ 1 \end{bmatrix} \quad (3)$$

$$\text{Met } M = K \begin{bmatrix} R & T \end{bmatrix} \quad (4)$$

In deze formule staat z_c voor de gemeten dieptewaarde, u voor de x-waarde van het punt in het pixel coördinatenstelsel en v voor de y-waarde van het punt in het pixel coördinatenstelsel. x_w, y_w en z_w zijn de x-, y- en z-waarde in het assenstelsel in de wereldruimte.

Matrix K is de intrinsieke matrix. Deze heeft volgende vorm:

$$K = \begin{bmatrix} \alpha_x & \gamma & u_0 & 0 \\ 0 & \alpha_y & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \quad (5)$$

In deze matrix staan α_x en α_y voor de focale lengte van de cameralens in pixelwaardes.

u_0 en v_0 staan respectievelijk voor de x-waarde en y-waarde van de oorsprong in pixelwaarden in het camera coördinatenstelsel. γ is de skew coëfficiënt tussen de x-as en y-as. Deze is meestal 0. Dit betekent dat de x- en y-as loodrecht op elkaar staan.

De matrices R en T staan respectievelijk voor de rotatie en translatie tussen het wereld coördinatensysteem en het camera coördinatensysteem. Samen worden ze de extrinsieke matrix genoemd.

Om de puntenwolk te reconstrueren wordt gebruik gemaakt van het pinhole camera model in de omgekeerde richting. Er wordt een transformatie van een 2D beeld naar een 3D ruimte uitgevoerd. Dit wordt bekomen door de inverse van de intrinsieke matrix te vermenigvuldigen met de inverse van de extrinsieke matrix en dit dan te vermenigvuldigen met de punten in de image assenstelsel. Op deze manier worden de 3D coördinaten in de wereld space van de 2D afbeelding gereconstrueerd.

2.3 Filteren van puntenwolken

Alvorens er berekeningen worden gedaan aan de hand van puntenwolken worden er vaak eerst filters toegepast. Deze filters hebben onder meer het doel om ruis te onderdrukken of de puntenwolk te downsamplen. Dit leidt tot minder foutieve metingen en minder vereiste rekenkracht. Ook het zoeken van keypoints kan gezien worden als een filtering van de puntenwolk. Het verschil tussen het downsamplen van een puntenwolk en het zoeken van keypoints is de soort data die wordt overgehouden uit de puntenwolk. Bij het downsamplen gaan details in de puntenwolk verloren. Bij het zoeken van keypoints worden echter de punten die opmerkelijke kenmerken in de scene beschrijven behouden. Hieronder worden enkele filters besproken:

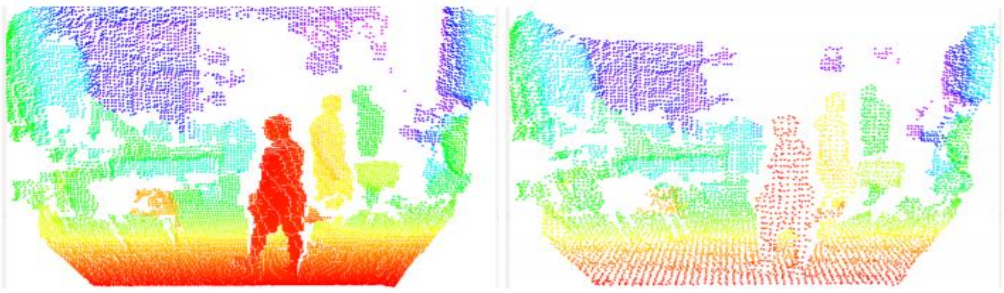
- *Passthrough filter*

  Bij deze filter worden de punten gefilterd op hun x-, y- en z-waardes. Wanneer deze waardes binnen de opgegeven afstand liggen worden de punten behouden, wanneer ze buiten deze range liggen worden ze weggefilterd. Een voorbeeld van een passthrough filter voor de z-waarde wordt gegeven in volgende code:

```
Foreach(Point point in pointcloud) {
        If ((point.z > maxZ) || (point.z < minZ)) {
                pointcloud.remove(point)
        }
}
```

- *Voxel grid filter*

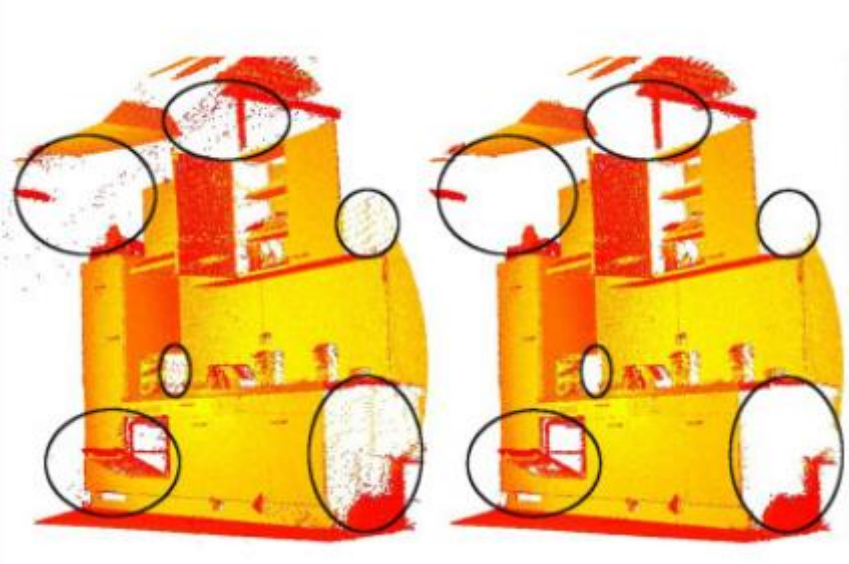
  Bij deze filter wordt de hele puntenwolk verdeeld in voxels. De puntenwolk wordt dan voorgesteld d.m.v. kubussen die een zeker volume beschrijven i.p.v. punten in de ruimte. Van elk punt dat in de kubus ligt wordt het zwaartepunt berekend en wordt dit zwaartepunt gebruikt als voorstelling van de voxel. Deze filter downsamplet de puntenwolk. Figuur 5 toont een puntenwolk voor en na er een voxel grid filter is op toegepast.



Figuur 5: Voxel grid filter: voor en na [2]

- *Statistical outlier removal*

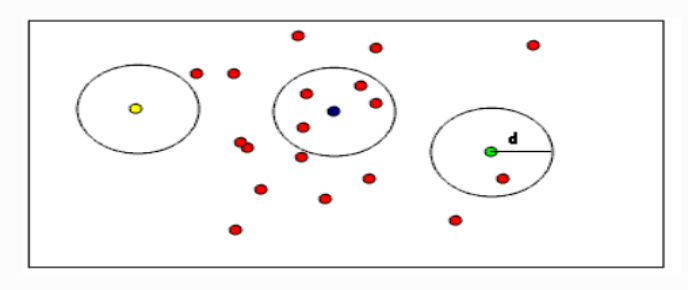
Bij deze filter worden de outliers weggefilterd met behulp van de gemiddelde afstand en standaarddeviatie van een punt met zijn buren. Voor elk punt wordt de gemiddelde afstand van zijn buren berekend. Er wordt verondersteld dat de verdeling van deze afstanden Gaussiaans is met een gemiddelde en een standaardafwijking. Alle punten die niet binnen het bereik van de gemiddelde afstand en de standaarddeviatie liggen worden weggefilterd. In figuur 6 is links een ongefilterde puntenwolk zichtbaar en rechts is dezelfde puntenwolk zichtbaar na het toepassen van de statistical outlier removal filter. De gefilterde zones zijn aangeduid met een zwarte kring.



Figuur 6: Statistical outlier removal: voor en na [3]

- *Radius Outlier Removal*

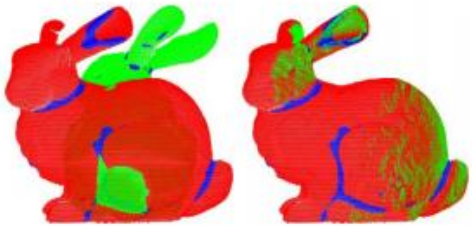
Bij deze filter worden alle punten die minder dan het opgegeven aantal buren heeft binnen een bepaalde straal weggefilterd. In figuur 7 is de werking van de radius outlier removal filter zichtbaar. Wanneer een punt minstens twee buren moet hebben binnen een opgelegde straal worden zowel het gele als het groene punt weggefilterd. Wanneer een punt slechts één buur moet hebben wordt enkel het gele punt weggefilterd. Het blauwe punt wordt behouden vanwege genoeg buren.



Figuur 7: Radius outlier removal: schematische voorsteling [4]

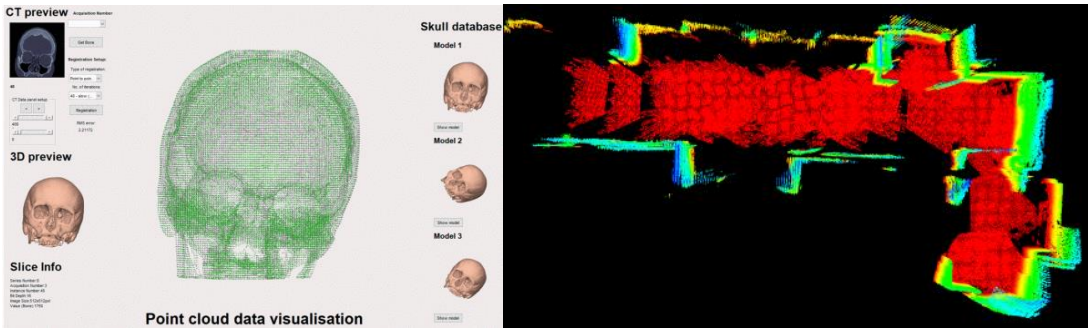
2.4 Registratie van puntenwolken

Registratie van puntenwolken is een synoniem voor het aligneren van twee puntenwolken die dezelfde scene (of delen hiervan) omvatten naar één puntenwolk. Er wordt dus gezocht naar de transformatie zodat twee opeenvolgende puntenwolken zo goed mogelijk aligneren. Deze puntenwolken zijn typisch gecapteerd vanop verschillende locaties of deze bevatten objecten die verplaatst zijn ten opzichte van de sensor. In beide gevallen kan een transformatie T berekend worden waarvoor geldt: T(X) = Y waarbij X de originele puntenwolk is en Y de puntenwolk waarin de sensor of de voorwerpen verplaatst zijn. In figuur 8 is links de puntenwolken zichtbaar van twee konijnen voor registratie en rechts dezelfde puntenwolken na registratie.



Figuur 8: Voorbeeld van registratie voor en na [5]

Registratie heeft verschillende toepassingen binnen het werkgebied en de academische wereld. Een van de toepassingen is het opbouwen van een map van de ruimte waarin een sensor zich bevindt. Hiervoor worden scans genomen van verschillende delen van een ruimte waarna deze met behulp van registratietechnieken deze puntenwolken combineert tot één puntenwolk. Een andere toepassing van registratie bevindt zich in de medische wereld. Bij het maken van scans is het tevens nodig om deze te combineren tot 1 puntenwolk. Nog een toepassing is het lokaliseren van een sensor in een ruimte. Wanneer een sensor beweegt in een ruimte is het nodig dat deze sensor kan berekenen welke verplaatsing hij gemaakt heeft. Deze berekening gebeurt op basis van de vorige captatie van een puntenwolk en de huidige captatie. Ook hiervoor wordt gebruikt gemaakt van registratietechnieken. In figuur 9 wordt links een voorbeeld van medische registratie gegeven en rechts een voorbeeld van map building.



Figuur 9: Registratievoorbeelden in praktijk [6][7]

In deze thesis wordt registratie gebruikt om de relatieve verplaatsing van de Human Interface Mate te bepalen om zo de extrinsieke matrix van het pinhole camera model te herdefiniëren.

2.5 Registratiealgoritmes

De twee grootste klassen van registratie algoritmes zijn feature-based registratie en iteratieve registratie. Tijdens feature based registratie worden geometrische feature descriptors berekend en gematcht in een hoog-dimensionale ruimte. Hoe meer uniek en meer beschrijvend deze descriptors zijn, hoe groter de kans dat alle gevonden matches tussen beide puntenwolken met elkaar overeenkomen. Bij iteratieve registratie worden geen feature descriptors berekend maar worden punten die in de Cartesiaanse ruimte het dichtst bij elkaar liggen verondersteld overeen te komen in beide puntenwolken. Deze thesis richt zich op de iteratieve algoritmes. In het volgende stuk worden de iteratieve algoritmes Iterative Closest Point (ICP) en Normal Distribution Transform (NDT) besproken. In het stuk over ICP worden tevens twee algoritmes besproken die een variant zijn op ICP. Deze algoritmes zijn Iterative Closest Point with Normals (NICP) en Generalized Iterative Closest Point (GICP).

2.5.1 Iteratieve algoritmes: Iterative Closest Point (ICP) en varianten

Bij iteratieve registratiealgoritmes wordt de translatie en rotatie iteratief berekend tussen twee puntenwolken. Het algoritme ICP en de hierop gebaseerde algoritmes berekenen deze parameters aan de hand van vijf verschillende stappen:

1. Zoek voor elk punt in de oorspronkelijke puntenwolk het dichtstbijzijnde punt in de verplaatste puntenwolk
2. Filter slecht gevonden corresponderende punten weg (optioneel)
3. Bereken de registratie
4. Pas de gevonden transformatie toe op de oorspronkelijke puntenwolk
5. Controleer of een convergentiecriterium is bereikt?

   5.1 Nee: nieuwe iteratie (stap 1). Ja: transformatie gevonden

2.5.1.1 ICP

Het algoritme Iterative Closest Point (ICP) werd een eerste keer uitvoerig beschreven door Besl en McKay in 1992[8]. Het doel van het Iterative Closest Point algoritme is een kostfunctie te minimaliseren. Deze kostfunctie wordt gegeven door volgende formule:

$$\sum_i w_i ||T \cdot b_i - m_i||^2 \quad (6)$$

Deze functie berekent de afstand van alle punten naar hun dichtstbijzijnde buur in de verplaatste puntenwolk. In deze formule staat m_i voor de punten in de verplaatste puntenwolk, b_i voor de punten in de originele puntenwolk en T voor de transformatie van de oorspronkelijke puntenwolk naar de verplaatste puntenwolk. De term w_i staat voor een gewicht dat kan worden gegeven aan ieder puntenpaar.

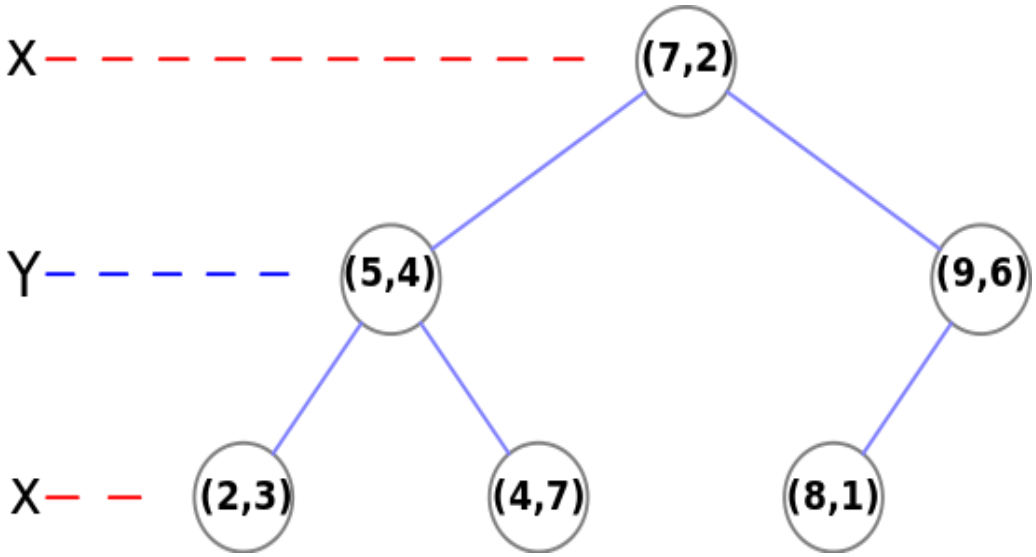
Dit algoritme volgt de stappen beschreven in 2.5.1 om de rotatie en translatie tussen twee puntenwolken te berekenen. Deze stappen worden verder in deze thesis toegelicht.

Stap 1: k-Nearest Neighbor search

In de eerste stap wordt de dichtstbijzijnde buur in de verplaatste puntenwolk gezocht voor ieder punt van de oorspronkelijke puntenwolk. De dichtstbijzijnde buur is het punt waarvan de euclidische afstand tot het andere punt het kleinst is. Het algoritme waarmee dit gebeurt is k-Nearest Neighbor (k-NN). Hiervoor wordt gebruikt gemaakt van een kd-tree om de verplaatste puntenwolk in op te slaan omdat het brute force checken van elk punt computationeel zwaar is. Een kd-tree is een k-dimensionale tree die de punten van een puntenwolk bevat. Voor puntenwolken wordt gebruik gemaakt van een 3-dimensionale tree. Het opstellen van een kd-tree met k = 3 van een puntenwolk gaat als volgt:

1) Bereken de mediaan van de x-waardes van de puntenwolk. Maak van het punt met deze mediaanwaarde een node van de boom. Zet alle punten met een kleinere x-waarde links van deze node en alle punten met een grotere x-waarde rechts van deze node
2) Zoek in de linker en rechterkant van iedere node de mediaan van de y-waarde. Maak van het punt met deze mediaanwaarde een node van de boom. Zet alle overgebleven punten met een kleinere y-waarde dan de nodes links van deze nodes en alle punten met een grotere y-waarde rechts van deze nodes
3) Zoek in de linker en rechterkant van iedere node de mediaan van de z-waarde. Maak van het punt met deze mediaanwaarde een node van de boom. Zet alle punten met een kleinere z-waarde links van deze node en alle punten met een grotere z-waarde rechts van deze node
4) Herhaal stappen 1 tot 3 tot alle punten zijn gesorteerd

In figuur 10 wordt een kd-tree met k=2 grafisch voorgesteld. In eerste instantie wordt de mediaanwaarde van de x-waardes genomen. De x-waardes zijn: 2, 4, 5, 7, 8 en 9. De mediaan van deze reeks getallen is 5 of 7. Er wordt geopteerd voor het grootste van deze twee getallen in deze figuur. Alle punten met een x-waarde kleiner dan 7 worden links van het punt met x-waarde 7 gezet en alle punten met een x-waarde groter dan 7 worden rechts gezet. In de tweede stap wordt zowel links als rechts van de node van de vorige stap de mediaanwaarde van de y-waarde genomen om een nieuwe node te bekomen. In de linkerkant is deze waarde 4 en in de rechterkant is deze waarde 6. Met de punten met deze waardes worden nieuwe nodes gemaakt. Links van deze nodes worden de punten met een kleinere y-waarde gezet en rechts de punten met een grotere y-waarde. Na deze stap is de tree gevormd.



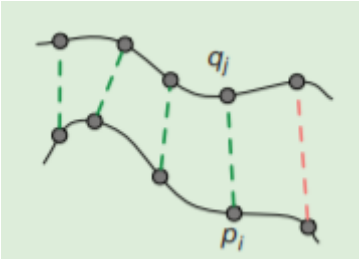
Figuur 10: Visuele voorstelling kd-tree [9]

Stap 2: filtering

Wanneer de punten die volgens het algoritme corresponderen in werkelijkheid slecht gecorrespondeerd zijn kan dit leiden tot een slechte registratie. In de tweede stap worden de corresponderende punten daardoor optioneel gefilterd. Hiervoor zijn er verschillende mogelijkheden. Enkele veelgebruikte technieken zijn de volgende:

- **Filteren op basis van afstand**

  Wanneer filteren op basis van afstand wordt gebruikt worden alle corresponderende punten waartussen de afstand groter is dan een opgegeven threshold waarde weggefilterd. Het nadeel van deze filter is dat de vaste waarde die gekozen wordt niet te groot mag zijn want dan verliest de filter zijn nut.



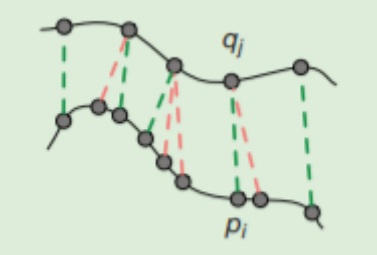
Figuur 11: Filtering op basis van vaste afstand [10]

- **Filteren op basis van mediaanafstand**

  Bij deze filterening techniek wordt geen vaste waarde gebruikt als threshold om corresponderende punten te filteren maar wordt de mediaanwaarde van de afstanden tussen alle corresponderende punten genomen. Het voordeel van deze techniek t.o.v. filteren op basis van een vaste afstand is dat de afstand waarop gefilterd wordt verkleint na iedere iteratie waarbij de berekende transformatie dichter komt bij de werkelijke transformatie. Een voordeel van deze techniek is dat deze naarmate de berekende transformatie beter wordt de outliers efficiënter worden verwijderd. Een nadeel van deze techniek is dat de registratie te vroeg kan worden afgebroken wanneer de mediaanafstand te klein wordt terwijl een deel van de puntenwolken nog niet volledig overlapt.

- **Filteren van dubbele matches**

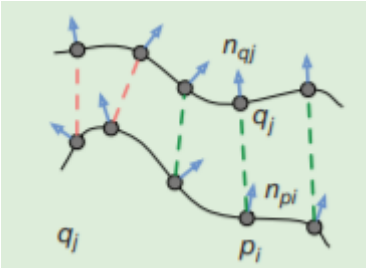
  Ieder punt in de oorspronkelijke puntenwolk wordt gematcht met zijn dichtste buur in de verplaatste puntenwolk. Het kan voorkomen dat een punt in de verplaatste puntenwolk wordt gematcht met meerdere punten uit de oorspronkelijke puntenwolk. Bij deze filtering techniek wordt enkel de het corresponderende paar met de kleinste euclidische afstand behouden. Deze filter is gebaseerd op de gedachte dat ieder punt in de oorspronkelijke puntenwolk maar één overeenkomstig punt heeft in de verplaatste puntenwolk. Hierdoor zal de registratie sneller convergeren. Een nadeel van deze techniek is dat de registratie vroegtijdig kan worden afgebroken wanneer beide puntenwolken niet evenveel punten bevatten.



