2020•2021
Faculteit Industriële ingenieurswetenschappen
master in de industriële wetenschappen: energie

Masterthesis

Studie naar autonoom transport van rolstoelgebruikers in zorgcentrum Sint Oda: concept, kaartopbouw, lokalisatie en taaksequentiëring

PROMOTOR :
Prof. dr. ir. Eric DEMEESTER

PROMOTOR :
Mevr. Leen HULSHAGEN

BEGELEIDER :
ing. Peter AERTS

BEGELEIDER :
Dhr. Kim SCHEEPERS

Hendrik Clijsters, Joris Ohmstede

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie, afstudeerrichting automatisering

Gezamenlijke opleiding UHasselt en KU Leuven

2020•2021

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: energie

Masterthesis

Studie naar autonoom transport van rolstoelgebruikers in zorgcentrum Sint Oda: concept, kaartopbouw, lokalisatie en taaksequentiëring

PROMOTOR :

Prof. dr. ir. Eric DEMEESTER

PROMOTOR :

Mevr. Leen HULSHAGEN

BEGELEIDER :

ing. Peter AERTS

BEGELEIDER :

Dhr. Kim SCHEEPERS

Hendrik Clijsters, Joris Ohmstede

Scriptie ingediend tot het behalen van de graad van master in de industriële wetenschappen: energie, afstudeerrichting automatisering

UHASSELT KU LEUVEN

Voorwoord

Deze masterproef is geschreven ter afsluiting van de opleiding Industriële Ingenieurswetenschappen met afstudeerrichting Energie-Automatisering gezamenlijk gegeven door de Katholieke Universiteit Leuven en Universiteit Hasselt te Diepenbeek.

Het onderwerp “Studie voor integreren van autonoom transport van zorgbehoevenden in zorgcentrum Sint Oda” is door ons gekozen omwille van de maatschappelijke relevantie. Wij wilden graag ons steentje bijdragen in de integratie van technologie in de zorgsector. Ook het technologische aspect rond AGV’s en ROS was een grote motivator.

Gedurende dit onderzoek hebben wij veel bijgeleerd over de huidige ontwikkelingen van de technologie omtrent AGV’s, ROS en navigatiesensoren. Er was ook een sterke motivatie gedurende het hele project. Dit dankzij een goede balans van theoretische studies en praktische werkzaamheden.

Graag zouden wij onze interne promotoren: Prof. dr. ir. Eric Demeester, ing. Peter Aerts en onze externe promotoren: mevr. Leen Hulshagen en mr. Kim Scheepers willen bedanken voor alle hulp gedurende het project. Ook bedanken we graag al het personeel bij ACRO die het voor ons mogelijk hebben gemaakt om tijdens deze Covid-19 pandemie verder te werken op het onderzoekscentrum. Ten slotte willen wij dr. Jeroen Lievens bedanken voor de begeleiding in het academisch schrijven van onze scriptie.

Ondanks de maatregelen tegen deze pandemie hebben we de samenwerkingen met iedereen als zeer vlot en aangenaam ervaren.

Hendrik Clijsters en Joris Ohmstede

22/01/2021

Diepenbeek

Inhoudsopgave

Lijst van tabellen

Lijst van figuren

Verklarende woordenlijst

AGV	*Automated guided vehicle*
BT	*Behavior tree*
DWA	*Dynamic Window Approach*
EKF	*Extended Kalman Filter*
FSM	*Finite state machine*
GPS	*Global Positioning System*
HMI	*Human machine interface*
IMU	*Inertial measurement unit*
LIDAR	*Light detecting And ranging*
SLAM	*Simultaneous localization and mapping*
ROS	*Robot Operating System*
RTK	*Real Time Kinematic*
ToF	*Time-of-Flight*
VPN	*Virtual private network*
WMS	*Warehouse management system*

Abstract

Sint Oda te Pelt is een zorgcentrum voor mensen met een handicap. De zorgbehoevenden in een rolstoel dienen dagelijks naar het activiteitencentrum gebracht te worden. Maar Sint Oda beschikt over onvoldoende personeel per leefgroep om alle zorgbehoevenden in één keer te vervoeren. Bovendien is dit fysiek een zware taak voor het personeel. Sint Oda wil de personeelsbelasting reduceren door de inzet van een autonoom voertuig. Deze masterproef heeft drie doelen: ten eerste, het opsommen en vergelijken van mogelijke bestaande oplossingen; ten tweede, het evalueren van outdoor kaartopbouw en lokalisatie; en ten slotte, de taaksequentiëring van het autonoom voertuig opstellen.

