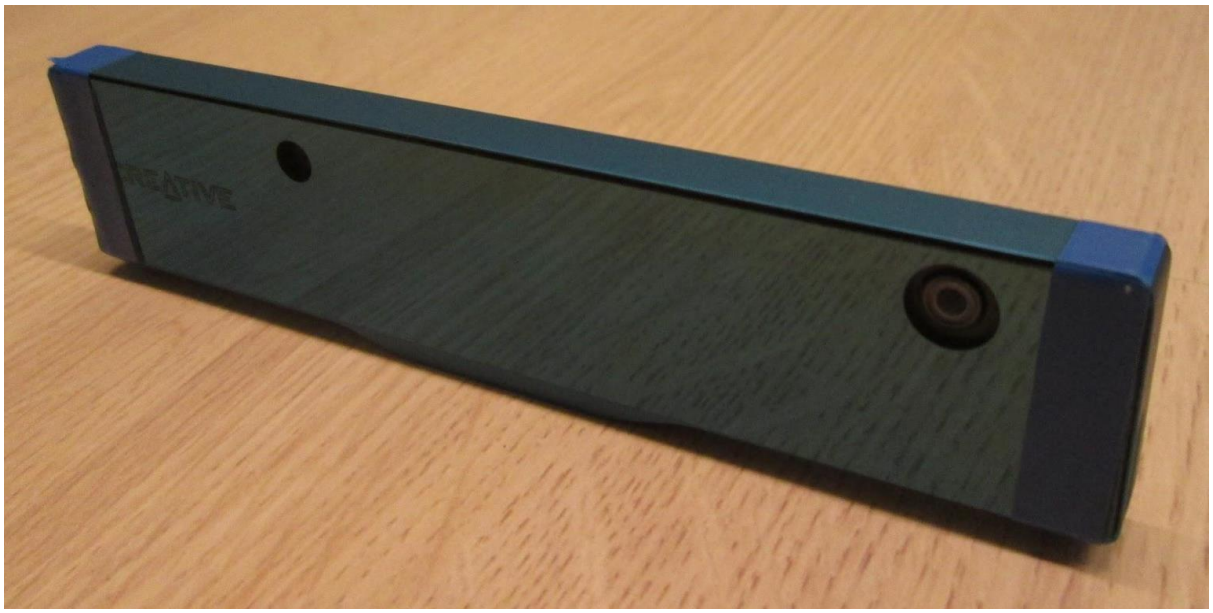
Ontwikkelen van 3D-SLAM voor de Slimme Rolstoel in ROS

Afstudeerverslag



3D-camera Intel RealSense ZR300

Student:
Richard Kokx
14139227

Opdrachtgever:
Dr. John Bolte
Lectoraat Smart Sensor Systems
Haagse Hogeschool

Examinatoren:
Expert examinator: J. van Peski
Begeleidend examinator: A.G.P. Pronk

Datum: 17 december 2018
Versie: 1.0

Referaat

Kokx, Richard
Ontwikkelen van 3D-SLAM voor de Slimme Rolstoel in ROS.

Afstudeerverslag van Richard Kokx, geschreven in het kader van afstuderen bij de opleiding Technische Informatica van de faculteit IT & Design aan de Haagse Hogeschool.

Het verslag behandelt het proces dat doorlopen is tijdens het afstuderen van Richard Kokx bij het lectoraat Smart Sensor Systems van de Haagse Hogeschool

Descriptoren

- ROS
- SLAM
- Intel Realsense ZR300
- RGBD-camera
- RGBDSLAMv2
- OctoMap
- 3D-mapping
- Voxel map
- Point cloud

Voorwoord

Beste lezer,

Voor u ligt het afstudeerdossier van Richard Kokx, dat geschreven is voor de opleiding Technische Informatica van de Haagse Hogeschool. Hierin wordt verslag gedaan van de werkzaamheden die ik heb uitgevoerd, en de producten die ik heb ontwikkeld, bij mijn afstuderen bij het lectoraat Smart Sensor Systems van de Haagse Hogeschool van 27 augustus tot 20 december 2018.

Ik wil graag het lectoraat bedanken voor de mogelijkheid die mij geboden is om bij het lectoraat af te studeren. Hierbij wil ik met name mijn begeleider, dhr. P.R. Fraanje, bedanken voor alle hulp en begeleiding.

Verder wil ik dhr. A.G.P. Pronk en dhr. J. van Peski, mijn examinatoren van de Haagse Hogeschool, bedanken voor de feedback die ze mij tijdens het afstuderen hebben gegeven. Ook wil ik mijn studieloopbaanbegeleider, dhr. R.A. van Neijhof bedanken voor zijn hulp en begeleiding, zowel tijdens mijn studie als tijdens deze afstudeeropdracht.

Tot slot wil ik graag mijn ouders bedanken voor hun onvoorwaardelijke hulp en steun.

Richard Kokx
Zoetermeer, 16 december 2018

Inhoudsopgave

1. Inleiding en leeswijzer

Zelfrijdende technologieën zijn steeds meer in opkomst. Auto's gebruiken allerlei technieken om de bestuurder te helpen en een deel van het autorijden over te nemen, en in fabrieken rijden robots al helemaal zelfstandig rond. Zelfrijdende shuttlebusjes zijn al een realiteit, en het zal niet lang meer duren voordat de eerste proeven in Nederland met volledig zelfrijdende auto's op de openbare weg zullen plaatsvinden.

Het lectoraat Smart Sensor Systems van de Haagse Hogeschool wil graag kennis opdoen over zelfrijdende systemen. Hiervoor is het lectoraat bezig met het ontwikkelen van een slimme rolstoel, die als platform kan dienen om te experimenteren met soft- en hardwaresystemen die te maken hebben met autonoom rijden.

Een van de technologieën die gebruikt wordt voor zelfrijdende systemen en robots is Simultaneous Localization and Mapping (SLAM). Door het gebruik van SLAM-algoritmen kan een robot op basis van sensoren een kaart van zijn omgeving maken, en tegelijkertijd bepalen waar op deze kaart hij zich bevindt.

Mijn afstudeeropdracht bij het lectoraat Smart Sensor Systems is het ontwikkelen van een systeem voor de slimme rolstoel dat met behulp van SLAM-technologie een driedimensionale kaart van de omgeving kan maken. Deze kaart moet gemaakt worden op basis van data uit een 3D-camera.

Het doel van dit verslag is om duidelijk te maken hoe mijn afstuderen bij het lectoraat Smart Sensor System is verlopen, en welke resultaten dit heeft opgeleverd.

In dit verslag zal eerst een beschrijving worden gegeven van de organisatiestructuur waarbinnen mijn afstudeeropdracht is uitgevoerd, en hoe deze opdracht eruit ziet. Vervolgens wordt de aanpak die ik heb gekozen om deze opdracht uit te voeren behandeld. Daarna wordt een overzicht gegeven van de gemaakte probleemanalyse, waarna zal worden toegelicht hoe het te ontwikkelen systeem is ontworpen, geïmplementeerd, en getest. Tot slot zal zowel een product- als een procesevaluatie worden uitgevoerd, en zal een toelichting worden gegeven op de door mij uitgevoerde beroepstaken. Na het verslag is een lijst met geraadpleegde referenties en een verklaring van gebruikte afkortingen te vinden. In de bijlagen zijn mijn Plan van Aanpak, afstudeeropdracht, testplan, opgeleverde broncode, en meetresultaten opgenomen.

2. Context en opdracht

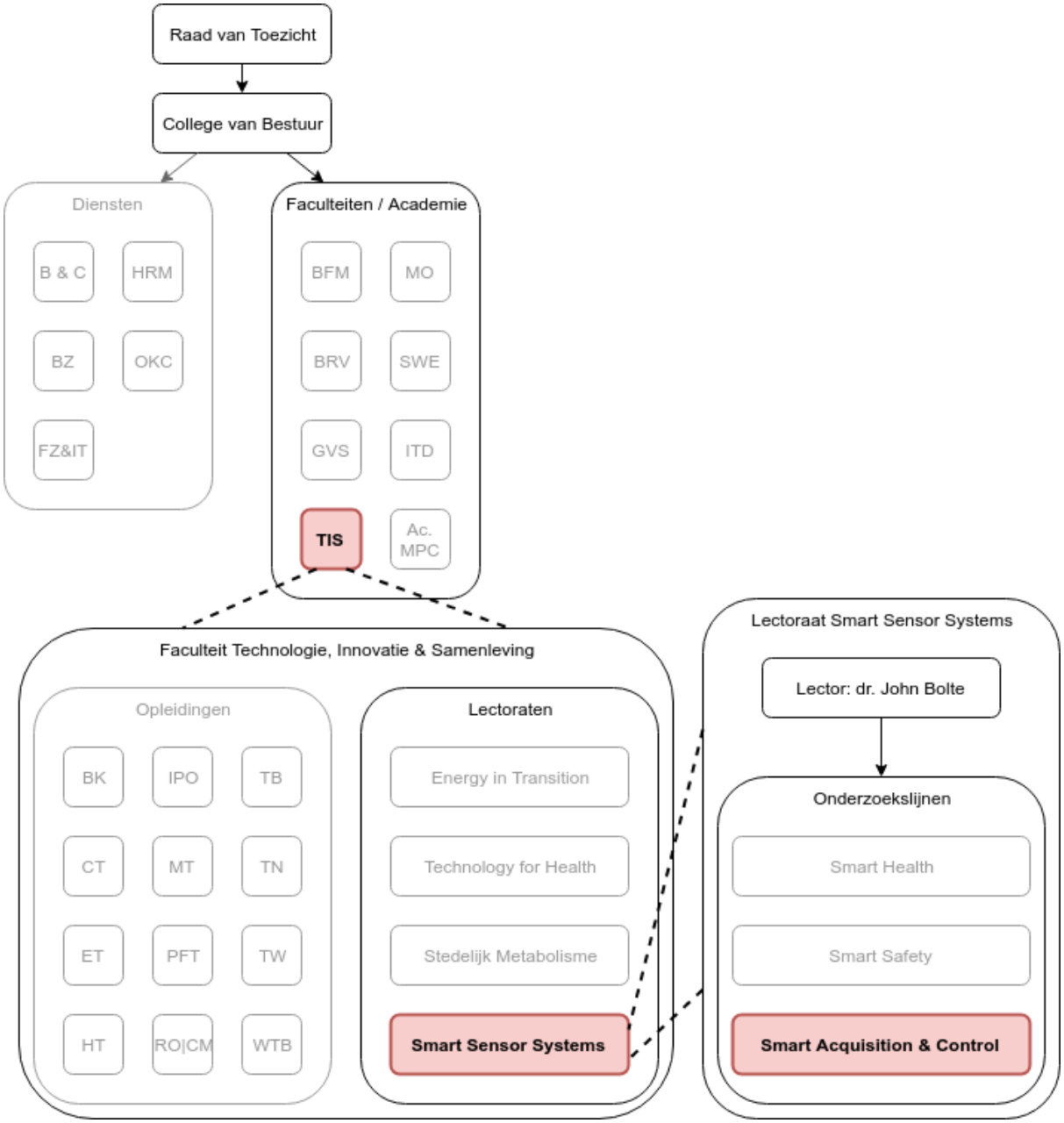
Mijn afstudeeropdracht heb ik uitgevoerd bij het lectoraat Smart Sensor Systems van de Haagse Hogeschool. In dit hoofdstuk zal beschreven worden wat de positie is van het lectoraat binnen de Haagse Hogeschool, en wat mijn positie is bij het lectoraat. Verder zal ook mijn afstudeeropdracht beschreven worden.

2.1. Organisatieomschrijving Haagse Hogeschool

De Haagse Hogeschool is een van de grootste onderwijsinstellingen voor hoger onderwijs in Nederland, met bijna 26.000 studenten en meer dan 2000 medewerkers [1]. De primaire taak van de Haagse Hogeschool is onderwijs te bieden aan al deze studenten. Hiervoor is de Haagse Hogeschool opgedeeld in zeven faculteiten met in totaal ca. 70 opleidingen. Naast onderwijs heeft de Haagse Hogeschool ook de taak om onderzoek te verrichten. Hiervoor is een aantal lectoraten ingesteld. Elk lectoraat wordt geleid door een lector, en bestaat verder uit een kenniskring van docenten en experts. Een overzicht van deze structuur, en mijn positie daarin, is te vinden in Figuur 1.

Elk lectoraat valt onder een van de faculteiten die de Haagse Hogeschool heeft, maar staat binnen de faculteit wel los van de opleidingen [2]. Doordat de lectoraten relatief onafhankelijk functioneren van de opleidingen, is het mogelijk om vakoverstijgend onderzoek te doen, door docenten van verschillende opleidingen en faculteiten, met elk hun eigen expertises, samen te laten werken.

Een van de lectoraten is het lectoraat Smart Sensor Systems, dat wordt geleid door dr. John Bolte. De onderzoeken van Smart Sensor Systems zijn opgedeeld in drie onderzoekslijnen: Smart Health, Smart Safety, en Smart Acquisition and Control. Smart Health houdt zich met name bezig met onderzoek naar draagbare sensoren zoals smartwatches, metingen doen in kassen, en andere onderzoeken op het gebied van gezondheid. Bij Smart Safety ligt de focus op bijvoorbeeld het meten van de mate waarin werknemers worden blootgesteld aan gevaren als straling, geluid, en fijnstof. Tot slot is er de onderzoekslijn Smart Acquisition and Control, die gedeeltelijk in dienst staat van de twee andere onderzoekslijnen. Hierin wordt onder andere onderzoek gedaan naar autonoom bewegen, wat weer gebruikt kan worden in de andere onderzoekslijnen om automatisch data te verzamelen. Onder dit onderzoek naar autonoom bewegen valt bijvoorbeeld het LearningLab URBINN, dat zich bezig houdt met zelfrijdende voertuigen, en het project Slimme Rolstoel, dat wordt ingezet als platform waarmee onderzoek gedaan wordt naar het gebruik van verschillende sensoren om kaarten van de omgeving te maken en hierop te navigeren. Bij dit project Slimme Rolstoel heb ik mijn afstudeeropdracht uitgevoerd.



Figuur 1: Organogram Haagse Hogeschool, met in rood dikgedrukt daarin aangegeven wat mijn positie is binnen de organisatiestructuur.

2.2. Opdrachtomschrijving

2.2.1. Aanleiding

Het lectoraat Smart Sensor Systems wil graag expertise in huis halen over zelfrijdende systemen. Daarom doet het lectoraat momenteel onderzoek naar indoor navigatiesystemen voor mobiele robots. Een van de toepassingen van dit onderzoek is het ontwikkelen van een slimme rolstoel. Deze rolstoel moet zelfstandig door een ruimte kunnen navigeren. Dit is bijvoorbeeld nuttig voor patiënten die vanwege motorische beperkingen moeite hebben met het zelf besturen van de rolstoel. Het doel hiervan is om met slimme navigatietechnieken de elektrische rolstoel veiliger en gebruiksvriendelijker te maken.

2.2.2. Probleemstelling

Op dit moment werkt de slimme rolstoel nog niet goed genoeg. De navigatie is nog niet goed genoeg, waardoor de botsdetectie (op basis van afstandsmetingen met ultrasoonsensoren) nog erg belangrijk is om botsingen te voorkomen. Zo kan de rolstoel bijvoorbeeld geen tafels detecteren, omdat de bestaande sensoren alleen de tafelpoten zien, maar niet het tafelblad.

2.2.3. Doelstelling

Het doel van mijn afstudeeropdracht is het verbeteren van de navigatie van de rolstoel met behulp van een 3D-camera. Op basis van deze 3D-camera moet de rolstoel een driedimensionale kaart van de omgeving gaan vormen, die gebruikt kan worden voor de navigatie en om obstakels te vermijden.

Ook moet er gemeten worden wat de nauwkeurigheid en betrouwbaarheid van de te ontwikkelen systemen is. Hiermee kan worden bepaald hoe accuraat het in kaart brengen van de omgeving is. Ook kunnen de resultaten van deze metingen gebruikt worden als basis om te bepalen wat het effect is van latere experimenten van het lectoraat, zoals het vervangen of aanpassen van deelsystemen.

2.2.4. Producten

Er is een aantal producten dat tijdens en aan het einde van de afstudeerperiode moet worden opgeleverd. Ten eerste is er het systeem voor het maken van de 3D-kaart en het gebruik van deze kaart voor de navigatie. Dit zal bestaan uit een analyse van de bestaande en te ontwikkelen systemen, een architectuurontwerp, en de implementatie zelf met documentatie hierover.

Een ander belangrijk product is een testverslag van het systeem. Dit testverslag zal bestaan uit een volledig reproduceerbaar testplan, dat ook door anderen gebruikt kan worden om vergelijkbare metingen uit te voeren. De resultaten van het uitvoeren van dit testplan zullen ook in dit testverslag komen.

Als basis voor de opdracht geldt het door mij opgestelde Plan van Aanpak, waarin beschreven wordt hoe ik te werk zal gaan tijdens mijn afstuderen. Ook zal er een onderzoeksverslag geschreven worden, met de resultaten van mijn literatuuronderzoek naar reeds bestaande systemen.

3. Aanpak

In dit hoofdstuk wordt beschreven hoe ik mijn afstudeeropdracht heb aangepakt. Delen van dit hoofdstuk zijn gebaseerd op mijn Plan van Aanpak. Het volledige PvA is te vinden in bijlage A.

3.1. Afbakening

Er is een aantal onderdelen die duidelijk wel bij mijn afstudeeropdracht horen, maar ook een aantal onderdelen die er in principe niet bij horen. Het is belangrijk om dit expliciet te noemen, om er voor te zorgen dat iedereen dezelfde verwachtingen heeft van mijn afstuderen.

Wat sowieso wel bij mijn opdracht hoort is het ontwikkelen van een 3D-mappingsysteem, dit is immers de kern van mijn afstuderen. Het definiëren van een reproduceerbare testomgeving hoort ook bij de afstudeeropdracht. Ook is het mijn verantwoordelijkheid om ervoor te zorgen dat het 3D-mappingsysteem uitbreidbaar is.

Het toevoegen van semantische informatie, of een mogelijkheid om semantische informatie toe te voegen tijdens het gebruik van het systeem, hoort in principe niet bij mijn afstudeeropdracht. Wat ook niet bij mijn opdracht hoort, is het aanbrengen van algemene verbeteringen aan de rolstoel, zoals het aanpassen van bedrading of hardware die niet expliciet met de 3D-camera te maken heeft.

3.2. Fasering

Om overzicht te krijgen over de uit te voeren taken, en om deze efficiënt in te kunnen plannen, is de opdracht opgedeeld drie primaire fases: een opstartfase; een hoofdfase, en een afrondingsfase. Elke primaire fase is weer in te delen in subfases.

De opstartfase begint met een onderzoeksfase, waarin ik onderzoek doe naar bestaande 3D-mappingsystemen door het lezen van bestaande literatuur en het vergelijken van de beschikbare mogelijkheden, systemen, algoritmen, etc. Vervolgens ga ik aan de slag met het kennismaken met en analyseren van de bestaande systemen van de rolstoel. In deze kennismakingsfase leer ik ook ROS (Robot Operating System, zie §4.5.) kennen. Als afsluiting van de opstartfase schrijf ik mijn Plan van Aanpak.

Hierop volgt de hoofdfase. Deze hoofdfase bestaat uit een globale ontwerpfase, een aantal ontwikkelfases (die elk weer bestaan uit een ontwerp- implementatie- en testfase), en tot slot een globale testfase. Ik maak eerst een globaal ontwerp, waarin het volledige systeem wordt ingedeeld in modules, die elk in een eigen ontwikkelfase geplaatst zullen worden. Het aantal modules bepaalt dus hoeveel ontwikkelfases er zullen zijn. Deze opdeling in modules helpt bij het maken van de gedetailleerde planning, en kan gebruikt worden om de voortgang te bewaken. Deze opdeling betekent ook een incrementeel proces, waarbij elke volgende module afhankelijk is van alle voorgaande modules. Het belangrijkste verschil met dit incrementele proces en een watervalproces is dat er nu niet vooraf een gedetailleerd ontwerp van elk subsysteem gemaakt wordt. Het voordeel hiervan is dat bij het maken van

een detailontwerp van een subsysteem gebruik gemaakt kan worden van de kennis die is opgedaan bij de implementatie van het voorgaande subsysteem. Hier is voor gekozen omdat deze methode beter past bij de manier waarop ROS-onderdelen werken, namelijk met losse ROS-nodes waarbij data van een eerste node wordt doorgegeven naar een volgende node.

In de ontwikkelfases ga ik deze modules een voor een ontwerpen, implementeren, en testen. Het ontwerpen zal vaak neerkomen op het analyseren van de beschikbare ROS-packages die de benodigde functionaliteit van die module vervullen. Het implementeren is dan het installeren van de betreffende package, deze werkend krijgen, en zorgen dat de verbinding tussen de nieuwe package en eerder geïnstalleerde packages werkt. Voor het testen is het vooral belangrijk dat alle software goed samenwerkt, het kwalitatieve testen komt pas later.

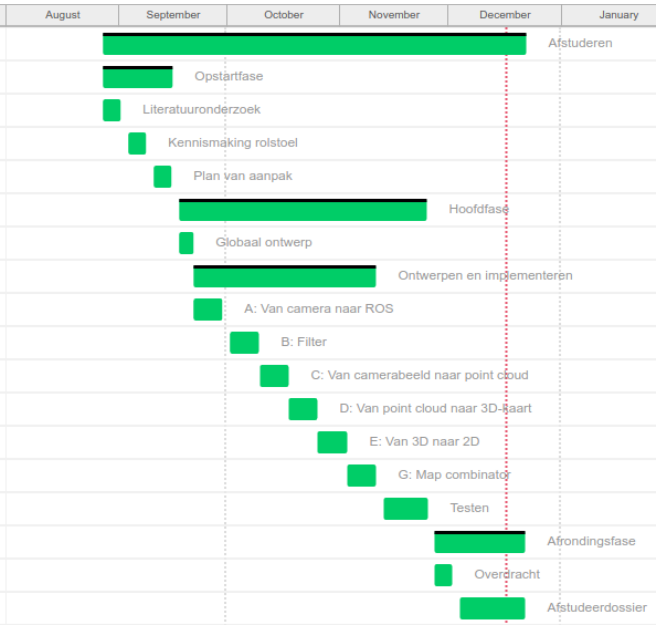
Als slot van de hoofdfase zal ik het volledige systeem kwalitatief en kwantitatief testen. Hiervoor zal ik een testplan schrijven waarin een reproduceerbare testomgeving wordt gedefinieerd. Vervolgens zal ik dit testplan uitvoeren om te bepalen hoe goed de rolstoel functioneert.