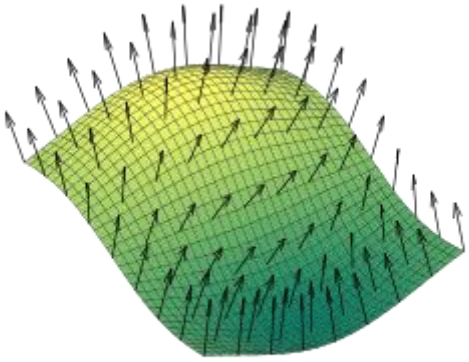
Figuur 12: Filteren van dubbele matches [10]

- **Filteren gebaseerd op normaalrichting van punten**

  Ieder punt in een puntenwolk heeft een normaalvector die loodrecht staat op het raakvlak aan dat punt in de puntenwolk. Deze normaalvector wordt berekend aan de hand van de buren van ieder punt. Een voorbeeld van een normaalvectoren wordt gegeven in figuur 14. Bij het filteren op basis van normaalrichting worden de corresponderende punten waarvan de hoek tussen de normaalvectoren groter is dan een opgegeven threshold waarde weggefilterd. Een mogelijk nadeel van deze filter is dat de berekening van de normalen kan beïnvloed worden door ruis waardoor goede puntenparen worden weggefilterd of verkeerde puntenparen worden aanvaard. Een ander mogelijk nadeel is dat er wordt gewerkt met een vaste waarde en deze bijgevolg zorgvuldig moet worden bepaald.



Figuur 13: Filteren op basis van normaalrichting [10]



Figuur 14: Normaalvectoren op een oppervlak [11]

Stap 3: zoeken van transformatie van corresponderende punten

In de derde stap wordt de registratie berekend. Hierbij worden de translatie en rotatie matrices die de transformatie tussen de oorspronkelijke puntenwolk en verplaatste puntenwolk bepalen gezocht. De berekening van deze transformatie kan wiskundig worden voorgesteld door volgende formule:

$$(q, d) = Q(P, Y) \quad (7)$$

In deze formule staat q voor de rotatieparameters, d voor de translatieparameters, P voor de punten uit de oorspronkelijke puntenwolk en Y voor de dichtstbijzijnde buren in stap 1. De berekening van de transformatieparameters q en d, die wordt voorgesteld door de letter Q, kan op verschillende manieren aangepakt worden. Enkele technieken die hiervoor worden gebruikt zijn Single Value Decomposition en BFGS.

Stap 4: transformatie van puntenwolk

In de vierde stap wordt de gevonden transformatie toegepast op de originele puntenwolk. Het transformeren van een puntenwolk is een veelgebruikte techniek. De transformatiematrix van een 3D puntenwolk bestaat typisch uit twee delen: een rotatie rond de de x-y- en z-as en een translatie in de x-, y- en z-richting.

De rotatiematrix wordt gegeven door de volgende formule:

$$R = \begin{pmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} \cos\beta & 0 & \sin\beta \\ 0 & 1 & 0 \\ -\sin\beta & 0 & \cos\beta \end{pmatrix} \begin{pmatrix} 1 & 0 & 0 \\ 0 & \cos\alpha & \sin\alpha \\ 0 & -\sin\alpha & \cos\alpha \end{pmatrix} \quad (8)$$

Waarbij alpha de hoek tussen de x- en y-as is, beta de hoek tussen de x- en z-as is en theta de hoek tussen de y- en z-as is.

De translatie wordt gegeven door:

$$T = \begin{bmatrix} t_x \\ t_y \\ t_z \end{bmatrix} \quad (9)$$

Het combineren van deze twee matrices geeft volgende homogene matrix:

$$M = \begin{bmatrix} r_1 & r_2 & r_3 & t_x \\ r_4 & r_5 & r_6 & t_y \\ r_7 & r_8 & r_9 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (10)$$

Elk punt van de puntenwolk wordt met deze matrix vermenigvuldigd en zo wordt het getransformeerde punt bekomen. Dit heeft volgende formulevorm:

$$P' = M.P \quad (11)$$

Waarbij P' de puntenwolk is na transformatie volgens M en P de oorspronkelijke puntenwolk is.

Stap 5: evaluatie van transformatie

In deze laatste stap wordt geëvalueerd of een van de convergentiecriteria bereikt is. Convergentiecriteria zijn parameters die aangeven wanneer de gevonden transformatie acceptabel is volgens de gebruiker. Het algoritme kan rekening houden met één of meerdere convergentiecriteria. Indien nog geen van de convergentiecriteria bereikt is keert het algoritme terug naar stap 1 en doet dit iteratief tot een van de convergentiecriteria wordt bereikt.

De iteratieve algoritmes hebben de volgende convergentiecriteria:

- **Absolute mean square error**

  Bij dit criterium stopt het algoritme wanneer de opgegeven mean square error wordt bereikt. Het nadeel van dit criterium alleen te gebruiken is dat als er een te kleine waarde gekozen wordt, het theoretisch oneindig lang kan duren voordat het algoritme convergeert. Een te grote waarde kan echter zorgen voor te vroeg afbreken van het algoritme.

- **Maximaal aantal iteraties**

  Dit criterium bepaalt na hoeveel iteraties het algoritme moet stoppen. Het aantal benodigde transformaties om de juiste transformatie te vinden is afhankelijk van hoe ver de twee puntenwolken die dienen gealigneerd te worden uit elkaar liggen. Ook het passeren langs een lokaal of globaal minimum bepaalt hoeveel iteraties er worden uitgevoerd. Een nadeel van dit criterium alleen te gebruiken is dat bij een te klein gekozen waarde de registratie te vroeg kan worden beëindigd en bij een te grote waarde de registratie te lang kan blijven doorgaan terwijl het aan het oscilleren is rond en globaal of lokaal minimum.

- **Absolute transformatie threshold**

  De iteraties worden gestopt wanneer de geschatte transformatie te ver verwijderd is van de initiële transformatie. Dit is een criterium voor wanneer de registratie divergeert. De intuïtie hierachter is dat twee puntenwolken die gealigneerd dienen te worden binnen een bepaalde bereik van elkaar liggen. Een nadeel van dit criterium alleen te gebruiken is dat bij een te kleine waarde de iteratie te vroeg kan worden afgebroken.

- **Relatieve transformatie threshold**

  Dit criterium specifieert het minimum transformatie verschil tussen twee iteraties dat wordt gezien als klein genoeg opdat het algoritme is geconvergeerd. Wanneer het verschil in transformatie kleiner is dan deze waarde is het algoritme dus geconvergeerd. Een nadeel van dit criterium alleen te gebruiken is dat de registratie kan eindigen bij een lokaal minimum.

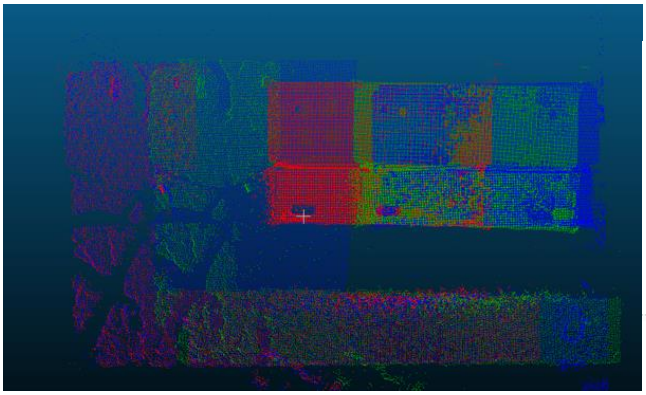
- **Maximaal aantal gelijkaardige iteraties**

  Het kan lijken dat het algoritme niet convergeert terwijl het mogelijk rond een lokaal minimum oscilleert. Dit criterium zorgt ervoor dat na het opgegeven aantal oscillaties rond dit minimum het algoritme stopt.

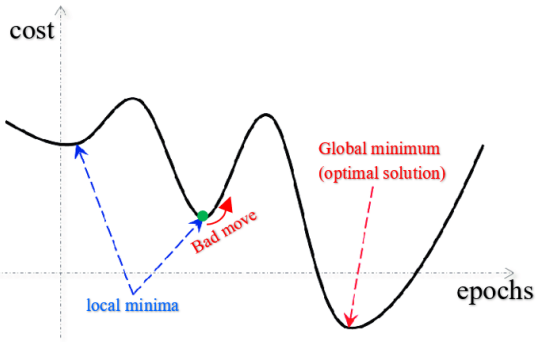
- **Relatieve mean square error**

  Dit criterium is gelijkaardig aan dat van de relatieve transformatie threshold. Wanneer de mean square error tussen 2 iteraties kleiner is dan de opgegeven waarde is het algoritme geconvergeerd. Een nadeel van dit criterium alleen te gebruiken is dat de registratie kan eindigen bij een lokaal minimum.

Bij enkele van bovengenoemde criteria werd aangehaald dat ze mogelijk zorgen voor het stoppen van de registratie bij een lokaal minimum. Een voorbeeld van een lokaal minimum wordt afgebeeld in figuur 15. In deze figuur is zichtbaar dat de getransformeerde puntenwolk (groen) niet volledig gealigneerd is met de verplaatste puntenwolk (blauw). De reden voor het eindigen in een lokaal minimum is dat bij een volgende iteratiestap de kostfunctie een hogere waarde heeft dan bij de vorige iteratie. Hierdoor zal de daaropvolgende iteratiestap de vorige berekende registratie ongedaan maken. Dit wordt voorgesteld in figuur 16.



Figuur 15: Lokaal minimum registratie



Figuur 16: Lokaal minimum voorstelling

Doordat elk criterium zijn nadelen heeft als ze individueel worden gebruikt, wordt er meestal gebruik gemaakt van een combinatie van bovenstaande criteria. Een maximaal aantal iteraties wordt in de meeste gevallen gebruikt omdat men niet eindeloos de registratie wil berekenen.

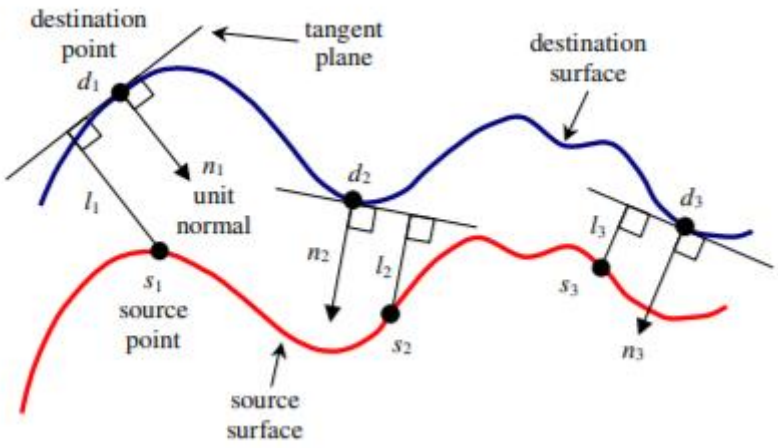
2.5.1.2 Iterative Closest Point with Normals (NICP)

Het algoritme NICP is een variant op het Iterative Closest Point algoritme. Het werd door Chen en Medioni voorgesteld in 1991 [12]. Een voordeel van dit algoritme volgens hun is dat vlakke regio's langs elkaar kunnen schuiven.

De stappen die in dit algoritme worden uitgevoerd sterk gelijkend op de stappen van Iterative Closest Point. Het verschil zit in de kostfunctie die wordt geminimaliseerd tijdens de iteraties. Waar bij ICP de afstand tussen twee matchende punten wordt genomen is dit bij NICP de afstand tussen het punt in de oorspronkelijke puntenwolk en het raakvlak van het punt in de verplaatste puntenwolk (zie figuur 17). Om deze reden wordt dit algoritme ook Point-To-Plane (punt tot vlak) Iterative Closest Point genoemd. De kostfunctie van NICP heeft volgende vorm:

$$\sum_i ((\mathbf{M} \cdot \mathbf{s}_i - \mathbf{d}_i) \bullet \mathbf{n}_i)^2 \quad (12)$$

In deze formule staat M voor de transformatiematrix, s_i voor de punten uit de oorspronkelijke puntenwolk, d_i voor de punten uit de verplaatste puntenwolk en n_i voor de normaalvector van de punten uit de verplaatste puntenwolk.



Figuur 17: Punt-tot-vlak afstand [13]

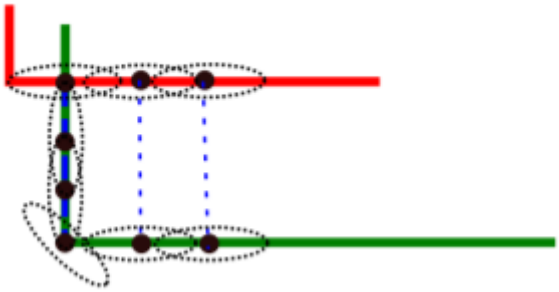
2.5.1.3 Generalized Iterative Closest Point (GICP)

Het algoritme GICP is eveneens een variant op het Iterative Closest Point algoritme. Het werd bedacht door Segal et al. in 2009 [14]. Volgens hun is het voordeel van deze techniek dat het meer robust is tegen dichtstbijzijnde buren tussen twee puntenwolken die fout gevonden zijn.

Dit algoritme volgt evenzeer de stappen die worden uitgevoerd tijdens Iterative Closest Point. Zoals bij NICP is het verschil de kostfunctie die wordt geminimaliseerd tijdens de iteraties. Bij GICP wordt de afstand tussen de twee raakvlakken genomen als afstand die geminimaliseerd dient te worden (zie figuur 18). Hierdoor wordt GICP ook wel Plane-To-Plane (vlak-tot-vlak) Iterative Closest Point genoemd. Het berekenen van de afstand tussen de twee vlakken gebeurt door een probabilistisch model te associëren met de kostfunctie van ICP. Hierbij wordt voor elk punt verondersteld dat het een hoge covariantie heeft in het vlak dat het ligt en een lage covariantie in de normaalrichting op dat vlak. De kostfunctie heeft volgende vorm:

$$\sum_i d_i^{(\mathbf{T})^T} (C_i^B + \mathbf{T} C_i^A \mathbf{T}^T)^{-1} d_i^{(\mathbf{T})} \tag{13}$$

In deze formule staat $d_i^{(T)}$ voor b_i - T*a_i waarbij b_i de punten zijn van de verplaatste puntenwolk en a_i de punten zijn van den de oorspronkelijke puntenwolk. C^B en C^A staan respectievelijk voor de covariantiematrix van de verplaatste puntenwolk en de oorspronkelijke puntenwolk.



Figuur 18: Vlak-tot-vlak afstand [14]

2.5.2 Normal Distribution Transform (NDT)

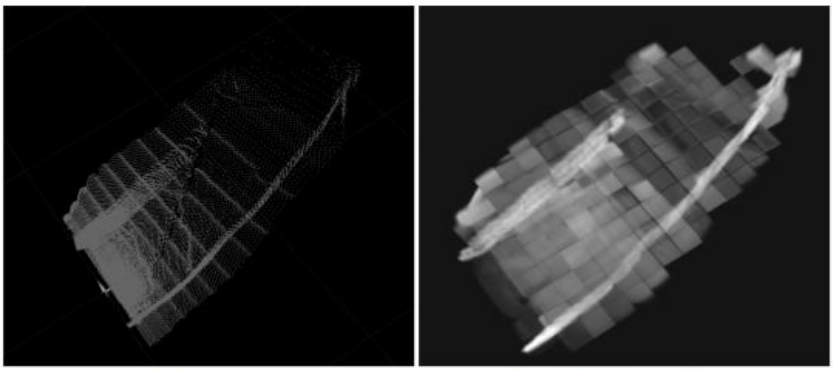
Het Normal Distribution Transform (NDT) algoritme werd de eerste keer beschreven door Biber en Straßer in 2003 [15].

Normal Distribution Transform is een iteratief algoritme waarbij de puntenwolk wordt opgedeeld in cellen. Elke cel is een kubus met een gegeven dimensie. Voor elke kubus wordt een probabiliteitsdichtheidfunctie (PDF) opgesteld. Deze functie beschrijft de kans om een sample te meten op een 3D punt x in deze cel. Deze functie heeft volgende vorm:

$$p(\vec{x}) = \frac{1}{(2\pi)^{D/2}\sqrt{|\boldsymbol{\Sigma}|}} \exp\left(-\frac{(\vec{x}-\vec{\mu})^{\mathrm{T}}\boldsymbol{\Sigma}^{-1}(\vec{x}-\vec{\mu})}{2}\right) \quad (14)$$

μ and Σ zijn in deze formule respectievelijk de vector met het gemiddelde van de punten in de gegeven cel en de covariantiematrix van de punten in de gegeven cel. De parameter D staat voor de dimensie van de NDT. De formules van μ and Σ zijn de volgende:

$$\vec{\mu} = \frac{1}{m}\sum_{k=1}^{m}\vec{y}_k,$$
$$\boldsymbol{\Sigma} = \frac{1}{m-1}\sum_{k=1}^{m}(\vec{y}_k-\vec{\mu})(\vec{y}_k-\vec{\mu})^{\mathrm{T}}, \quad (15)$$



(a) Original point cloud. (b) NDT representation.

Figuur 19: NDT [16]

Nadat deze PDF's zijn opgesteld voor elke cel in de verplaatste puntenwolk worden de translatie- en rotatieparameters geïnitialiseerd. Hierna begint het iteratieve algoritme. Het doel van dit algoritme is om de likelihood functie te maximaliseren. Deze functie heeft volgende vorm:

$$\Psi = \prod_{k=1}^{n} p(T(\vec{p},\vec{x}_k)) \quad (16)$$

De stappen die worden uitgevoerd zijn de volgende:

1. Transformeer oorspronkelijke puntenwolk met de transformatiematrix;
2. Zoek voor elk getransformeerd punt de overeenkomstige PDF;
3. Bereken de score van de huidige transformatiematrix door voor elk getransformeerd punt de waarde van de PDF te sommeren;
4. Kijk of convergentiecriterium is bereikt (Zie ICP). Indien ja, transformatie gevonden. Indien nee, bereken een nieuwe set parameters via Newton's algoritme en ga terug naar stap 1.

Het voordeel van dit algoritme t.o.v. ICP is dat de computationeel zware nearest neighbour search niet bij iedere stap toegepast dient te worden. Tevens is het mogelijk om standaard numerieke optimalisatiemethodes toe te passen die bewezen snel en betrouwbaar te zijn. Een nadeel t.o.v. ICP is dat de parameters van dit algoritme sterk schaalafhankelijk zijn.

2.6 Feature detectie

Feature detectie is een verzamelnaam voor methodes die als doel hebben om regio's met bepaalde eigenschappen binnen een afbeelding te lokaliseren. Of een bepaalde regio binnen een afbeelding beschikt over de nodige eigenschappen, hangt af van de toepassing waarin de feature detectie gebeurt. Voorbeelden van features zijn randen, hoeken, blobs, ribbels, enzovoort. Er bestaan verschillende algoritmes om deze kenmerken te extraheren uit een afbeelding. In de volgende paragrafen worden deze verder besproken. Het zoeken van deze features in afbeeldingen heeft op zichzelf beperkte toepassingen. Deze algoritmes zijn dan ook vaak de eerste stap in een bepaald proces waarbij afbeeldingen geanalyseerd en gemanipuleerd worden.

2.6.1 Harris corner detection

Het Harris hoek [18] is een detectie algoritme, dat verder bouwt op het hoek algoritme van Moravec. Formulematig ziet het algoritme er als volgt uit:

$$E_{u,v} = \sum_{x,y} w_{x,y} \, [I_{x+u,y+v} - I_{x,y}]^2 \quad (17)$$

Waarbij $w_{x,y}$ de voorstelling van het window is. Een window is een regio met een vaste afmeting, dat stapsgewijs over de hele afbeelding wordt verplaatst. Het kiezen van een geschikt window heeft invloed op de waargenomen ruis. Zo geeft een Gaussiaanse cirkel bijvoorbeeld een beter resultaat dan een rechthoekig window. $I_{x+u,y+v}$ stelt de verplaatste intensiteit voor en $I_{x,y}$ de intensiteit van de huidige pixel die gecontroleerd wordt. De intensiteit van een pixel is de grijswaarde die aangeeft hoe donker of hoe licht de pixel is. Een lage waarde wijst op een donkere pixel en een hoge waarde wijst op een lichte pixel. Het doel is om de functie $E_{u,v}$ te maximaliseren om zo een hoek te detecteren. Een hoek wordt gedefinieerd als de regio waar twee randen elkaar kruisen en als gevolg kan hier een plotse verandering in intensiteit waargenomen worden. De functie kan verder vereenvoudigd worden tot:

$$E(u,v) = (u,v)M(u,v)^T \quad (18)$$

Waarbij:

$$M = \sum_{x,y} w(x,y)\begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} (19)$$

I_x en I_y zijn respectievelijk de afgeleiden in de x en y richting. Om te evalueren of er sprake is van een hoek, kan de volgende formule gebruikt worden:

$$R = det(M) - k(trace(M))^2$$

$$det(M) = \lambda_1 \lambda_2 \; en \; trace(M) = \lambda_1 + \lambda_2 \; (20)$$

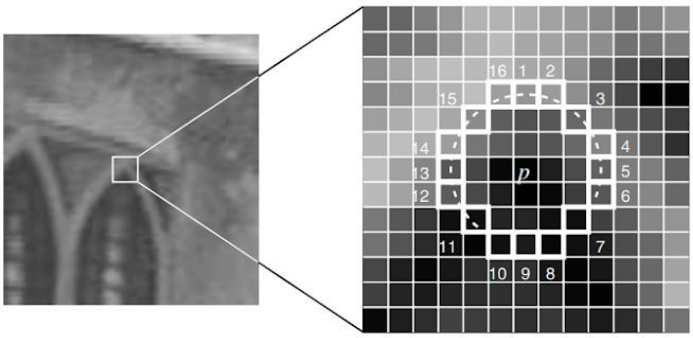
λ_1 en λ_2 zijn de eigenwaarden van matrix M. Vervolgens is het belangrijk om de bekomen waarde voor R juist te interpreteren. Bij een kleine waarde voor λ_1 en λ_2 gaat $|R|$ klein zijn en wijst dit op een vlak gebied. Wanneer één van de eigenwaarden significant groter is dan de andere zal R kleiner zijn dan nul en betekent dit dat het punt zich op een rand bevindt. Als laatste kunnen de eigenwaarden ongeveer even groot zijn, dan gaat R groot zijn en wordt dit geïnterpreteerd als een hoek.