Het vergelijken van oplossingen is uitgevoerd met een “van den Kroonenberg” analyse. Hieruit blijkt dat een autonoom gemaakt voertuig met LIDAR, stereo camera, GPS-RTK en veiligheidssensoren een haalbare oplossing is.

Om de kaartopbouw en lokalisatie te evalueren is een sensorplatform opgebouwd en uitgelezen m.b.v. ROS. Hiermee zijn in-/outdoor testen uitgevoerd. De outdoor testen zijn rondom het onderzoekscentrum uitgevoerd omwille van de Covid-19 pandemie. Door ongeschikte sensoren bleek het platform onbruikbaar in outdoor omstandigheden. De gebruikte algoritmes in ROS zijn wel veelbelovend voor kaartopbouw en lokalisatie.

Voor de taaksequentiëring zijn hulpmiddelen geëvalueerd a.d.h.v. de literatuur en is besloten om met een behavior tree te werken. Deze is opgesteld maar vanwege een beperkt tijdskader is de onderliggende software niet geïmplementeerd.

Abstract in English

Sint Oda is a care centre for disabled people located in Pelt. The patients in a wheelchair need to be taken to the activity center daily. But Sint Oda does not have enough staff to transport all patients in one go. Also this is a physically tiresome task. Sint Oda wants to relieve the staff by implementing an autonomous vehicle. This thesis consists of three main goals. Firstly an assessment of all possible solutions, secondly the evaluation of outdoor mapping and localization and lastly the task sequencing of the autonomous vehicle.

The assessment of all possible solutions was done using the van den Kroonenberg method. The assessment concluded that an autonomous vehicle using a LIDAR, camera, GPS-RTK and safety sensors is the most feasible solution.

To evaluate the mapping and localization a sensor platform was constructed. This platform was tested in in-/outdoor environments. Due to the Covid-19 pandemic outdoor testing was done around the research center. Due to unsuitable sensors the sensor platform is not suited for outdoor use but the algorithms used in ROS are promising for outdoor mapping and localization.

After analyzing tools for task sequencing the choice was made to use a “Behavior Tree”. The behavior tree was made but never implemented in the software due to time restrictions.

1 Inleiding

1.1 Situering

Sint Oda is een dienstencentrum voor mensen met een ernstige meervoudige en matige- tot diepverstandelijke beperking. Het hoofdcentrum, gelegen in Pelt, telt ongeveer 300 zorgbehoevenden [1]. Dit centrum beschrijft een oppervlakte van 17 hectare groot en is opgedeeld in meerdere leefgroepen. Elke leefgroep bestaat uit minstens acht zorgbehoevenden en minstens twee begeleiders. In het midden van het centrum bevindt zich het activiteitencentrum Sens-City. Hier kunnen de bewoners leuke ervaringen opdoen in onder andere een ballenbad, snoezelruimte en zwembad. De bewoners, die vaak rolstoelgebruikers zijn, pendelen dagelijks tussen hun leefgroep en Sint Oda.

Momenteel gebeurt dit transport door één begeleider die één of twee rolstoelgebruikers kan voortbewegen. Figuur 1-1 toont hiervan een huidig gebruikte methode. In deze methode zijn twee rolstoelen aan elkaar verbonden door de handvaten van de voorste rolstoel aan de wielas van de achterste te bevestigen met behulp van stangen.

Figuur 1-1: Een van de methoden van transport

1.2 Probleemstelling

Momenteel zijn er te weinig begeleiders om een hele leefgroep in één keer naar Sens-City te brengen. Door de grootte van het terrein bevinden sommige leefgroepen zich ook op aanzienlijke afstand van Sens-City. Het gevolg is dat de begeleiders meermaals dezelfde lange route moeten wandelen, zeker met twee rolstoelgebruikers is dit een zeer belastende taak. Hierdoor zijn de faciliteiten van Sens-City vaak niet optimaal benut.