Tot slot zal ik in de afrondingsfase mijn opdracht afronden en overdragen aan het lectoraat. Ook zal ik in deze fase de laatste hand leggen aan mijn afstudeerdossier.

3.3. Planning

In §3.2. zijn drie primaire fases genoemd. Voor de eerste fase, de opstartfase, is ingeschat dat deze ongeveer drie weken zal duren. Hiervoor is een week voor het literatuuronderzoek, een week voor het kennismaken met de bestaande systemen van de rolstoel, en een week voor het schrijven van het plan van aanpak. De volgende primaire fase, de hoofdfase, zal significant langer duren, hiervoor zijn tien weken gerekend. Acht weken hiervan zijn voor het ontwerpen en implementeren, en de laatste twee weken zijn voor het uitvoerig testen van het volledige systeem. Tot slot is er de afrondingsfase van vier weken, waarvan een week is voor de afronding en overdracht van de afstudeeropdracht, en drie weken voor het afronden en controleren van het afstudeerdossier. Een vereenvoudigd schematisch overzicht van deze planning is te vinden in Figuur 2. Hierin worden zes modules (A t/m E en G) genoemd, een uitleg van wat deze modules inhouden is te vinden in §5.1.

name	start	End	dur.
Afstuderen	8/27/2018	12/20/2018	84
Opstartfase	8/27/2018	9/14/2018	15
Literatuuronderzoek	8/27/2018	8/31/2018	5
Kennismaking rolstoel	9/3/2018	9/7/2018	5
Plan van aanpak	9/10/2018	9/14/2018	5
Hoofdfase	9/17/2018	11/23/2018	50
Globaal ontwerp	9/17/2018	9/20/2018	4
Ontwerpen en implementeren	9/21/2018	11/9/2018	36
A: Van camera naar ROS	9/21/2018	9/28/2018	6
B: Filter	10/1/2018	10/8/2018	6
C: Van camerabeeld naar point cloud	10/9/2018	10/16/2018	6
D: Van point cloud naar 3D-kaart	10/17/2018	10/24/2018	6
E: Van 3D naar 2D	10/25/2018	11/1/2018	6
G: Map combinator	11/2/2018	11/9/2018	6
Testen	11/12/2018	11/23/2018	10
Afrondingsfase	11/26/2018	12/20/2018	19
Overdracht	11/26/2018	11/30/2018	5
Afstudeerdossier	12/3/2018	12/20/2018	14



Figuur 2: Gantt-chart oorspronkelijke planning

3.4. Wijziging opdracht en planning

Tijdens het uitvoeren van de opdracht bleek al vrij snel dat ik me niet aan de oorspronkelijke planning kon gaan houden. In deze sectie wordt beschreven welke wijzigingen er in de oorspronkelijke planning gemaakt zijn en waarom deze wijzigingen nodig waren. In Figuur 3 is in een Gantt-chart aangegeven wanneer welke onderdelen daadwerkelijk zijn uitgevoerd. De eerste wijziging in de planning was bij het literatuuronderzoek, dat langer bleek te duren dan ingepland. Hier zijn uiteindelijk drie weken voor gebruikt in plaats van een, waarbij in de laatste week tegelijkertijd het plan van aanpak is geschreven. Ook is de kennismaking met de rolstoel verplaatst naar ná het PvA. Uiteindelijk heeft de opstartfase hierdoor vier in plaats van drie weken geduurd.

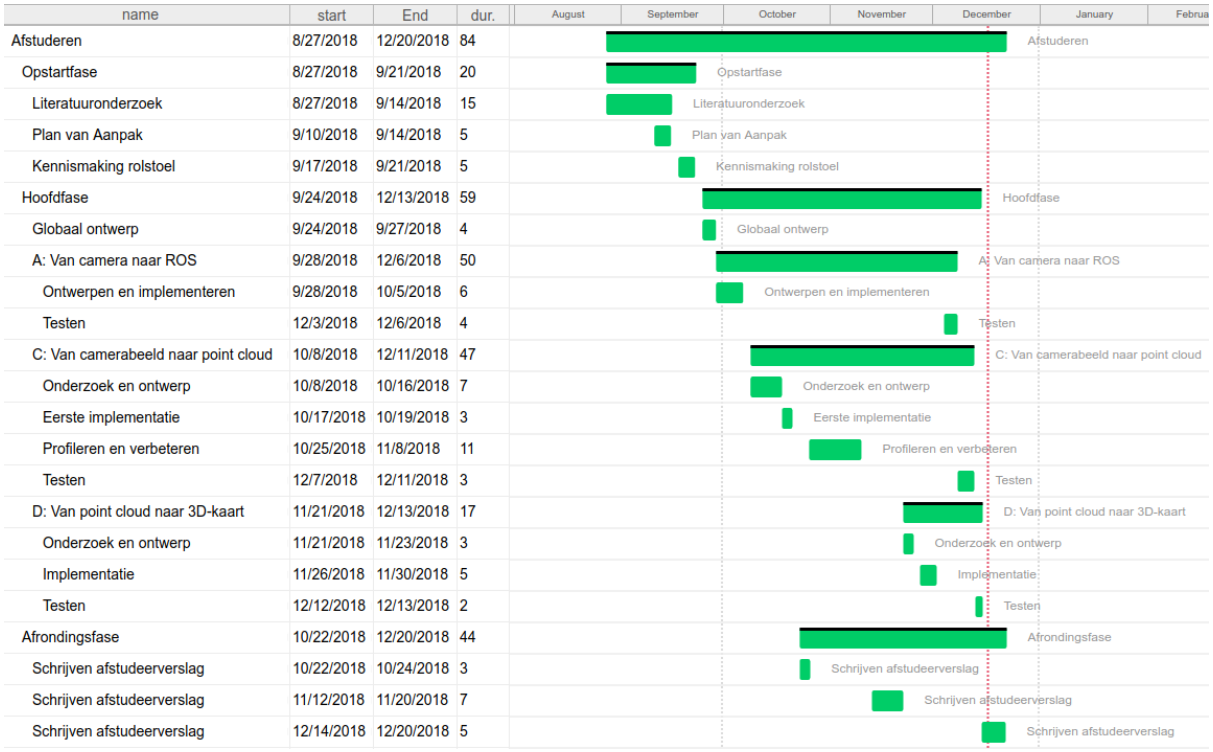
Het begin van de hoofdfase is uitgevoerd zoals gepland, met vier dagen voor het globale ontwerp, en vervolgens zes dagen voor de eerste module. Daarna is echter in overleg met de bedrijfsmentor besloten dat het nog niet nuttig was om te beginnen met module B, omdat pas als er meerdere volgende modules geïmplementeerd waren bepaald kon worden of module B überhaupt nodig was. Daarom is eerst begonnen aan module C.

Voor module C was eerst meer onderzoek nodig voordat een ontwerp gemaakt kon worden, dit heeft een week geduurd. Het ontwerpen en implementeren kostte vervolgens nog een week. Hierna ben ik drie dagen bezig geweest om mijn eerste conceptafstudeerverslag te schrijven. Vervolgens ben ik bezig geweest met het profileren en verbeteren van module C, omdat de performance van deze module niet voldoende was.

Het profileren en verbeteren van module C kostte redelijk wat tijd. Hierdoor kwam de rest van de planning in gevaar, en leek het erop dat het niet zou lukken om de volledige opdracht af te maken. In overleg met de bedrijfsmentor, is toen nagedacht over waar de prioriteit moest liggen: welke onderdelen zijn het belangrijkst om een nuttige bijdrage te leveren aan

het werk van het lectoraat? Er is toen besloten om er in ieder geval voor te zorgen dat modules A, C, en D afgemaakt zouden worden, om ook tijd te hebben deze modules uitgebreid te testen. Bij dit testen moet dan niet alleen op functionaliteit worden gelet, maar moet er ook kwantitatief getest worden, zodat er informatie komt over de nauwkeurigheid en betrouwbaarheid van het systeem.

Het ontwerpen en implementeren van module D ging redelijk volgens planning. Hierna zijn de tests van alle drie de modules uitgevoerd. Het belangrijkste verschil met de planning hier is dat alle tests aan het einde zijn uitgevoerd, in plaats van meteen na de bijbehorende implementatiefases. Tot slot is er nog de afrondingsfase, deze is niet geheel volgens planning uitgevoerd omdat tijdens het afstudeerproces al grote delen van het verslag geschreven moesten worden ten behoeve van de conceptbespreking en het tussentijds assessment. Hierdoor kon de laatste schrijffase ook ingekort worden.



Figuur 3: Gantt-chart zoals uitgevoerd

4. Analyse bestaande en nieuwe systemen

4.1. Vooronderzoek

Er is eerst onderzoek gedaan naar bestaande literatuur over semantic mapping. Uit dit onderzoek blijkt dat verschillende onderzoeksgroepen verschillende definities van semantic mapping gebruiken. Zo wordt in het werk van Sunderhauf et al. per camerabeeld bepaald welk type ruimte zichtbaar is, en op basis hiervan wordt een kaart gemaakt met per ruimte een indeling in een categorie (zoals keuken, kantoor, gang, etc.) [4]. Door Blodow et al. wordt iets heel anders bedoeld met semantic mapping, hier wordt namelijk in een enkele ruimte (in dit geval een keuken) bepaald waar kastdeurtjes, lades, en tafeloppervlakten zijn. Vervolgens wordt ook ingeschat hoe bepaalde kastdeuren open zullen gaan aan de hand van de positie van handgrepen, en wordt dit gecontroleerd door daadwerkelijk te proberen de kasten te openen [5].

Een hele andere aanpak is te vinden in de thesis van Gallart del Burgo [6]. Hierin wordt op basis van laserscans een tweedimensionale kaart van de omgeving gemaakt, die op basis van een deurdetectiealgoritme wordt omgezet in een topologische kaart. Vervolgens wordt per kamer op basis van objectdetectie bepaald wat voor soort kamer het waarschijnlijk is. Zo is bijvoorbeeld de kans vrij groot dat een kamer met veel bekers een keuken is, terwijl een kamer waar een bureau en een beeldscherm gevonden wordt waarschijnlijk een kantoor is.

Al deze semantic mapping-technieken bieden wellicht interessante mogelijkheden voor het verkrijgen van semantische informatie, maar ze doen geen van allen wat wij zoeken, namelijk het maken van een driedimensionale kaart waarmee genavigeerd kan worden door een ruimte en obstakels vermeden kunnen worden. Daarom is besloten om eerst aan de slag te gaan met het maken van, en navigeren met, een driedimensionale kaart. Om een beter beeld te krijgen van wat er nodig was, zijn eerst de bestaande systemen van de rolstoel bekeken. Hoewel deze systemen redelijk werkten voor de doelen van de projectgroep die eerder aan de rolstoel had gewerkt, zijn ze niet bruikbaar voor de doeleinden van mijn afstudeeropdracht (zie §4.3). Daarom is besloten dat er een eigen systeem ontworpen zou worden, onafhankelijk van de eerder ontwikkelde rolstoelsystemen.

4.2. Eisen

Het nieuw te ontwikkelen systeem moet aan een aantal eisen voldoen. Deze eisen zijn te vinden in Tabel 1. De eisen zijn genummerd, nummers tussen haakjes in deze paragraaf verwijzen naar eisen in de tabel. In de tabel is ook aangegeven hoe belangrijk elke eis is volgens de MoSCoW-methode [7]. Deze eisenlijst is opgesteld aan de hand van gesprekken met de bedrijfsmentor, gebaseerd op de doelstellingen van de opdracht (zie §2.2.) en de reeds beschikbare kennis. Deze lijst is tijdens het proces ook nog aangepast waar nodig, toen bleek dat de doelstellingen van de opdracht veranderden. Er wordt verwacht dat als aan al deze eisen is voldaan, er een goed functionerend systeem is.

De eerste eis is dat het systeem data uit een 3D-camera moet uitlezen (eis 1). Hiervoor is een Intel RealSense ZR300 RGBD-camera[1] beschikbaar, maar als blijkt dat deze niet

[1] Een RGBD-camera ziet zowel kleur (rood, groen, blauw) als diepte

geschikt is, is het eventueel mogelijk een andere camera aan te schaffen (eis 7). Op basis van de data moet het systeem een driedimensionale kaart kunnen maken van de omgeving (eis 2). Vervolgens is het plan dat de rolstoel op basis van de 3D-kaart obstakels kan gaan vermijden (eis 5). Er is een sterke voorkeur vanuit het lectoraat om dit systeem in ROS te maken (eis 8). Dit is onder andere omdat de huidige rolstoel al gebruik maakt van ROS, maar vooral omdat het gebruik van ROS ervoor zorgt dat het systeem uit zichzelf erg modulair wordt, waardoor het eenvoudig uit te breiden is. Dit is nodig omdat het lectoraat het systeem graag uitgebreid ziet worden met een systeem dat semantische informatie levert (eis 6). Dit is echter geen onderdeel van mijn afstudeeropdracht. Ook is het in ROS mogelijk om subsystemen te vervangen, bijvoorbeeld door een systeem dat voor een bepaalde berekening een ander algoritme toepast, of dat data uit een ander hardwarecomponent levert,

Voor het testen is het belangrijk dat er een testomgeving wordt gedefinieerd. Deze testomgeving moet reproduceerbaar zijn, zodat het bijvoorbeeld mogelijk is om verschillende algoritmes of hardware met elkaar te vergelijken (eis 3). Ook moet er met behulp van deze testomgeving kwantitatieve informatie worden verzameld over het functioneren van de onderdelen van het systeem, met name over de nauwkeurigheid van afstands- en afmetingenbepaling (eis 4).

#	Eis	MoSCoW
	Functionele eisen	
1	De data uit de 3D-camera wordt uitgelezen.	M
2	Er wordt een 3D-kaart gemaakt op basis van de data uit de camera.	M
3	De testomgeving is reproduceerbaar.	M
4	Het uitvoeren van het testplan leidt tot kwantitatieve informatie over de nauwkeurigheid van afstands- en afmetingenbepaling van het systeem	M
5	De rolstoel kan op basis van de 3D-kaart obstakels vermijden.	C
6	De rolstoel gebruikt semantische informatie voor slimmere navigatie.	W
	Niet-functionele eisen	
7	De 3D-camera is een Intel RealSense ZR300.	S
8	Het systeem is gebaseerd op ROS-modules.	S

Tabel 1: Eisenlijst.

4.3. Bestaand systeem

Om te bepalen welke opties er zijn om de afstudeeropdracht uit te voeren, is het bestaande rolstoelsysteem geanalyseerd. Er zijn twee relevante documenten hierover. "Het grote boek van de rolstoel" is het kennisoverdrachtsdocument dat geschreven is door de projectgroep

die aan het eind van het schooljaar 2016/2017 aan de rolstoel heeft gewerkt. Het bevat uitleg over hoe de rolstoel opgestart en gebruikt moet worden. Vervolgens worden eerst de verschillende hardwaremodules beschreven, met daarbij welke software daarop draait. Daarna wordt uitgelegd welke packages er op de Intel NUC (de computer die de kern vormt van het rolstoelsysteem) gebruikt worden. Hierbij wordt van sommige packages gezegd dat ze niet meer in gebruik zijn, en ook worden er hier en daar vreemde "hacks" beschreven. Dit geeft geen overtuigend beeld van een stabiel systeem dat goed doordacht ontwikkeld is.

Het andere document is het eindverslag van de projectgroep uit 2016/2017. Dit verslag gaat uitgebreider in op verschillende onderdelen, maar heeft het voornamelijk over het gevolgde proces. Voor specifieke onderdelen kunnen delen van het verslag wellicht nuttig zijn als achtergrondinformatie, maar dit zal dan vooral over de gebruikte navigatiesystemen gaan.

Naast de documenten is ook de bestaande soft- en hardware bekeken. De bestaande code focust met name op het gebruik van de ultrasoonsensoren en de LIDAR[2], en het wordt niet duidelijk of de 3D-camera überhaupt gebruikt wordt, en zo ja op welke manier dat gebeurt. Bij de analyse van de hardware bleek dat de ultrasoonsensoren gedemonteerd waren, en alleen de Arduino's waarop deze waren aangesloten nog aanwezig waren. Ook was er bekabeling aanwezig die niet aangesloten was en/of waarvan niet duidelijk was wat de functie hiervan was.

Gezien de hierboven beschreven staat van de rolstoel is in overleg met de bedrijfsmentor besloten om alle overbodige bekabeling en onderdelen voorlopig te verwijderen, zodat alleen de onderdelen overbleven waarvan zeker was dat ze werkten en nodig waren. Ook is besloten om alle bestaande code te negeren, en de nieuwe code onafhankelijk hiervan te ontwikkelen. Dit heeft als nadeel dat het later wellicht extra werk kan zijn om de losse systemen samen te laten werken, maar het grote voordeel is dat als er in het oude systeem onderdelen niet werken, dat geen invloed heeft op de afstudeeropdracht of het nieuwe systeem.

4.4. Werking 3D-mapping

Om een driedimensionale kaart van een omgeving te maken op basis van een 3D-camera is een aantal stappen nodig. Het proces begint bij de 3D-camera, die op basis van twee infraroodcamera's een dieptekaart (*depth map*) maakt. Dit is een monochrome (zwart-wit) afbeelding waarbij de waarde van elke pixel staat voor de afstand tussen de camera en een waargenomen object. Een voorbeeld van een depth map is te vinden in Figuur 8. Deze dieptekaart moet vervolgens worden omgezet in een puntenwolk (*point cloud*), zoals te zien is in Figuur 12. Deze puntenwolk is een verzameling datapunten, waarvan voor elke punt een X-, Y-, en Z-coördinaat wordt opgeslagen. Omdat er niks bekend is over de positie van de camera, zijn deze coördinaten relatief aan een assenstelsel dat gebaseerd is op de camera. Dat wil zeggen dat de camera op een vast punt (bijvoorbeeld (0,0,0)) geplaatst wordt. Als gebruik gemaakt wordt van een RGBD-camera, kan voor elk punt ook de kleur worden opgeslagen.

[2]Samenvoeging van *Light* en *Radar*, radar die zichtbaar licht in plaats van radiogolven gebruikt. De gebruikte LIDAR werkt alleen in een enkel horizontaal vlak.

Om de puntenwolk om te zetten naar een assenstelsel dat niet met de camera mee beweegt, maar in plaats daarvan gebaseerd is op de omgeving, moet bepaald worden hoe de camera heeft bewogen. Dit kan op veel verschillende manieren, maar hoe dit exact gebeurt is nu niet relevant. Op basis van de beweging van de camera kunnen de coördinaten van de puntenwolk worden omgezet naar het omgevings-assenstelsel. Meerdere puntenwolken die hetzelfde assenstelsel gebruiken kunnen vrij eenvoudig worden samengevoegd. Door elke nieuwe puntenwolk samen te voegen met de eerste puntenwolk ontstaat uiteindelijk één grote puntenwolk van de volledige omgeving (of tenminste, dat deel van de omgeving dat de camera gezien heeft).

Een belangrijk nadeel van zo'n grote puntenwolk is dat deze enorm veel geheugen in beslag neemt. Stel bijvoorbeeld dat de resolutie van de dieptekaart 640x480 pixels is, en elk punt drie 32-bits waarden (X, Y, Z) en drie 8-bits waarden (R, G, B) bevat. Elk frame bestaat dan uit 307200 punten van 15 bytes aan de puntenwolk toegevoegd. Uitgaande van 30 frames per seconden levert dit een datastroom op van ca. 132 MB/s. Met deze hoeveelheid data kan bijvoorbeeld 4GB RAM in ca 30 seconden opgevuld zijn. In de praktijk zal dit waarschijnlijk lager zijn, maar deze berekening geeft wel aan dat het langdurig opslaan van de volledige puntenwolk geen optie is.

Een oplossing hiervoor is het omzetten van de data in een *voxel map*. Bij een voxel map wordt een ruimte opgedeeld in een aantal even grote kubussen (*voxels*), waarbij van elke kubus is vastgelegd of deze bezet of vrij is. In Figuur 17 is een voorbeeld te zien van een voxel map. In dit geval betekent dit dat er één voxel map gemaakt wordt, en er bij een nieuwe point cloud voor elk punt bepaald wordt in welke voxel deze zich bevindt. Deze voxel kan vervolgens als "bezet" worden gemerkt.

4.5. Werking ROS

Om de context van mijn afstudeeropdracht goed te begrijpen, is het belangrijk enige kennis over Robot Operating System (ROS) te hebben. ROS is een framework voor het ontwikkelen van software voor robots [3]. Een ROS-systeem bestaat uit verschillende nodes, die met elkaar communiceren met behulp van topics, messages en service calls. De door mij te ontwikkelen systemen zullen bestaan uit nodes en packages die op het ROS-framework draaien, en die met elkaar communiceren via ROS.

Elke node kan luisteren naar een of meerdere topics, en zal dan alle messages op die topic ontvangen. Een node kan ook een of meerdere topics aanmaken, en hier messages op publiceren. Communicatie tussen nodes kan ook op basis van service calls. Hierbij kan een node een request indienen bij een andere node, en deze kan hierop reageren met een reply. Voor de distributie van ROS-software wordt gebruik gemaakt van packages. Deze zijn vergelijkbaar met Ubuntu-packages, en kunnen op Ubuntu ook op dezelfde manier geïnstalleerd worden.

5. Ontwerp, implementatie en tests

In dit hoofdstuk wordt eerst beschrijven hoe er tot een globaal ontwerp gekomen is. Vervolgens wordt per module uitgelegd hoe deze ontworpen is, hoe dit ontwerp geïmplementeerd is, en hoe deze getest is. Tot slot wordt toegelicht hoe het volledige systeem getest is.