2.6.2 Features from Accelerated Segment Test

FAST oftewel Features from Accelerated Segment Test [19] is ook een hoek detectie algoritme, dat op een snellere manier hoek detectie tracht uit te voeren dan de traditionele algoritmes zoals bijvoorbeeld het Harris algoritme. Het detecteren van een hoek kan opgedeeld worden in een aantal stappen:

- Drempelwaarde t bepalen
- 16 pixel grootte cirkel selecteren rond p
- Pixelintensiteit van de 16 pixels vergelijken met het bereik

Een centrale pixel p wordt gekozen en de drempelwaarde t wordt vastgelegd. Het intensiteitsbereik van p wordt gedefinieerd als $[I_p - t, I_p + t]$. Vervolgens wordt een cirkel van 16 pixels rondom p bestudeerd, zoals voorgesteld in figuur 20. Wanneer 12 aaneengesloten pixels (n) buiten het vooropgestelde bereik liggen, is er sprake van een hoek. Om de snelheid te verhogen worden de pixels in een bepaalde volgorde getest. Eerst worden pixels 1, 5, 9 en 13 getest. Indien minstens drie van de vier geteste pixels buiten het intensiteit bereik liggen, is er sprake van een hoek. Om de nauwkeurigheid te verhogen is het mogelijk om de pixels die vastgesteld werden als een hoek verder te testen en nagaan of deze ook voldoen aan n >= 12.



Figuur 20: Gekozen centrale pixel met 16 pixel grootte cirkel er rond [19, p. 434]

De test is door deze methode significant sneller dan traditionele methodes, maar daarbij komen wel andere nadelen bij kijken. Zo kan het zijn dat er vals positieve hoeken worden vastgesteld bij de hoge snelheid variant, dus m.a.w. een situatie waarbij $n < 12$. De keuze en ordening van de snelle test, dus het enkel testen van pixels 1, 5, 9 en 13, is gebaseerd op impliciete veronderstellingen over de distributie van features. Een andere nadeel is dat resultaten van de hoge snelheidstest achteraf nietig verklaard kunnen worden. Het laatste nadeel is dat meerdere features gedetecteerd worden die langs elkaar liggen. Dit leidt tot extra features, die geen meerwaarde bieden over het interpreteren van de scène. Ook heeft dit als gevolg dat er veel meer punten vergeleken moeten worden tijdens het matchen van twee scènes en zal het matchingproces dus langer duren.

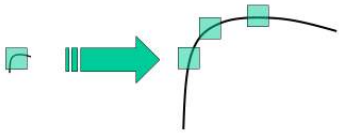
De eerste drie aangehaalde nadelen kunnen verholpen worden door gebruik te maken van machine learning. Machine learning houdt in dat op basis van aangeleverde data, een model getraind wordt. Dit model is dan in staat om voor nieuwe data voorspellingen te doen. In dit geval werd een model gecreëerd dat aangeeft of een pixel al dan niet een hoek is. Om een zo goed mogelijk resultaat te bekomen uit de training, is het aangeraden om gelijkaardige afbeeldingen te gebruiken van de omgeving waarin het algoritme later gebruikt zal worden. Tijdens het trainen wordt dezelfde methode gebruikt als eerder uitgelegd om aan hoek detectie te doen. Het enige verschil is dat nu wel alle 12 pixels getest worden. Hierdoor wordt er een nauwkeuriger model verkregen. Het uiteindelijke resultaat van de training is dan een boomstructuur, die aangeeft of de geteste pixel een hoek is of niet.

Om het vierde aangehaalde probleem op te lossen kan er gewerkt worden met een score functie. Hierbij wordt voor elke hoek een score V berekend, waarbij V de som is van het absolute verschil tussen P en de 16 omliggende pixels. Vervolgens worden de scores van aanliggende features met elkaar vergeleken. De feature met de lagere score van de twee wordt dan uiteindelijk verwijdert.

2.6.3 SIFT

Scale Invariant Feature Transform (SIFT) [20] is een feature detectie algoritme dat blobs, ook wel keypoints genoemd, extraheert uit afbeeldingen. In tegenstelling tot het Harris en Fast algoritme is dit algoritme in staat om dezelfde keypoints te vinden bij verschillende schalen. Eerder aangehaalde algoritmes als Harris en FAST zijn niet schaal invariant. Dit komt omdat deze algoritmes enkel de coördinaten bijhouden van de features. Schaal invariantie houdt dus in dat het algoritme correct blijft presteren ongeacht een verandering in schaal. Om dit probleem aan te pakken werd dus SIFT voorgesteld in de paper “Distinctive Image Features from Scale-Invariant Keypoints” door G. Lowe in 2004. Het algoritme is in staat om dezelfde hoek te detecteren tussen afbeelding met een andere schaalgrootte. Dit wordt gevisualiseerd in figuur 21. De werking van het algoritme kan opgesplitst worden in vier stappen:

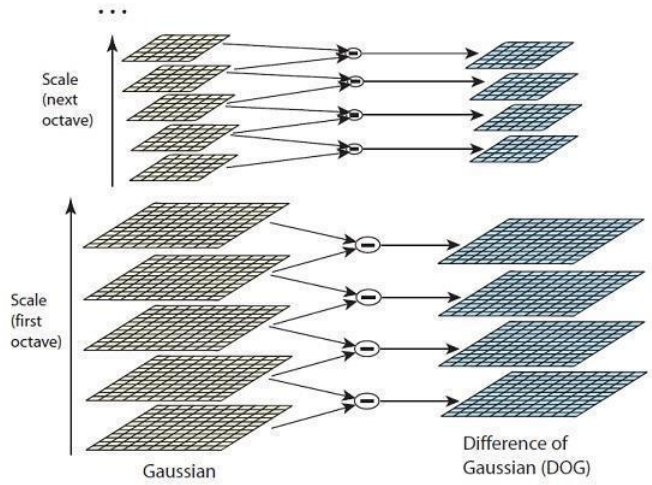
1. Schaal invariante extrema detecteren
2. Lokaliseren van keypoints binnen de afbeelding
3. De keypoints voorzien van een oriëntatie
4. Descriptor opstellen voor ieder keypoint



Figuur 21: Dezelfde hoek in een andere schaal [21]

2.6.3.1 Schaal invariante extrema detectie

In de eerste stap van het algoritme wordt de afbeelding meerdere malen onderworpen aan Gaussiaanse vervaging. Bij iedere stap van de Gaussiaanse vervaging wordt de afbeelding onduidelijker. Dit heeft onder andere als voordeel dat de hoeveelheid ruis binnen de afbeelding verminderd wordt. Vervolgens wordt de resolutie van de originele afbeelding gehalveerd. Hierna wordt deze afbeelding ook meermaals vervaagd. Dit proces herhaalt zich vervolgens meermaals. De afbeeldingen met een verschillende hoeveelheid vervaging, maar met dezelfde schaal worden octaven genoemd. In de originele paper worden vier octaven en 5 hoeveelheden aan vervaging aanbevolen om tot een zo goed mogelijk resultaat te komen. Het tweede deel van de eerste stap bestaat uit het berekenen van de differences of Gaussian (DOG). Om de DOG te bekomen, wordt het verschil in pixelintensiteit tussen twee opeenvolgende schalen berekend. Dit proces is te zien in figuur 22. Een voordeel van DOG is dat de frames nu schaal invariant zijn.

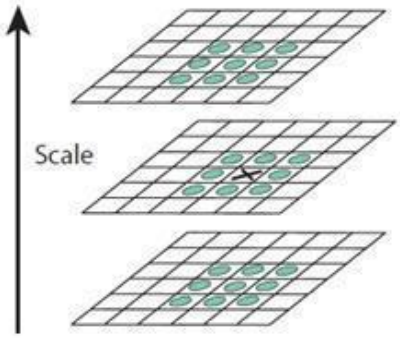


Figuur 22: Difference of Gaussian [21]

2.6.3.2 Keypoint lokalisatie

Tijdens de tweede stap worden de keypoints gevonden. Dit gebeurt aan de hand van een iteratief proces waarbij iedere pixel vergeleken wordt zijn naburige pixels. Niet enkel de buren binnen dezelfde schaal worden vergeleken, maar ook de buren in schaal erboven en eronder. Als het huidige punt de hoogste of laagste intensiteitswaarde van al zijn buren heeft, wordt dit punt beschouwd als een keypoint. Afbeelding 23 visualiseert de werking hiervan. Om de nauwkeurigheid te verhogen van de gedetecteerde keypoints, wordt er een Taylor expansie gebruikt om subpixels te genereren. Een nieuw minima of maxima wordt vervolgens gevonden in de omgeving van het voorgaande keypoint. Deze nieuwe subpixel keypoints worden hierna gefilterd op twee verschillende manieren. Eerst wordt de grootte van de intensiteit van de subpixel vergeleken met een vooropgestelde drempelwaarde. Als de waarde lager is dan de drempelwaarde wordt de keypoint verworpen. De tweede filter die toegepast wordt, gaat op dezelfde manier te werk als het Harris algoritme. De eigenwaardes worden berekend en

gebaseerd op deze resultaten wordt de locatie van deze regio bepaald. Als het resultaat een hoek is, wordt het keypoint behouden en anders wordt het verworpen.



Figuur 23: Vinden van lokale minima/maxima [21]

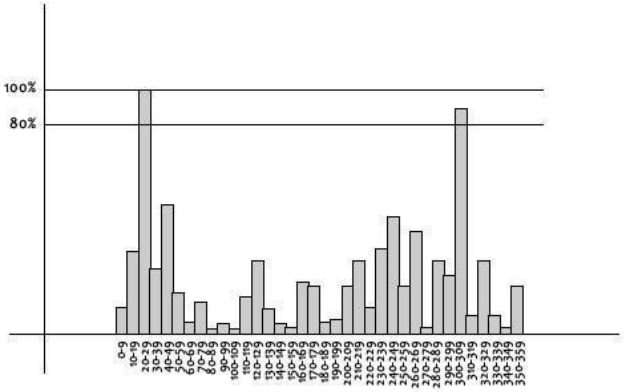
2.6.3.3 Oriëntatie toewijzen

Bij de derde stap van het algoritme wordt de oriëntatie van de keypoints bepaald. Als eerste stap wordt een regio rondom het keypoint geselecteerd. Een grotere schaal leidt tot een grotere regio. Van deze regio's worden de groottes van de gradiënt en de oriëntaties berekend. De formules voor het vinden van de gradiënt en de oriëntatie zijn:

$$m(x,y) = \sqrt{(L(x+1,y) - L(x-1,y))^2 + (L(x,y+1) - L(x,y-1))^2}$$
$$\theta(x,y) = tan^{-1}((L(x,y+1) - L(x,y-1))/(L(x+1,y) - L(x-1,y))), \quad (21)$$

waarbij $L(x,y)$ duidt op de intensiteit voor een pixel met coördinaten (x,y) behorende tot de Guassiaans vervaagde afbeelding L.

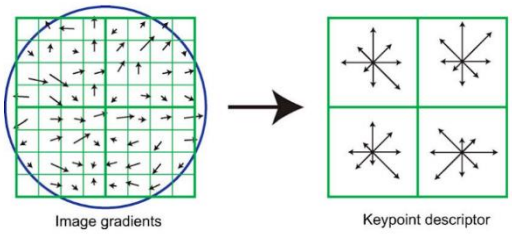
Hierna wordt een histogram opgesteld van de berekende waardes. Het 360 graden histogram wordt onderverdeeld in 36 bin van elk 10 graden, zoals te zien op figuur 24. De bin met het hoogste percentage wordt geselecteerd als oriëntatie van het keypoint. Als er meerdere bins met een percentage boven de 80 procent zijn, wordt er voor ieder een nieuw keypoint aangemaakt met dezelfde locatie en schaal, maar met een andere oriëntatie.



Figuur 24: 360 graden histogram [22]

2.6.3.4 Keypoint descriptor

Als laatste wordt een keypoint descriptor aangemaakt voor ieder keypoint. Dit als doel om een onderscheid te kunnen maken tussen verschillende keypoints. Om te beginnen wordt een window van 16x16 genomen rondom het keypoint. Vervolgens wordt elke cel opnieuw opgesplitst in een 4x4 window. De gradiënt en de oriëntatie wordt daarna voor elk 4x4 window berekend. Dit wordt visueel gepresenteerd in figuur 25. In tegenstelling tot stap 3, wordt het histogram opgesplitst in 8 bins. Dit resulteert in 128 bin waardes per keypoint. Deze waarden worden dan gerepresenteerd als een vector. Deze vector wordt dan de keypoint descriptor genoemd.

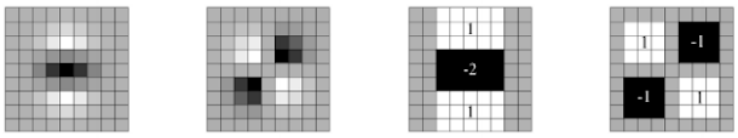


Figuur 25: Overgang gradient matrix naar keypoint descriptor [22]

2.6.4 SURF

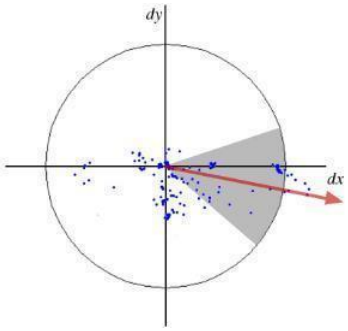
Speeded Up Robust Features (SURF) [23] is een variant die verder bouwt op het SIFT algoritme. Het doel van SURF was om een snellere variant te creëren van het SIFT algoritme, zodat het geschikt is voor real time toepassingen. Om een sneller algoritme te bekomen, werd er gebruik gemaakt van een andere methode voor het opstellen van de schaalruimte, voor het bepalen van de oriëntatie en voor het opmaken van de descriptors.

Bij SIFT werd de laplaciaan van Gauss benaderd met DOG voor het vinden van schaal ruimtes. SURF daarentegen benadert de laplaciaan van Gauss door middel van box filters. Figuur 26 weergeeft de functionaliteit van zo een box filter. Een van de voordelen van deze werkwijze is dat de convolutie met box filters gemakkelijk berekend kan worden met de hulp van integral images. Dit proces kan parallel gebeuren voor verschillende schalen.



Figuur 26: Box filters [23, p. 408]

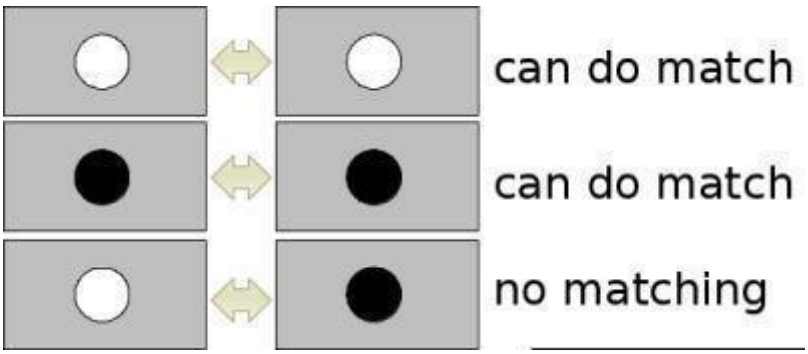
Bij het bepalen van de oriëntatie van een keypoint maakt SURF gebruik van wavelet reacties in de horizontale en verticale richting voor een omgeving met een grootte van 6*s*, waarbij *s* de huidige schaal voorstelt. Adequate gaussiaanse gewichten worden er ook op toegepast. Dit wordt vervolgens geplot zoals weergeven in figuur 27. De meest overheersende richting wordt gevonden door het berekenen van de som van alle reacties binnen een bewegend window van 60 graden. Het vinden van de wavelet reactie kan gebeuren door gebruik van integral images ongeacht de schaal. Indien de toepassing het niet vereist kan de rotatie invariantie achterwege gelaten worden. Dit resulteert in het sneller vinden van keypoints. Deze variant van SURF wordt ook wel Upright-SURF of U-SURF genoemd.



Figuur 27: Window voor een wavelet reactie [24]

Voor het opstellen van de feature descriptor maakt SURF gebruik van wavelet reacties in zowel de horizontale en verticale richting. Een regio van $20s$x$20s$ rond het keypoint wordt genomen. Deze wordt opgedeeld in subregio's van 4x4. Voor iedere subregio wordt de horizontale en verticale wavelet reactie genomen en hieruit wordt een vector afgeleid. De gevonden vector heeft een dimensie van 64. Deze is dus lager dan bij SIFT, wat leidt tot een hogere snelheid van het berekenen en matchen. Er is ook een alternatieve variant waarbij een vector met een dimensie van 128 gevonden wordt. Ondanks de verdubbeling aan features, verhoogt de berekeningstijd marginaal.

Een laatste verbetering maakt gebruikt van het teken van de laplaciaan van het onderliggende keypoint. Hiervoor is geen extra rekenkracht nodig, aangezien dit al berekend wordt tijdens de detectiefase. Door het teken kan er een onderscheid gemaakt worden tussen heldere keypoints met een donkere achtergrond en keypoints die dezelfde intensiteit hebben als hun achtergrond. Tijdens het matchen worden dan enkel nog features met hetzelfde soort contrast met elkaar vergeleken. Door gebruik te maken van de informatie kan het algoritme sneller presteren zonder dat er extra berekeningen nodig zijn.



Figuur 28: Checken van hetzelfde contrast [24]

2.6.5 BRIEF

Het SIFT algoritme vereist een grote computationele kost, wat leidt tot een algoritme dat minder efficiënt is op vlak van snelheid. De aanpak van SURF was om dit probleem te verhelpen door andere benaderingen te gebruiken voor het detecteren van de keypoints en voor het opstellen van de descriptors. Een andere alternatief is om de descriptors anders voor te stellen, zodat deze minder groot zijn en leiden tot een sneller matching proces. Een voorbeeld van zo een methode is Binary Robust Independent Elementary Features algoritme (BRIEF) [25]. Het is dus geen algoritme dat keypoints vindt, maar één dat de descriptors voor de gevonden keypoints aanmaakt.

BRIEF bestaat uit drie verschillende stappen:
Ten eerste wordt de grootte van de window gekozen. De pixels die binnen dit window vallen worden onderworpen aan smoothing. Tijdens het smoothing proces wordt de gemiddelde pixelintensiteit van een geselecteerde pixel en de omliggende pixels berekend. De resulterende waarde wordt dan de nieuwe waarde van de geselecteerde pixel. Dit proces wordt op iedere pixel toegepaste en heeft als gevolg dat de overgangen van donker naar lichte regio's geleidelijker verloopt. Het gevolg hiervan is dat de hoeveelheid ruis vermindert, wat leidt tot een verbetering van resultaten bij moeilijkere matching scenario's.
Vervolgens worden binnen de geselecteerde window n_d aantal (x,y)-locatie paren gekozen, waarbij n_d gelijk gesteld kan worden aan 128, 256 of 512. De waarde van n_d wordt ook wel uitgedrukt in bytes en om dit te verkrijgen moet n_d door acht gedeeld worden.
Als laatste stap worden de intensiteiten van de pixel paren vergeleken. Dit wordt dan als volgt uitgedrukt:

$$\tau(p; x, y) \coloneqq \{1 \; als \; p(x) < p(y) \; 0 \; anders \quad (22)$$

Hierbij stelt τ de test voor voor window p en is $p(x)$ de intensiteit van de pixel. Deze n_d testen samen vormen dan een binaire string die dan dient als feature descriptor voor het gekozen keypoint. Deze binaire strings kunnen dan gematcht worden door middel van een Hamming test. Tijdens deze test worden bij beide strings de bits op dezelfde positie met elkaar vergeleken. Het aantal bits dat verschillend zijn, bepaald de resultaat van deze test. Wanneer er geen bits verschillen wijst dit op een perfecte match.

2.6.6 ORB

SIFT en SURF zijn beide zeer goede algoritmes om keypoints te identificeren. Het nadeel is dat SIFT relatief traag is en dat SURF gepatenteerd is. Om dit te omzeilen werd er een gelijkaardig presterend algoritme bedacht genaamd Oriented FAST and Rotated BRIEF (ORB) [26]. Het algoritme is een combinatie van enerzijds FAST en anderzijds BRIEF. Er werden meerdere aanpassingen aangebracht aan deze methodes om de prestatie te verhogen.

Als eerste worden de keypoints gevonden aan de hand van het FAST algoritme. Vervolgens wordt Harris corner gebruikt om de top N keypoints onder hen te vinden. Ook maakt het gebruik van een piramide structuur om features op verschillende schalen te genereren. Een van de verschillen tussen FAST en SURF is dat FAST geen oriëntaties berekent. Om alsnog een oriëntatie mee te geven aan de gegenereerde FAST keypoints, maakt ORB gebruik van het zwaartepunt van de hoek. Dit houdt in dat er van uitgegaan wordt dat de intensiteit van de hoek op een bepaalde afstand ligt van zijn middelpunt. De vector tussen deze twee punten geeft dan de oriëntatie van het keypoint aan. Het berekenen van de oriëntatie gebeurt in drie stappen:
Eerst worden de momenten van het gebied van het keypoints berekend volgens de definitie van Rosin:

$$m_{pq} = \sum_{x,y} x^p y^q I(x, y) \; (23)$$

Aan de hand van de gevonden momenten wordt het zwaartepunt bepaald:

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right) \; (24)$$

De vector $\overrightarrow{OC}$ is dan verbinding tussen het centrum van de hoek en het zwaartepunt. De oriëntatie wordt dan berekend door volgende formule:

$$\theta = atan2(m_{01}, m_{10}) \text{ (25)}$$

Om de rotatie invariantie te verbeteren, wordt het moment berekend met een x en y waarde die binnen een cirkel met straal r liggen en r is hierbij de grootte van de regio.

Voor het genereren van de descriptors maakt ORB gebruik van BRIEF. Het nadeel bij BRIEF is dat het slecht presteert bij rotaties. Om dit te verhelpen geeft ORB de gevonden oriëntatie van het keypoint door, zodat de juiste richting al meegenomen kan worden tijdens het BRIEF proces. Het toewijzen van een oriëntatie aan de BRIEF descriptor gebeurt dan als volgt:

- Voor iedere feature set van n binaire testen op locatie (x_i, y_i) wordt een 2xn matrix opgesteld. Deze bevat de coördinaten van deze pixels.
- Vervolgens wordt deze matrix vermenigvuldigd met de eerder gevonden oriëntatiematrix. De bekomen matrix is hierdoor gestuurd volgens de oriëntatie van het keypoint.
- ORB rond de hoek naar een veelvoud van $2\pi/30$ af en creëert een opzoektabel van eerder berekende BRIEF patronen.