Het zorgcentrum beschikt over drie handmatig bestuurbare elektrische wagentjes die tot nu toe louter een logistieke functie hebben. Figuur 1-2 toont een van deze wagentjes, in een vorig leven was de functie van dit wagentje het vervoeren van bagage op het vliegveld van Zaventem. Een logische oplossing zou het gebruik van één van de elektrische wagentjes zijn waaraan een kar voor rolstoelgebruikers bevestigd is. Het inhuren of opleiden van extra chauffeurs is echter niet gewenst, omdat deze personen altijd beschikbaar moeten zijn. Ook moet Sint Oda rekening houden met mogelijke menselijke fouten. Om deze redenen wilt Sint Oda weten welke autonome oplossingen er zijn alsook wat nodig is om deze oplossingen te implementeren.

Figuur 1-2: Elektrische wagens

1.3 Doelstellingen

Het hoofddoel van de gegeven opdracht is het implementeren van een veilige manier om meerdere bewoners in rolstoel tegelijkertijd van de leefgroep naar het activiteitencentrum te verplaatsen. Dit doel is behaald wanneer de oplossing autonoom vier of meer bewoners kan vervoeren tussen de verschillende haltes. Dit vervoer moet vlot en veilig verlopen over de verschillende wegen en ondergronden, namelijk asfalt en beklinkering, maar ook bij verschillende weersomstandigheden, zoals beperkt licht, lichte mist en regenval. Het systeem moet functioneren met een begeleider die instaat voor het aan- en afkoppelen van de rolstoelen. De begeleider moet ook kunnen aanduiden aan welke haltes de oplossing moet stoppen en moet een noodstop ter beschikking hebben.

Deze masterproef bestaat slechts uit 20 studiepunten. Dit in combinatie met de beperkte middelen maakt het onmogelijk voor ons om het hoofddoel tot een succesvol eind te brengen. Deze masterproef zal dus dienen als fundering waarop andere masterproeven kunnen verder bouwen. De focus van deze masterproef ligt op een studie van de verschillende onderdelen die nodig zijn om een autonoom voertuig in Sint Oda te integreren. Er zijn hiervoor de volgende doelstellingen gemaakt:

- een conceptuele studie met als doel het vergelijken van de mogelijke oplossingen voor het bekomen van het hoofddoel;
- data verzamelen van het centrum bij wisselende weersomstandigheden om kwaliteit van lokalisatie en het bouwen van outdoor kaarten te bepalen;
- een taaksequentiëring opstellen om het dagelijks gedrag van het autonoom voertuig te implementeren.

Om deze doelstellingen tot een goed eind te brengen zijn er verschillende gesprekken met Sint Oda gehouden. Op basis hiervan is er een lijst met eisen opgesteld waarvan de belangrijkste hieronder te zien zijn (zie Tabel 1-1). Het volledige eisenpakket is in deel 5.2 terug te vinden.

Tabel 1-1: Belangrijkste eisen

EISEN	Beschrijving
Rolstoel capaciteit	De oplossing moet minstens 4 rolstoelen tegelijkertijd verplaatsen van punt A naar B
Wegbreedte	De oplossing moet over wegen van minimaal twee meter breed kunnen rijden
Opstapmogelijkheden	Er moeten tussen de 20 en 30 “haltes” voorzien zijn waar de oplossing de bewoners kan ophalen
Belasting begeleider	Fysieke belasting moet niet zwaarder zijn dan nu en er moet een reductie van belastingstijd zijn van minimaal 30%
Gemak zorgbehoevenden	Comfort, veiligheid en aantal handelingen van de zorgbehoevenden mogen niet verslechteren ten opzichte van de vorige situatie
Veiligheidsnormen	Het aantal veiligheidsnormen die Sint Oda moet implementeren zodat het een CE-keuring kan verkrijgen

1.4 Materiaal en methode

Een vergelijking van de verschillende oplossingen volgt in *hoofdstuk 5 Conceptuele studie*. Een literatuurstudie geeft hiervoor de nodige achtergrondinformatie. Deze literatuurstudie is opgesplitst in drie hoofdstukken. *Hoofdstuk 2 Transportmogelijkheden* gaat over welke voertuigen in aanmerking komen voor het autonoom vervoer. *Hoofdstuk 3 Navigatie-algoritmen en -hardware* bekijkt welke navigatie-algoritmen autonome voertuigen toepassen en welke sensoren bij deze voorkomen. *Hoofdstuk 4 Systeeminfrastructuur* bespreekt hoe de communicatie tussen de verschillende onderdelen van Sint Oda kan verlopen. De literatuurstudie in combinatie met de vereisten van Sint Oda heeft ons toegelaten om, aan de hand van de methode van den Kroonenberg [2], verschillende oplossingen te vergelijken. Er is, zeker in het begin, een wekelijkse vergadering met de interne promotor en begeleider gehouden. Zij hebben namelijk de expertise om te zien of de conceptuele studie de juiste richting uitging en of de gemaakte keuzes verantwoord waren. De terugkoppeling met Sint Oda in dit proces was zeer belangrijk. Enerzijds om de bijkomende vragen te beantwoorden en anderzijds om miscommunicatie tegen te gaan.