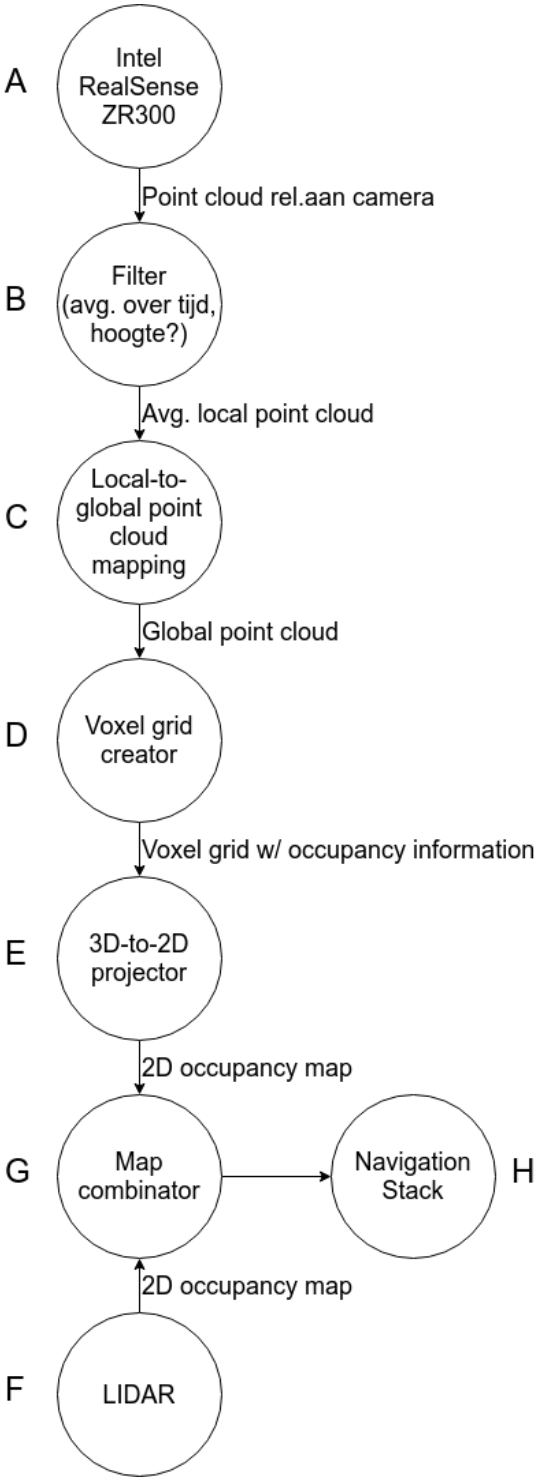
5.1. Globaal ontwerp

Om de rolstoel te laten navigeren op basis van data van de 3D-camera, moet deze data uiteindelijk bij de *Navigation Stack* uitkomen. De Navigation Stack is een verzameling ROS-nodes die een robot kan laten navigeren door een ruimte. Hoe deze Navigation Stack intern werkt valt buiten de scope van de afstudeeropdracht, omdat deze als gesloten systeem gebruikt kan worden. Wat wel belangrijk is om te weten is dat de Navigation Stack alleen werkt in twee dimensies. De Navigation Stack moet dan ook voorzien worden van een tweedimensionale kaart van de omgeving.

Om de volledige datastroom van de camera tot aan de Navigation Stack op te delen in definieerbare subsystemen is een globaal ontwerp gemaakt, waarin het systeem is opgedeeld in acht modules, A t/m H. Elk van deze modules is in principe een ROS-package, die kan bestaan uit een of meerdere ROS-nodes. Een schematische weergave van dit ontwerp is te vinden in Figuur 4.

Op basis van deze indeling kan per onderdeel besloten worden hoe deze geïmplementeerd gaat worden. Voor elke module moet namelijk opnieuw worden afgewogen of er bestaande systemen zijn die gebruikt kunnen worden, welke algoritmen er gebruikt moeten worden, en of er eventueel eigen code geschreven moet worden.

De eerste module is module A. Deze bestaat uit de cameradriver en de interpretatie van de data uit de camera. Hieruit komt een point cloud die relatief is aan de camera. module B kan deze data vervolgens filteren, bijvoorbeeld door een gemiddelde van meerdere frames te nemen. De taak van module C is daarna om elk frame de



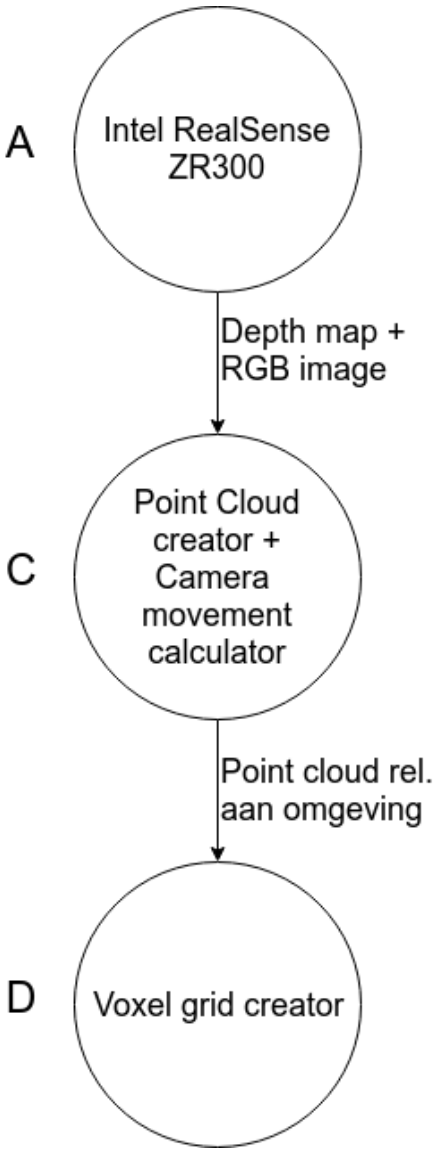
Figuur 4: Schematisch overzicht globaal ontwerp

nieuwe point cloud toe te voegen aan een bestaande point cloud, om hier uiteindelijk een globale point cloud van te maken waarvan de posities niet meer relatief zijn aan de positie van de camera, maar relatief aan de positie van de omgeving. In module D wordt van deze point cloud een voxel grid gemaakt, die in module E naar twee dimensies wordt geprojecteerd. Hieruit komt een tweedimensionale kaart.

Module F is geen echte module, maar stelt in dit ontwerp de bestaande systemen voor die van de LIDAR-data een tweedimensionale kaart maken. Deze kaart wordt in module G gecombineerd met de kaart uit module E om een kaart te maken die betrouwbaarder is dan de twee kaarten uit modules E en F. Deze kaart gaat vervolgens naar module H, wat ook geen echte module is maar in dit ontwerp de bestaande Navigation Stack voorstelt.

5.2. Gewijzigd ontwerp

Zoals in §3.5. is beschreven, is de opdracht tijdens het uitvoeren significant veranderd. Hierdoor heeft het ontwerp ook mee moeten veranderen. Ook is er ontdekt dat sommige functionaliteit anders moest worden geïmplementeerd, of beter bij een andere module ondergebracht kon worden. Een schematische weergave van de definitieve versie van het globale ontwerp (vergelijkbaar met het schema in Figuur 4) zoals dat uiteindelijk is geïmplementeerd is te vinden in Figuur 5.



Figuur 5: Schematisch overzicht definitief ontwerp

In deze schematische weergave is te zien dat modules E t/m H weggelaten zijn. Er is namelijk besloten dat het belangrijker was om tot en met module D te ontwikkelen en dit uitgebreid te testen. Ook is module B weggelaten. De filtering die deze module zou bieden bleek namelijk niet nodig te zijn, omdat de data van module A rechtstreeks naar module C kan, en module C en D voldoende capaciteit hebben om deze datastroom realtime te verwerken.

Wat ook veranderd is, is dat de data vanuit module A geen point cloud meer is, maar in plaats daarvan een depth map en een RGB afbeelding. Dit is omdat de omzetting van een depth map in een point cloud al ingebouwd zat in de software die voor module C gekozen was.

Module C is ook veranderd. In plaats van een grote global point cloud maakt deze module nu

alleen een point cloud uit de depth map, en wordt deze omgezet in een point cloud die relatief is in de omgeving. Meer hierover wordt beschreven verderop in dit hoofdstuk, in §5.4.

Module D is nog steeds de Voxel grid creator, maar deze krijgt nu losse point clouds relatief aan de omgeving binnen, in plaats van een volledige global point cloud.

5.3. Module A: Van camera naar ROS

5.3.1. Keuzeproces en ontwerp

Module A moet er voor zorgen dat de data uit de camera wordt uitgelezen, en dat deze wordt omgezet in een formaat waar ROS mee kan werken. Dit systeem moet aan een aantal eisen voldoen:

- Het systeem mag niet te veel overhead hebben. De data moet van de camera naar ROS omgezet worden zonder dat dit overbodig veel rekenkracht of werkgeheugen kost. Er zijn hier geen vaste cijfers aan te verbinden, maar het belangrijkste is dat er voldoende rekenkracht en werkgeheugen overblijft voor de overige modules, en voor de andere taken die de rolstoel tegelijkertijd moet uitvoeren.
- Het systeem moet redelijk eenvoudig en snel geïmplementeerd kunnen worden. Er is namelijk geen tijd om wekenlang met deze module bezig te zijn. Volgens de planning moet het systeem binnen zes dagen volledig ontworpen en geïmplementeerd kunnen worden.
- Het systeem moet stabiel en betrouwbaar werken. Tijdens normaal gebruik mag het systeem niet crashen, en moet een constante datastroom geleverd worden.
- De module moet alle functionaliteit bieden die nodig is in het grotere systeem en waar de camera over beschikt, of anders moet de module eenvoudig uitbreidbaar zijn om deze functionaliteit toe te voegen.

Dit systeem moet bestaan uit twee delen: een USB-driver die de communicatie met de camera regelt, en een ROS-node die gebruik maakt van de driver om de data te verkrijgen, deze omzet in een ROS-compatibel formaat en publiceert. Een schematisch overzicht hiervan is te zien in Figuur 6. Hieruit zijn drie mogelijkheden af te leiden om deze module te implementeren. Een overzicht van deze mogelijkheden is weergegeven in Tabel 2.

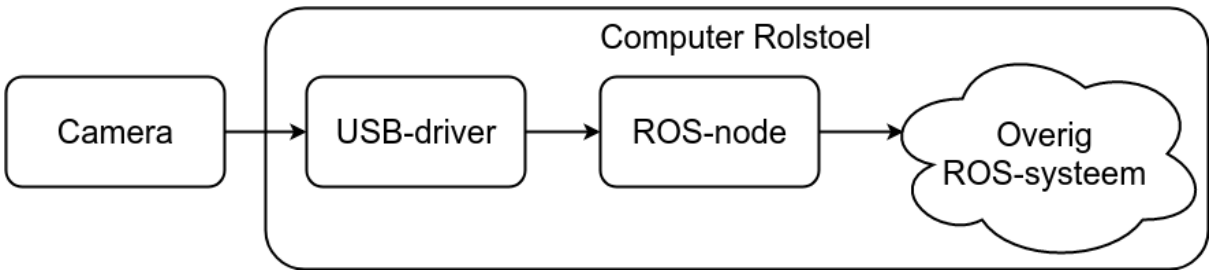
De eerste mogelijkheid is het volledig zelf schrijven van beide delen. Enerzijds geeft dit de vrijheid om precies die functionaliteit te implementeren die nodig is, zonder extra overhead. Dit zorgt ervoor dat de code heel efficiënt kan zijn. Een nadeel is echter dat het schrijven van een driver geen eenvoudige opgave is, en fouten in drivers kunnen gevolgen hebben voor de stabiliteit van het besturingssysteem. Ook zal voor elk stukje extra benodigde functionaliteit nieuwe code geschreven moeten worden, wat het ingewikkeld maakt om het systeem uit te breiden.

De volgende mogelijkheid is om gebruik te maken van de bestaande driver, librealsense. Deze driver biedt een API die gebruikt kan worden om zelf een ROS-node te schrijven. Hiermee kan nog steeds exact dat gedeelte van mogelijke functionaliteit geïmplementeerd worden die nodig is voor de rolstoel, met slechts een kleine hoeveelheid overhead van de API. Het schrijven van een eigen ROS-node is niet triviaal, maar een stuk makkelijker dan het schrijven van een driver. Het gevolg van het gebruiken van deze API is ook dat het risico

voor de stabiliteit van het besturingssysteem weggenomen wordt. Wel moet er voor het toevoegen van nieuwe functionaliteit nog steeds nieuwe code geschreven worden.

De laatste optie is om gebruik te maken van door Intel ontwikkelde ROS-node, realsense_camera. Deze ROS-node maakt op zijn beurt weer gebruik van de librealsense-driver. Deze ROS-node zal wat overhead opleveren, maar omdat er geen ingewikkelde berekeningen uitgevoerd hoeven worden, zal deze overhead redelijk weinig zijn. Omdat er geen eigen code geschreven hoeft te worden, is de implementatie waarschijnlijk vrij eenvoudig. De stabiliteit zal ook vrij hoog zijn, omdat gebruik gemaakt wordt van code die al langere tijd in productieomgevingen gebruikt is. Een nadeel van het gebruik van de bestaande ROS-node is dat deze vrij lastig uitbreidbaar is. Dit is echter niet erg, omdat alle functionaliteit van de camera beschikbaar gemaakt wordt door de ROS-node. Deze hoeft dus überhaupt niet uitgebreid te worden.

Al met al lijkt deze laatste optie de beste te zijn. In een meerjarenproject met een groot team kan het wellicht de moeite waard zijn om een eigen driver en ROS-node te schrijven, maar met mijn kennis en de tijd die beschikbaar is wegen de voordelen van een eigen driver en ROS-node niet op tegen de enorme tijdsinvestering die nodig zou zijn om het wiel opnieuw uit te vinden. Er is dus gekozen om gebruik te maken van de realsense_camera-ROS-node en librealsense.



Figuur 6: Schematisch overzicht module A

	Eigen driver + ROS-node	**Librealsense + eigen ROS-node**	**Librealsense + realsense_camera**
Geringe overhead	++	+	+
Eenvoudig implementeren	--	+/-	+
Stabiliteit	-	+	++
Uitbreidbaarheid	-	-	n.v.t.
Conclusie	**-**	**+/-**	**+**

Tabel 2: overzicht mogelijkheden module A. De conclusie is dat de optie librealsense + realsense_camera de beste optie is.

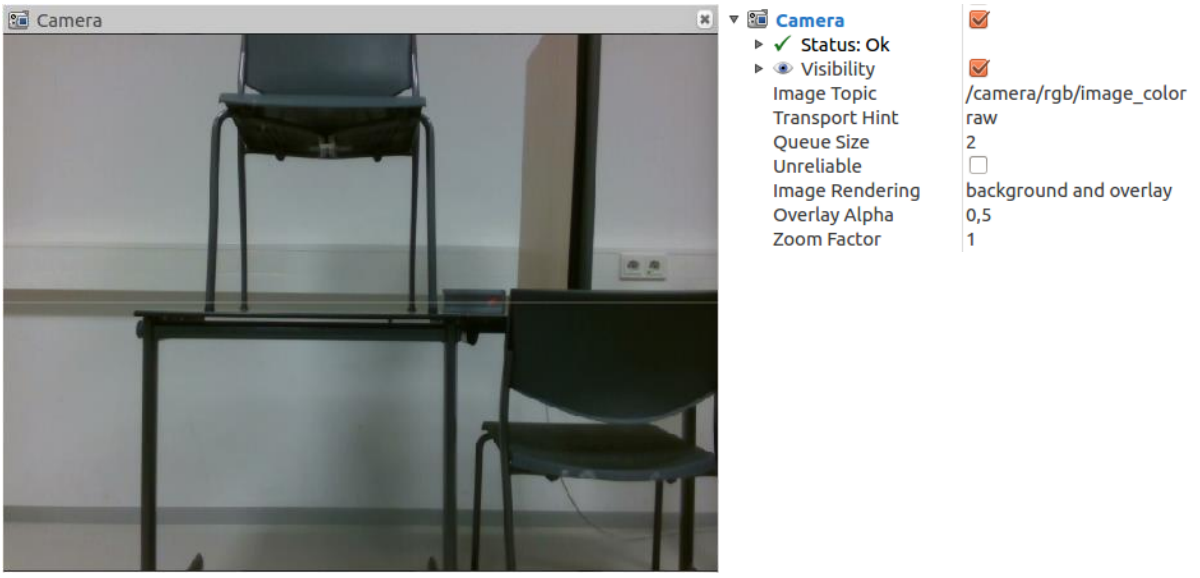
5.3.2. Implementatie

Hoewel het werkend krijgen van realsense_camera en librealsense relatief eenvoudig zou moeten zijn, had dit nog wel wat voeten in aarde. Er was namelijk een aantal problemen waar tegenaan gelopen werd. Ten eerste wordt de ZR300 niet meer ondersteund door Intel, en moet er dus een oudere versie van librealsense en realsense_camera gebruikt worden, namelijk librealsense 1.12.1 en realsense_camera 1.8.1. Dit levert echter het probleem op dat librealsense afhankelijk is van een kernelmodule, uvcvideo, die gepatcht (aangepast) moet worden om librealsense te laten werken. Deze aanpassingen zijn echter alleen gemaakt voor Ubuntu 16.04 of ouder. Ubuntu 16.04 maakt gebruik van versie 4.4 van de Linuxkernel, en ik gebruik Ubuntu 18.04, dat gebaseerd is op kernel 4.15.

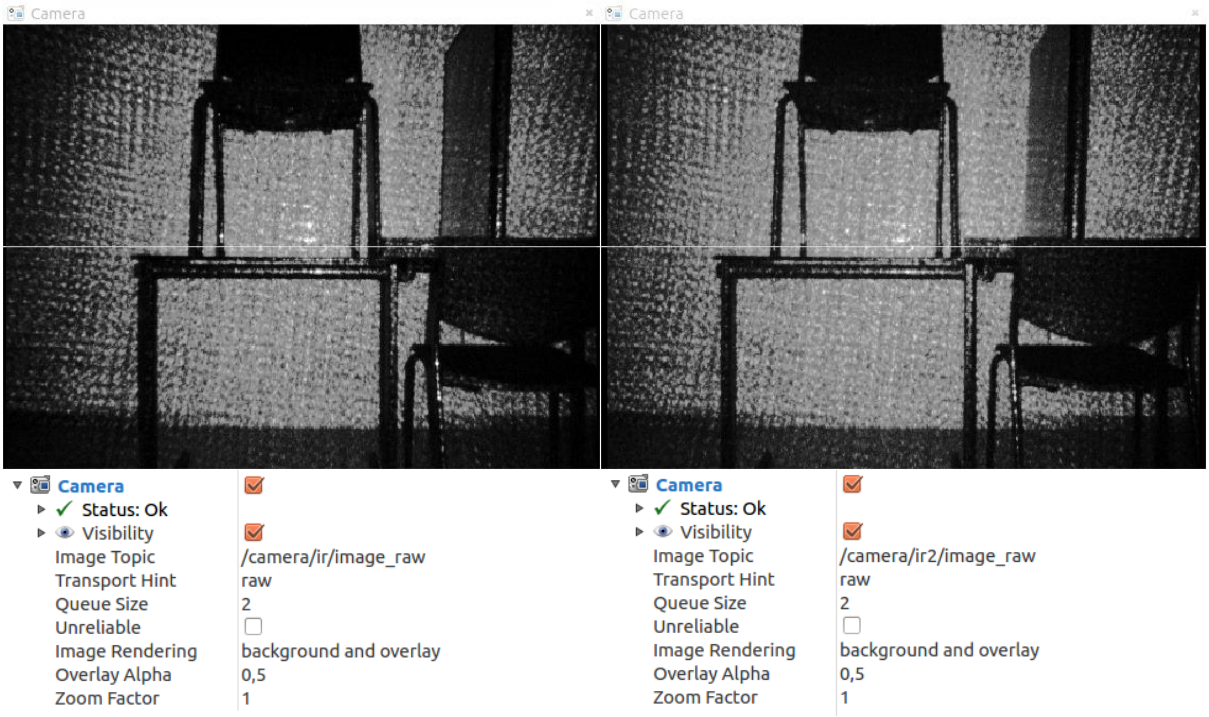
Uiteindelijk is het wel gelukt om librealsense en realsense_camera werkend te krijgen, door Ubuntu 16.04 te installeren. Om de data uit de camera te visualiseren, is ook RViz geïnstalleerd, de standaard visualisatieapplicatie voor ROS.

In Figuur 6 is te zien hoe de data van de RGB-camera eruit ziet in RViz. Links is het camerabeeld te zien, en rechts is te zien wat RViz weet over deze data. Het belangrijkste hier is de “Image Topic”, het datakanaal waarop RViz de data ontvangt. In dit geval is dit /camera/rgb/image_color. Figuur 7 is vergelijkbaar met Figuur 6, in dit geval zijn de beelden van de beide infraroodcamera’s zichtbaar, links het linkerbeeld en rechts het rechterbeeld. Hieronder is opnieuw de metadata te zien, waarbij de Image Topics /camera/ir/image_raw en /camera/ir2/image_raw zijn. Uit deze beide infraroodbeelden genereert de camera een dieptekaart, die in Figuur 8 te zien is. Hier is te zien dat deze data beschikbaar is op /camera/depth/image.