Zolang de rotatie consistent is bij verschillende zichten, wordt de juist georiënteerde matrix gebruik bij het berekenen van de descriptor.

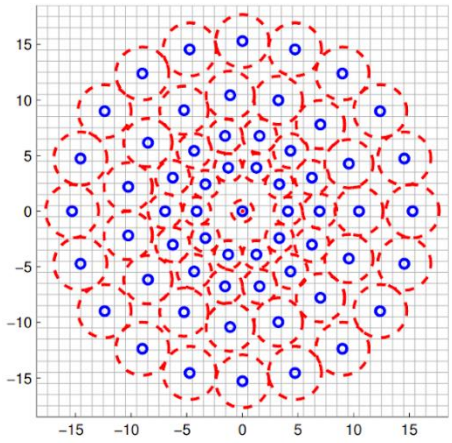
2.6.7 BRISK

Een ander alternatief voor SIFT en SURF is Binary Robust Invariant Scalable Keypoints (BRISK) [27]. Het vooropgestelde doel van dit algoritme was om een sneller alternatief te bieden voor SIFT en SURF. Daarbij moet het algoritme ook beschikken over hoog kwalitatieve keypoints en matching, die zowel schaal als rotatie invariant zijn. Net als ORB wordt er snelheidswinst geboekt door de detectie te doen door middel van een variant van het FAST algoritme en door een bit string descriptor te gebruiken, die opgebouwd wordt aan de hand van pixel intensiteit vergelijkingen. Ondanks het gelijkaardige idee zijn er toch enkele verschillen op vlak van implementatie.

Voor het vinden van features binnen de afbeelding, wordt er gebruik gemaakt van het AGAST [28] algoritme. De is een verdere verbetering van FAST, die de methode meer veralgemeend en ook versneld. Om BRISK schaal invariant te maken wordt er niet enkel een maxima gezocht binnen het beeldvlak, maar ook voor de schaalruimte. Dit wordt gedaan door gebruik te maken van de FAST score s als waarde voor hoe opvallend een feature is. Bij het opstellen van de piramide van de schaalruimte worden n octaven c_i en n intra-octaven d_i gecreëerd. De aangewezen standaard waarde voor n is 4. Het creëren van de octaven gebeurt op analoge manier als bij SIFT en dus wordt de resolutie meermaals gehalveerd. De intra-octaven bevinden zich tussen twee opeenvolgende octaven en kunnen bekomen worden door een octaaf te vermenigvuldigen met 1,5. Om de keypoints te vinden, wordt vervolgens FAST toegepast op alle octaven en intra-octaven. Daarna wordt de FAST score s berekend. Om niet verworpen te worden, moeten zowel de octaaf erboven als eronder eveneens een lagere s score hebben dan de drempelwaarde. Als laatste stap wordt het keypoint verder verfijnd door subpixels te creëren.

Het voorstellen van de descriptors wordt gedaan door middel van een binaire string. Net als bij ORB wordt de richting van het keypoint meegegeven aan de descriptor om zo rotatie invariantie te bekomen. Net als bij BRIEF wordt de bit string opgesteld door het vergelijken van

pixelintensiteiten binnen het gebied van het keypoint. Er zijn enkel wel drie fundamentele verschillen tussen de aanpak van BRISK en BRIEF. Zo gebruikt BRISK een deterministisch bemonsterpatroon, zie figuur 29, dat zorgt voor een uniforme puntdichtheid op een bepaalde afstand rond het keypoint. Ook maakt BRISK gebruik van minder punten, dit omdat een punt meermaals kan gebruikt worden in het vergelijken van de intensiteiten. Dit in tegenstelling tot BRIEF dat werkte met paren. Dit zorgt ervoor dat het vergelijkingsproces merkbaar sneller is bij BRISK. Als laatste wordt de vergelijking beperkt op basis van afstand, zodat de helderheid variaties enkel lokaal consequent moeten zijn.



Figuur 29: Bemonsterpatroon BRISK [27, p. 2551]

2.6.8 AKAZE

ORB en BRISK hebben beide een gelijkaardige aanpak om aan feature detectie en descriptie te doen, maar dit is niet de enige manier om een alternatief te bieden voor SIFT en SURF. Zo is er bijvoorbeeld ook Accelerated-KAZE (A-KAZE) [29], dat een versnelde versie is van het KAZE [30] algoritme. Bij SIFT en SURF wordt er gebruikt gemaakt van een Gaussiaanse schaalruimte en wordt er dus Gaussiaanse vervaging toegepast op de afbeelding. Hierdoor wordt dus de gehele afbeelding op een uniforme wijze vervaagd. Het nadeel hiervan is dat de randen binden de afbeelding minder uitgesproken zijn, wat een negatieve invloed heeft op de nauwkeurigheid van het algoritme. AKAZE maakt gebruik van een niet lineaire schaalruimte om deze uitdaging te overwinnen. Om deze niet lineaire schaalruimte te creëren, worden Fast Explicit Diffusion (FED) schema's gebruikt. Dit maakt dat dit algoritme sneller is dan SURF, SIFT en KAZE en dat het een betere detectie en descriptie biedt dan ORB en BRISK.

Om te beginnen wordt een set van evolutie tijden gedefinieerd. De schaalruimte bestaat uit verschillende O octaven en S sub-levels. De octaaf en sub-level indices worden gekoppeld aan de bijhorende schaal σ aan de hand van de volgende formule:

$$\sigma_i(o,s) = 2^{o+s/S}, o \in [0 \dots O-1], s \in [0 \dots S-1], i \in [0 \dots M] \quad (26)$$

Hierbij stelt M het totaal aantal gefilterde afbeeldingen voor. Vervolgens worden de verschillende lagen van de schaalruimte omgezet naar een tijdseenheid in plaats van een pixel eenheid. Dit omdat de niet lineaire diffusie filter werkt met tijdseenheden. Het omzetten gebeurt door middel van de volgende formule:

$$t_i = \frac{1}{2}\sigma_i^2, i = \{0 \dots M\} \quad (27)$$

Daarna vindt een convolutie plaats op de afbeelding om de hoeveelheid ruis te onderdrukken. De laatste input van het FED schema is de contrastwaarde λ, die gelijk is het 70% percentiel van het gradient histogram. De stappen van het FED schema worden voorgesteld via figuur 30.

Algorithm 1 Pyramidal FED approach for nonlinear diffusion filtering

Input Image L^0, contrast parameter λ, τ_{max} and set of evolution times t_i
Output Set of filtered images L^i, $i = 0 \ldots M$

for $i = 0 \rightarrow M - 1$ **do**
1. Compute diffusivity matrix $A\left(L^i\right)$
2. Set FED outer cycle time $T = t_{i+1} - t_i$
3. Compute number of FED inner steps n
4. Compute step sizes τ_j
5. Set Prior $L^{i+1,0} = L^i$

L^{i+1} = **FEDCYCLE**$(L^{i+1,0}, A\left(L^i\right), \tau_j)$
if $o_{i+1} > o_i$ **then**
Downsample L^{i+1} with mask $\left(\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right)$
$\lambda = \lambda \cdot 0.75$
end if
end for

Algorithm 2 FED Cycle

function FEDCYCLE$(L^{i+1,0}, A\left(L^i\right), \tau_j)$
for $j = 0 \rightarrow n - 1$ **do**
$L^{i+1,j+1} = \left(I + \tau_j A\left(L^i\right)\right) L^{i+1,j}$
end for
Return $L^{i+1,n}$
end function

Figuur 30: Stappenplan FED schema [29, p. 5]

Nadat de niet lineaire schaalruimte voor de afbeelding aangemaakt, vindt de feature detectie plaats. Als eerste stap wordt de determinant van de Hessiaan voor iedere afbeelding L^i berekend. De formule ziet er als volgt uit:

$$L^i_{Hessian} = \frac{\sigma_i}{2^{o^i}} \left(L^i_{xx} L^i_{yy} - L^i_{xy} L^i_{xy}\right) \quad (28)$$

Voor het berekenen van de tweede afgeleiden, wordt een Scharr filter gebruikt. Deze filter werd specifiek gekozen, omdat deze rotatie invariantie beter benadert. Daarna wordt er door middel van een 3x3 window gezocht naar maxima's. Voor ieder maxima wordt ook gecontroleerd of de laag erboven en eronder ook over een maxima beschikt. Van de overgebleven keypoints wordt in de laatste stap de locatie verfijnd door extra subpixels aan te maken rondom het keypoint.

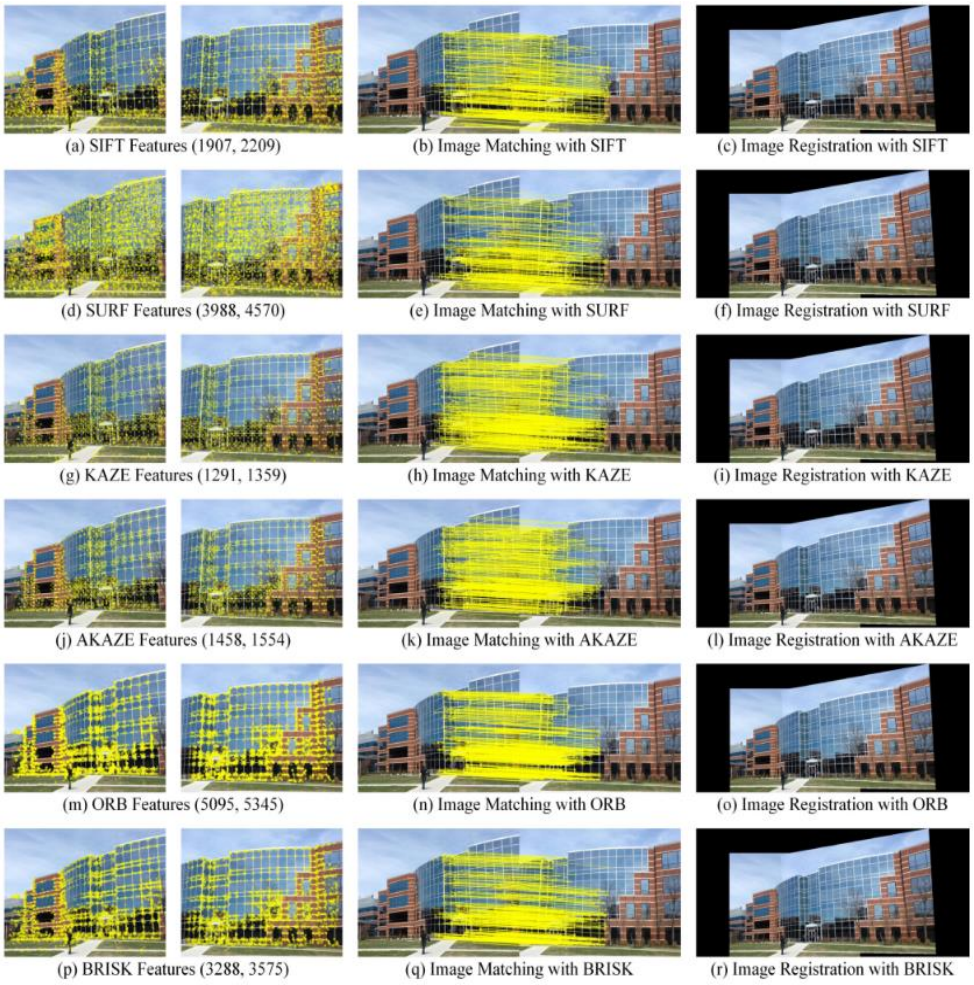
Om de gevonden keypoints te beschrijven maakt AKAZE gebruik van Modified-Local Difference Binary (M-LDB). Dit is net als BRIEF een binaire descriptor, maar verschilt op een aantal vlakken. M-LDB vergelijkt de gemiddelde intensiteit van regio's in plaats van pixels zoals bij BRIEF. Ook wordt niet enkel de intensiteit vergeleken, maar ook het gemiddelde van de horizontale en verticale afgeleide van de regio. Dit resulteert in een drie bit string voor iedere vergelijking. Het gebruiken van deze afgeleiden tijdens de vergelijking resulteert in een marginale extra computationele kost, aangezien deze al berekend worden tijdens de feature detectie stap.

2.6.9 Vergelijking algoritmes

Bij het kiezen van het geschikte algoritme om feature detectie uit te voeren, is het dus belangrijk om rekening te houden in wat voor applicatie het gebruikt zal worden. Ieder algoritme heeft dan ook zijn sterktes en zwaktes en de soort afbeelding die gebruikt wordt kan ook een invloed hebben op de prestatie van het algoritme. Vandaar deze vergelijking tussen de besproken

algoritmes. De vergelijking is gebaseerd op de resultaten uit de paper "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB and BRISK" [31]. Hierin werden de implementaties van deze algoritmes binnen OpenCV 3.0 met elkaar vergeleken.

In de paper werden er ook varianten van deze algoritmes gebruikt. Zo werden $ORB_{(1000)}$ en $BRISK_{(1000)}$ ook meegenomen in de vergelijking. De 1000 geeft aan dat maximum 1000 features gedetecteerd worden. Het lager aantal aan gedetecteerde features resulteert in een lagere nauwkeurigheid tijdens het matchen. Het voordeel is echter dat het matchen van de keypoints veel sneller voltooid is. Een ander onderscheid werd gemaakt tussen $SURF_{(64D)}$ en $SURF_{(128D)}$. De twee verschillen in het feit dat ze respectievelijk een vectorgrootte hebben van 64 en 128. Een kleinere vectorgrootte wijst op een discriptor, die beschikt over een beperktere beschrijving van het keypoint. Het voordeel is dat de matching daardoor sneller verloopt, maar de nauwkeurigheid is hierdoor minder kwalitatief. Figuur 31 visualiseert de detectie en matching voor de verschillende algoritmes en tabel 1 geeft een numeriek overzicht van de resultaten.



Figuur 31: Visuele vergelijking algoritmes [31, p. 6]

Tabel 1: Numerieke vergelijking algoritmes [31, p. 7]

Algorithm	Features Detected in the Image Pairs		Features Matched	Outliers Rejected	Feature Detection & Description Time (s)		Feature Matching Time (s)	Outlier Rejection & Homography Calculation Time (s)	Total Image Matching Time (s)
	1st Image	2nd Image			1st Image	2nd Image			
Building Dataset (Image Pair # 1)									
SIFT	1907	2209	384	51	0.1812	0.1980	0.1337	0.0057	0.5186
SURF(128D)	3988	4570	319	58	0.1657	0.1786	0.5439	0.0058	0.8940
SURF(64D)	3988	4570	612	73	0.1625	0.1734	0.2956	0.0052	0.6367
KAZE	1291	1359	465	26	0.2145	0.2113	0.0613	0.0053	0.4924
AKAZE	1458	1554	475	36	0.0695	0.0715	0.0307	0.0055	0.1772
ORB	5095	5345	854	149	0.0213	0.0220	0.1586	0.0067	0.2086
ORB(1000)	1000	1000	237	21	0.0103	0.0101	0.0138	0.0049	0.0391
BRISK	3288	3575	481	47	0.0533	0.0565	0.1236	0.0056	0.2390
BRISK(1000)	1000	1000	190	33	0.0188	0.0191	0.0158	0.0049	0.0586
Bricks Dataset (Image Pair # 2)									
SIFT	1404	1405	427	16	0.1571	0.1585	0.0680	0.0053	0.3889
SURF(128D)	2855	2332	140	34	0.1337	0.1191	0.2066	0.0051	0.4645
SURF(64D)	2855	2332	327	63	0.1307	0.1137	0.1173	0.0055	0.3672
KAZE	366	705	88	6	0.1930	0.1988	0.0105	0.0047	0.4070
AKAZE	278	289	153	9	0.0541	0.0536	0.0037	0.0049	0.1163
ORB	978	942	323	17	0.0075	0.0079	0.0124	0.0049	0.0327
ORB(1000)	731	734	240	32	0.0067	0.0073	0.0088	0.0050	0.0278
BRISK	796	752	285	18	0.0146	0.0144	0.0119	0.0049	0.0458
BRISK(1000)	796	752	285	18	0.0146	0.0144	0.0119	0.0049	0.0458
Mountain Dataset (Image Pair # 3)									
SIFT	1867	2033	170	45	0.1943	0.2017	0.1197	0.0047	0.5204
SURF(128D)	1890	2006	175	33	0.0979	0.1130	0.1208	0.0051	0.3368
SURF(64D)	1890	2006	227	62	0.0978	0.1113	0.0689	0.0053	0.2833
KAZE	972	971	131	20	0.2787	0.2826	0.0343	0.0050	0.6006
AKAZE	960	986	187	16	0.0841	0.0842	0.0151	0.0050	0.1884
ORB	4791	5006	340	78	0.0228	0.0236	0.1397	0.0052	0.1913
ORB(1000)	1000	1000	113	29	0.0118	0.0118	0.0117	0.0049	0.0402
BRISK	3153	3382	287	22	0.0520	0.0555	0.1083	0.0052	0.2210
BRISK(1000)	1000	1000	143	7	0.0201	0.0213	0.0160	0.0046	0.0620
Graffiti Dataset (Image Pair # 4)									
SIFT	2654	3698	99	51	0.2858	0.3382	0.2940	0.0073	0.9253
SURF(128D)	4802	5259	44	31	0.2166	0.2184	0.7399	0.0222	1.1971
SURF(64D)	4802	5259	62	42	0.2072	0.2142	0.3951	0.0169	0.8334
KAZE	2232	2302	30	9	0.3311	0.3337	0.1594	0.0052	0.8294
AKAZE	2064	2205	23	13	0.1158	0.1185	0.0491	0.0081	0.2915
ORB	5527	7517	38	22	0.0277	0.0341	0.2129	0.0083	0.2830
ORB(1000)	1000	1000	16	7	0.0146	0.0157	0.0114	0.0059	0.0476
BRISK	3507	5191	54	22	0.0669	0.0953	0.1687	0.0077	0.3386
BRISK(1000)	1000	1000	18	10	0.0227	0.0239	0.0143	0.0073	0.0682
Roofs Dataset (Image Pair # 5)									
SIFT	2303	3550	423	154	0.1983	0.2665	0.2475	0.0063	0.7186
SURF(128D)	2938	3830	171	95	0.1173	0.1523	0.3349	0.0084	0.6129
SURF(64D)	2938	3830	247	143	0.1165	0.1504	0.1847	0.0090	0.4606
KAZE	1260	1736	172	85	0.2119	0.2265	0.0710	0.0068	0.5162
AKAZE	1287	1987	175	59	0.0686	0.0806	0.0294	0.0053	0.1839
ORB	7660	11040	498	157	0.0296	0.0407	0.4131	0.0065	0.4899
ORB(1000)	1000	1000	91	32	0.0106	0.0113	0.0111	0.0055	0.0385
BRISK	5323	7683	436	207	0.0899	0.1260	0.3672	0.0074	0.5905
BRISK(1000)	1000	1000	90	44	0.0189	0.0199	0.0156	0.0069	0.0613
River Dataset (Image Pair # 6)									
SIFT	8619	9082	1322	192	0.6795	0.7083	2.2582	0.0092	3.6552
SURF(128D)	9434	10471	223	63	0.3768	0.4205	2.8521	0.0055	3.6549
SURF(64D)	9434	10471	386	66	0.3686	0.4091	1.4905	0.0049	2.2731
KAZE	2984	2891	391	78	0.5119	0.5115	0.2669	0.0059	1.2962
AKAZE	3751	3635	376	65	0.2048	0.1991	0.1341	0.0056	0.5436
ORB	34645	35118	2400	553	0.1219	0.1276	5.3813	0.0092	5.6400
ORB(1000)	1000	1000	40	6	0.0235	0.0235	0.0109	0.0046	0.0625
BRISK	23607	24278	1725	366	0.3813	0.4040	4.8117	0.0089	5.6059
BRISK(1000)	1000	1000	39	8	0.0251	0.0248	0.0155	0.0040	0.0694
Mean Values for All Image Pairs									
SIFT	3125.7	3662.8	470.8	84.8	0.2827	0.3119	0.5202	0.0064	**1.1212**
SURF(128D)	4317.8	4744.7	178.7	52.3	0.1847	0.2003	0.7997	0.0087	**1.1934**
SURF(64D)	4317.8	4744.7	310.2	74.8	0.1806	0.1954	0.4254	0.0078	**0.8092**
KAZE	1517.5	1660.7	212.8	37.3	0.2902	0.2941	0.1006	0.0055	**0.6904**
AKAZE	1633.0	1776.0	231.5	33.0	0.0995	0.1013	0.0437	0.0057	**0.2502**
ORB	9782.7	10828.0	742.2	162.7	0.0385	0.0427	1.0530	0.0068	**1.1410**
ORB(1000)	955.2	955.7	122.8	21.2	0.0129	0.0133	0.0113	0.0051	**0.0426**
BRISK	6612.3	7476.8	544.7	113.7	0.1097	0.1253	0.9319	0.0066	**1.1735**
BRISK(1000)	966.0	958.7	127.5	20.0	0.0200	0.0206	0.0149	0.0054	**0.0609**

Uit de conclusies van de paper volgt dat SIFT, SURF en BRISK de meest schaal invariante feature detectors zijn. ORB was het algoritme dat hierin het slechtste scoorde. $ORB_{(1000)}$, $BRISK_{(1000)}$ en AKAZE zijn het meest rotatie invariant. De nauwkeurigheid tijdens een rotatie is het hoogste bij SIFT, KAZE, AKAZE en BRISK en algemeen zijn SIFT en BRISK het nauwkeurigst. Zoals al meermaals aangehaald is SIFT zeer nauwkeurig, maar wel trager. Uit deze paper bleek ook dat SIFT over het algemeen het meest nauwkeurige algoritme was. Op de volgende pagina volgen nog enkele algemene rangschikkingen voor alle algoritmes.