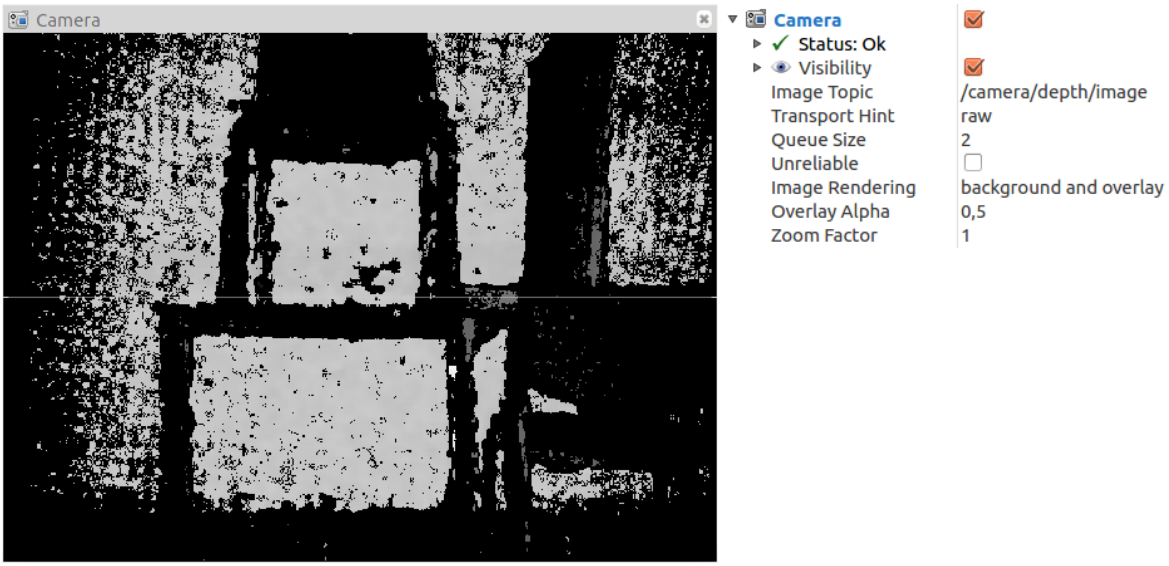
Om te zien hoe data tussen verschillende ROS-nodes kan bewegen, kan gebruik gemaakt worden van rqt_graph. Dit programma, dat standaard met ROS wordt meegeleverd, analyseert alle nodes en topics, en maakt hier een directionele graaf van. In Figuur 9 is te zien hoe deze graaf eruit ziet als alleen de camera en RViz in gebruik zijn. Hierop is te zien dat er van de node /camera/camera_nodelet_manager data via veel verschillende topics naar de node /rviz wordt gestuurd. Hier zijn onder andere de ir/image_raw, ir2/image_raw, rgb/image_color en depth/image topics te zien die hiervoor zijn genoemd. Wat ook opvalt is dat voor elk van deze topic een bijbehorend camera_info topic beschikbaar is. Deze topic bevat informatie over bijvoorbeeld het aantal pixels, de framerate, en het dataformaat van de beelden.



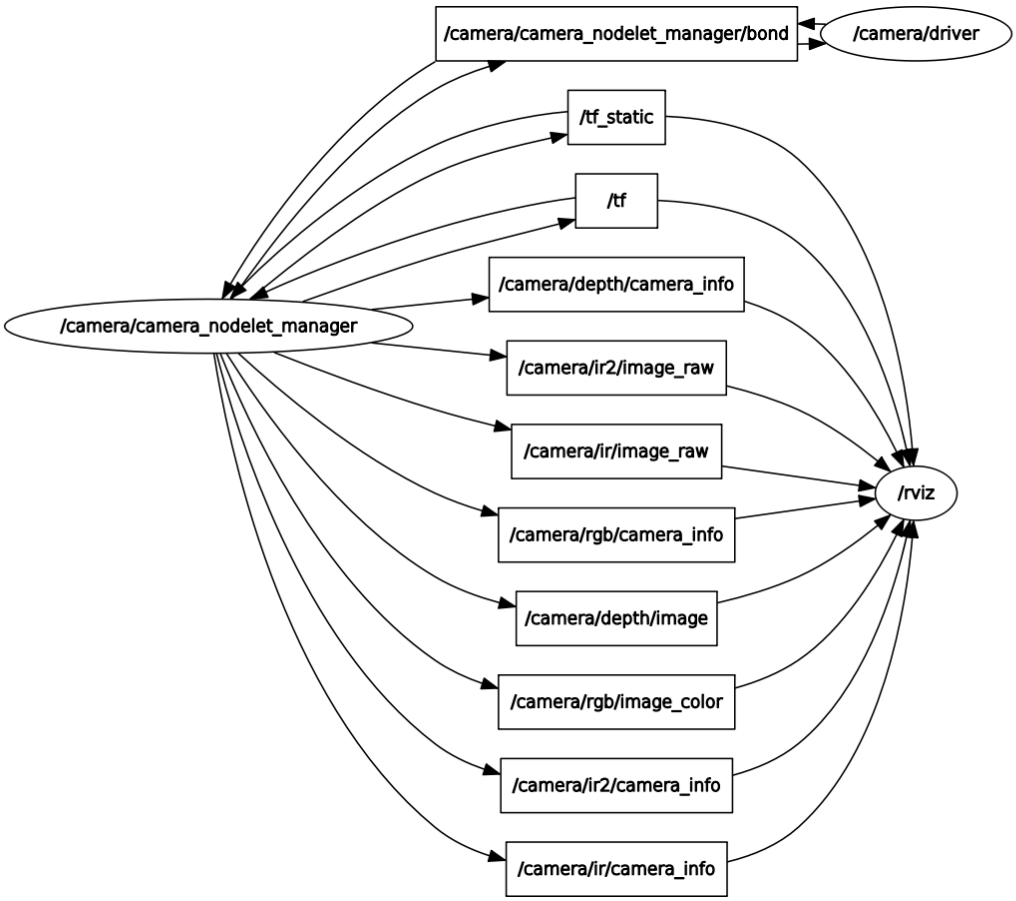
Figuur 6: RGB-beeld in RViz



Figuur 7: Infraroodbeelden links en rechts in RViz. De helderheid en het contrast van deze afbeeldingen zijn aangepast om details beter zichtbaar te maken.



Figuur 8: dieptekaart in RViz. De helderheid en het contrast van deze afbeelding zijn aangepast om details beter zichtbaar te maken.



Figuur 9: rqt_graph bij gebruik camera en RViz

5.3.3. Testen

Voor het testen van deze module is eerst een testplan gemaakt. Dit testplan is te vinden in Bijlage C. Hierbij wordt de camera voor een blinde muur geplaatst, en wordt bij verschillende afstanden tussen de camera en de muur bepaalt welke afstand tot de muur gemeten wordt door de camera. Door vervolgens de gemeten waarden met de werkelijke afstanden te vergelijken, kan bepaald worden hoe nauwkeurig en betrouwbaar de camera is.

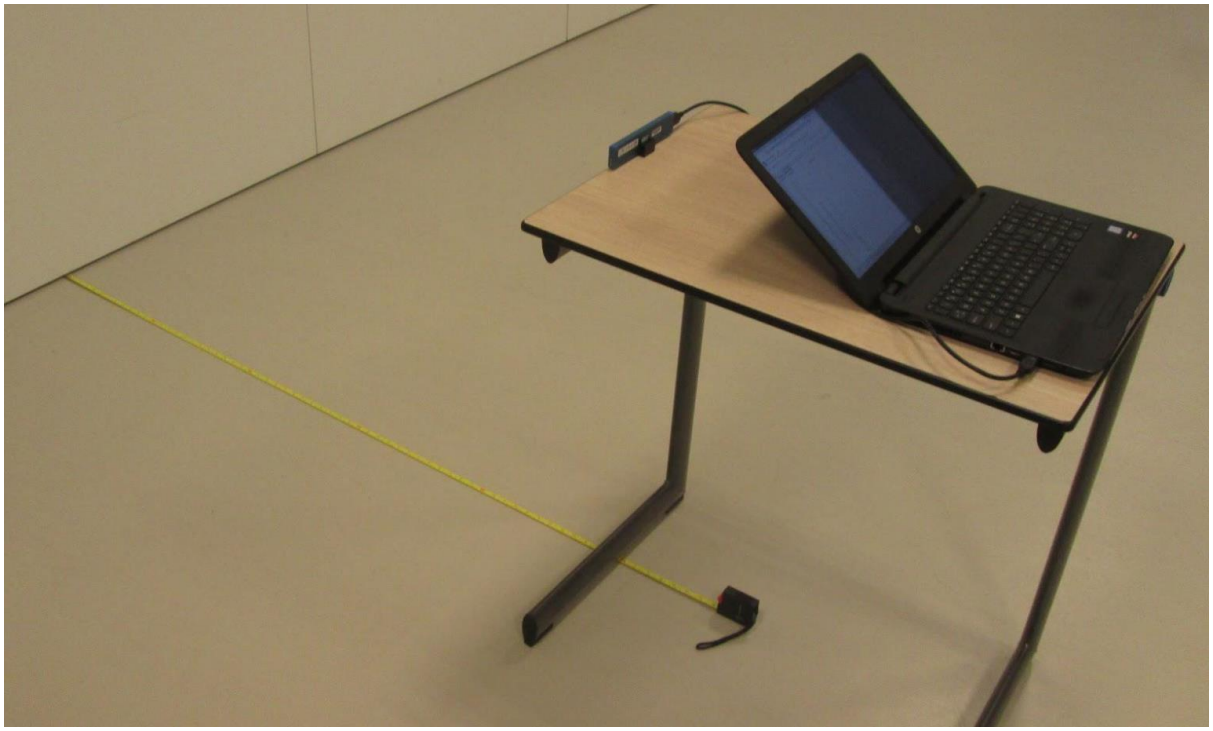
Om eenvoudig de door de camera gegenereerde waarde af te lezen, is een python-script geschreven, dat te vinden is in Bijlage D.2. Dit script start een ROS-node op dat subscribed aan de topic /camera/depth/image_raw. Vervolgens wordt voor elk ontvangen bericht de callback()-functie aangeroepen, waarin van de ontvangen afbeelding uitgerekend wordt wat de middelste pixel is. Vervolgens wordt de waarde van deze middelste pixel weergegeven met behulp van de rospy.loginfo()-functie.

Vervolgens is een blinde muur gevonden, en is de testopstelling opgebouwd zoals deze in Figuur 10 te zien is. De camera is op verschillende afstanden van de muur geplaatst, en er is opgeschreven welke waarden er vervolgens uit de camera kwamen. Deze resultaten zijn te vinden in Bijlage E.1. Hier is redelijk wat informatie over de camera af te leiden.

Ten eerste kon er geen afstand korter dan ca. 610 mm gemeten worden. Als de camera dichter bij de muur geplaatst wordt, zal de camera de waarde "0" geven. Ook is te zien dat de gemeten waarde altijd hoger is dan de werkelijke waarde. Op een afstand onder de 2 meter ligt deze afwijking bijna altijd onder de 2 centimeter, maar met het toenemen van de afstand loopt ook de afwijking snel op, met op 5 meter afstand zelfs een afwijking van 56 centimeter. Omdat de afwijking zo groot werd, is besloten niet verder te meten op afstanden groter dan 5 meter.

Vervolgens is nog een test uitgevoerd, om te bepalen hoe waardevast de data uit de camera is. Hiervoor is de camera op vier verschillende afstanden van de blinde muur gezet, en zijn voor elk van deze afstanden vijf metingen gedaan. De resultaten hiervan zijn te vinden in Bijlage E.2.

Ook uit deze resultaten kan een aantal conclusies getrokken worden. Ten eerste laat deze data dezelfde trend zien als de vorige test, namelijk een consequente afwijking naar boven. Wat wel opvalt is dat bij een afstand van een meter de gemiddelde afwijking slechts 3,2 millimeter is. Ook is te zien dat, hoewel de gemiddelde afwijkingen op 61 centimeter en 2 meter vrij dicht bij elkaar liggen, de standaarddeviatie op 2 meter afstand veel groter is. Bij een grotere afstand tussen de camera en de muur is dus niet alleen de afwijking groter (zoals in de vorige test was vastgesteld), maar is ook de betrouwbaarheid van deze meting veel kleiner.



Figuur 10: Foto testopstelling dieptemeting

5.4. Module C: Van camerabeelden naar point cloud

5.4.1. Keuzeproces en ontwerp

In het begin van het ontwerpen van deze module was het niet precies duidelijk hoe de module moest gaan werken. Alles wat bekend was, was dat er op de een of andere manier gebruik gemaakt moest worden van de data uit module A om een volledige point cloud van de omgeving te maken. Na een eerste onderzoek zijn er drie opties gevonden die nader bekeken zijn: Google Cartographer, ORB-SLAM2 en OctoMap. Na het analyseren van deze opties is bij verder onderzoek nog een vierde optie gevonden: RGBDSLAMv2. Een overzicht van deze opties is te vinden in Tabel 3.

Google Cartographer viel al vrij snel af, omdat de 3D-functionaliteiten hiervan nog lang niet goed genoeg werken om ze te gebruiken. Cartographer is namelijk oorspronkelijk ontwikkeld voor 2D-mapping, en pas recent is er begonnen met het toevoegen van 3D-functionaliteit bovenop het 2D-systeem.

De volgende optie was ORB-SLAM2 [8]. ORB-SLAM2 maakt gebruik van beeldherkenningstechnieken om *keypoints* te vinden, en bouwt op basis hiervan een *sparsely populated*[3] kaart van de omgeving op. Hoewel de resultaten hiervan veelbelovend zijn, valt ORB-SLAM2 toch af, omdat de resulterende data niet compatibel is met de verder geplande modules, ORB-SLAM2 levert namelijk geen point cloud.

[3] *Sparsely populated* wil zeggen dat niet van elk punt in de ruimte informatie wordt opgeslagen, maar dat alleen van belangrijke punten wordt opgeslagen wat zich daar bevindt.

Tot slot was er de optie om OctoMap te gebruiken [9]. OctoMap is een framework voor het verwerken van *point cloud*-data tot een *voxel map*, en voor het efficiënt opslaan van deze voxel map. Ook kan er in de voxel map informatie worden opgeslagen over de onzekerheid van de data. Wat OctoMap echter niet kan, is het bepalen welke beweging de camera gemaakt heeft, om een nieuwe point cloud van de camera op de juiste positie in de bestaande voxel map toe te voegen.

De opslagalgoritmes van OctoMap leken echter wel de beste optie, omdat deze erg efficiënt een voxel map kunnen opslaan, en net zo efficiënt deze data weer kunnen ophalen en verwerken. Er is dus verder gezocht naar systemen die wel de beweging van de camera kunnen bepalen, maar die ook gebruik maken van het OctoMap-framework. De enige goede optie die hier gevonden is, is RGBDSLAMv2 [10]. RGBDSLAMv2 maakt gebruik van zowel de RGB-data als de diepte-informatie van de camera om te bepalen hoe de camera bewogen heeft. Ook worden de recentste camerabeelden vergeleken met oudere beelden, om te bepalen of de camera wellicht op een plek is waar deze eerder is geweest. Hiermee kunnen cumulatieve fouten worden gevonden en verbeterd. Doordat continu bekend is hoe de camera heeft bewogen, kan er een globaal consistente point cloud worden gemaakt, die met behulp van OctoMap wordt omgezet in een voxel map. Er is ook nog verder gezocht naar alternatieven voor RGBDSLAMv2, maar deze zijn niet gevonden. Er is dus gekozen om RGBDSLAMv2 te installeren.

	Cartographer	**ORB-SLAM2**	**OctoMap**	**RGBDSLAMv2**
3D	Nee	Ja	Ja	Ja
Camerabeweging	Ja	Ja	Nee	Ja
Point cloud	Nee	Nee	Ja	Ja
Voxel map	Nee	Nee	Ja	Ja
Conclusie	**Nee**	**Nee**	**Nee**	**Ja**

Tabel 3: overzicht mogelijkheden module C. De conclusie is dat RGBDSLAMv2 de beste optie is.

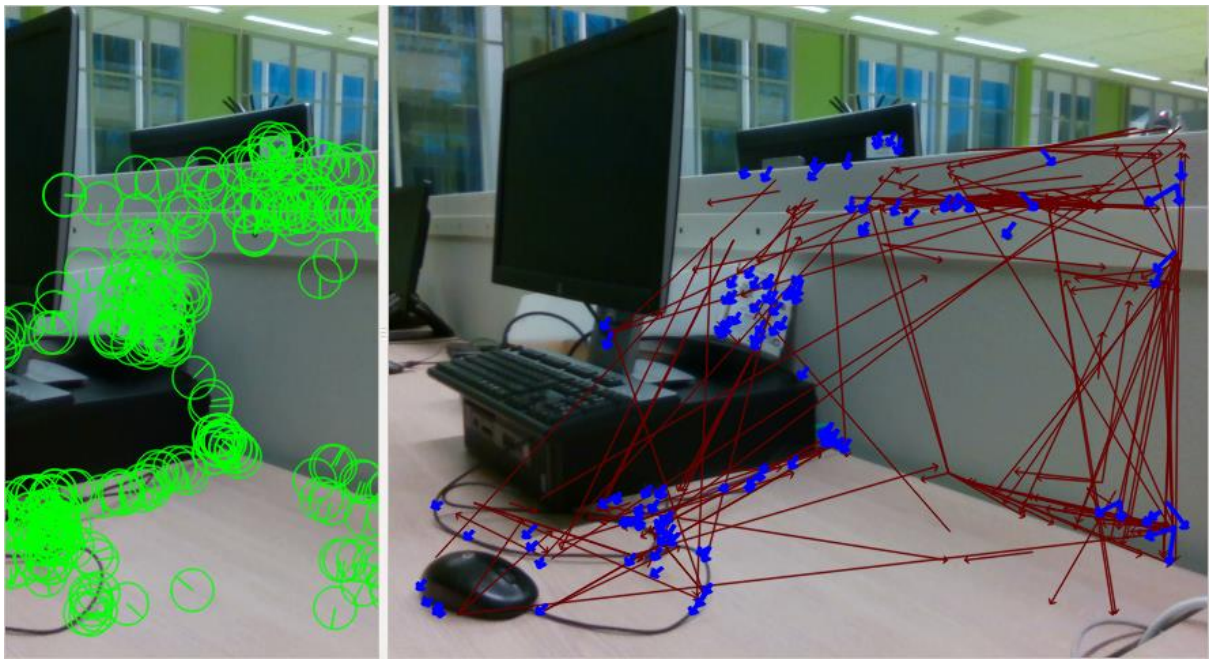
5.4.2. Implementatie

De installatie van RGBDSLAMv2 ging niet zonder slag of stoot. De installatie zelf leek eerst te lukken, maar de `rgbdslam` executable crashte bijna meteen. Ook als er geen gebruik gemaakt werd van de camera, maar in plaats daarvan van de test-dataset die bij RGBDSLAMv2 wordt geleverd, crashte `rgbdslam` met dezelfde foutcode. Na analyse met `gdb`[4] bleek dat dit om een *segmentation fault* ging. De oplossing hiervoor bleek te bestaan uit het zelf compileren van de `g2o`-library, de `PCL`-library, en RGBDSLAMv2 [11]. Toen dit gedaan was, werkte RGBDSLAMv2 goed, en werd er een duidelijke point cloud zichtbaar van de omgeving.

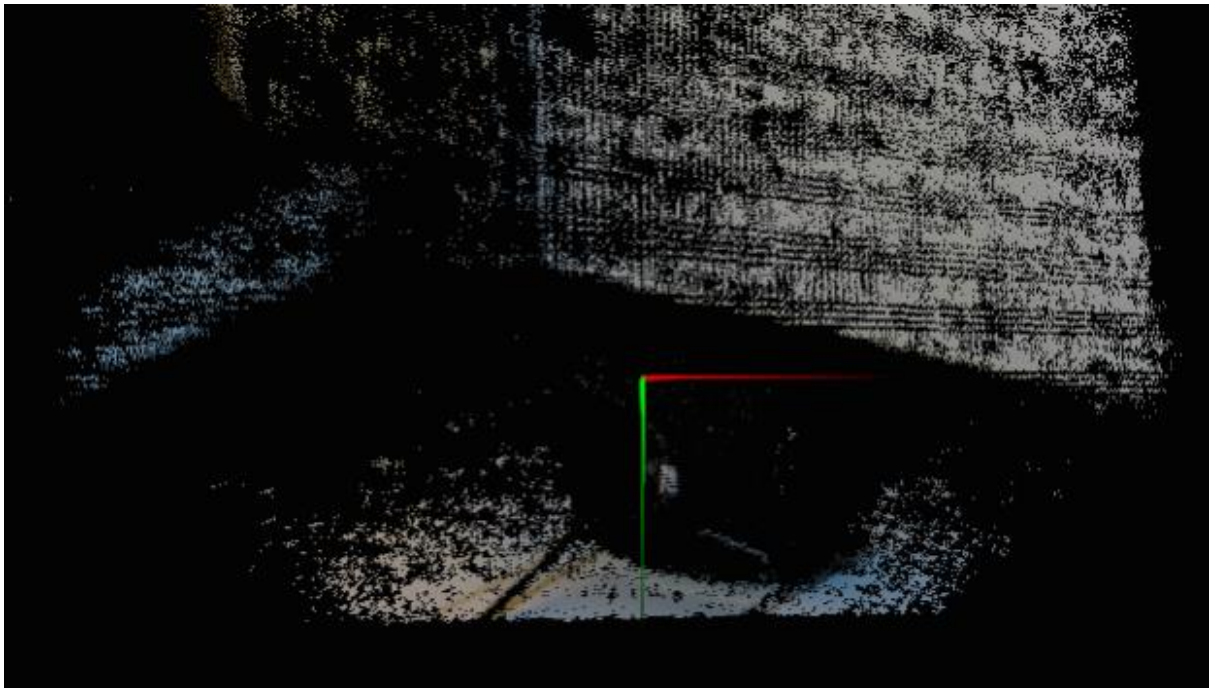
[4] `Gdb` is de GNU Debugger, een programma dat gebruikt kan worden om andere programma's te analyseren terwijl ze draaien.

RGBDSLAM heeft een eigen GUI waarin de door RGBDSLAM gegenereerde data zichtbaar is. In Figuur 11 is te zien welke hoeken en andere opvallende punten RGBDSLAM detecteert in een RGB-beeld. Deze informatie wordt gebruikt om te bepalen hoe de camera bewogen heeft. Op basis van de RGB-data en de dieptekaart wordt vervolgens een point cloud gemaakt. In Figuur 12 is te zien hoe deze point cloud eruit ziet als de camera nauwelijks bewogen heeft. Het snijpunt van de groene en rode lijn is de positie van de camera, de groene lijn wijst vanaf de camera loodrecht naar beneden, de rode lijn wijst pal naar rechts. Als de camera bewogen wordt, wordt een beeld zoals in Figuur 13 zichtbaar. In dit geval is de camera van rechts naar links bewogen, en is voor elke positie van de camera een groene en rode lijn getekend.

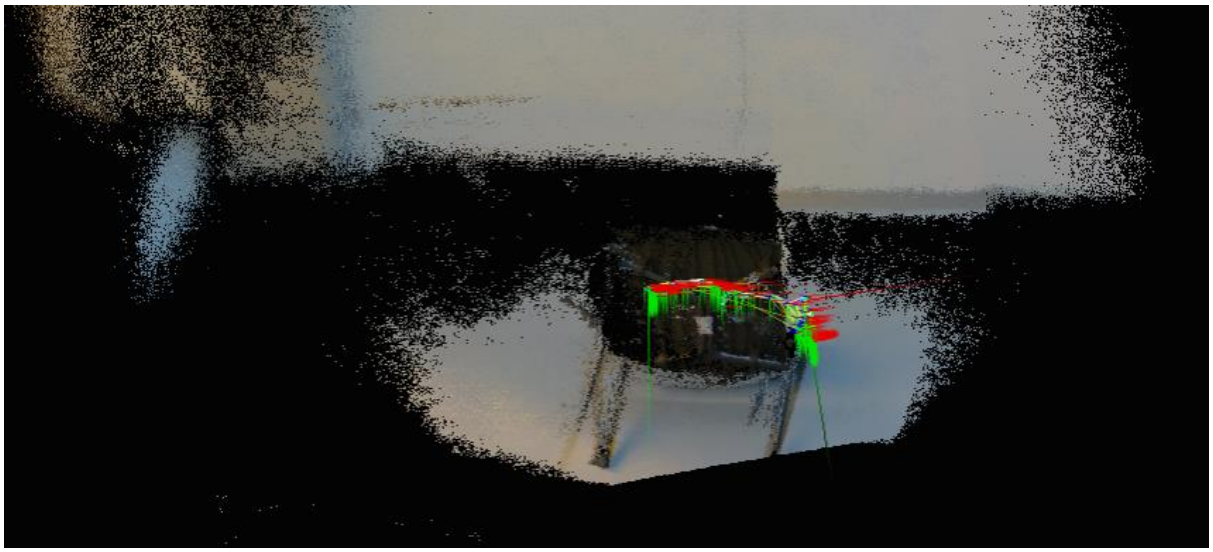
Net zoals bij module A wordt er hier weer gebruik gemaakt van rqt_graph om de datastromen tussen de camera en RGBDSLAM weer te geven. In Figuur 14 is te zien hoe in dit geval de graaf eruit ziet. Wat opvalt is dat de node /rgbdslam maar gesubscribed is aan een beperkt aantal topics. /rgbdslam maakt gebruik van /camera/rgb/image_color en de bijbehorende /camera/rgb/camera_info voor de beelden van de RGB-camera, op basis waarvan opvallende punten gedetecteerd worden. De enige andere belangrijke topic waar /rgbdslam naar luistert is /camera/depth_registered/sw_registered/image_rect_raw. Dit is in feite dezelfde data als de bij module A genoemde /camera/depth/image, maar dan is deze door de cameradriver getransformeerd om in een rechthoekig grid te passen.



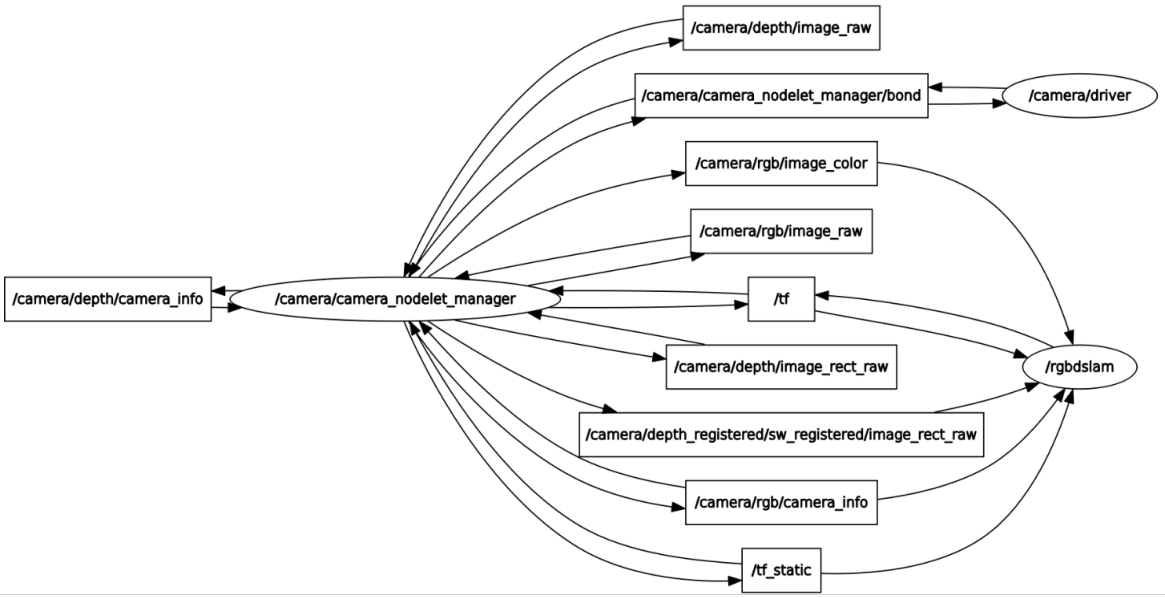
Figuur 11: Feature recognition door RGBDSLAM



Figuur 12: Point cloud zonder camerabeweging



Figuur 13: Point cloud met camerabeweging



Figuur 14: rqt_graph bij gebruik camera en RGBDSLAM

5.4.3. Probleem met geheugengebruik

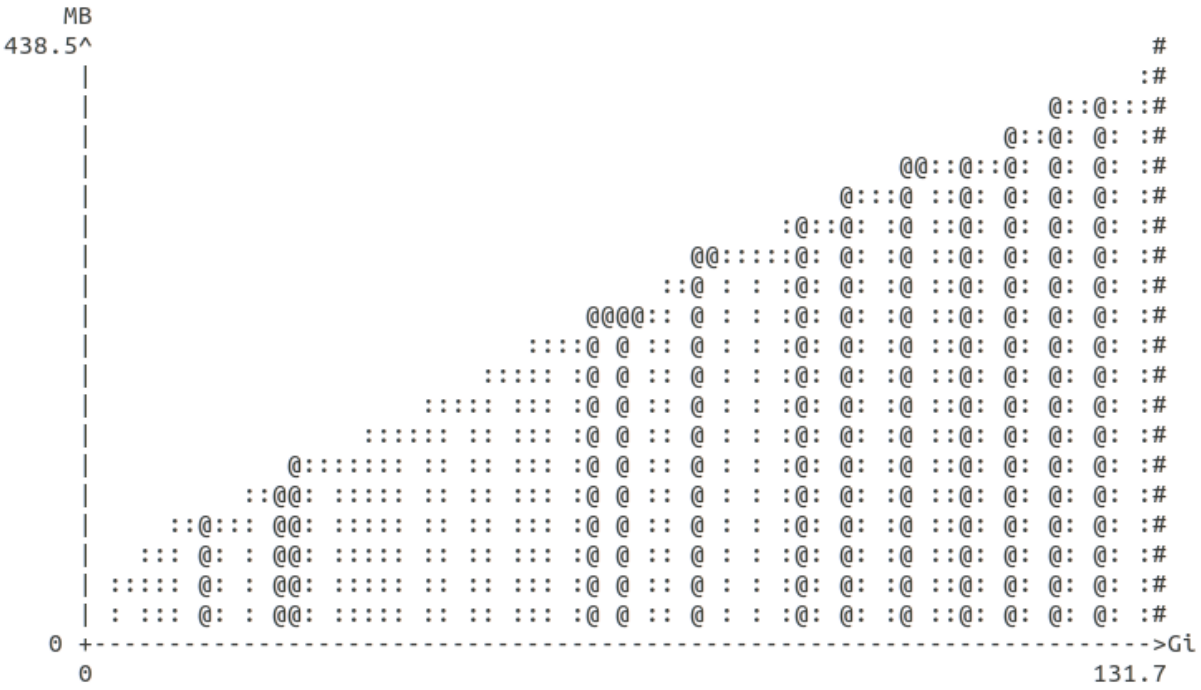
Tijdens het gebruik van RGBDSLAM bleek dat deze na enkele minuten gebruik de volledige computer liet crashen. Met behulp van de Linux-applicatie `top` was te zien dat de `rgbdslam`-executable langzaam steeds meer geheugen ging gebruiken, totdat het volledige werkgeheugen van de computer vol was. Om te bepalen wat hier de oorzaak van was, is een profiler geïnstalleerd, Valgrind [12]. Deze profiler beschikt over een aantal verschillende tools om programma's op verschillende manieren te analyseren. De tool die in dit geval nodig was heet `massif`. Na het uitvoeren van RGBDSLAM in Valgrind met als tool Massif is een rapport gegenereerd van het geheugengebruik van RGBDSLAM. Dit rapport is niet goed door mensen te lezen, maar kan met behulp van de met Valgrind meegeleverde tool `ms_print` omgezet worden in een leesbaarder formaat. In dit rapport is eerst een grafiek te zien van het geheugengebruik tijdens de looptijd van het programma. Een voorbeeld van deze grafiek is te vinden in Figuur 15, waarin te zien is dat het geheugengebruik vrij snel is opgelopen tot 438,5MB. Vervolgens wordt in het rapport van een aantal momenten een uitgebreid beeld gegeven van welke functies geheugen hebben gealloceerd.

Hier werd duidelijk dat een groot deel (per keer verschillend, maar meestal ca. 50 tot 90 procent) van het geheugen door een bepaalde functie werd gebruikt. In het rapport werd wel de functienaam genoemd, namelijk `createXYZRGBPointCloud()`, maar er was niet te zien hoe deze functie in de code wordt gebruikt, of vanaf waar deze precies wordt aangeroepen. Om hierachter te komen moet de code namelijk gecompileerd zijn met debugging-informatie, wat niet gebeurd was omdat dit eerder niet nodig was. Door het toevoegen van de `-g`-optie aan de lijst met compileropties in de configuratie van RGBDSLAM, en het opnieuw compileren van RGBDSLAM, was deze debugging-informatie wel beschikbaar.

Met de debugging-informatie beschikbaar werd RGBDSLAM opnieuw gedraaid, en werd een nieuw Massif-rapport gegenereerd. In dit nieuwe rapport was te vinden dat `createXYZRGBPointCloud()` wordt aangeroepen vanaf regel 131 van het bestand "node.cpp",

dat onderdeel is van de RGBDSLAM-broncode. In Code 1 is deze regel en de relevante directe omgeving te zien. In deze code is te zien dat createXYZRGBPointCloud() wordt aangeroepen als aan een aantal voorwaarden wordt voldaan. In de code wordt het object ps gebruikt, dit is de ParameterServer die de informatie uit de configuratiebestanden van RGBDSLAM interpreteert. Als dus de juiste configuratiewaarden worden ingesteld, wordt createXYZRGBPointCloud() niet aangeroepen, en zal RGBDSLAM minder geheugen gebruiken.

Nadat deze code was begrepen, zijn de configuratiewaarden van RGBDSLAM aangepast. store_pointclouds is op false gezet en emm__skip_step is op 0 gezet. In de code voor de ParameterServer bleek dat use_icp standaard al op false stond. Tot slot is glwidget_without_clouds op true gezet. Na al deze aanpassingen is RGBDSLAM opnieuw gedraaid, en is een nieuw Massif-rapport gegenereerd. Zoals te zien in Figuur 16 was het geheugengebruik nu slechts 61,29MB. Ook bleef het geheugengebruik nu vrij constant, waar dit eerder (zoals in Figuur 15 te zien is) continu bleef stijgen.



Figuur 15: Grafiek van geheugengebruik van RGBDSLAM. Op de horizontale as staat de tijdsduur (in aantal instructies), op de verticale as staat het geheugengebruik.
Massif meet de verlopen tijd in het aantal uitgevoerde processorinstructies, in dit geval zijn dat 131,7 miljard instructies. Elke kolom in de grafiek stelt een snapshot voor, waarin Massif het geheugengebruik heeft gemeten. Kolommen bestaande uit een @ of een # zijn gedetailleerde snapshots, waarin ook is onderverdeeld welke functies het geheugen hebben gebruikt. De kolom bestaande uit # stelt het snapshot voor met het grootste geheugengebruik.

```
126  if(ps->get<bool>("store_pointclouds") ||
127    ps->get<int>("emm__skip_step") > 0 ||
128    ps->get<bool>("use_icp") ||
129    (ps->get<bool>("use_glwidget") && ps->get<bool>("use_gui") && ! ps-
     >get<bool>("glwidget_without_clouds")))
130  {
131    pc_col = pointcloud_type::Ptr(createXYZRGBPointCloud(depth, visual, cam_info));
132  }
```

Code 1: uittreksel node.cpp uit RGBDSLAM

Figuur 16: Grafiek van geheugengebruik van RGBDSLAM na aanpassingen in configuratie. Op de horizontale as staat de tijdsduur (in aantal instructies), op de verticale as staat het geheugengebruik. Voor uitleg bij de grafiek zie het bijschrift bij Figuur 13.

5.4.4. Testen

Ook voor het testen van deze module is een testplan gemaakt, te vinden in Bijlage C. Omdat de taak van RGBDSLAM is om te bepalen hoe de camera bewogen heeft, is besloten om te testen hoe goed deze bewegingsbepaling is. RGBDSLAM maakt voor de bewegingsbepaling gebruik van beeldverwerkingsalgoritmen die hoeken en randen van objecten herkennen. Hiervoor moeten er in de testopstelling dus voldoende objecten aanwezig zijn om te kunnen herkennen.

Zowel translatie als rotatie van de camera moeten getest worden.[5] Voor de translatietests is een testopstelling gebouwd waarbij een aantal tafels en stoelen voor een egale muur zijn gezet. De camera is vervolgens op ca. anderhalve meter van deze opstelling geplaatst. Ook is een python-script geschreven (Bijlage D.3.) dat continu weergeeft wat op dat moment de gemeten translatie en rotatie zijn vanaf het punt waar de camera zich bevond aan het begin van de test.

Vervolgens is de translatie getest in alle zes de richtingen (naar links, naar rechts, vooruit, achteruit, omhoog, omlaag). Voor elke test is de camera een meter in de desbetreffende richting verplaatst, en is na de verplaatsing genoteerd wat de gemeten translatie was. Elke test is drie keer uitgevoerd. De resultaten van deze tests zijn te vinden in Bijlage E.3. Hierin is te zien dat de translatiemetingen redelijk accuraat zijn, de translatie wordt meestal overschat met een afwijking van minder dan 5 centimeter. Ook liggen de resultaten vrij dicht bij elkaar, wat laat zien dat de metingen redelijk consistent zijn.

Voor het testen van de rotatiemeting zijn vergelijkbare opstellingen gemaakt, maar hierbij is er voor gezorgd dat tijdens de volledige rotatie objecten in beeld waren. Voor de rotaties naar boven en beneden (pitch omhoog en omlaag) was dit niet mogelijk voor een volledige rotatie van een halve omwenteling (π rad), dus is er besloten om slechts een kwart omwenteling te roteren ($\pi/2$ rad). Ook deze tests zijn over alle drie de assen in beide richtingen drie keer uitgevoerd. De resultaten van deze tests zijn te vinden in Bijlage E.4. Hieruit is af te leiden dat de rotatie over het algemeen iets onderschat wordt, maar dat deze afwijking over het algemeen vrij klein is. Ook is aan de relatief lage standaarddeviatie te zien dat er niet veel variatie is in de meetresultaten, en dat de metingen dus vrij consistent zijn.

5.5. Module D: Van point cloud naar 3D-kaart

5.5.1. Ontwerp

Voor het maken van een driedimensionale kaart moet een systeem gevonden worden dat een reeks point clouds kan omzetten naar een kaart van voxels (driedimensionale pixels). Bij het zoeken naar mogelijkheden voor module C was hier al een optie voor gevonden, namelijk OctoMap. Het bij module C gebruikte systeem, RGBDSLAM, is ontworpen om samen te werken met OctoMap, en levert alle data die OctoMap nodig heeft om een voxel map te maken. Ook gebruikt OctoMap een slimme datastructuur waarbij het vrij eenvoudig is om bijvoorbeeld een gedetailleerde kaart van een deel van de ruimte op te vragen, of juist

[5] Translatie is verplaatsing van de camera zonder dat deze gedraaid wordt. Rotatie is draaiing van de camera zonder dat deze verplaatst wordt.

een minder gedetailleerde kaart van een grote ruimte. Al deze factoren maken OctoMap tot een zeer geschikte oplossing voor module D. Uit de literatuur bleek dat vaak gebruik gemaakt werd van OctoMap, en omdat er geen aanwijzingen zijn gevonden dat OctoMap niet de beste voxel map-applicatie van dit moment is, is er niet verder gezocht naar eventuele andere opties.

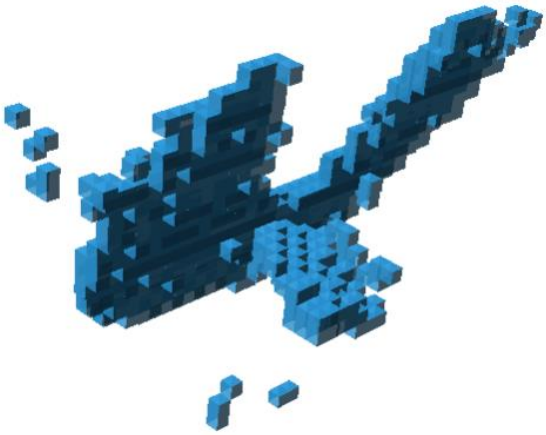
5.5.2. Implementatie

De installatie van OctoMap was vrij eenvoudig, deze kon namelijk direct vanuit de ROS repository geïnstalleerd worden. Om vervolgens de data uit RGBDSLAM bij OctoMap te krijgen, moet de topic waarop RGBDSLAM de point cloud publiceert verbonden worden aan de topic waar OctoMap naar luistert. Volgens de documentatie van RGBDSLAM zijn er drie topics met point clouds beschikbaar: /rgbdslam/batch_clouds, /rgbdslam/aggregate_clouds, en /rgbdslam/online_clouds. Hiervan wordt bij /rgbdslam/batch_clouds expliciet vermeld dat deze voor OctoMap gebruikt kan worden. Om de link te leggen tussen /rgbdslam/batch_clouds en de topic waarop OctoMap data verwacht, /cloud_in, is de regel <remap from="/rgbdslam/batch_clouds" to="/cloud_in" /> toegevoegd aan de launchfile (zie bijlage D.1. regel 3).

Met deze configuratie zou OctoMap moeten werken, maar dit was helaas niet het geval. Er bleek namelijk dat RGBDSLAM geen data publiceerde op een van de point cloud-topics. Hierop is uitgebreid onderzoek gedaan naar de interne werking van RGBDSLAM, om te achterhalen wat hier de oorzaak van is en hoe dit op te lossen was. Hiervoor is eerst de broncode van RGBDSLAM gelezen en onderzoek gedaan op internet. Daarnaast is gdb gebruik om beter te begrijpen hoe RGBDSLAM werkt. Uiteindelijk is toen gevonden dat RGBDSLAM alleen point clouds kan versturen als de configuratieoptie store_pointclouds op true staat, precies de optie die eerder op false is gezet om het probleem met het geheugengebruik van RGBDSLAM op te lossen. Ook is gevonden dat RGBDSLAM alleen point clouds naar /rgbdslam/batch_clouds stuurt als hier expliciet om gevraagd wordt. Dit kan handmatig met een menu-optie in de GUI van RGBDSLAM, of door de service /rgbdslam/ros_ui aan te roepen met het bericht "send_all". Om deze service continu aan te roepen is het bash-script caller.sh geschreven (zie bijlage D.4.), waar met behulp van catkin_create_package een ROS-package van gemaakt is. Dit script voert een keer per seconde de opdracht rosservice call /rgbdslam/ros_ui send_all uit. Het script wordt gestart vanuit de launchfile (zie bijlage D.1. regel 6). Om te voorkomen dat het geheugengebruik van RGBDSLAM uit de hand loopt is, aan de hand van een suggestie van de maker van RGBDSLAM [13], de regel node->clearPointCloud(); toegevoegd aan het broncodebestand graph_mgr_io.cpp. In bijlage D.5. is een patchfile voor deze aanpassing te vinden. Door deze aanpassing worden nu point clouds direct weggegooid als ze verzonden zijn.

Na het opnieuw compileren van RGBDSLAM is deze opnieuw getest met de Massif-tool voor geheugenanalyse (zoals beschreven in §5.4.3.). Hieruit bleek dat het geheugengebruik stabiel bleef tijdens de loopduur van het programma. Ook was nu te zien dat er inderdaad point clouds verstuurd werden door RGBDSLAM, en dat deze ontvangen werden door OctoMap. Het enige probleem nu nog was dat OctoMap nog geen voxel map maakte. Dit bleek te komen doordat OctoMap niet wist relatief aan welke positie de voxel map gemaakt moest worden. De oplossing hiervoor stond in de launchfile voor OctoMap, hier moest de

parameter "frame_id" op "map" worden gezet, in plaats van op de standaardwaarde "odom_combined". Na deze aanpassing werd wel een voxel map gemaakt, die ook zichtbaar was in Rviz. In Figuur 17 is een voorbeeld te zien van hoe een voxel map eruit ziet.



Figuur 17: Voxelmap gemaakt door OctoMap. Elke voxel stelt een volume van 5x5x5 centimeter voor.

5.5.3. Testen

Bij het testen van OctoMap was het plan om een voxel map te maken van een testomgeving, op te meten hoe groot objecten in deze testomgeving waren, en achteraf in de gemaakte voxel map te meten hoe groot de objecten in de voxel map zijn. In Figuur 18 is te zien hoe deze testomgeving eruit ziet. Deze bestaat uit een enkele stoel die op zijn rug ligt. Het bleek namelijk tijdens het testen dat dit de enige manier was om een voxel map te genereren die niet volledig uit ruis bestond. De voxel map die in deze testomgeving is gegenereerd, is te vinden in Figuur 17. Deze voxel map laat een diagonaal vlak zien van ca. 23 voxels hoog en 34 voxels breed. Aangezien elke voxel een volume voorstelt van 5x5x5 centimeter, houdt dit in dat het vlak ca. 115x170 cm is. Gezien de positie van de camera lijkt het aannemelijk dat het vlak de zitting van de stoel moet voorstellen, maar deze is slechts 43,5 cm breed en 44,5 cm hoog. De voxel map is dus geen accurate voorstelling van de werkelijkheid. De enige conclusie die uit deze test getrokken kan worden, is dus ook dat er nog meer werk nodig is om OctoMap goed te laten functioneren.