Totaal aantal features detecteren:
ORB>BRISK>SURF>SIFT>AKAZE>KAZE

Efficiëntie feature detectie + descriptie per keypoint:
ORB>ORB$_{(1000)}$>BRISK>BRISK$_{(1000)}$>SURF$_{(64D)}$>SURF$_{(128D)}$>AKAZE>SIFT>KAZE

Efficiëntie feature matching:
ORB$_{(1000)}$>BRISK$_{(1000)}$>AKAZE>KAZE>SURF$_{(64D)}$>ORB>BRISK>SIFT>SURF$_{(128D)}$

Snelheid feature detectie + descriptor voor matching volledige afbeelding:
ORB$_{(1000)}$>BRISK$_{(1000)}$>AKAZE>KAZE>SURF$_{(64D)}$>SIFT>ORB>BRISK>SURF$_{(128D)}$

2.7 Optical flow tracking

Optical flow is een algoritme dat de beweging weergeeft van pixels tussen twee opeenvolgende frames, die veroorzaakt is door een beweging van het object zelf of door een verplaatsing van de camera. De optical flow wordt weergeven aan de hand van een vectorveld, waarbij iedere vector aangeeft hoe een punt verplaatst is tussen de eerste en de tweede frame. Bij het bereken van de optical flow worden er twee veronderstellingen gemaakt.

Ten eerste verandert de intensiteit van de pixel niet tussen twee opeenvolgende frames en ten tweede maken de naburige pixels een gelijkaardige beweging. De berekening ziet er dan uit als volgt:

$I(x, y, t)$ is de intensiteit van pixels in de eerste frame. De pixels bewegen dan volgens (dx, dy) na een tijd dt en geeft:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \quad (29)$$

Vervolgens wordt de Taylorreeks van de rechter term genomen, de gemeenschappelijke termen worden verwijderd en als laatste wordt de vergelijking gedeeld door dt. De vergelijking ziet er dan als volgt uit:

$$f_x u + f_y v + f_t = 0$$

$$f_x = \frac{\partial f}{\partial x} \; ; \; f_y = \frac{\partial f}{\partial y} \; ; \; u = \frac{dx}{dt} \; ; \; v = \frac{dy}{dt} \quad (30)$$

Dit is de Optical Flow vergelijking. Het probleem met deze vergelijking is dat u en v beide ongekend zijn. Met andere woorden zijn er twee onbekende en één vergelijking, dus de vergelijking is niet oplosbaar. Er zou verschillende methodes om deze vergelijking toch op te lossen.

De verschillende optical flow methodes kunnen opgedeeld worden in twee categorieën: sparse en dense optical flow. Bij sparse optical flow worden enkel bepaalde features binnen een frame gevolgd. Dit in tegenstelling tot dense optical flow waarbij alle pixels opgenomen worden in de berekening. Het voordeel van sparse optical flow is dat de berekening sneller uitgevoerd kan worden, maar dit gaat ten koste van nauwkeurigheid. Dense optical flow is dan weer nauwkeuriger maar trager.

2.7.1 Lucas-Kanade methode

Het doel van de Lucas-Kanade [32] methode is om op een efficiënte wijze de beweging van interessante features te volgen. Deze methode gebruikt dezelfde veronderstellingen die gemaakt werden bij het opstellen van de Optical Flow vergelijking. Dit heeft als gevolg dat deze methode beter presteert wanneer de grootte van de verplaatsing tussen twee frames klein is. Om de twee onbekende u en v uit de Optical Flow vergelijking te vinden, wordt er gebruikt gemaakt van een 3x3 window. Hieruit volgen dan negen vergelijkingen en twee onbekende en dus kunnen u en v berekend worden. Aangezien het bereken van negen vergelijkingen niet efficiënt is, wordt de berekening nog verder vereenvoudigd. Dit wordt gedaan aan de hand van de kleinste kwadratenmethode. Het resultaat hiervan is dan:

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} \sum_i f_{x_i}{}^2 & \sum_i f_{x_i} f_{y_i} \\ \sum_i f_{x_i} f_{y_i} & \sum_i f_{y_i} \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i f_{x_i} f_{t_i} \\ -\sum_i f_{y_i} f_{t_i} \end{bmatrix} \quad (31)$$

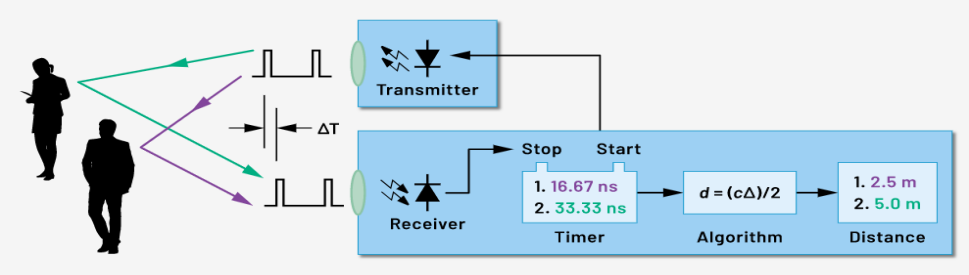
3 Hardware en bibliotheken

Voor deze thesis werd gebruik gemaakt van de Human Interface Mate van Arkite. Dit toestel bevat een Kinect v2. In de Kinect zit een infraroodcamera en een dieptecamera. De werking hiervan wordt uitgelegd in hoofdstuk 3.1

De code werd geschreven in C++ opdat de eindoplossing gemakkelijk in de Human Interface Mate kan geïntegreerd worden. Voor de registraties te berekenen werd daarom gebruik gemaakt van de Point Cloud Library (PCL). Voor het zoeken van de keypoints werd geopteerd voor de OpenCV bibliotheek.

3.1 Camera's

De Kinect v2 bevat een diepte camera en een infrarood camera. De dieptesensor werkt volgens het time-of-flight principe. Dit wil zeggen dat de sensor constant infrarood licht met gemoduleerde golven uitstuurt en de verschoven fase van de golf dan detecteert. Hierdoor kan de sensor de diepte van een punt berekenen door te meten hoe lang het duurt voor deze uitgezonden lichtgolven om de sensor terug te bereiken. Indien de sensor een onbetrouwbare meting waarneemt, wordt een specifieke waarde van 65526 teruggegeven. In figuur 32 wordt de werking van een time-of-flight camera grafisch voorgesteld. De lichtstralen worden voorgesteld door de groene en paarse pijlen. De tijd die zit tussen het uitsturen van het lichtsignaal en de reflectie ervan terug op de camera kan worden gemeten door de faseverschuiving van de lichtgolf. Voor de groene straal is dit 16.67 ns en voor de paarse straal 33.3 ns. Op basis van de formule in de figuur kan aan de hand van de lichtsnelheid de afgelegde afstand bepaald worden.



Figuur 32: Time-of-flight camera [17]

De infraroodsensor van de Kinect meet de intensiteit van de teruggekaatste lichtstraal. Deze intensiteit is een waarde tussen 0 en 8395 waarbij 0 staat voor geen reflectie en 8395 voor volledige reflectie. In figuur 33 wordt links een dieptebeeld getoond en rechts een infraroodbeeld. Beide foto's werden gemaakt met de Kinect. Deze scene is opgenomen als realistische setting van een werkcel.

Figuur 33: Diepte- en infraroodbeeld

3.2 Bibliotheken

3.2.1 Point Cloud Library (PCL)

Voor de praktische implementatie van de eindoplossing werd gebruik gemaakt van de Point Cloud Library (PCL). Dit is een open source bibliotheek voor het verwerken van 2D afbeeldingen en 3D puntenwolken. Deze bibliotheek bevat onder meer algoritmes voor het filteren van puntenwolken, algoritmes om registraties uit te voeren, algoritmes voor het detecteren van keypoints en tools om puntenwolken te visualiseren. Er werd gebruik gemaakt van PCL omwille van de bekendheid en anciënniteit van deze library en omwille van de verschillende registratie algoritmes die aanwezig zijn. Ook de community en de open source waren doorslaggevende factoren in deze beslissing. Twee andere bibliotheken voor het verwerken van puntenwolken zijn Open3D en LibPointMatcher. Deze libraries werden niet gebruikt wegens het beperkt aantal registratie algoritmes die beschikbaar zijn in deze bibliotheken.

3.2.2 OpenCV

OpenCV (Open Source Computer Vision Library) is een open source computer visie en machine learning software bibliotheek. De bibliotheek valt onder de BSD-licentie, wat betekent dat bedrijven de software gemakkelijk kunnen gebruiken en aanpassen. OpenCV bevat verschillende geoptimaliseerde feature detectie algoritmes, feature descriptor algoritmes en matching methodes. Voor deze masterproef werd versie 4.5.0 gebruikt. Buiten de officiële bibliotheek is er ook nog de opencv_contrib bibliotheek. Deze bevat algoritmes die nog niet volledig geoptimaliseerd zijn en algoritmes die niet gratis te gebruiken zijn. SURF is een voorbeeld van een algoritme dat gepatenteerd is en dus niet zonder toestemming industrieel geïmplementeerd gebruikt mag worden. Aangezien SURF een van de meest populaire algoritmes is, werd ook de opencv_contrib bibliotheek gebruikt voor deze masterproef. Naast het feit dat OpenCV beschikt over verschillende goed geoptimaliseerde methodes, werd er ook geopteerd om deze bibliotheek te gebruiken, aangezien Arkite ook al gebruik maakt van OpenCV voor verschillende methodes van de HIM.

4 Experimenten

De thesis werd uitgevoerd in drie fases. In de eerste fase werden de algoritmes GICP, NICP en ICP toegepast op data van een voorbeeld van een puntenwolk waarin geen ruis aanwezig was en die een beperkt aantal punten bevatte. Voor deze puntenwolk werd nagegaan tot welke rotaties en translaties de algoritmes een acceptabele transformatie berekenen.

In de tweede fase werden de algoritmes ICP, GICP en NICP toegepast op data afkomstig van de Human Interface Mate. Hierbij werd gezocht naar welke data nodig is om een correcte transformatie te berekenen en welke algoritmes het best presteren op vlak van tijd en nauwkeurigheid.

In de derde fase werden na de testen uit de tweede fase de algoritmes ICP en NICP geselecteerd om te testen op keypoints die werden geëxtraheerd uit de 2D beelden van de infraroodcamera. Ook hier werd gezocht naar welk algoritme het best presteerde.

Er werd gekozen om NDT niet toe te passen in deze thesis gezien de parameters voor dit algoritme sterk afhankelijk zijn van de schaal van de puntenwolk en dit voor de automatisering voor verschillende werkomgevingen voor Arkite niet interessant is. De algoritmes ICP, NICP en GICP zijn niet afhankelijk van schaalafhankelijke parameters.

4.1 Registratie perfecte puntenwolk

In het eerste deel van het onderzoek werden de algoritmes ICP, GICP en NICP toegepast op de puntenwolk van een beeldhouwwerk van een konijn. Deze puntenwolk is afkomstig van de Stanford University [33] en wordt vaak gebruikt voor het meten van de performantie van registratie algoritmes op ruisloze puntenwolken. Het doel van dit deel is dan ook de performantie van de bovengenoemde algoritmes te meten. Hiervoor werden verschillende translaties en rotaties toegepast op deze puntenwolk en daarna de registratie uitgevoerd tussen de oorspronkelijke en verplaatste puntenwolk. De verschillende transformaties worden weergegeven in tabel 2.

Tabel 2: Transformaties bij testen perfecte puntenwolk

Test	Translatie x (m)	Translatie y (m)	Translatie z (m)	Rotatie (°)
1	0 - 4	0	0	0
2	0 - 4	0 - 4	0	0
3	0 - 4	0 - 4	0 - 4	0
4	0	0	0	0-90
5	0.5	0	0	0-90
6	0.5	0.5	0	0-90
7	0.5	0.5	0.5	0-90

4.1.1 Parameters van algoritmes

Voor de parameters van elk algoritme werden de volgende waardes gekozen:

Filtering:

Er werd geen filtering toegepast aangezien de puntenwolk maar 397 punten bezit en er geen ruis aanwezig is.

Convergentiecriteria:

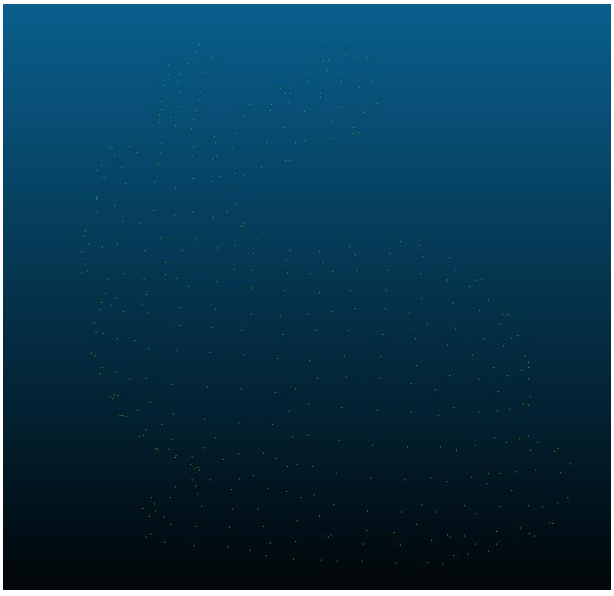
Als convergentiecriteria werd er geopteerd voor een relatieve transformatie threshold parameter. Als waarde werd gekozen voor 1e-7. Dit wil zeggen dat het algoritme wordt beëindigd wanneer het verschil in gekwadrateerde translatie afstand tussen twee transformaties kleiner is dan deze waarde. Als tweede waarde werd gekozen voor een relatieve rotatie threshold. Wanneer de rotatie tussen twee transformaties kleiner was dan 0.5° werd het algoritme beëindigd. Als maximaal aantal iteraties werd voor 1000 gekozen.

Filteren corresponderende punten:

Er werd niet geopteerd om matchende punten te filteren gezien de grote afstand die mogelijk was tussen twee corresponderende punten.

4.1.2 Resultaten mean square error perfecte puntenwolk

De puntenwolk die werd gebruikt om de transformaties te berekenen is een puntenwolk van Stanford University. Deze puntenwolk is afkomstig van een scan van een beeldhouwwerk van een konijn. Deze puntenwolk wordt weergegeven in figuur 34. Deze puntenwolk is een perfecte puntenwolk zonder ruis.



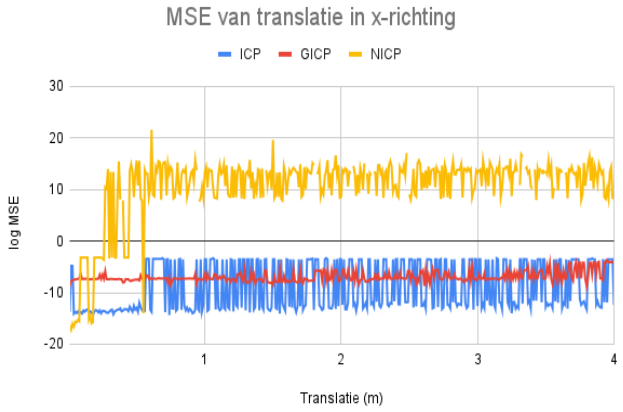
Figuur 34: Puntenwolk konijn

Voor elk van de registraties voorgesteld in tabel 2 werd de mean square error berekend. De mean square error wordt berekend volgens volgende formule:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(x_i - x')^2 + (y_i - y')^2 + (z_i - z')^2 \tag{32}$$

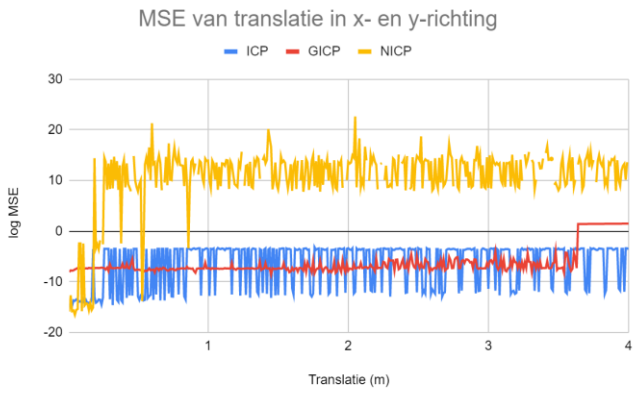
In deze formule staat n voor het aantal punten in de oorspronkelijke puntenwolk dat een corresponderend punt heeft gevonden in de verplaatste puntenwolk zoals beschreven in hoofdstuk 2.5.1.1. x_i, y_i en z_i zijn de coördinaten van de punten in de originele puntenwolk en x', y' en z' zijn de coördinaten van hun corresponderende gevonden punten in de verplaatste puntenwolk.

In figuren 35 t.e.m. 37 worden de grafieken gegeven van het logaritme van de MSE i.f.v de translatie. In figuren 38 t.e.m. 41 worden de grafieken gegeven van het logaritme van de MSE i.f.v de rotatie.



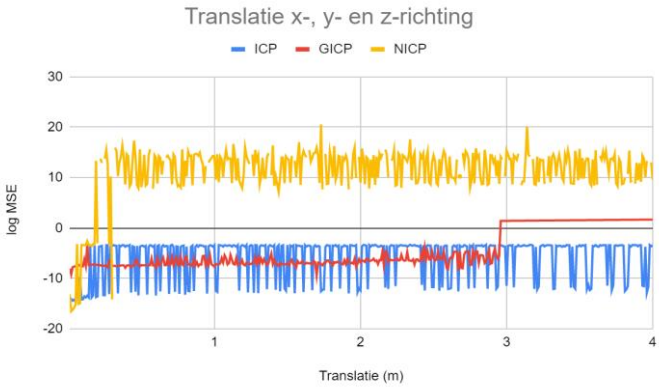
Figuur 35: log MSE i.f.v. translatie (translatie x-richting)

In figuur 35 is zichtbaar dat voor een zuivere verplaatsing in de x-richting de MSE-waarde klein is voor ICP tot verplaatsingen van ±0.5 m. Na 0.5 m schommelt de MSE waarde tussen grote en kleine waardes en is bijgevolg de berekening van de transformatie niet meer betrouwbaar. Voor NICP wordt de MSE al snel groot en de log van de MSE zelfs positief wat duidt op heel grote afstanden tussen corresponderende punten in beide puntenwolken. De transformatie wordt vanaf ±0.25 m dus volledig foutief berekend. GICP heeft voor translaties tussen 0 m en ±0.5 m hogere waardes voor de MSE dan ICP maar deze zijn klein genoeg om een goede transformatie te vinden. Na ± 0.5 m blijft de MSE bij GICP ongeveer constant wat duidt op een blijvende goede berekening van de transformatie.



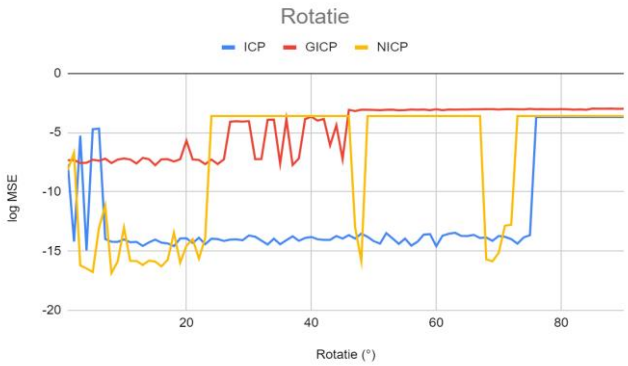
Figuur 36: log MSE i.f.v. translatie (translatie x- en y-richting)

In figuur 36 is zichtbaar dat voor een verplaatsing in de x- en y-richting de MSE-waarde klein is voor ICP tot verplaatsingen van ±0.3 m. Na 0.3 m schommelt de MSE waarde tussen grote en kleine waardes en is bijgevolg de berekening van de transformatie niet meer betrouwbaar. Voor NICP wordt de MSE al snel groot en de log van de MSE zelfs positief wat duidt op heel grote afstanden tussen corresponderende punten in beide puntenwolken. De transformatie wordt vanaf ±0.2 m dus volledig foutief berekend. GICP heeft voor translaties tussen 0 m en ±0.3 m hogere waardes voor de MSE dan ICP maar deze zijn klein genoeg om een goede transformatie te vinden. Na ± 0.3 m blijft de MSE bij GICP ongeveer constant wat duidt op een blijvende goede berekening van de transformatie.



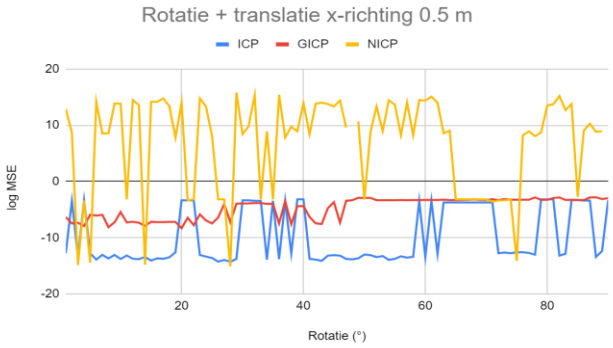
Figuur 37: log MSE i.f.v. translatie (translatie x-, y- en z-richting)

In figuur 37 is zichtbaar dat voor een verplaatsing in de x-, y- en z-richting de MSE-waarde klein is voor ICP tot verplaatsingen van ±0.2 m. Na 0.2 m schommelt de MSE waarde tussen grote en kleine waardes en is bijgevolg de berekening van de transformatie niet meer betrouwbaar. Voor NICP wordt de MSE al snel groot en de log van de MSE zelfs positief wat duidt op heel grote afstanden tussen corresponderende punten in beide puntenwolken. De transformatie wordt vanaf ±0.2 m dus volledig foutief berekend. GICP heeft voor translaties tussen 0 m en ±0.2 m hogere waardes voor de MSE dan ICP maar deze zijn klein genoeg om een goede transformatie te vinden. Na ± 0.2 m blijft de MSE bij GICP ongeveer constant wat duidt op een blijvende goede berekening van de transformatie.



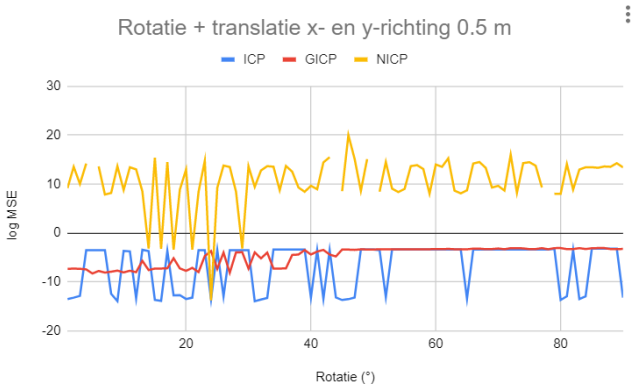
Figuur 38: log MSE i.f.v. rotatie (zuivere rotatie)

In figuur 38 is zichtbaar dat voor een zuivere rotatie de MSE-waarde klein is voor ICP bij rotaties tussen ±5 en ±70 graden. Na ±70 graden wordt de MSE waarde te groot en wordt de rotatie slecht berekend. Voor NICP wordt de MSE na ±20 graden te groot en is de berekende rotatie niet meer correct. GICP heeft voor rotaties tussen 0 graden en ±20 graden een hogere MSE dan NICP en ICP maar de berekende rotaties zijn hier nog correct. Na ±20 graden zijn de berekende rotaties niet meer correct.