Figuur 18: Testopstelling OctoMap: een liggende stoel

6. Conclusies en aanbevelingen

Om dit verslag te concluderen zal worden bepaald in hoeverre het systeem aan de in §4.2 genoemde eisen is voldaan. Tot slot zullen enkele aanbevelingen worden gedaan voor potentiële toekomstige verbeteringen van de rolstoel en de 3D-mappingsystemen.

6.1. Conclusie

Eerst zal gekeken worden naar in hoeverre er aan de in §4.2 genoemde eisen is voldaan. Een overzicht hiervan is te vinden in Tabel 4. De eerste eis is dat de data uit de 3D-camera wordt uitgelezen. Dit gebeurt door module A. Eis 2 is dat er op basis van de cameradata een 3D-kaart gemaakt wordt. modules C en D samen vervullen deze eis. Eis 3 wordt vervuld door het testplan, zoals dat te vinden is in Bijlage C. Eis 4 wordt ook vervuld, de kwantitatieve informatie over de nauwkeurigheid van het systeem is te vinden in Bijlage E, en wordt uitgelegd in §5.3.3, §5.4.4, en §5.5.3. De vijfde eis was dat de rolstoel op basis van de 3D-kaart obstakels kan vermijden. Bij het wijzigen van de opdracht (zoals beschreven in §3.4) is besloten dat aan deze eis niet meer voldaan hoeft te worden. Over eis 6, het gebruik van semantische informatie, was bij het formuleren van de eisenlijst al besloten dat deze buiten de scope van de opdracht valt. Aan beide niet-functionele eisen is voldaan.

#	Eis	MoSCoW	Voldaan?
	Functionele eisen		
1	De data uit de 3D-camera wordt uitgelezen.	M	Ja
2	Er wordt een 3D-kaart gemaakt op basis van de data uit de camera.	M	Ja
3	De testomgeving is reproduceerbaar.	M	Ja
4	Het uitvoeren van het testplan leidt tot kwantitatieve informatie over de nauwkeurigheid van afstands- en afmetingenbepaling van het systeem	M	Ja
5	De rolstoel kan op basis van de 3D-kaart obstakels vermijden.	C	Nee
6	De rolstoel gebruikt semantische informatie voor slimmere navigatie.	W	Nee
	Niet-functionele eisen		
7	De 3D-camera is een Intel RealSense ZR300.	S	Ja
8	Het systeem is gebaseerd op ROS-modules.	S	Ja

Tabel 4: Eisenlijst.

6.2. Aanbevelingen

De eerste aanbeveling is het verder ontwikkelen van het maken van de voxel map. Zoals in §5.5.3. is beschreven, geeft OctoMap nu nog geen goede voorstelling van de werkelijkheid. Om verder te kunnen werken met het tot nu toe ontwikkelde product, zal er onderzoek gedaan moeten worden naar de oorzaak hiervan, en moeten er verbeteringen doorgevoerd worden om OctoMap beter te laten werken.

Een andere belangrijke aanbeveling voor een eventueel vervolgproject is het overstappen naar een nieuwe RGBD-camera, omdat de ZR300 niet meer ondersteund wordt door Intel. Met een nieuwe camera kan ook geüpgraded worden naar de nieuwste versie van librealsense en realsense_camera, en daarmee ook naar de nieuwste versie van ROS en Ubuntu. Dit zal een aantal voordelen met zich meebrengen, zowel in performance (nieuwere versies van software zijn vaak beter geoptimaliseerd) als in functionaliteit. Ook zal de hier gebruikte versie van Ubuntu (16.04) niet meer ondersteund worden na april 2021, wat te zijner tijd beveiligingsrisico's met zich mee kan brengen.

De laatste aanbeveling is om de meetresultaten van de tests beter te analyseren. Dit moet dan gebeuren door iemand die een grotere theoretische basis in de statistiek heeft, zodat bepaald kan worden of de conclusies die nu uit de data zijn getrokken statistisch verantwoord zijn. Op basis van deze statistische analyse kan ook bepaald worden of de gedane metingen voldoende zijn om nuttige conclusies te trekken, of dat er wellicht nog meer of andere metingen gedaan moeten worden.

7. Evaluatie

In dit hoofdstuk zal ik terugkijken op mijn afstuderen. Als eerste zal ik de opgeleverde producten evalueren. Vervolgens zal ik de wijze waarop ik tot deze producten ben gekomen evalueren in de procesevaluatie. Daarna volgt een reflectie op mijn eigen kunnen en handelen. Tot slot zal ik evalueren op welke manier ik de beroepstaken, zoals die in de afstudeeropdracht waren vastgelegd, heb vervuld.

7.1. Productevaluatie

In deze sectie zullen de verschillende opgeleverde producten worden geëvalueerd. Het gaat hierbij om de gemaakte analyse van de bestaande en te ontwikkelen systemen, de gemaakte ontwerpen en de implementatie hiervan, en de testopstellingen met de hieruit voortkomende testresultaten. Hoewel het Plan van Aanpak ook beschouwd kan worden als een opgeleverd product, zal dit niet in deze sectie worden geëvalueerd, maar in §7.2.

Over het algemeen ben ik tevreden over het opgeleverde eindresultaat. Het product werkt goed, en levert bruikbare data op. Ook is aan alle eisen voldaan die belangrijk waren om tot een goed product te komen. Met behulp van de testscripts zijn de verschillende onderdelen ook uitvoerig getest. Uit gesprekken met de bedrijfsmentor blijkt ook dat hij over het algemeen tevreden is met het uitgevoerde werk.

7.1.1. Analyse bestaande en nieuwe systemen

In hoofdstuk 4 is beschreven hoe de huidige state-of-the-art en de reeds bestaande systemen zijn geanalyseerd, en is besproken aan welke eisen het te ontwikkelen systeem moest voldoen. Het oorspronkelijke literatuuronderzoek was op zich nuttig om een beeld te krijgen van wat andere onderzoeksgroepen doen op het gebied van semantic mapping, maar dit bleek uiteindelijk niet direct van toepassing op het te ontwikkelen systeem. Dit kwam omdat met het literatuuronderzoek begonnen was op basis van de oorspronkelijke afstudeeropdracht, maar het uiteindelijke doel van het afstuderen later aangepast was (zie hiervoor §3.4.). Het onderzoek heeft wel geholpen om mij te leren hoe wetenschappelijke artikelen gelezen moeten worden, wat nuttig was bij het maken van het ontwerp van module C (§5.4.1.).

Tijdens het ontwerpen, implementeren en testen bleken de in §4.2. opgestelde eisen goed te kloppen. Zo waren ze erg nuttig bij het maken van keuzes in de ontwerpfases, en kon er bij het maken van het testplan goed op teruggegrepen worden. Zoals in hoofdstuk 6 te lezen is, is aan bijna al deze eisen voldaan. Omdat door te voldoen aan deze eisen een goed werkend systeem is ontwikkeld, kan geconcludeerd worden dat de eisenlijst goed aansloot bij de opdracht.

De analyse van het bestaande systeem, zoals die in §4.3. is beschreven, had als conclusie dat het nieuw te ontwikkelen systeem volledig los zou staan van de bestaande systemen. Hierdoor was het mogelijk om het nieuwe systeem volledig op mijn laptop te ontwikkelen, in plaats van dat hiervoor de rolstoel nodig was. Het effect hiervan was dat latere hardwareproblemen van de rolstoel (met name met de accu) niet relevant waren voor mij, en geen effect hebben gehad op de voortgang van mijn afstuderen.

7.1.2. Ontwerpen

In §5.1 is beschreven hoe het eerste globale ontwerp is gemaakt. Hoewel dit ontwerp uiteindelijk flink is aangepast, ben ik hier toch tevreden mee. Het ontwerp gaf een goed beeld van hoe het systeem eruit moest gaan zien, en was denk ik het beste ontwerp wat, gegeven de kennis van dat moment, gemaakt kon worden. Het heeft een nuttige houvast gegeven tijdens het ontwerpen en implementeren van de eerste module, en was een belangrijk hulpmiddel in zowel het maken van de eerste planning als in de communicatie met de opdrachtgever over deze planning en de voortgang hiervan.

Na het wijzigen van de opdracht is het ontwerp aangepast, zoals te vinden is in §5.2. Dit gewijzigde ontwerp komt zeer goed overeen met het uiteindelijk ontwikkelde systeem, en heeft goed geholpen om de detailontwerpen te kunnen maken.

Ook de detailontwerpen zijn goed. In §5.3.1. is het keuzeproces te vinden dat gevolgd is voor module A. Hierin zijn verschillende opties onderzocht, en is de conclusie getrokken om gebruik te maken van librealsense en realsense_camera. Deze conclusie is goed uitgevallen, het uiteindelijke systeem maakt inderdaad gebruik van deze systemen, en ze werken goed. Ook het keuzeproces voor module C, zoals beschreven in §5.4.1., heeft tot een goed resultaat geleid. Hier is na onderzoek gekozen voor RGBDSLAMv2, een systeem dat uiteindelijk essentieel is geworden in het eindproduct, en dat hierin naar behoren werkt. Het ontwerpproces voor module D, te vinden in §5.5.1., was redelijk eenvoudig, en het gebruik van OctoMap heeft tot een positief resultaat geleid, waarbij realtime voxel maps gemaakt kunnen worden. Al met al kijk ik positief terug op de gemaakte ontwerpen en keuzes.

7.1.3. Implementatie

Over de implementaties ben ik over het algemeen zeer tevreden. De implementatie van module A had, zoals beschreven in §5.3.2., nogal wat voeten in aarde. Dankzij mijn probleemoplossend vermogen, enkele aanwijzingen van mijn bedrijfsmentor, en een grote hoeveelheid online te vinden documentatie is het uiteindelijk gelukt om module A goed werkend te krijgen. Toen dit systeem eenmaal werkte is het ook altijd betrouwbaar blijven werken, ik kon er op vertrouwen dat module A later nooit de oorzaak van een fout of probleem zou zijn. Daarom ben ik blij met mijn implementatie van module A.

Voor module C lagen de zaken iets anders. Zoals beschreven in §5.4.2. was er hier ook een probleem met de installatie, wat gelukkig vrij eenvoudig op te lossen was. Daarna kwam echter nog een groter probleem, namelijk dat van het geheugengebruik zoals beschreven in §5.4.3. Het oplossen van dit probleem heeft vrij veel tijd gekost, maar uiteindelijk is het, mede dankzij mijn kennis van C++ en mijn vaardigheid om snel een complex systeem te doorgronden, gelukt om de oorzaak van dit probleem te achterhalen en hier een oplossing voor te vinden. Hoewel ik later nog terug moest naar module C om module D werkend te krijgen, is de voornaamste oorzaak hiervan de ontwerpkeuzes die gemaakt zijn door de oorspronkelijke ontwikkelaar van RGBDSLAM. Aangezien ik hier niets aan had kunnen veranderen, en het mij toch gelukt is om module C goed werkend te krijgen, ben ik tevreden over mijn implementatie van deze module.

In §5.5.2. is te vinden hoe de implementatie van module D is gegaan. Hoewel dit relatief eenvoudig had moeten zijn, bleek dat hiervoor aanpassingen nodig waren in de code van module C. De kennis die ik had opgedaan bij het oplossen van probleem met het geheugengebruik van module C was hiervoor erg nuttig, en heeft er voor gezorgd dat het gelukt is om module D werkend te krijgen. Uiteindelijk werkt module D goed. Al met al heb ik drie implementaties gemaakt, voor modules A, C, en D. Al deze implementaties functioneren naar verwachting, en werken goed met elkaar samen.

7.1.4. Tests

Er is uitgebreid kwantitatief getest hoe nauwkeurig met name modules A en C zijn. De resultaten van deze tests laten zien hoe accuraat de metingen en berekeningen van deze modules en systemen zijn. Ook module D is getest, deze test laat zien waar in later onderzoek nog meer tijd aan besteed moet worden.

Zoals in §5.3.3. is uitgelegd, is er gemeten hoe nauwkeurig de dieptemeting van de camera is. De resultaten hiervan geven een goed beeld van de nauwkeurigheid en betrouwbaarheid van de camera. Ook is de gebruikte testopstelling eenvoudig reproduceerbaar, waardoor het eenvoudig is om bijvoorbeeld een vergelijkbare test te doen met een andere camera. De gevonden resultaten vormen een goede basis voor verder onderzoek. Daarbij is het goed dat de testopstelling eenvoudig en reproduceerbaar is.

In §5.4.4. is beschreven hoe de tests van module C zijn uitgevoerd, en wat hier de resultaten van zijn. Deze resultaten laten duidelijk zien hoe consistent en nauwkeurig RGBDSLAM translatie en rotatie kan bepalen. Ook hier is de gebruikte testopstelling redelijk eenvoudig en goed reproduceerbaar, wat het mogelijk maakt om een vergelijking te maken tussen RGBDSLAM en een eventuele ander algoritme. Zowel de testresultaten als de testopstelling vormen een nuttig product dat in de toekomst goed gebruikt kan worden.

De test van module D, zoals beschreven in §5.5.3, is minder waardevol. Uit deze test is vooral gebleken dat er nog meer werk nodig gaat zijn om ervoor te zorgen dat OctoMap een accurate weergave van de werkelijkheid gaat tonen, die daadwerkelijk bruikbaar is voor navigatie.

7.2. Procesevaluatie

In de evaluatie van het proces zijn er twee onderdelen die geëvalueerd moeten worden. Ten eerste moet er gekeken worden naar hoe het proces is verlopen, en ten tweede moet het gevolgde proces vergeleken worden met het Plan van Aanpak om te bepalen in hoeverre het PvA gevolgd is.

In het algemeen ben ik tevreden met het verloop van het proces. De onderzoeks- en kennismakingsfases aan het begin zijn goed verlopen en nuttig geweest om een basis te vormen voor de rest van het project. De hoofdfase is goed begonnen, met het maken van een globaal ontwerp op basis waarvan een heldere planning gemaakt is. Het ontwerpen en implementeren van module A ging vervolgens ook vlot. Module C heeft langer geduurd dan oorspronkelijk gepland, maar deze tijd was nodig vanwege de complexiteit van module C. Met de mogelijkheid dat deze implementatie langer ging duren was oorspronkelijk geen

rekening gehouden in de risicoanalyse, wat wellicht wel had moeten gebeuren. Dan had namelijk eerder bedacht kunnen worden hoe hiermee omgegaan kon worden.

Het implementeren van module D ging ook goed. Hoewel ik hier tegen wat inhoudelijke problemen aanliep, ben ik wel continu bezig kunnen zijn met het oplossen van deze problemen. Uiteindelijk zijn de ontwerp- en implementatiefases goed verlopen, er was altijd duidelijk wat me te doen stond en hoe ik dit moest gaan doen, of wat ik moest gaan onderzoeken om hierachter te komen.

Ook het testproces is goed verlopen. Het schrijven van de testscripts ging redelijk snel, en het uitvoeren van de tests liep vervolgens goed. Hoewel het resultaat van de test van module D tegenviel, laat dit wel goed het contrast zien tussen de positieve resultaten van de tests van modules A en C en het negatievere resultaat van de test van module D.

De afrondingsfase verliep soepel, met name omdat ik tijdens het proces al grote delen van mijn afstudeerverslag had geschreven. Ook de meerdere feedbackmoment van zowel mijn bedrijfsmentor als begeleidend examinator hebben hierbij geholpen om een kwalitatief hoogwaardig afstudeerdossier samen te stellen.

7.2.1. Plan van Aanpak

Betreffende het Plan van Aanpak lijkt het in eerste oogopslag alsof ik hier sterk van afgeweken ben. Verdere analyse laat echter zien dat de hoofdlijnen van het PvA vrij goed gevolgd zijn, en dat de afwijking alleen zichtbaar is in de details. De opdracht is, zoals dat in het PvA beschreven was, opgedeeld in een opstart- hoofd- en afrondingsfase, waarbij de opstartfase een week langer heeft geduurd dan gepland. De uiteindelijke afrondingsfase is korter geworden, maar dit heeft niet tot problemen geleid omdat tijdens de implementatiefase al veel geschreven is aan het verslag. De verlenging van de opstartfase komt met name door het literatuuronderzoek, waar twee weken voor zijn genomen in plaats van een. Dit was nodig om de huidige state-of-the-art voldoende te doorgronden.

In het begin van de hoofdfase is het PvA gevolgd wat betreft de duur van het maken van het globale ontwerp en het ontwikkelen van module A. Voor module C was veel meer tijd nodig dan gepland, vanwege de complexiteit van deze module. Ook bleek module C een veel essentiëlere module voor het volledige systeem dan oorspronkelijk gedacht, waardoor het het ook waard was om hier zo lang mee bezig te zijn als nodig om deze module goed werkend te krijgen. Module D heeft wel weer ongeveer even lang geduurd als wat was gepland, waardoor ik op tijd kon beginnen aan de testfase. In de testfases ben ik vrij significant afgeweken van het PvA, door alle tests aan het einde van de hoofdfase uit te voeren, in plaats van meteen na de implementatiefase van elke module. Omdat er kwantitatief getest werd op nauwkeurigheid en betrouwbaarheid van de modules, in plaats van of de modules überhaupt werkten (wat al tijdens de implementatiefases was bepaald), waren de testresultaten niet nodig om te beginnen aan een volgende ontwerp- en implementatiefase.

De afrondingsfase is niet helemaal verlopen zoals gepland in het PvA. Dit was met name omdat in het PvA geen rekening was gehouden met het feit dat er veel eerder dan de laatste drie weken al veel tijd besteed moest worden aan het schrijven van het afstudeerverslag, ten

behoeve van de conceptbespreking en het tussentijds assessment. Hierdoor heeft de afrondingsfase deels parallel plaatsgevonden met de hoofdfase.

Wat betreft de gevolgde methodiek ben ik hier over het algemeen tevreden over. Dankzij de opdeling in fases had ik snel in de gaten wanneer ik niet meer op schema liep, en de fasering hielp ook om hierover duidelijk te communiceren. Ook wist ik altijd in welke fase ik me bevond, en dus wat ik op een bepaalde dag moest doen. De keuze om sequentieel te werken (beginnen bij de 3D-camera, en in elke stap de datastroom verder verwerken) was ook goed, omdat hierdoor altijd een werkend tussenresultaat zichtbaar was waarop verder gebouwd kon worden.

7.3. Persoonlijke evaluatie

Bij het literatuuronderzoek aan het begin van het project heb ik niet alleen inhoudelijk veel geleerd over *semantic mapping*, maar heb ik ook geleerd om wetenschappelijke artikelen te lezen, het verband tussen verschillende onderzoeken te leggen, en een beeld te vormen van de huidige state-of-the-art van een bepaald vakgebied. Dit onderzoek is op zich goed gegaan, en de samenvattingen die ik maakte van de gelezen artikelen hebben mij geholpen met het beter begrijpen van de artikelen. Wat ik beter had kunnen doen was dat ik te lang met dit onderzoek bezig ben geweest, en dat ik wellicht eerder had moeten inzien dat de artikelen die ik aan het lezen was niet relevant genoeg waren voor mijn opdracht.

Vervolgens heb ik bij het kennismaken met de rolstoel en het maken van het globale ontwerp veel geleerd over hoe ROS werkt, kennis die ik in de rest van het project heb kunnen toepassen en uitbreiden. In het algemeen ben ik tevreden over hoe ik hier te werk ben gegaan, maar ik heb wel gemerkt dat het maken van het globale ontwerp in het begin vrij moeizaam ging. Dit is een vaardigheid die ik nog verder moet en wil ontwikkelen.

Bij de implementatie van module A heb ik de ervaring die ik al met Linux had kunnen toepassen, en tegelijkertijd heb ik hierbij nieuwe kennis opgedaan over de werking van de Linux-kernel. Ook bij de latere implementatiefases heb ik goed gebruik kunnen maken van mijn Linux-ervaring. In de implementatiefases heb ik soms wat langzamer gewerkt dan ik had gewild, maar is het me uiteindelijk wel gelukt om een implementatie te maken waar ik tevreden over kon zijn.

Bij de analyse van de performance van module C heb ik veel geleerd over *profiling*-tools, een onderwerp waar ik eerder nog niet mee in aanraking was gekomen maar wat ik wel erg interessant vond. Hier heb ik mijn reeds beschikbare kennis over de programmeertaal C++ goed kunnen toepassen, maar heb ik wel geleerd dat het lezen en analyseren van code een hele andere vaardigheid is dan het schrijven van code. Ik ben vooral blij met de snelheid waarmee ik leerde werken met tools en systemen die me eerst nog totaal onbekend waren. Deze vaardigheid om snel nieuwe technieken onder de knie te krijgen is voor mij erg nuttig.

Bij het implementeren van module D was het erg nuttig dat ik inmiddels al vrij goed bekend was geworden met ROS en de configuratie van ROS-nodes. Ook kon ik hier goed gebruik maken van mijn kennis over C++ en Bash-scripts. Een vaardigheid die ik ook hard nodig had om tot een werkende implementatie te komen was mijn vermogen om op internet

oplossingen te vinden voor problemen waar ik tegenaan loop, en uit meerdere (niet altijd even coherent geschreven) forumberichten uiteindelijk een goed werkende oplossing te destilleren.

Bij het testen van de verschillende modules heb ik veel nieuwe vaardigheden moeten leren. Dit was nodig omdat ik nog nooit eerder op deze manier kwantitatief een systeem heb getest. Zo had ik nog geen ervaring met het doen van metingen, en kan ik dus ook niet inschatten of mijn meetmethoden wetenschappelijk verantwoord zijn. Ook bij het statistisch analyseren van de gevonden resultaten heb ik slechts gebruik kunnen maken van het kleine beetje basiskennis statistiek dat ik heb. Als ik hier ooit meer mee wil gaan doen, zal ik hier nog veel meer over moeten leren, maar voor deze opdracht lijkt het mij dat de kennis die ik zelf al had voldoende was.