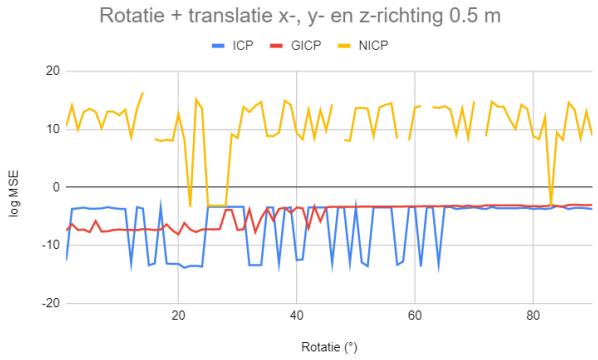
Figuur 39: log MSE i.f.v. rotatie (rotatie + translatie x 0.5 m)

In figuur 39 is zichtbaar dat voor een rotatie in combinatie met een translatie in de x-richting de MSE-waarde klein is voor ICP bij rotaties tussen ±5 en ±20 graden. Na ±20 graden wordt de MSE waarde te groot en wordt de rotatie slecht berekend. GICP heeft voor rotaties tussen 0 graden en ±25 graden een hogere MSE dan ICP maar de berekende rotaties zijn hier nog correct. Na ±25 graden zijn de berekende rotaties via GICP niet meer correct. Voor NICP wordt de transformatie verkeerd berekend doordat bij een zuivere translatie in de x-richting van 0.5m deze translatie ook al verkeerd berekend werd.



Figuur 40: log MSE i.f.v. rotatie (rotatie + translatie x en y 0.5 m)

In figuur 40 is zichtbaar dat voor een rotatie in combinatie met een translatie in de x- en y-richting de MSE-waarde klein is voor ICP bij rotaties tussen 0 en ±3 graden. Na ±3 graden wordt de MSE waarde te groot en wordt de rotatie slecht berekend. GICP heeft voor rotaties tussen 0 graden en ±3 graden een hogere MSE dan ICP maar de berekende rotaties zijn hier nog correct. Na ±20 graden zijn de berekende rotaties via GICP niet meer correct. Voor NICP wordt de transformatie verkeerd berekend doordat bij een translatie in de x- en y-richting van 0.5m deze translatie ook al verkeerd berekend werd.



Figuur 41: log MSE i.f.v. rotatie (rotatie + translatie x, y en z 0.5 m)

In figuur 41 is zichtbaar dat voor een rotatie in combinatie met een translatie in de x-, y- en z-richting de MSE-waarde klein is voor ICP bij rotaties van 0 en 1 graad. Na 1 graad wordt de MSE waarde te groot en wordt de rotatie slecht berekend. GICP heeft voor rotaties tussen 0 graden en 2 graden een hogere MSE dan ICP maar de berekende rotaties zijn hier nog correct. Na ±25 graden zijn de berekende rotaties via GICP niet meer correct. Voor NICP wordt de transformatie verkeerd berekend doordat bij een translatie in x-, y- en z-richting van 0.5m deze translatie ook al verkeerd berekend werd.

4.1.3 Conclusie

Uit de tests op de puntenwolk van het konijn blijkt dat GICP het meest constant presteert voor alle scenario's. ICP en NICP presteren voor kleine translaties over het algemeen beter dan GICP maar bij grote translaties presteren deze algoritmes beduidend slechter. Bij de pure rotatie is zichtbaar dat ICP het best presteert bij hoeken tussen ±5 en ±70 graden. NICP presteert bij een zuivere rotatie beter dan GICP voor hoeken tussen de ±5 en ±20 graden maar daarna presteren ze beide slecht. Bij de combinatie van rotatie en translatie presteert GICP constanter dan ICP

voor rotaties kleiner dan ±20 graden. NICP presteert in alle gevallen van de combinatie van rotatie en translatie slecht doordat bij de pure translatie dit algoritme ook al slecht presteerde bij verplaatsingen van 0.5m.

4.2 Registratie puntenwolk lege tafel

In het tweede deel van dit onderzoek werd gewerkt met echte data van een proefopstelling van Arkite. De waardes van de dieptesensor en de infraroodsensor worden uitgelezen uit recordings gemaakt met de Human Interface Mate. Deze recordings hebben voor zowel de dieptesensor als de infraroodsensor een resolutie van 512 x 424 pixels. Iedere frame bevat dus 217088 punten. De onbetrouwbare waardes van de dieptemetingen worden door de reader voor de recordings op 0 gezet. Hierdoor worden deze in het camera coördinatenstelsel allemaal afgebeeld op 1 punt in de ruimte, namelijk het punt met coördinaat (0,0,0). In het wereld assenstelsel wordt dit punt dus afgebeeld op de locatie bepaald door de translatie en rotatie bepaald door de extrinsieke matrix. In deze thesis wordt verondersteld dat de sensor loodrecht boven de werkbank staat waardoor de z-waarde van deze onbetrouwbare punten overeenkomt met de z-translatie in de extrinsieke matrix.

Voor deze opnames werd de extrinsieke matrix die gebruikt wordt voor de reconstructie van de puntenwolk bepaald via kalibratie in de software van de Human Interface Mate.

De testsituatie bestond uit een werkcel waarbij een werkbank in beeld is. Deze werkbank werd manueel verschoven en het doel was om de registratie te berekenen. Dit werd gedaan in verschillende stappen waarbij manueel stukken uit de puntenwolk werden geknipt met als doel te achterhalen welke algoritmes het best presteren en welk soort data nodig is om een correcte registratie te bekomen.

4.2.1 Parameters van algoritmes

Voor de parameters van ieder algoritme werden de volgende waardes gekozen:

Filtering:

Er werd een conditionele filtering toegepast om de onbetrouwbare punten van de sensor weg te filter. Tevens werd wanneer de vloer werd weggefilterd ook gebruik gemaakt van een conditionele filter. Bij de eerste twee tests werden enkel de punten die afkomstig zijn van de projectie van onbetrouwbare punten weggefilterd. Hiervoor werden de z-waardes die meer dan 10 cm kleiner zijn dan de hoogte van de sensor t.o.v. de tafel verwijderd. Bij derde en vierde test werden tevens de punten met een z-waarde kleiner dan -20 cm verwijderd (z= 0 cm ter hoogte van de tafel).

Convergentiecriteria:

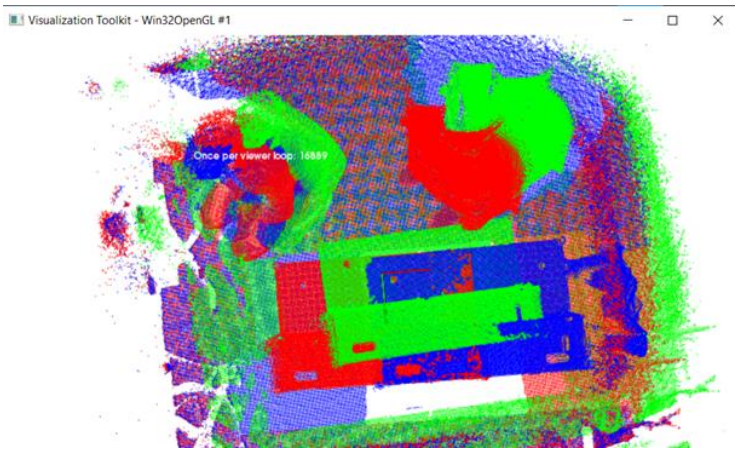
Als convergentiecriteria werd er geopteerd voor een relatieve transformatie threshold parameter. Als waarde werd gekozen voor 1e-7. Dit wil zeggen dat het algoritme wordt beëindigd wanneer het verschil in gekwadrateerde translatie afstand tussen twee transformaties kleiner is dan deze waarde. Als tweede waarde werd gekozen voor een relatieve rotatie threshold. Wanneer de rotatie tussen twee transformaties kleiner was dan 0.1° werd het algoritme beëindigd. Er werd voor een kleine waarde gekozen doordat een zuivere rotatie een kleine kans heeft om voor te komen. Als maximaal aantal iteraties werd voor 100 gekozen.

Filteren corresponderende punten:

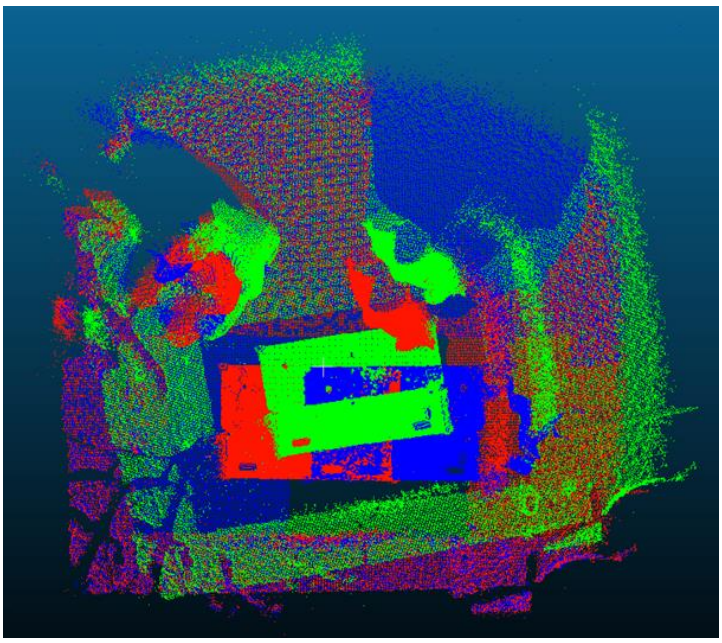
Er werd gekozen om punten waartussen de afstand groter is dan 0.5 meter weg te filteren uit de corresponderende punten. Deze waarde werd gekozen doordat de transformatie in reële situaties niet kan worden ingeschat maar er wordt verwacht dat deze nooit 0.5 m zal overschrijden. Het doel hiervan is om de ruispunten te elimineren.

4.2.2 Registratie volledige puntenwolk

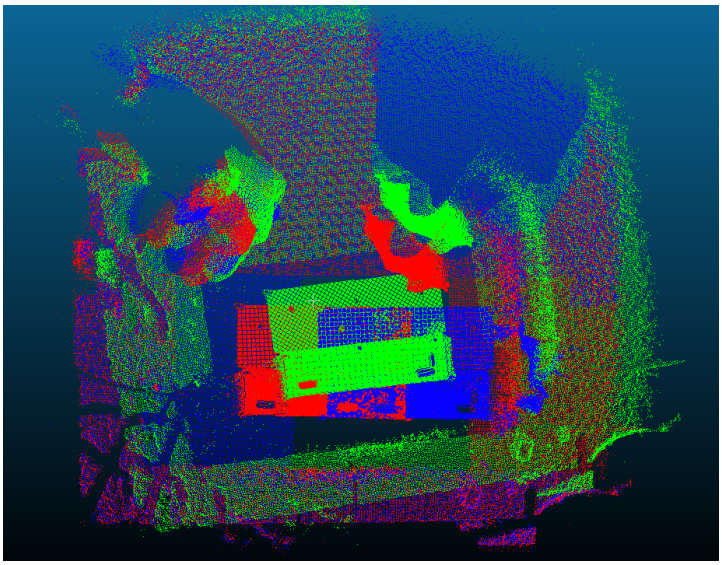
Als eerste werden de algoritmes ICP, NICP en GICP, beschreven in hoofdstuk 2.5.1, getest op de volledige puntenwolken die werden bekomen uit de opnames van de Kinect sensor in de Human Interface Mate. Het resultaat van de gevonden transformaties wordt weergegeven in figuren 42, 43 en 44. In deze figuren is de rode puntenwolk de oorspronkelijke puntenwolk, de blauwe puntenwolk de verplaatste puntenwolk en de groene puntenwolk de oorspronkelijke puntenwolk getransformeerd via de gevonden transformatie van de algoritmes.



Figuur 42: Registratie volledige puntenwolk: ICP



Figuur 43: Registratie volledige puntenwolk: NICP



Figuur 44: Registratie volledige puntenwolk: GICP

Bij zowel ICP, NICP en GICP werd de beoogde translatie van de werkbank niet bereikt. Dit is zichtbaar doordat de groene en blauwe puntenwolk niet volledig overlappen met elkaar. Tevens trad er bij elk van voorgenoemde algoritmes een rotatie op in de gevonden transformatie, dewelke niet wenselijk is.

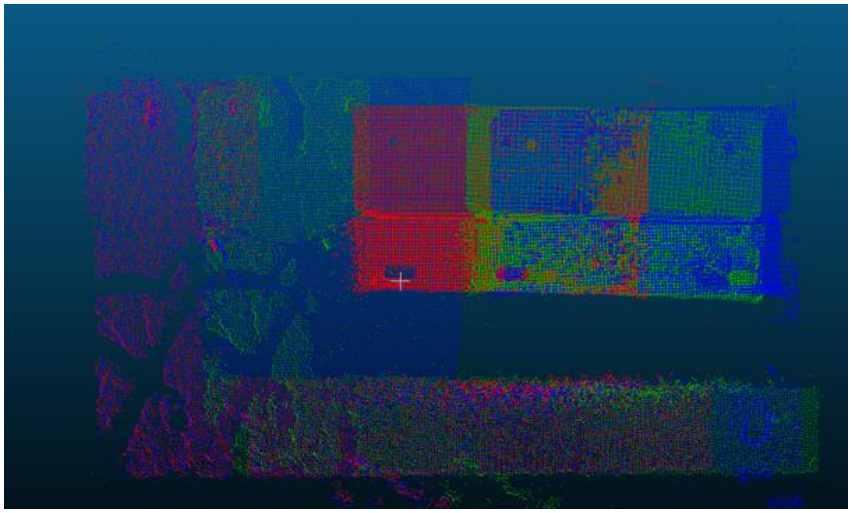
Het niet behalen van de beoogde transformatie is tevens zichtbaar in tabel 3, dewelke de MSE en de benodigde tijd voor de registratie bevat. Hierin is zichtbaar dat ieder algoritme ongeveer even slecht presteert maar dat ICP beduidend meer tijd nodig heeft om de transformatie te bepalen.

Tabel 3: MSE en registratietijd voor volledige puntenwolk

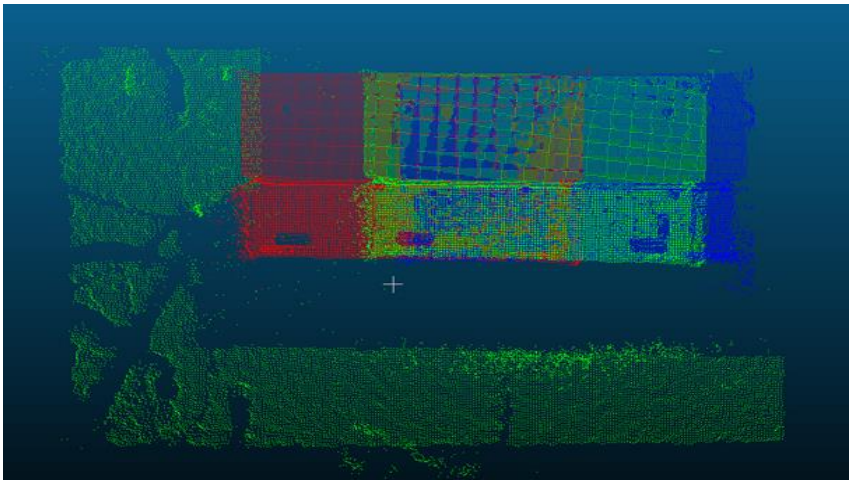
	ICP	GICP	NICP
MSE (m)	0,0113928	0,0122722	0,0125727
TIJD (ms)	72996	9262	9062

4.2.3 Registratie puntenwolk zonder personen

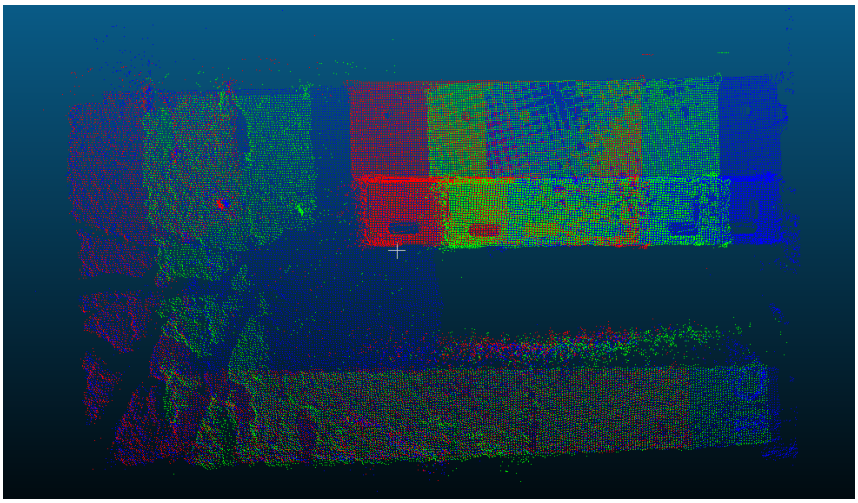
Doordat bij de vorige testen op de volledige puntenwolk er een ongewenste rotatie in de gevonden transformatie werd bekomen werd er gezocht naar welk deel van de puntenwolk verantwoordelijk was voor deze rotatie. Bij een volgende test werd de puntenwolk uit hoofdstuk 4.2.2 manueel bijgeknipt zodat alle personen uit de puntenwolk waren weggefilterd. Op deze gefilterde puntenwolk werden de algoritmes ICP, NICP en GICP, die beschreven staan in hoofdstuk 2.5.1, weer toegepast. De figuren 45, 46 en 47 tonen de gevonden transformaties van deze algoritmes. De rode puntenwolk is de oorspronkelijke puntenwolk, de blauwe puntenwolk is de verplaatste puntenwolk en de groene puntenwolk is de oorspronkelijke puntenwolk getransformeerd via de gevonden transformatie van de algoritmes.



Figuur 45: Registratie puntenwolk zonder personen: ICP



Figuur 46: Registratie puntenwolk zonder personen: NICP



Figuur 47: Registratie puntenwolk zonder personen: GICP

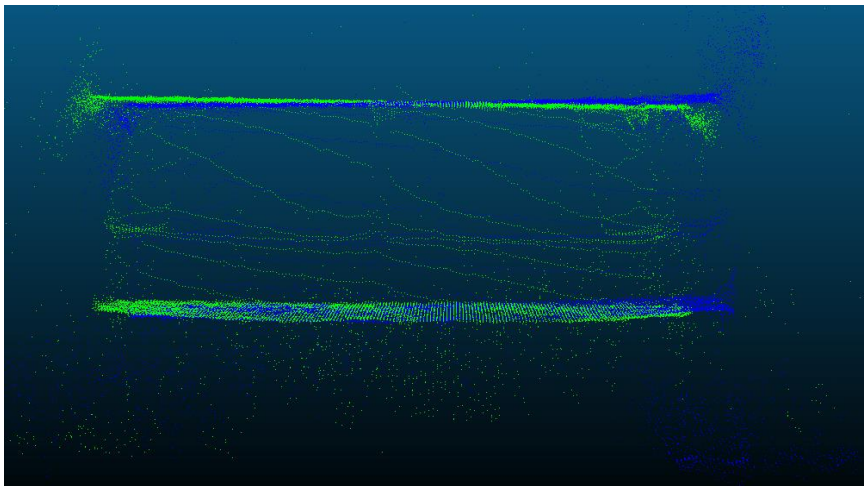
In zowel figuur 45, 46 als 47 is zichtbaar dat de rotatie die bij de volledige puntenwolk werd gevonden verdwenen is. Hieruit valt af te leiden dat de personen in de scene deze rotatie veroorzaakten. De translatie werd echter nog steeds bij geen van de algoritmes correct gevonden. Dit is zichtbaar doordat de groene en blauwe puntenwolk niet volledig overlappen met elkaar. In figuur 45 is zichtbaar dat bij ICP de translatie net niet correct gevonden wordt, in figuur 46 is zichtbaar dat NICP de translatie minder goed benadert dan ICP en in figuur 47 is zichtbaar dat GICP de translatie minder goed benadert dan NICP. In tabel 4 worden de MSE en benodigde registratietijd gegeven. Uit deze waardes blijkt dat ICP een lagere MSE heeft dan NICP en NICP een lagere MSE heeft dan GICP. Dit komt overeen met de visuele bevindingen op de figuren 45, 46 en 47. Tevens blijkt dat GICP ±4.5 keer trager is dan NICP en ±2.5 keer trager is dan ICP.