De afrondingsfase, en met name het schrijven van het afstudeerdossier, vond ik erg lastig. Ik weet van eerdere projecten dat ik met name het beschrijven van mijn werk- en denkprocessen erg lastig vind, en hier had ik bij het schrijven van dit verslag opnieuw last van. Het schrijven van het verslag heeft dan ook relatief veel tijd gekost, maar uiteindelijk ben ik wel blij met de kwaliteit van datgene dat ik geschreven heb.

7.4. Evaluatie beroepstaken

Deze beroepstaken zijn overgenomen uit de oorspronkelijke afstudeeropdracht, en vormen de basis van de vaardigheden die ik tijdens mijn afstuderen moet laten zien. De tekst in *italics* komt uit de afstudeeropdracht, waarin ik had uitgelegd hoe ik toen dacht aan de beroepstaken zou gaan voldoen. Hier reflecteer ik op, en ik geef aan hoe ik uiteindelijk aan de beroepstaken heb voldaan.

7.4.1. G1: Praktische aspecten hanteren in (internationale) projecten:

Onder taak G1 valt onder andere het opstellen van een plan van aanpak, en het plannen van de eigen werkzaamheden. Om er voor te zorgen dat ik op schema blijf is het belangrijk dat ik in mijn plan van aanpak duidelijke deadlines voor mijzelf stel. Hiermee plan ik mijn werkzaamheden op zo'n manier in dat ik het maximale uit mijzelf kan halen.
In het begin van mijn afstuderen heb ik inderdaad een plan van aanpak opgesteld en mijn werkzaamheden ingepland (zie hoofdstuk 3 en bijlage A). Toen bleek dat ik niet de volledige opdracht af kon maken, is in overleg met de bedrijfsmentor de scope van de opdracht verkleind. Hoewel er vrij significant van de oorspronkelijke planning is afgeweken, heb ik wel altijd een up-to-date planning bijgehouden, zodat ik altijd wist welke taken ik nog had en wat hiervoor de deadlines waren.

7.4.2. A1: Analyseren van het probleemdomein:

Om te bepalen welke semantic mapping-systemen beschikbaar en bruikbaar kunnen zijn, is het belangrijk de huidige literatuur goed te analyseren. Om ook ontwerpen te maken die daadwerkelijk zullen leiden tot een werkend systeem rond de 3D-camera, is het belangrijk om de randvoorwaarden en systeemeisen goed en duidelijk in beeld te brengen.

Zoals beschreven in §4.1. ben ik begonnen met het doen van een literatuuronderzoek naar semantic mapping-systemen. Voor het ontwerpen van module C heb ik ook literatuuronderzoek gedaan, namelijk naar 3D-mappingsystemen (zie §5.4.1.). Ik heb ook de randvoorwaarden en systeemeisen in beeld gebracht, en veel geleerd over hoe 3D-mappingsystemen kunnen werken. Dit is onder andere te zien in §4.2., 3., en 4.

7.4.3. C6: Ontwerpen van software

Om de individuele onderdelen van de software correct, testbaar en integreerbaar te implementeren, moeten deze onderdelen nauwkeurig en gedetailleerd ontworpen worden. Hierdoor zal het eenvoudig zijn om deze onderdelen te implementeren, te testen, en ze met elkaar te combineren tot een goed werkend eindproduct.
Op basis van mijn globale ontwerp heb ik de verschillende modules tot in detail ontworpen. Hierbij is bepaald wat elke module moet kunnen, en is op basis van deze eisen gezocht naar bestaande systemen die de functionaliteit van deze module konden vervullen. Dit is onder andere te lezen in §5.3.1. En §5.4.1.

7.4.4. C10: Ontwerpen van een systeemarchitectuur:

Om de te realiseren software efficiënt en zonder problemen te realiseren, is het belangrijk dat er een goede globale systeemarchitectuur wordt ontworpen. Deze globale systeemarchitectuur zal het ook vereenvoudigen om de architectuur van subsystemen en onderdelen te ontwerpen en te implementeren.
Zoals beschreven in §5.1. heb ik op basis van de opgedane kennis over 3D-mapping en ROS, en gegeven de opgestelde eisen, een globaal ontwerp gemaakt dat het volledige systeem opdeelt in individueel te ontwikkelen modules. Vervolgens heb ik aan de hand van wijzigingen in de scope van de opdracht dit ontwerp sterk aangepast (zie §5.2.), waarbij de oorspronkelijke module-indeling gedeeltelijk behouden is gebleven.

7.4.5. D16: Het realiseren van software:

Bij het implementeren van het semantic mapping-systeem rond de 3D-camera, en om deze te integreren in het bestaande rolstoelsysteem, zal ik uiteindelijk een significante hoeveelheid code moeten schrijven.
Tijdens het implementeren van de verschillende modules heb ik niet heel veel code zelf hoeven schrijven. De belangrijkste code die ik hiervoor geschreven heb is de launchfile die te vinden is in Bijlage D.1. en het Bash-script caller.sh (Bijlage D.4.), zoals beschreven in §5.5.2. Tijdens de testfases is veel meer code geschreven, namelijk de pythonscripts die gebruikt zijn om de tests uit te voeren. Deze scripts zijn beschreven in §5.3.3 en §5.4.4. en zijn te vinden in Bijlagen D.2. en D.3.

7.4.6. D17: Testen van softwaresystemen:

Om te bewijzen dat de 3D-camera toegevoegde waarde heeft, en dat de semantic mapping correct werkt, moeten de systemen uitvoerig getest worden.
Met name in het laatste deel van mijn afstudeeropdracht heb ik veel getest. In §5.3.3. is uitgebreid beschreven hoe module A getest is, met de resultaten van deze tests in Bijlagen E.1. en E.2. Vervolgens is in §5.4.4. uitgelegd wat er gedaan is om module C te testen.

Hiervan zijn de resultaten te vinden in Bijlagen E.3. en E.4. In §5.5.3. is besproken hoe module D is getest. Het testplan van al deze tests is te vinden in Bijlage C.

8. Referenties

[1] L. Geluk, S. Menéndez and H. Camps, “Jaarstukken 2017 - Jaarverslag en jaarrekening”, Dehaagsehogeschool.nl, 2018. [Online]. Beschikbaar: https://www.dehaagsehogeschool.nl/docs/default-source/documenten-over-de-haagse/organisatie/180626_de_haagse_hogeschool_jaarstukken-2017.pdf. [Geraadpleegd: 4 november 2018].

[2] “Organogram”, Dehaagsehogeschool.nl, 2018. [Online]. Beschikbaar: https://www.dehaagsehogeschool.nl/over-de-haagse/organisatie/organogram. [Geraadpleegd: 4 november 2018].

[3] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, “Ros: an open-source robot operating system,” in *ICRA workshop on open source software*, vol. 3, no. 3.2. Kobe, Japan, 2009, p. 5.1

[4] N. Sunderhauf, F. Dayoub, S. Mcmahon, B. Talbot, R. Schulz, P. Corke, G. Wyeth, B. Upcroft, and M. Milford, “Place categorization and semantic mapping on a mobile robot,” *2016 IEEE International Conference on Robotics and Automation (ICRA)*, 2016.

[5] N. Blodow, L. C. Goron, Z.-C. Marton, D. Pangercic, T. Ruhr, M. Tenorth, and M. Beetz, “Autonomous semantic mapping for robots performing everyday manipulation tasks in kitchen environments,” *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2011.

[6] X. Gallart Del Burgo, “Semantic Mapping in ROS”, MSc., KTH Royal Institute of Technology, 2013.

[7] H. van. Vliet, *Software Engineering: Principles and Practice*, 3rd ed. Wiley, 2008.

[8] R. Mur-Artal and J. Tardos, “ORB-SLAM2: An Open-Source SLAM System for Monocular, Stereo, and RGB-D Cameras”, *IEEE Transactions on Robotics*, vol. 33, no. 5, pp. 1255-1262, 2017.

[9] A. Hornung, K. Wurm, M. Bennewitz, C. Stachniss and W. Burgard, “OctoMap: an efficient probabilistic 3D mapping framework based on octrees”, *Autonomous Robots*, vol. 34, no. 3, pp. 189-206, 2013.

[10] F. Endres, J. Hess, J. Sturm, D. Cremers and W. Burgard, "3-D Mapping With an RGB-D Camera", *IEEE Transactions on Robotics*, vol. 30, no. 1, pp. 177-187, 2014.

[11] S. Rap, "Instructions for Compiling Rgbdslam (V2) on a Fresh Ubuntu 16.04 Install (Ros Kinetic) in Virtualbox", *GitHub*, 2017. [Online]. Beschikbaar: https://github.com/felixendres/rgbdslam_v2/wiki/Instructions-for-Compiling-Rgbdslam-(V2)-on-a-Fresh-Ubuntu-16.04-Install-(Ros-Kinetic)-in-Virtualbox. [Geraadpleegd: 4 november 2018].

[12] “Profiling roslaunch nodes”, *Wiki.ros.org*, 2018. [Online]. Beschikbaar: http://wiki.ros.org/roslaunch/Tutorials/Profiling%20roslaunch%20nodes. [Geraadpleegd: 04 november 2018].

[13] F. Endres, 'RGBDSLAM & Octomap for continuous realtime mapping - ROS Answers: Open Source Q&A Forum', 2012. [Online]. Beschikbaar: https://answers.ros.org/question/33909/. [Geraadpleegd: 2 december 2018].

9. Verklaring van afkortingen

BRIEF: Binary Robust Independent Elementary Features; een feature detection algoritme.
FAST: Features from Accelerated Segment Test; een hoekdetectiealgoritme.
GUI: Graphical User Interface
LIDAR: Samenvoeging van *Light* en *Radar*, radar die zichtbaar licht in plaats van radiogolven gebruikt.
ORB: Oriented FAST and Rotated BRIEF
RGBD: Red-Green-Blue-Depth
ROS: Robot Operating System
SLAM: Simultaneous Localization and Mapping

Bijlagen

Plan van aanpak

Ontwikkelen van 3D-mapping voor de Slimme Rolstoel

Richard Kokx - 14139227

Inleiding

De vraag naar slimme apparaten is de afgelopen jaren enorm toegenomen. Voor het vervoer van zowel personen als goederen is bijvoorbeeld steeds meer vraag naar zelfrijdende apparaten, die zonder tussenkomst van mensen hun eigen weg kunnen vinden. Ook in de zorg wordt steeds meer gebruik gemaakt van slimme technologieën, bijvoorbeeld om het werk van artsen en verpleegkundigen te verlichten, of om patiënten te helpen. Het lectoraat Smart Sensor Systems van de Haagse Hogeschool heeft besloten deze ontwikkelingen te combineren: om onderzoek te doen naar technologieën voor zelfrijdende apparaten is besloten een slimme rolstoel te bouwen.

Deze rolstoel werkt al wel, maar de navigatie hiervan laat nog te wensen over. Voor mijn afstuderen heb ik de opdracht om de navigatie van de rolstoel te verbeteren. Hiervoor is een 3D-camera beschikbaar die aan de rolstoel gekoppeld kan worden. Met behulp van deze camera moet de rolstoel een driedimensionaal beeld van de omgeving krijgen.

1. Aanleiding en context

Het lectoraat Smart Sensor Systems doet momenteel onderzoek naar indoor navigatiesystemen voor mobiele robots. Een van de toepassingen van dit onderzoek is het ontwikkelen van een slimme rolstoel. Het doel hiervan is om met slimme navigatietechnieken de elektrische rolstoel veiliger en gebruiksvriendelijker te maken. Er is reeds een prototype slimme rolstoel ontwikkeld, die automatisch kan navigeren en routes kan plannen. De navigatie van deze rolstoel moet echter flink verbeterd worden voor deze praktisch bruikbaar is. Hiervoor wil het lectoraat graag dat gebruik gemaakt wordt van *3D mapping*-technieken.

Bij 3D mapping wordt niet alleen een plattegrond gemaakt van wat er op de grond staat, maar wordt een driedimensionaal totaalbeeld van de omgeving verkregen. Hierdoor kan bijvoorbeeld ook een tafelblad gezien worden, waar laag bij de grond alleen de tafelpoten zichtbaar zijn.

2. Probleemanalyse en probleemstelling

De navigatie van de rolstoel is nog niet goed genoeg, waardoor de rolstoel kan botsen op obstakels. Het probleem is dat de rolstoel obstakels niet goed kan herkennen, omdat de rolstoel momenteel vooral een laser range finder gebruikt, die vrij laag bij de grond is geplaatst en die alleen in een horizontaal vlak obstakels kan zien. Hierdoor kan het bijvoorbeeld zijn dat de rolstoel tussen twee tafelpoten door probeert te rijden, omdat niet bekend is dat hier een tafelblad tussen zit. Vervolgens kan de rolstoel (of de inzittende) tegen het tafelblad aan botsen. Om de navigatie te verbeteren heeft het lectoraat een 3D-camera aangeschaft. Deze moet de rolstoel meer inzicht geven in de omgeving, zodat de navigatie beter wordt. Momenteel wordt deze wel gebruikt, maar er is geen informatie over hoe bruikbaar of betrouwbaar de data van de 3D-camera is.

3. Doelstelling en eindresultaat

Het doel van de afstudeeropdracht is om de nauwkeurigheid en betrouwbaarheid van de navigatie van de slimme rolstoel te verbeteren, door gebruik te maken van beelden van de 3D-camera om obstakels te herkennen. Met behulp van deze informatie moet de robot zelfstandig kunnen navigeren op zo'n manier dat de botsdetectie niet in werking hoeft te komen.

Het resultaat van de afstudeeropdracht is een beter werkende slimme rolstoel. Hiervoor worden de beelden van de 3D-camera gebruikt om een driedimensionale kaart van de omgeving te maken. Deze kaart wordt vervolgens gebruikt in de navigatie-algoritmen.

Het lectoraat wil ook graag dat de rolstoel objecten niet alleen kan zien en ontwijken, maar ze ook kan herkennen. Als tijdens het afstuderen blijkt dat hier tijd voor over is, kan ik eventueel onderzoeken of het mogelijk is deze objectherkenning toe te voegen.

Om kwantitatief te kunnen bepalen hoe goed de rolstoel functioneert moet er ook een testopstelling gedefinieerd worden, die gebruikt kan worden voor het testen van de rolstoel, en die reproduceerbaar is door anderen die later aan de rolstoel gaan werken.

De algoritmen voor de interpretatie van de beelden, het omzetten van deze beelden in een 3D-kaart, en het gebruik van deze kaart in de navigatiesystemen zullen, samen met alle overige gebruikte algoritmen, technieken en andere informatie, vastgelegd en uitgebreid gedocumenteerd worden, zodat deze ook gebruikt kunnen worden bij andere projecten met intelligente navigatiesystemen van het lectoraat Smart Sensor Systems.

4. Randvoorwaarden en risicoanalyse

De rolstoel maakt gebruikt van Robot Operating System (ROS), een open-source systeem voor de aansturing van robots. De toe te voegen functionaliteit moet binnen de bestaande ROS-architectuur geplaatst worden, en hier een aanvulling op zijn.

Een mogelijk probleem is de staat van de huidige code. Deze is het resultaat van verschillende projectgroepen die over de jaren heen verschillende delen hebben toegevoegd en bewerkt, waarbij de overdracht niet altijd vlekkeloos is gegaan. Hierdoor is het lastig om na te gaan welke delen van de code nog relevant zijn, en hoe goed deze delen werken. Ook zijn er geen geautomatiseerde tests geschreven. Dit alles brengt het risico met zich mee dat het uitbreiden van de huidige systemen moeizaam zou kunnen gaan.

5. Aanpak

Ik zal beginnen met het onderzoek doen naar 3D mapping, door het zoeken, lezen, en samenvatten van bestaande literatuur en artikelen. Hier zal ik een vergelijking over schrijven van de beschikbare mogelijkheden, systemen, en algoritmen. Vervolgens ga ik kennismaken met de bestaande systemen van de rolstoel, Robot Operating System, en andere gebruikte technologieën en sensoren. Hiervoor zal ik onder andere documentatie lezen, mijn eigen development-toolchain opzetten en testen, en hands-on werken met de rolstoel. Tijdens deze stappen, en op basis van de informatie die ik in deze eerste stappen heb verkregen, zal ik een plan van aanpak schrijven. Dit PvA wordt ook gebaseerd op mijn afstudeeropdracht en op gesprekken met de opdrachtgever en de lector.

Nadat de hiervoor genoemde analyses en het PvA af zijn, zal ik ontwerpen gaan maken voor de systemen die nodig zijn voor het gebruik van de 3D-camera en 3D mapping. Hiervoor zal ik eerst een globale systeemarchitectuur ontwerpen, waarin de integratie met de bestaande systemen wordt meegenomen. Vervolgens zal ik deze globale architectuur in delen ontwerpen, implementeren, en testen. Uit een eerste analyse blijkt dat deze globale systeemarchitectuur waarschijnlijk uit circa zes losse onderdelen zal bestaan. Voor elk onderdeel zal ik hiervoor een detailontwerp maken, en dit ontwerp implementeren. Hier hoort uiteraard ook het implementeren en uitvoeren van unit tests bij. Na de implementatiefase moeten de systemen getest worden. Hiervoor zal ik een testomgeving definiëren en het volledige systeem, inclusief integratie met de bestaande systemen, uitvoerig testen. Eventueel kan het blijken dat het handiger is om de fases ontwerpen–implementeren–testen in kleine iteratieve stappen uit te voeren, in plaats van volgens de hier genoemde watervalmethode.

Tot slot zal ik mijn werk afronden en overdragen. Dit zal bestaan uit het afronden van de documentatie en het geven van een demonstratie aan de opdrachtgever en de lector. Daarna zal ik mijn afstudeerdossier opbouwen, door alle eerder geproduceerde documenten (logboeken, verslagen, etc.) samen te voegen tot een samenhangend afstudeerdossier.

Hieronder wordt een overzicht gegeven van de geplande fases. In bijlage A is een Gantt-grafiek te zien die deze planning per week weergeeft.

Overzicht fases met tijdsaanduiding:

1. Onderzoek doen naar bestaande 3D mapping-systemen (5 dagen)
    a. Lezen van bestaande literatuur, vergelijking schrijven van beschikbare mogelijkheden, systemen, algoritmen, etc.

2. Kennismaken met en analyseren van navigatiesystemen rolstoel, Robot Operating System (ROS), en andere technologieën en sensoren. (5 dagen)
    a. Lezen van documentatie, opzetten en testen development-toolchain, hands-on werken met rolstoel.
3. Plan van aanpak schrijven (5 dagen)
    a. Dit plan wordt gebaseerd op deze lijst van uit te voeren werkzaamheden en overleg met de opdrachtgever en lector.
4. Ontwerpen en implementeren systeem 3D mapping / 3D-camera (40 dagen)
    a. Ontwerpen globale systeemarchitectuur, integratie met bestaande systemen. (4 dagen)
    b. Per onderdeel: Ontwerpen gedetailleerde architectuur, implementeren van ontworpen systeem, inclusief implementeren en uitvoeren van unit tests. (per onderdeel: 2 dagen ontwerpen, 3 dagen implementeren, 1 dag testen. 6 dagen per onderdeel x 6 onderdelen = 36 dagen)
5. Testen systeem 3D mapping / 3D-camera (10 dagen)
    a. Definiëren testomgeving, testen van volledige systeem, inclusief integratie met bestaande systemen.
6. Afronding en overdracht (5 dagen)
    a. Afronden documentatie
    b. Demonstratie geven aan opdrachtgever, bedrijfsmentor, etc.
7. Opbouwen afstudeerdossier (15 dagen)
    a. Samenvoegen eerder geproduceerde documenten, logboeken, verslagen etc. tot een samenhangend afstudeerdossier.

Bijlage: Gantt chart

					1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
	Fase	start	duur	eind																
1	Onderzoek	27-8	5	31-8																
2	Analyse	3-9	5	7-9																
3	PvA	10-9	5	14-9																
4	Ontw+Impl.	17-9	40	9-11																
6	Testen	12-11	10	23-11																
7	Afronding	26-11	5	30-11																
8	Afstudeerdossier	3-12	15	20-12																
			85																	

Bijlage B. Afstudeerplan

Afstudeerplan

Informatie afstudeerder en gastbedrijf

Afstudeerblok: 2018-2.1 (start uiterlijk 27 augustus 2018)
Startdatum uitvoering afstudeeropdracht: 27 augustus 2018
Inleverdatum afstudeerdossier volgens jaarrooster: 20 december 2018

Studentnummer: 14139227
Achternaam: dhr. Kokx
Voorletters: R
Roepnaam: Richard
Adres: Ernst Casimirstraat 35
Postcode: 2713BD
Woonplaats: Zoetermeer
Telefoonnummer:
Mobiel nummer: 06-24924340
Privé emailadres: richardkokx@gmail.com

Opleiding: Technische Informatica
Locatie: Delft
Variant: voltijd

Naam studieloopbaanbegeleider: Ron van Neijhof
Naam begeleidend examinator:
Naam expert examinator:

Naam bedrijf: De Haagse Hogeschool
Afdeling bedrijf: Lectoraat Smart Sensor Systems
Bezoekadres bedrijf: Rotterdamseweg 137
Postcode bezoekadres: 2628AL
Postbusnummer:
Postcode postbusnummer:
Plaats: Delft
Telefoon bedrijf: 015-2606200
Telefax bedrijf: 015-2606201
Internetsite bedrijf: www.dehaagsehogeschool.nl

Achternaam opdrachtgever: dhr. Bolte
Voorletters opdrachtgever: J. F. B.
Titulatuur opdrachtgever: PhD
Functie opdrachtgever: Professor Smart Sensor Systems
Doorkiesnummer opdrachtgever: 015-2606386
Email opdrachtgever: J.F.B.Bolte@hhs.nl

Achternaam bedrijfsmentor: dhr. Fraanje
Voorletters bedrijfsmentor: P.R.
Titulatuur bedrijfsmentor: Dr.ir.
Functie bedrijfsmentor: Hoofddocent
Doorkiesnummer bedrijfsmentor: 015-2606362
Email bedrijfsmentor: P.R.Fraanje@hhs.nl

Doorkiesnummer afstudeerder:
Functie afstudeerder (deeltijd/duaal):

Titel afstudeeropdracht:
Ontwikkelen van Semantic Mapping voor de Slimme Rolstoel

Opdrachtomschrijving

1. Bedrijf

Op de Haagse Hogeschool wordt naast het geven van onderwijs ook praktijkgericht onderzoek gedaan. Dit onderzoek wordt gedaan in lectoraten, die met hun onderzoek de verbinding leggen tussen het onderwijs en de beroepspraktijk. Een van deze lectoraten is Smart Sensor Systems. Dit lectoraat richt zich op het ontwerp en de ontwikkeling van slimme meetinstrumenten, sensoren, en meetnetwerken, en ook op het verwerken en terugkoppelen van metingen.

2. Probleemstelling

Het lectoraat Smart Sensor Systems doet momenteel onderzoek naar indoor navigatiesystemen voor mobiele robots. Een van de toepassingen van dit onderzoek is het ontwikkelen van een slimme rolstoel. Het doel hiervan is om met slimme navigatietechnieken de elektrische rolstoel veiliger en gebruiksvriendelijker te maken. Er is reeds een prototype slimme rolstoel ontwikkeld, die automatisch kan navigeren en routes kan plannen. De navigatie is echter nog niet nauwkeurig en sommige obstakels worden niet goed gezien, waardoor de rolstoel kan botsen op obstakels.

3. Doelstelling van de afstudeeropdracht

Het doel van de afstudeeropdracht is om de nauwkeurigheid en betrouwbaarheid van de navigatie van de slimme rolstoel te verbeteren, door gebruik te maken van beelden van de 3D-camera om vaste en verplaatsbare obstakels te herkennen. Deze obstakels moeten vervolgens in een *semantic map* geplaatst worden, waarmee de rolstoel zo moet kunnen navigeren dat de botsdetectie niet in werking hoeft te komen.

Er zijn ook nog drie optionele doelen: het uitbreiden van de verwerking van de beelden van de 3D-camera zodat deze ook mensen en dieren kan herkennen; het ontwikkelen van een gebruikersinterface voor de 3D-camera; of het optimaliseren van de al bestaande navigatiesystemen van de rolstoel. Als tijdens het afstuderen blijkt dat het primaire doel zeer eenvoudig te realiseren is, kan in overleg met de opdrachtgever en examinator een of meerdere van deze doelen toegevoegd worden aan de afstudeeropdracht. Dit besluit zal uiterlijk in de vierde week van het afstuderen genomen worden.

4. Resultaat

Het resultaat van de afstudeeropdracht is een beter werkende slimme rolstoel. Hiervoor worden de beelden van de 3D-camera geïnterpreteerd, en er wordt continu een semantische kaart gevormd van de omgeving. Deze kaart wordt vervolgens gebruikt in de navigatie-algoritmen, waardoor de navigatie nog nauwkeuriger wordt.

De algoritmen voor de interpretatie van de beelden, het vormen van de semantische kaart, en het toepassen van de kaart in de navigatiesystemen zullen, samen met alle overige gebruikte algoritmen, technieken en andere informatie, vastgelegd en uitgebreid gedocumenteerd worden, zodat deze ook gebruikt kunnen worden bij andere projecten met intelligente navigatiesystemen van het lectoraat Smart Sensor Systems.

5. Uit te voeren werkzaamheden, inclusief een globale fasering, mijlpalen en bijbehorende activiteiten

- Plan van aanpak schrijven (5 dagen)
  - In de eerste dagen van mijn stage zal ik een plan van aanpak schrijven. Dit plan wordt gebaseerd op deze lijst van uit te voeren werkzaamheden en overleg met de opdrachtgever en lector.
- Kennismaken met en analyseren van navigatiesystemen rolstoel, Robot Operating System (ROS), en andere technologieën en sensoren. (5 dagen)
  - Lezen van documentatie, opzetten en testen development-toolchain, hands-on werken met rolstoel.
- Onderzoek doen naar bestaande semantic mapping-systemen (5 dagen)

    - Lezen van bestaande literatuur, vergelijking schrijven van beschikbare mogelijkheden, systemen, algoritmen, etc.
- Ontwerpen systeem semantic mapping / 3D-camera (20 dagen)
    - Ontwerpen globale systeemarchitectuur, integratie met bestaande systemen, gedetailleerde architectuur.
- Implementeren systeem semantic mapping / 3D-camera (20 dagen)
    - Implementeren van ontworpen systeem, inclusief implementeren en uitvoeren van unit tests.
- Testen systeem semantic mapping / 3D-camera (10 dagen)
    - Definiëren testomgeving, testen van volledige systeem, inclusief integratie met bestaande systemen.
- Afronding en overdracht (5 dagen)
    - Afronden documentatie
    - Demonstratie geven aan opdrachtgever, bedrijfsmentor, etc.
- Opbouwen afstudeerdossier (15 dagen)
    - Samenvoegen eerder geproduceerde documenten, logboeken, verslagen etc. tot een samenhangend afstudeerdossier.

De stappen ontwerpen – implementeren – testen kunnen eventueel iteratief uitgevoerd worden in plaats van na elkaar.

6. Op te leveren (tussen)producten

- Plan van aanpak
    - Het plan van aanpak wordt opgeleverd zodra dit af is. Eventuele gewenste wijzigingen hiervan kunnen nog worden doorgevoerd.
- Onderzoeksverslag vergelijking semantic mapping-systemen
    - Verslag van welke semantic mapping-systemen er zijn, vergelijking tussen deze systemen, verantwoording voor keuze voor een bepaald systeem. Dit wordt opgeleverd na het onderzoek.
- Ontwerp systeem semantic mapping / 3D-camera
    - Het architectuurontwerp van het systeem rond de semantic mapping en de 3D-camera
- Implementatie systeem semantic mapping / 3D-camera
    - Code en documentatie van het systeem rond de semantic mapping en de 3D-camera
- Testverslag volledige systeem
    - Definitie testomgeving, testplan, resultaten uitgevoerde tests

De onderdelen ontwerp / implementatie / testverslag worden aan het einde van de afstudeeropdracht opgeleverd in een publiceerbaar formaat.

7. Te demonstreren competenties en wijze waarop

- G1: Praktische aspecten hanteren in (internationale) projecten:
    - Onder taak G1 valt onder andere het opstellen van een plan van aanpak, en het plannen van de eigen werkzaamheden. Om er voor te zorgen dat ik op schema blijf is het belangrijk dat ik in mijn plan van aanpak duidelijke deadlines voor mijzelf stel. Hiermee plan ik mijn werkzaamheden op zo'n manier in dat ik het maximale uit mijzelf kan halen.
- A1: Analyseren van het probleemdomein:
    - Om te bepalen welke semantic mapping-systemen beschikbaar en bruikbaar kunnen zijn, is het belangrijk de huidige literatuur goed te analyseren. Om ook ontwerpen te maken die daadwerkelijk zullen leiden tot een werkend systeem rond de 3D-camera, is het belangrijk om de randvoorwaarden en systeemeisen goed en duidelijk in beeld te brengen.
- C6: Ontwerpen van software
    - Om de individuele onderdelen van de software correct, testbaar en integreerbaar te implementeren, moeten deze onderdelen nauwkeurig en gedetailleerd ontworpen worden. Hierdoor zal het eenvoudig zijn om deze onderdelen te implementeren, te testen, en ze met elkaar te combineren tot een goed werkend eindproduct.
- C10: Ontwerpen van een systeemarchitectuur:
    - Om de te realiseren software efficiënt en zonder problemen te realiseren, is het belangrijk dat er een goede globale systeemarchitectuur wordt ontworpen. Deze globale

systeemarchitectuur zal het ook vereenvoudigen om de architectuur van subsystemen en onderdelen te ontwerpen en te implementeren.

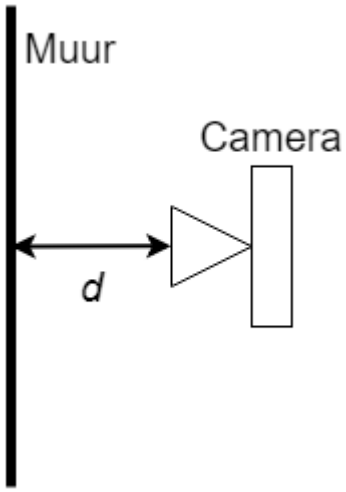
- D16: Het realiseren van software:
  - Bij het implementeren van het semantic mapping-systeem rond de 3D-camera, en om deze te integreren in het bestaande rolstoelsysteem, zal ik uiteindelijk een significante hoeveelheid code moeten schrijven.
- D17: Testen van softwaresystemen:
  - Om te bewijzen dat de 3D-camera toegevoegde waarde heeft, en dat de semantic mapping correct werkt, moeten de systemen uitvoerig getest worden.

Bijlage C. Testplan

1. Module A - camera

1. Testen diepte-resolutie:

- camera voor een blinde muur plaatsen, op minimale afstand dat de camera de diepte kan meten.
- Afstand meten tussen muur en camera
- Gemeten waarde uit camera noteren
- Camera naar achter verplaatsen, voor elk punt afstand tussen muur en camera meten en door camera waargenomen afstand noteren
- Punten: 0-100cm: elke 5 cm
- 100-250cm: elke 10 cm
- 250-500cm: elke 25 cm
- Indien >5,0m nog steeds enigszins betrouwbare waarden waargenomen worden: elke 0,5m.
- Indien >10m nog steeds enigszins betrouwbare waarden: elke 1,0m tot 20m.



Schematisch bovenaanzicht test diepte-resolutie

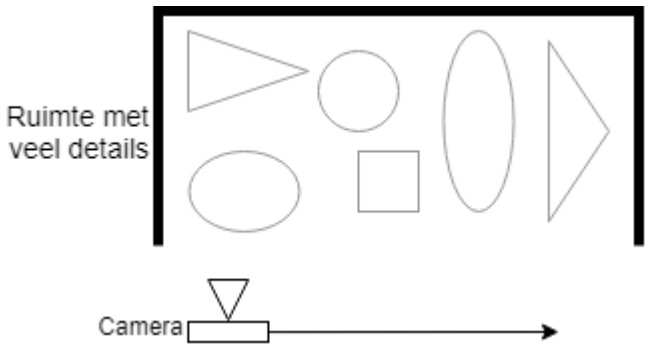
2. Testen betrouwbaarheid diepte-resolutie:

- camera voor een blinde muur plaatsen, op minimale afstand dat de camera de diepte kan meten.
- Afstand meten tussen muur en camera
- Gemeten waarde uit camera noteren
- IR-camera’s tijdelijk afdekken (door bijv. een vinger o.i.d.)
- Nieuwe meting doen
- Dit proces herhalen, in totaal 5 metingen doen op deze afstand
- Volledige experiment herhalen op afstanden 50cm, 1,0m, 2,0m, 5,0m, 10m, 20m.
- Op basis van metingen berekenen standaardafwijkingen etc.

2. Module C - RGBDSLAMv2

1. Testen nauwkeurigheid translatiemeting

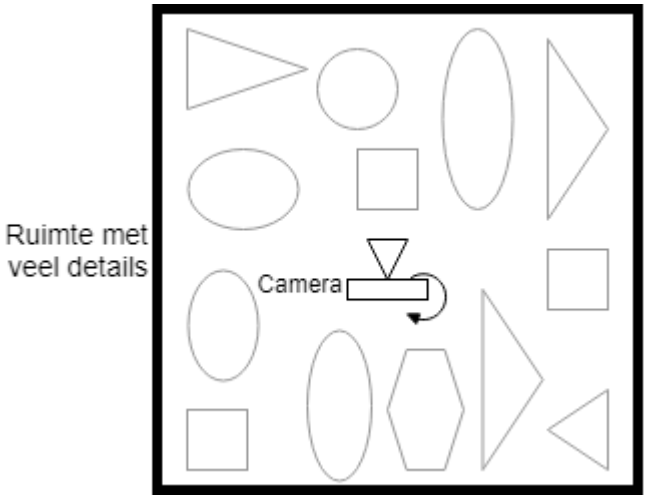
- Camera in een ruimte met veel details plaatsen
- Positie camera vastleggen - werkelijk en volgens RGBDSLAM
- Camera langzaam in één richting verplaatsen (ca. 0,1m/s), over een afstand van 1,00m
- Bepalen hoeveel de camera volgens RGBDSLAM heeft bewogen
- Deze meting nog twee keer doen, voor drie resultaten.
- Volledig experiment in totaal zes keer uitvoeren: links, rechts, vooruit, achteruit, boven, onder
- Volledig experiment ook uitvoeren met een snellere beweging (ca. 0,5m/s) over een afstand van 5,0m



Schematisch bovenaanzicht test translatie

2. Testen nauwkeurigheid rotatiemeting

- Camera in een ruimte met veel details plaatsen
- Rotatie camera vastleggen - werkelijk en volgens RGBDSLAM
- Camera langzaam over één as draaien (ca. pi/4 rad/s - 8 seconden per volledige omwenteling) over een rotatie van pi rad (halve omwenteling, dus in 4 seconden)
- Bepalen hoeveel de camera volgens RGBDSLAM heeft gedraaid
- Deze meting nog twee keer doen, voor drie resultaten
- Volledig experiment in totaal zes keer uitvoeren: in beide richtingen voor yaw, pitch, en roll.
- Volledig experiment ook uitvoeren met een rotatiesnelheid van ca. pi/2 rad/s over een rotatie van 2pi rad (dus hele omwenteling in 4 seconden). Hierbij opletten dat de omgeving niet veranderd, bus bijv. zorgen dat degene die het experiment uitvoert uit beeld blijft.



Schematisch bovenaanzicht test rotatie

3. Module D - Voxel map

- Eenvoudige testopstelling maken met goed meetbare afmetingen
- Voxel map maken van deze testopstelling
- Vergelijken daadwerkelijke afmetingen met afmetingen in voxel map

Bijlage D. Code

1. Launchfile

```
1  <launch>
2   <env name="ROSCONSOLE_CONFIG_FILE" value="/home/richard/catkin_ws/rosconsole.conf" />
3   <remap from="/rgbdslam/batch_clouds" to="/cloud_in" />
4   <include file="$(find realsense_camera)/launch/zr300_nodelet_rolstoel.launch" />
5   <include file="$(find rgbdslam)/launch/rgbdslam_rolstoel.launch" />
6   <node name="caller" pkg="caller" type="caller.sh" />
7   <include file="$(find octomap_server)/launch/octomap_mapping.launch" />
8   <node type="rviz" name="rviz" pkg="rviz" />
9  </launch>
```

Code 2: Launchfile

2. A-tester.py

```
1  #!/usr/bin/env python
2  import rospy
3  import cv2
4  from sensor_msgs.msg import Image
5  from cv_bridge import CvBridge, CvBridgeError
6
7
8  class middle_value_distance:
9
10     def __init__(self):
11         self.bridge = CvBridge()
12         self.image_sub = rospy.Subscriber('/camera/depth/image_raw',
13                                           Image,
14                                           self.callback)
15         rospy.loginfo("Listening to /camera/depth/image_raw")
```

```
16
17      def callback(self, data):
18          rospy.logdebug(rospy.get_caller_id() +
19                         " Received Image with time: %s\nencoding:\n%s",
20                         data.header.stamp,
21                         data.encoding)
22          try:
23              cv_image = self.bridge.imgmsg_to_cv2(data, "16UC1")
24          except CvBridgeError as e:
25              rospy.logerr(e)
26
27          (rows, cols) = cv_image.shape
28          rospy.logdebug(rospy.get_caller_id() +
29                         " Rows: %d\tCols: %d", rows, cols)
30
31          rospy.loginfo("d: %d", cv_image[rows/2, cols/2])
32
33
34  def main():
35      middle_value_distance()
36      rospy.init_node('tester', anonymous=True)
37      rospy.loginfo("Initialized tester")
38      try:
39          rospy.spin()
40      except KeyboardInterrupt:
41          rospy.logerr("Shutting down")
42      cv2.destroyAllWindows()
43
44
45  if __name__ == '__main__':
46      main()
```

Code 3: a-tester.py

3. C-tester.py

```
1  #!/usr/bin/env python
2  import rospy
3  import tf
4  from time import sleep
5
6
7   class transform_rotate:
8
9       def __init__(self):
10          self.tf = tf.TransformListener()
11          rospy.loginfo("Listening to /camera/depth/image_raw")
12
13      def run(self):
14          if (self.tf.frameExists("camera_rgb_optical_frame") and
15                  self.tf.frameExists("map")):
16              rospy.logdebug("Both frames exist")
17              t = self.tf.getLatestCommonTime("/camera_rgb_optical_frame",
18                                              "/map")
19              rospy.loginfo("t = %s", t)
20              pos, quat = self.tf.lookupTransform("/camera_rgb_optical_frame",
21                                                  "/map", t)
22              (roll, pitch, yaw) = tf.transformations.euler_from_quaternion(quat)
23  #             rospy.logdebug("pos: %s\teuler: %s", pos, eul)
24              rospy.loginfo("x: %.5f\ty: %.5f\tz: %.5f", pos[0], pos[1], pos[2])
25              rospy.loginfo("r: %.5f\tp: %.5f\ty: %.5f", roll, pitch, yaw)
26          else:
27              rospy.logerr("One of the frames doesn't exist.")
28
29
30   def main():
31      rospy.init_node('tester', anonymous=True)
32      rospy.loginfo("Initialized tester")
33      tr = transform_rotate()
34      while True:
```

```
35          tr.run()
36          sleep(0.05)
37
38
39  if __name__ == '__main__':
40      main()
```

Code 4: c-tester.py

4. Caller.sh

```
1  #!/bin/bash
2  while :
3  do
4          rosservice call /rgbdslam/ros_ui send_all
5          sleep 1
6  done
```

Code 5: Caller.sh

5. Graph_mgr_io.cpp patch

```
1  diff --git a/src/graph_mgr_io.cpp b/src/graph_mgr_io.cpp
2  index 8f53f12..2516e95 100644
3  --- a/src/graph_mgr_io.cpp
4  +++ b/src/graph_mgr_io.cpp
5  @@ -179,6 +179,7 @@ void GraphManager::sendAllCloudsImpl()
6      tf::StampedTransform base_to_fixed = this->computeFixedToBaseTransform(node, true);
7      br_.sendTransform(base_to_fixed);
8      publishCloud(node, base_to_fixed.stamp_, batch_cloud_pub_);
9  +   node->clearPointCloud(); // per https://answers.ros.org/question/33909/
10
11     QString message;
12     Q_EMIT setGUIInfo(message.sprintf("Sending pointcloud and map transform (%i/%i) on topics %s and
   /tf", it->first, (int)graph_.size(), ParameterServer::instance()-
   >get<std::string>("individual_cloud_out_topic").c_str()) );
```

Code 6: Patchfile voor graph_mgr_io.cpp in RGBDSLAM

Bijlage E. Meetresultaten

1. Diepteresolutie camera

Afstand	Meting	Verschil	Afstand	Meting	Verschil
610	630	20	1900	1903	3
650	664	14	2000	2026	26
700	712	12	2100	2130	30
750	767	17	2200	2234	34
800	812	12	2300	2339	39
850	860	10	2400	2471	71
900	912	12	2500	2580	80
950	960	10	2750	2773	23
1000	1013	13	3000	3070	70
1100	1106	6	3250	3325	75
1200	1207	7	3500	3640	140
1300	1317	17	3750	3988	238
1400	1426	26	4000	4241	241
1500	1512	12	4250	4550	300
1600	1614	14	4500	4835	335
1700	1720	20	4750	5107	357
1800	1820	20	5000	5561	561

Tabel 5: Resultaten meting diepteresolutie camera.
Alle waarden in millimeters.

2. Betrouwbaarheid diepteresolutie camera

Afstand	610	1000	2000	5000
Meting 1	627	1003	2022	5592
Meting 2	626	1005	2006	5561
Meting 3	628	1002	2010	5720
Meting 4	627	1003	2022	5500
Meting 5	628	1003	2010	5687
Gemiddelde	627.2	1003.2	2014	5612
Verschil	17.2	3.2	14	612
Std. dev.	0.837	1.095	7.483	90.60

Tabel 6: Resultaten betrouwbaarheidsmeting.
Alle waarden in millimeters.

3. Betrouwbaarheid translatiemeting

Richting	Links	Rechts	Voor	Achter	Boven	Onder
Afstand	1.00	1.00	1.00	1.00	1.00	1.00
Meting 1	1.01625	-1.04351	-1.09126	0.98700	1.02320	-1.14157
Meting 2	1.01566	-0.97377	-1.00588	1.01624	1.01768	-1.06651
Meting 3	1.04950	-1.02456	-1.00866	1.04363	1.01764	-0.94989
Gemiddelde	1.02714	1.01395	1.03527	1.01562	1.01951	1.05266
Gem. afwijking	0.02714	0.01395	0.03527	0.01562	0.01951	0.05266
Std.dev.	0.01937	0.03606	0.04851	0.02832	0.00320	0.09659

Tabel 7: Resultaten betrouwbaarheidsmeting translatie. Alle waarden in meters.

4. Betrouwbaarheid rotatiemeting

Draairichting	Yaw links	Yaw rechts	Pitch omhoog	Pitch omlaag	Roll links	Roll rechts
Hoek	3.14159	3.14159	1.57080	1.57080	3.14159	3.14159
Meting 1	3.13949	-3.12048	-1.44540	1.37058	-3.13874	3.11083
Meting 2	3.06450	-3.11497	-1.62473	1.53064	-3.10461	3.13503
Meting 3	3.10734	-3.10530	-1.51415	1.59249	-3.13439	3.13488
Gemiddelde	3.10378	3.11358	1.52809	1.49790	3.12591	3.12691
Gem. afwijking	0.03782	0.02801	0.04270	0.07289	0.01568	0.01468
Std.dev.	0.03762	0.00768	0.09047	0.11452	0.01858	0.01393

Tabel 8: Resultaten betrouwbaarheidsmeting rotatie. Alle waarden in radialen.
Yaw is rotatie rond de verticale as.
Pitch is rotatie rond de horizontale as van links naar rechts.
Roll is rotatie rond de horizontale as in de kijkrichting van de camera.