Tabel 4: MSE en registratietijd voor puntenwolk zonder personen

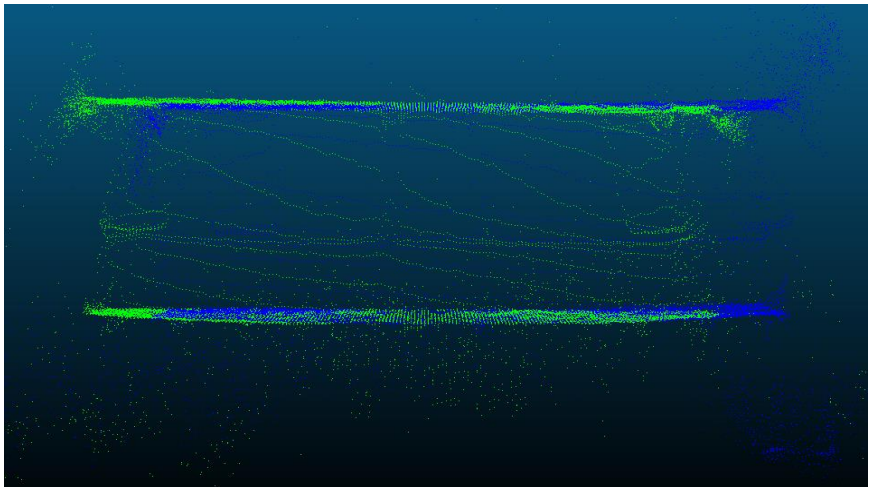
	ICP	GICP	NICP
MSE (m)	0,000890277	0,00160265	0,000932435
TIJD (ms)	2971	7404	1676

4.2.4 Registratie puntenwolk zonder personen en vloer

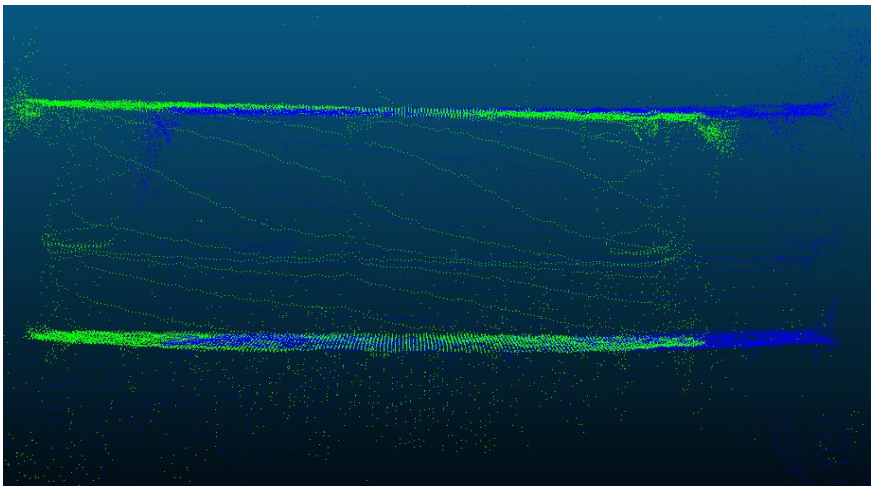
Uit de voorgaande test waarbij de personen werden verwijderd uit de data bleek dat de gevonden translatie nog niet nauwkeurig genoeg was. Om te trachten de translatie nauwkeuriger te bepalen werd verder gezocht naar de oorzaak van de foutieve registratie. Uit de puntenwolk van het vorige deel (hoofdstuk 4.2.3) werd als test de vloer weggefilterd. Op deze gefilterde puntenwolk werd de registratie weer berekend via ICP, NICP en GICP, beschreven in hoofdstuk 2.5.1. In figuren 48, 49 en 50 zijn de gevonden transformaties zichtbaar voor respectievelijk ICP, NICP en GICP. De blauwe puntenwolk is van de verplaatste werkbank en de groene puntenwolk is de getransformeerde puntenwolk.



Figuur 48: Registratie puntenwolk zonder personen en vloer: ICP



Figuur 49: Registratie puntenwolk zonder personen en vloer: NICP



Figuur 50: Registratie puntenwolk zonder personen en vloer: GICP

Bij de transformaties die op deze manier werden gevonden was de translatie beter benaderd maar trad er echter een fout op in de rotatie. Dit is zichtbaar doordat de getransformeerde puntenwolk langs de linkerkant boven de verplaatste puntenwolk uitkomt en langs de rechterkant de getransformeerde puntenwolk onder de verplaatste puntenwolk uitkomt. Uit deze figuren kan tevens worden afgeleid dat ICP beter de transformatie berekent dan NICP en NICP op zijn beurt beter de transformatie berekent dan GICP.

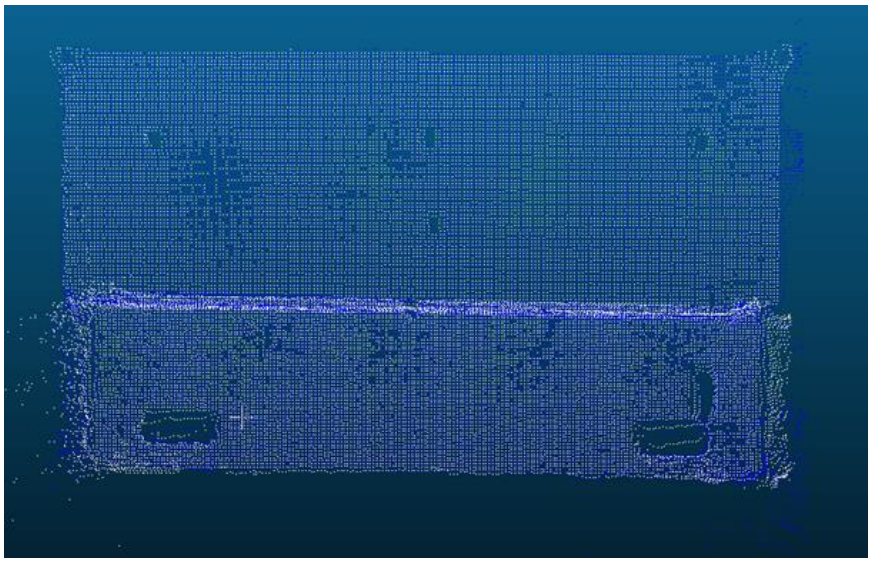
In tabel 5 worden de MSE en benodigde registratietijd gegeven. Ook uit deze waardes voor de MSE blijkt dat ICP de transformatie beter berekent dan NICP en NICP de transformatie beter berekent dan GICP. GICP is bij deze test ±1.5 keer trager dan ICP en ±4 keer trager dan NICP

Tabel 5: MSE en registratietijd voor puntenwolk met weggefilterde vloer

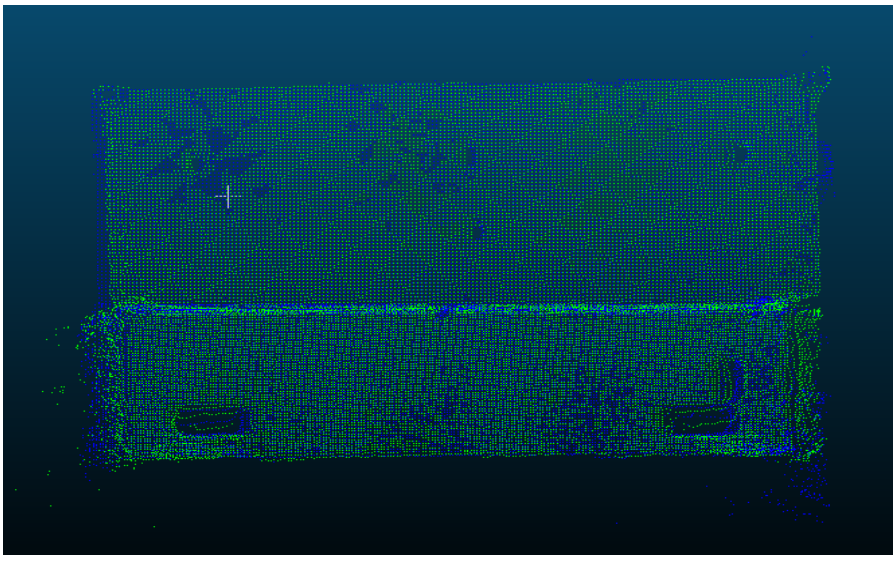
	ICP	GICP	NICP
MSE (m)	0,00101227	0,00212299	0,00126207
TIJD (ms)	2038	3241	758

4.2.5 Registratie puntenwolk zonder ruis

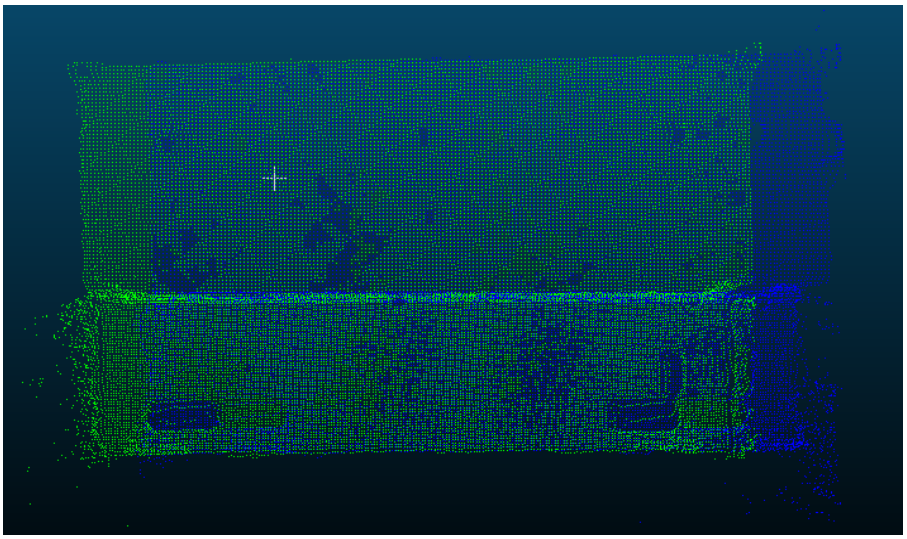
Uit de testen uit het vorige stuk bleek dat enkel de vloer wegfilteren niet leidt tot het vinden van een correcte transformatie. In dit deel werd de ruis weggeknipt uit de puntenwolk (hoofdstuk 4.2.4) om te achterhalen of deze verantwoordelijk is voor het niet vinden van de correcte transformatie in de vorige stap. Op deze nieuwe puntenwolk werden de algoritmes ICP, NICP en GICP, beschreven in hoofdstuk 2.5.1, toegepast. De resultaten van de gevonden transformaties zijn zichtbaar in figuren 51, 52 en 53. In figuur 51 is de getransformeerde puntenwolk de witte puntenwolk en de verplaatste puntenwolk de blauwe puntenwolk. In figuur 52 en 53 zijn de getransformeerde puntenwolken de groene puntenwolken en de verplaatste puntenwolken de blauwe puntenwolken.



Figuur 51: Registratie puntenwolk zonder ruis: ICP



Figuur 52: Registratie puntenwolk zonder ruis: NICP



Figuur 53: Registratie puntenwolk zonder ruis: GICP

In figuur 51 is zichtbaar dat voor ICP de gevonden transformatie een nauwkeurige benadering is voor de reële transformatie. Voor NICP is zichtbaar in figuur 52 dat de gevonden transformatie de reële transformatie licht overschat. Bij GICP wordt de transformatie echter nog niet goed berekend. Dit is zichtbaar in figuur 53.

In tabel 6 worden de MSE en benodigde registratietijd gegeven. Deze waardes voor de MSE tonen tevens aan dat ICP en NICP de transformatie beduidend beter berekenen dan GICP. ICP is bij deze test ±3 keer trager dan GICP en ±5 keer trager dan NICP.

Tabel 6: MSE en registratietijd voor puntenwolk ruisloze tafel

	ICP	GICP	NICP
MSE (m)	3,60E-05	0,0003521	3,40E-05
TIJD (ms)	2236	685	409

4.2.6 Conclusie

Uit de resultaten van de testen in dit deel valt af te leiden dat de beste resultaten voor het vinden van de transformatie worden bekomen bij het wegfilteren van de ruis rond de tafel in combinatie met het wegfilteren van de vloer. Uit bovenstaande resultaten wordt ook besloten om verder te werken met NICP en ICP gezien deze 2 de beste transformatie vonden bij de tafel zonder ruis en vloer. GICP valt dus af bij volgende testen. Het verkrijgen van ruisarme data van enkel de tafel is dus van belang voor het vinden van een goede transformatie.

In de literatuurstudie werden verschillende technieken waaronder onder meer statistical outlier removal en radius outlier removal beschreven als manier om ruis te onderdrukken. Doordat dit echter traag is voor grote puntenwolken werd er besloten om over te gaan naar het vinden van keypoints en hierop de transformatie te berekenen. Hiervoor diende er nieuwe opnames gemaakt te worden met een realistischer scenario van een werkbank met allerhande materialen en voorwerpen op.

4.3 Registratie gevulde werkbank met behulp van keypoints

In het derde deel worden de algoritmes Iterative Closest Point en Iterative Closest Point with Normals gebruikt. Generalized Iterative Closest Point wordt niet meer gebruikt. In dit deel wordt gebruik gemaakt van keypoint detectie om features te vinden in de puntenwolk. De keypoints worden gezocht in de infraroodbeelden van de sensor. Deze keypoints worden dan uit de beelden van de dieptesensor gehaald. Rond de keypoints wordt een mediaanfilter van 3x3 toegepast om ruis te onderdrukken. Met deze gevonden keypoints wordt de registratie dan berekend.

4.3.1 Parameters van algoritmes

Voor de parameters van ieder algoritme werden de volgende waardes gekozen:

Filtering:

Er werd een conditionele filtering toegepast om de onbetrouwbare punten van de sensor en de vloer weg te filteren. Ook werden keypoints gezocht in de infraroodbeelden en werden deze punten in de puntenwolk behouden en de andere punten weggefilterd.

Convergentiecriteria:

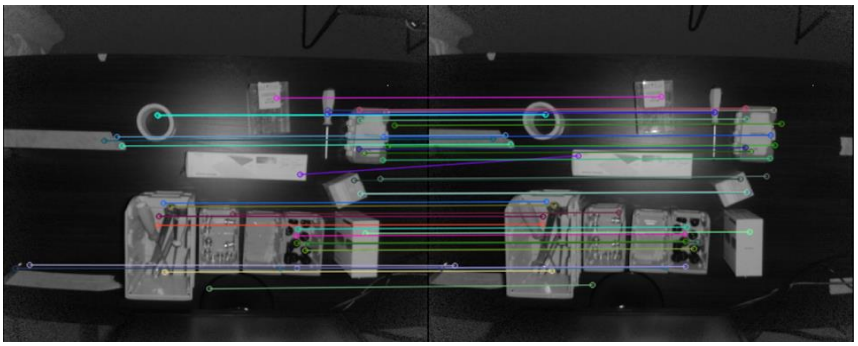
Als convergentiecriteria werd er geopteerd voor een relatieve transformatie threshold parameter. Als waarde werd gekozen voor 1e-8. Dit wil zeggen dat het algoritme wordt beëindigd wanneer het verschil in gekwadrateerde translatie afstand tussen twee opeenvolgende transformaties kleiner is dan deze waarde. Deze waarde is kleiner dan bij de voorgaande delen gezien het hier om minder punten gaat en de registratie bijgevolg sneller kan worden berekend. Als tweede waarde werd gekozen voor een relatieve rotatie threshold. Wanneer de rotatie tussen twee transformaties kleiner was dan 0.1° werd het algoritme beëindigd. Er werd voor een kleine waarde gekozen doordat een zuivere rotatie een kleine kans heeft om voor te komen. Als maximaal aantal iteraties werd voor 100 gekozen.

Filteren corresponderende punten:

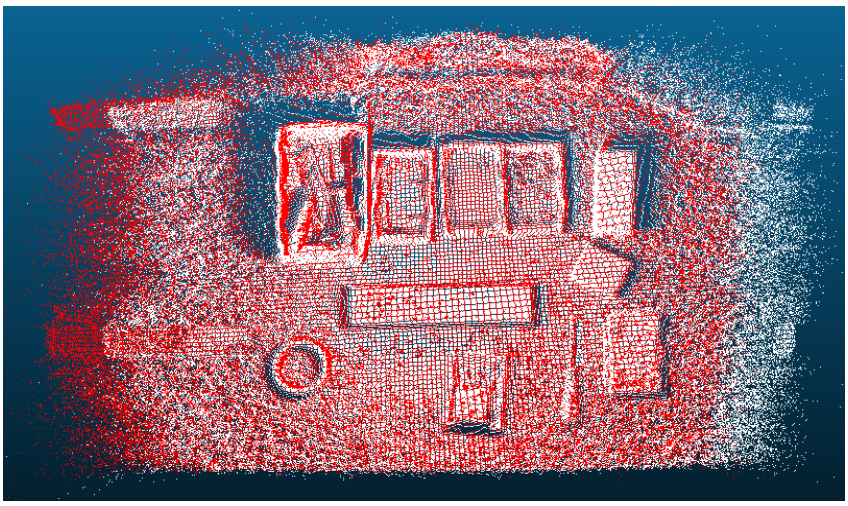
Er werd gekozen om punten waartussen de afstand groter is dan 0.5 meter weg te filteren uit de corresponderende punten. Deze waarde werd gekozen doordat de transformatie in reële situaties niet kan worden ingeschat maar er wordt verwacht dat deze nooit 0.5 m zal overschrijden. Het doel hiervan is om de ruispunten te elimineren.

4.3.2 Registratie bij verplaatsingen in x-richting

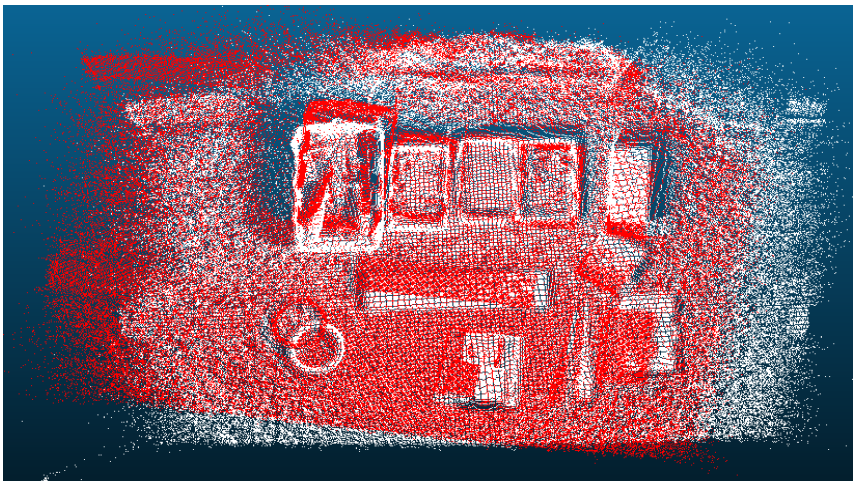
Als eerste werd de transformatie berekend van een werktafel waarbij de sensor 10 cm was verplaatst in de x-richting. De algoritmes ICP en NICP, beschreven in hoofdstuk 2.5.1, werden hiervoor gebruikt. Om deze transformatie te berekenen werd gebruikt gemaakt van keypoints in de puntenwolk die geëxtraheerd werden uit de 2D infraroodbeelden. De keypoints werden berekend met behulp van het SURF algoritme, beschreven in hoofdstuk 2.6.4. Het resultaat van de gevonden keypoints is zichtbaar in figuur 54. In figuur 55 is de via ICP gevonden transformatie zichtbaar. In deze figuur wordt de getransformeerde puntenwolk voorgesteld in het rood en de verplaatste puntenwolk in het wit. De gevonden transformatie via NICP is zichtbaar in figuur 56. De getransformeerde puntenwolk wordt voorgesteld door de rode en de verplaatste puntenwolk door de witte.



Figuur 54: Gevonden keypoints bij x-translatie: SURF



Figuur 55: Registratie m.b.v. keypoints en ICP : x-translatie



Figuur 56: Registratie m.b.v. keypoints en NICP : x-translatie

Het is zichtbaar dat ICP een goede transformatie vindt doordat in de rode en witte puntenwolken de voorwerpen op tafel grotendeels overlappen. NICP berekent echter een verkeerde transformatie. In figuur 56 is zichtbaar dat er foutief een rotatie wordt gevonden en dat de voorwerpen in de puntenwolken niet goed overlappen. Dit is onder meer zichtbaar aan het cirkelvormig voorwerp linksonder op de tafel en de opbergbak linksboven. De reden voor deze slechte transformatie is te wijten aan het feit dat de normaalrichtingen van de keypoints moeilijk kunnen worden berekend door het verschil in afstand tussen de keypoints en hun ligging op verschillende vlakken. Op basis van de resultaten van de gevonden transformaties via ICP en NICP in dit deel werd besloten om voor de overige verplaatsingen enkel nog ICP te gebruiken.

Het ICP algoritme werd tevens gebruikt voor andere translaties in de x-richting. Tabel 7 geeft een overzicht van deze registraties hun bijhorende registratietijd, hun MSE waarde en het aantal keypoints waarop de registratie werd uitgevoerd. In deze tabel is zichtbaar dat de MSE waarde voor iedere transformatie ongeveer gelijk is buiten bij deze van 15 cm. De verklaring hiervoor is dat bij de filtering van de keypoints enkele punten verkeerd gematcht werden. Dit had echter geen invloed op de kwaliteit van de gevonden transformatie. Tevens toont deze tabel dat het aantal keypoints waarop de registratie berekend wordt ondergeschikt is aan de kwaliteit van de keypoints.

Tabel 7: MSE, registratietijd en aantal keypoints bij translaties in x-richting

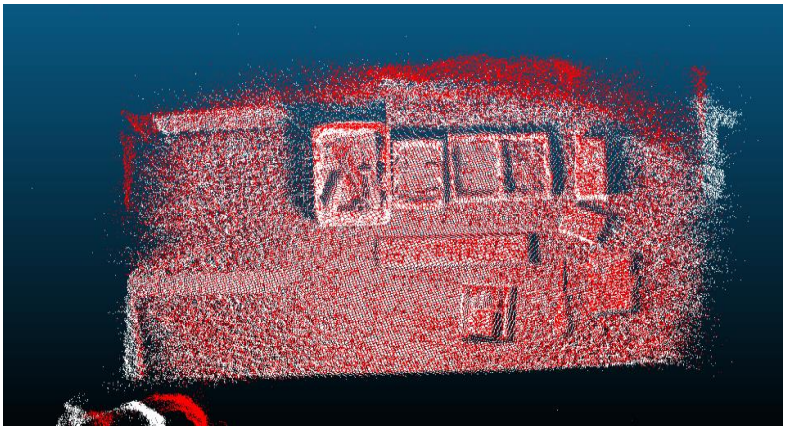
translatie x (cm)	5	10	15	20
MSE (m)	0,000792028	0,000797384	0,0012012	0,000635607
Time (ms)	36	26	21	17
# keypoints	91	41	86	45

4.3.3 Registratie bij rotaties

Als tweede test werden de transformaties van verschillende rotaties berekend om te achterhalen of de combinatie van keypoints met ICP voor het vinden van een transformatie in verschillende situaties goed presteert.

In figuur 58 wordt de gevonden transformatie getoond van een puntenwolk van een werkbank waarin de sensor een middelgrote rotatie heeft ondergaan. De gevonden transformatie is zichtbaar in de rode puntenwolk en de geroteerde werkbank is zichtbaar in de witte puntenwolk. De keypoints van de werktafel werden gevonden in de 2D beelden van de infraroodsensor. Om deze te vinden werd .. gebruikt, uitgelegd in hoofdstuk ... De gevonden keypoints zijn weergegeven in figuur 57.

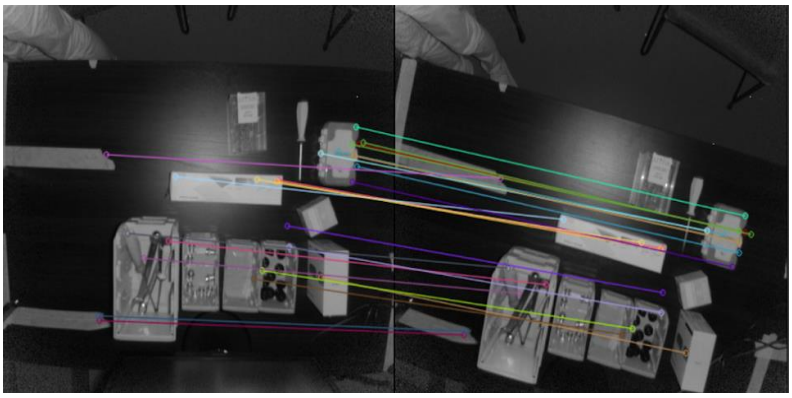
Figuur 57: Gevonden keypoints bij middelgrote rotatie: SURF



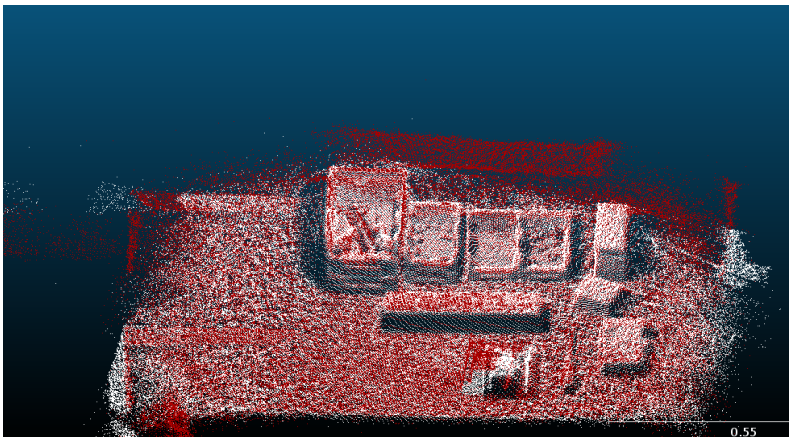
Figuur 58: Registratie m.b.v. keypoints en ICP : middelgrote rotatie

De gevonden transformatie gaf een goede benadering van de reële transformatie. Dit is zichtbaar doordat de voorwerpen in de rode en witte puntenwolken goed overlappen met elkaar. De transformatie werd berekend via ICP, uitgelegd in hoofdstuk 2.5.1.

De transformatie werd eveneens berekend voor een puntenwolk waarbij de sensor een grote rotatie had ondergaan. De resultaten van deze transformatie zijn zichtbaar in figuur 60. De witte puntenwolk stelt de puntenwolk voor van de werkbank nadat de sensor geroteerd is en de rode puntenwolk stelt de gevonden transformatie voor. De transformatie werd berekend via ICP, uitgelegd in hoofdstuk 2.5.1 . De keypoints van de werktafel werden gevonden in de 2D beelden van de infraroodsensor. Om deze te vinden werd SURF gebruikt, uitgelegd in hoofdstuk 2.6.4. De gevonden keypoints zijn weergegeven in figuur 59.



Figuur 59: Gevonden keypoints bij grote rotatie: SURF



Figuur 60: Registratie m.b.v. keypoints en ICP : grote rotatie

Uit figuur 60 is af te leiden dat de gevonden transformatie een goede benadering is voor de reële transformatie. Dit is zichtbaar doordat in beide puntenwolken de voorwerpen op de tafel goed overlappen. Het bakje onderaan in beeld overlapt niet goed. Dit wordt echter veroorzaakt door het accidenteel verplaatsen van dit voorwerp tussen de twee opnames in.

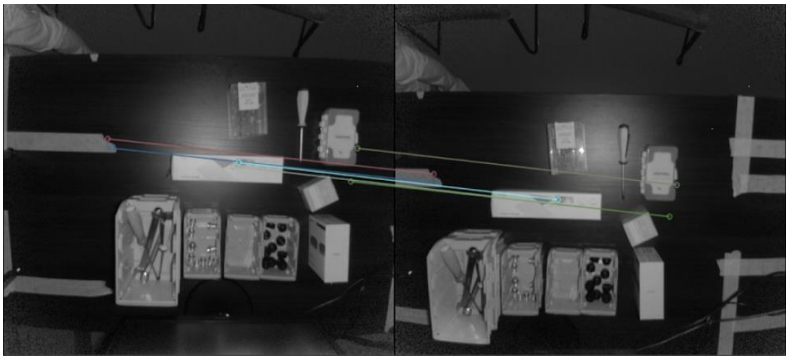
Het berekenen van de transformatie werd tevens gedaan voor een kleine rotatie. De resultaten van de MSE, registratietijd en aantal keypoints is zichtbaar in tabel 8. Deze tabel bevat eveneens deze waardes voor de middelgrote en grote rotatie. In deze tabel is zichtbaar dat de transformatie bij iedere rotatie even goed berekend werd, onafhankelijk van het aantal gevonden keypoints.

Tabel 8: MSE, registratietijd en aantal keypoints bij rotaties

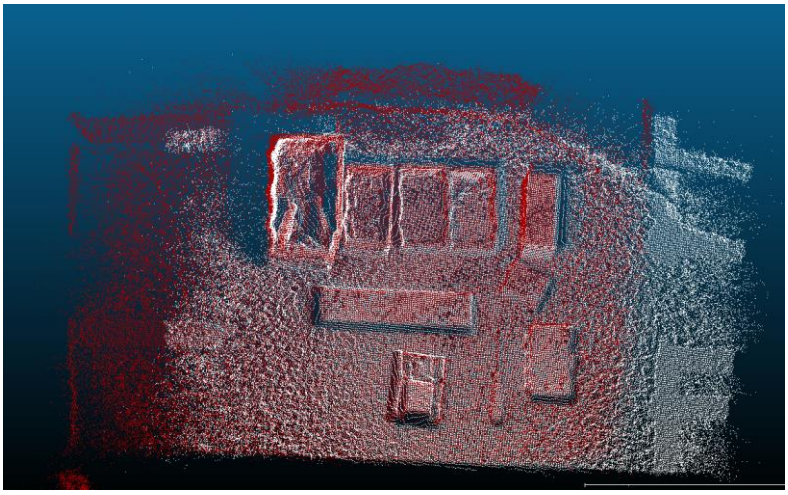
rotatie	middel	groot	klein
MSE (m)	2,93E-05	2,68E-05	6,72E-05
Time (ms)	33	22	18
# keypoints	49	25	130

4.3.4 Registratie bij verplaatsingen in x- en y-richting

Als derde test met keypoints werd een transformatie berekend van situaties waarbij de sensor in zowel de x-richting als de y-richting verplaatst was. Het resultaat van de gevonden transformatie is zichtbaar in figuur 62. In deze figuur wordt de getransformeerde puntenwolk voorgesteld in het rood en de puntenwolk na het verplaatsen van de sensor in het wit. De transformatie werd berekend via ICP, uitgelegd in hoofdstuk 2.5.1. De keypoints van de werktafel werden gevonden in de 2D beelden van de infraroodsensor. Om deze te vinden werd SURF gebruikt, uitgelegd in hoofdstuk 2.6.4. De gevonden keypoints zijn weergegeven in figuur 61.



Figuur 61: Gevonden keypoints bij x- en y-translatie: SURF



Figuur 62: Registratie m.b.v. keypoints en ICP : x- en y-translatie

In beide puntenwolken van figuur 62 overlappen de voorwerpen op de werkbank goed met elkaar. Dit toont aan dat de gevonden transformatie goed overeenkomt met de reële transformatie. Op figuur 61 is te zien dat er een beperkt aantal keypoints gevonden worden maar ondanks het lage aantal keypoints is de berekende transformatie nog steeds correct.

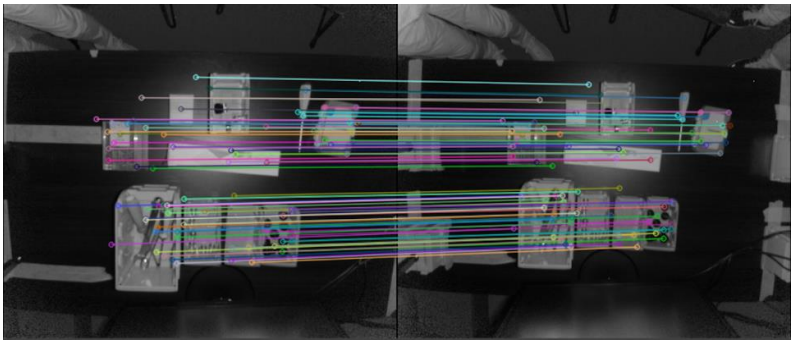
De transformatie werd tevens berekend voor andere combinaties van x- en y-waardes voor de translatie. In tabel 9 worden de resultaten van de MSE, registratietijd en aantal keypoints van elk van de registraties getoond. In deze tabel is zichtbaar aan de MSE dat voor elke registratie de transformatie ongeveer even goed berekend wordt buiten bij de translatie in beide richtingen van 10 cm. Bij deze registratie werd de verkeerde transformatie berekend. Verder was de correctheid van de berekende transformatie niet afhankelijk van het aantal gevonden keypoints.

Tabel 9: MSE, registratietijd en aantal keypoints bij translaties in x- en y-richting (y = 10 cm)

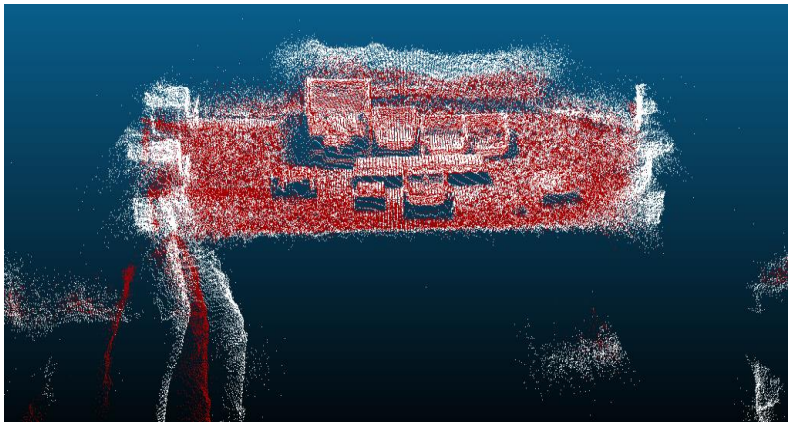
translatie x (cm)	5	10	15	20
MSE (m)	0,000180908	0,00689097	0,000245106	0,0000302
Time (ms)	17	15	16	18
# keypoints	36	17	15	6

4.3.5 Registratie bij verplaatsing in z-richting

Als laatste test werd de transformatie berekend van een situatie waarbij de hoogte van de sensor werd veranderd. Het resultaat van de gevonden transformatie is zichtbaar in figuur 64. De rode puntenwolk stelt de getransformeerde puntenwolk voor en de witte puntenwolk stelt de puntenwolk voor na de verandering in hoogte van de sensor. De transformatie werd berekend via ICP, uitgelegd in hoofdstuk 2.5.1. De keypoints van de werktafel werden gevonden in de 2D beelden van de infraroodsensor. Om deze te vinden werd SURF gebruikt, uitgelegd in hoofdstuk 2.6.4. De gevonden keypoints zijn weergegeven in figuur 63.



Figuur 63: Gevonden keypoints bij z-translatie: SURF



Figuur 64: Registratie m.b.v. keypoints en ICP : z-translatie

Om na te gaan of de correcte transformatie werd gevonden werd het vooraanzicht van de puntenwolk bekeken. Hierin is zichtbaar dat de voorwerpen op de tafel en het tafelvlak zelf goed overlappen. Dit betekent dat de gevonden transformatie goed overeenkomt met de reële transformatie.

In tabel 10 worden de resultaten van de MSE, registratietijd en aantal keypoints getoond voor de berekening van de transformatie in de z-richting. In deze tabel is zichtbaar aan de MSE waarde dat de transformatie goed berekend is.

Tabel 10: MSE, registratietijd en aantal keypoints bij translatie in z-richting

translatie z (cm)	11
MSE (m)	0,0000735
Time (ms)	16
# keypoints	79

4.3.6 Conclusie

Uit de testen uitgevoerd in het voorgaande deel bleek bij de eerste test bij een verplaatsing van 10 cm ICP beduidend beter presteerde dan NICP. Hieruit werd besloten om de overige testen enkel nog met ICP te doen. Van de 12 verschillende situaties die werden getest werd er 11 keer een correcte transformatie gevonden. Dit komt overeen met een succespercentage van 92% waarbij het vinden van een verkeerde transformatie wordt gezien als falen. Tevens kan worden geconcludeerd dat het aantal keypoints ondergeschikt is aan de kwaliteit van de keypoints voor het berekenen van een correcte transformatie

5 Besluit

Uit de testen van de algoritmes op de puntenwolk van een konijn van Stanford University bleek dat GICP het meest constant presteerde in het voldoende nauwkeurig berekenen van de transformatie. Dit bij zowel translaties, rotaties als een combinatie van beide. Hieruit wordt geconcludeerd dat GICP het beste algoritme is om transformaties te berekenen voor punten van 3D scans.

Op basis van de resultaten voor de experimenten, waarbij de puntenwolk van een lege werktafel werd gebruikt, bleek dat personen in beeld een grote invloed hebben op het vinden van de correcte transformatie. Het is dus noodzakelijk dat er enkel delen van de werkbank zitten in de punten waarop de transformatie berekend wordt. Tevens werd vastgesteld dat ruis rond de werkbank een grote invloed heeft op het correct berekenen van de transformatie. Bij ruisarme data van de werkbank werd de transformatie goed benaderd door ICP en NICP, maar niet door GICP. Voor de data van een werkbank geldt dus een andere conclusie dan voor de puntenwolk van het konijn. Bij dit soort data presteren ICP en NICP beter dan GICP.

Uit de testen van de werkbank gevuld met voorwerpen, waarbij de registratie niet meer berekend werd via puntenwolken maar op basis van keypoints uit de puntenwolken, volgde dat er in 11 van de 12 scenario's een correcte transformatie berekend werd via ICP. De keypoints in dit deel werden geëxtraheerd uit 2D beelden, die gecapteerd waren via de infraroodsensor van de Human Interface Mate. Uit de testen bleek tevens dat het aantal keypoints geen invloed heeft op de berekende transformatie zolang de kwaliteit van de keypoints goed is. De tijd voor het berekenen van de transformatie via ICP bedroeg gemiddeld 21.25 ms, dit t.o.v. 2236 ms voor het geval van de puntenwolk van de werkbank zonder ruis uit deel 2.

Aan de hand van de resultaten van de uitgevoerde experimenten in deze thesis kan gesteld worden, dat de meest potentiële methode bestaat uit een combinatie van feature detectie en ICP. Dit is dan ook de methode, die best verder geëxploreerd wordt om een robuuste herkalibratie van de extrinsieke parameters te bekomen.

Een mogelijke pad dat onderzocht kan worden in een volgende fase is een waarbij er een schatting gemaakt wordt van de verplaatsing, voordat het registratiealgoritme de verplaatsingsmatrix berekend heeft. Dit zou bijvoorbeeld kunnen door een accelerometer te bevestigen aan de HIM. Het doel hiervan is om deze schatting dan door te geven aan het registratiealgoritme. Een initiële schatting kan het registratiealgoritme ten goede komen in moeilijker te aligneren scenes.

De andere mogelijkheid die voorgesteld wordt om betere resultaten te verkrijgen, draait rond de keuze van het feature detectie algoritme. De kwaliteit van de keypoints en de bijhorende matching heeft een grote invloed op de nauwkeurigheid van de berekening van het registratiealgoritme. Het is enkel niet eenduidig te zeggen welk algoritme in elk scenario het beste presteert. Vandaar dat het interessant zou kunnen zijn om een automatische kalibratie methode te creëren, die enkele algoritmes uittest op testbeelden van de werkcel waarin de HIM geplaatst zal worden. Dit zou dan eenmalig moeten gebeuren bij het kalibreren van de HIM. Het best presterende algoritme, voor deze specifieke omgeving, kan dan vervolgens verder gebruikt worden als filter voor het registratiealgoritme.

Bibliografie

[1] L. W. Kheng, "Principal Component Analysis Leow Wee Kheng CS4243 Computer Vision and Pattern Recognition CS4243 Principal Component Analysis 1," [Online]. Available: https://www.comp.nus.edu.sg/~cs4243/lecture/camera.pdf.

[2] M. Munaro, F. Basso, S. Michieletto, E. Pagello, and E. Menegatti, "A software architecture for RGB-D people tracking based on ros framework for a mobile robot," *Stud. Comput. Intell.*, vol. 466, no. December 2015, pp. 53–68, 2013, doi: 10.1007/978-3-642-35485-4_5.

[3] R. B. Rusu, "Removing outliers using a StatisticalOutlierRemoval filter," *Point Cloud Library*. 2020, [Online]. Available: https://pointclouds.org/documentation/tutorials/statistical_outlier.html#statistical-outlier-removal.

[4] G. O'Leary, "Removing outliers using a Conditional or RadiusOutlier removal; PCL 1.1." 2012, [Online]. Available: http://pointclouds.org/documentation/tutorials/remove_outliers.php#remove-outliers.

[5] R. B. Rusu, N. Blodow, and M. Beetz, "Fast Point Feature Histograms (FPFH) for 3D Registration."

[6] M. Sinko, P. Kamencay, R. Hudec, and M. Benco, "3D registration of the point cloud data using ICP algorithm in medical image analysis," *12th Int. Conf. ELEKTRO 2018, 2018 ELEKTRO Conf. Proc.*, pp. 1–6, 2018, doi: 10.1109/ELEKTRO.2018.8398245.

[7] E. P. Bonnal, "3D Mapping of indoor environments using RGB-D Kinect camera for robotic mobile application," no. July, 2011.

[8] P. J. Besl and N. D. McKay, "A method for registration of 3D shapes," 1992.

[9] "k-d tree - Wikipedia." [Online]. Available: https://en.wikipedia.org/wiki/K-d_tree.

[10] D. Holz, A. E. Ichim, F. Tombari, R. B. Rusu, and S. Behnke, "Registration with the point cloud library: A modular framework for aligning in 3-D," *IEEE Robot. Autom. Mag.*, vol. 22, no. 4, pp. 110–124, 2015, doi: 10.1109/MRA.2015.2432331.

[11] "Normal (geometry) - Wikipedia." 2020, [Online]. Available: https://en.wikipedia.org/wiki/Normal_(geometry).

[12] Y. Chen and G. Medioni, "Object modeling by registration of multiple range images," *Proceedings - IEEE International Conference on Robotics and Automation*. pp. 2724–2729, 1991, [Online]. Available: https://graphics.stanford.edu/~smr/ICP/comparison/chen-medioni-align-rob91.pdf.

[13] K. Low, "Linear Least-squares Optimization for Point-to-plane ICP Surface Registration," *Chapel Hill, Univ. North Carolina*, no. February, pp. 2–4, 2004, [Online]. Available: https://www.iscs.nus.edu.sg/~lowkl/publications/lowk_point-to-plane_icp_techrep.pdf.

[14] A. V Segal, D. Haehnel, and S. Thrun, "Generalized-ICP," *Robot. Sci. Syst.*, vol. 2, p. 435, 2009.

[15] P. Biber, "The Normal Distributions Transform: A New Approach to Laser Scan Matching," *IEEE Int. Conf. Intell. Robot. Syst.*, vol. 3, no. June, pp. 2743–2748, 2003, doi: 10.1109/iros.2003.1249285.

[16] M. Martin, *The Three-Dimensional Normal-Distributions Transform*, vol. 10. 2009.

[17] "3D Time of Flight (3D ToF)," *Analog Devices*. 2020, [Online]. Available: https://www.analog.com/en/applications/technology/3d-time-of-flight.html.

[18] C. Harris, M. Stephens, "A combined corner and edge detector", Alvey Vision Conference, 1988.

[19] E. Rosten, T. Drummond, "Machine learning for high-speed corner detection", ECCV 2006, 2006, Vol.3951, p.430-443.

[20] D. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints", International journal of computer vision, 2004-11, Vol.60 (2), p.91-110.

[21] OpenCV, "Introduction to SIFT", [Online]. Available:https://docs.opencv.org/master/da/df5/tutorial_py_sift_intro.html, [Geopened 13 maart 2021].

[22] U. Sinha, AIShack, "SIFT: Theory and practice", [Online]. Available: https://aishack.in/tutorials/sift-scale-invariant-feature-transform-introduction/, [Geopened 13 maart 2021].

[23] H. Bay, T. Tuytelaars, L. Van Gool, "SURF: Speeded Up Robust Features", ECCV, 2006, Vol.3951, p.404-417

[24] OpenCV, "Introduction to SURF (Speeded-Up Robust Features)", [Online]. Available: https://docs.opencv.org/master/df/dd2/tutorial_py_surf_intro.html, [Geopened 13 maart 2021].

[25] M. Calonder, V. Lepetit, C. Strecha, P. Fua, "BRIEF: Binary Robust Independent Elementary Features", ECCV 2010, Vol.6314 (4), p.778-792

[26] E. Rublee, V. Rabaud, K. Konolige, G. Bradski, "ORB: an efficient alternative to SIFT or SURF", International Conference on Computer Vision, 2011-11, p.2564-2571

[27] S. Leutenegger, M. Chli, Y. Siegwart, "BRISK: Binary Robust Invariant Scalable Keypoints", International Conference on Computer Vision, 2011-11, p.2548-2555

[28] E. Mair, D. Hager, D. Burschka, M. Suppa, G. Hirzinger, "Adaptive and Generic Corner Detection Based on the Accelerated Segment Test", ECCV 2010, Vol.6312 (2), p.183-196

[29] F. Alcantarilla, J. Nuevo, A. Bartoli, "Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces", BMVC 2013

[30] P. Alcantarilla, A. Bartoli, A. Davison, "KAZE Features", ECCV, 2012, Vol.7577 (6), p.214-227

[31] S. Tareen, Z. Saleem, "A Comparative Analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK", International Conference on Computing, Mathematics and Engineering Technologies (iCoMET), 2018-03, p.1-10

[32] D. Lucas, T. Kanade, "An iteratieve image registration technique with an application to stereo vision", DARPA image understanding workshop, April 1981, p.121-130

[33] "The Stanford 3D Scanning Repository," *Online*. [Online]. Available: http://graphics.stanford.edu/data/3Dscanrep/.