De bespreking van de verzamelde data gebeurt in *hoofdstuk 6 Evaluatie outdoor kaartopbouw en lokalisatie*. Het maken en testen van een sensorplatform is eerst gebeurt, dit op het onderzoekcentrum ACRO. Het platform bestaat uit verschillende etages waarop de gebruikte sensoren bevestigd zijn (zie Figuur 6-1). De gebruikte sensoren zijn een camera, 2D LIDAR en inertail measurement unit (IMU) met GPS-RTK. Hiernaast is er ook een MOXA MC-7000 computer gebruikt om de gegevens te verwerken en op te slaan. Omwille van het beperkte budget van Sint Oda is alle gebruikte hardware van de onderzoeksgroep ACRO afkomstig. Om de gegevens op een correcte manier te verwerken zijn ook navigatie-algoritmen nodig. Een bespreking van deze algoritmen gebeurt in *hoofdstuk 3 Navigatie-algoritmen en -hardware*. Na het afronden van de testen zou de opstelling op één van de elektrische wagentjes van Sint Oda komen te staan. Door met dit voertuig rond te rijden is het mogelijk om datasets op te stellen. Omwille van onder andere de Covid-19 pandemie is dit niet gebeurt. In plaats hiervan zijn datasets opgesteld in en rond de ACRO om de kwaliteit van lokalisatie en het opbouwen van outdoor kaarten te evalueren.

De laatste doelstelling is het opstellen van een taaksequentiëring om het dagelijks gedrag van het voertuig te implementeren. Dit gebeurt in *hoofdstuk 7 Taaksequentiëring*. In de eerste stap is een dagoverzicht gemaakt in samenwerking met Sint Oda. Dit dagoverzicht laat zien wanneer het autonoom voertuig wat moet doen. De tweede stap was het maken van een flowchart op basis van dit dagoverzicht. Deze beschrijft de samenhang tussen de verschillende activiteiten van het voertuig. Tenslotte is in de derde stap deze flowchart gebruikt om de taaksequentiëring op te stellen die het voertuig zal sturen. De complexiteit van het gedrag stijgt snel omwille van de grote if-else structuren die ontstaan. Daarom gebeurt er in *hoofdstuk 4 Systeeminfrastructuur* een onderzoek naar gedragsalgoritmen in ROS die deze derde stap kunnen vereenvoudigen.

2 Transportmogelijkheden

2.1 Automated Guided Vehicle

Een eerste mogelijke oplossing is een Automated Guided Vehicle (AGV). Een AGV is een robot die zich autonoom kan verplaatsen met als doel het transport van goederen of personen [3]. Deze beschrijving is zeer algemeen. De oorzaak hiervoor is dat er veel verschillende soorten AGV's bestaan die op verschillende manieren werken. De volgende secties zullen enkele soorten AGV's beschrijven, hoe ze zich in de omgeving zullen navigeren, de verschillende manieren om een veilige werking te garanderen. Tenslotte volgt een analyse van de haalbaarheid voor Sint Oda.

2.1.1 Soorten AGV

Er zijn veel verschillende soorten AGV's, maar niet alle soorten zijn geschikt voor het vervoer van personen. De focus van deze masterproef ligt op personenvervoer, om echter een zo volledig mogelijk beeld van de AGV te geven komen ook de types voor goederentransport aan bod.

Tow vehicle

Een tow vehicle kan één of meerdere karren transporteren (zie Figuur 2-1). Dit stelt hem in staat om grote volumes te verplaatsen. Er zijn twee modellen tow vehicle, degenen die enkel autonoom kunnen rijden en degenen die zowel autonoom als handgestuurd kunnen rijden.

Figuur 2-1: Industriële tow vehicle [4]

Unit load carrier

De unit load carrier draagt de last rechtstreeks op zichzelf. Bij goederenvervoer zijn deze vaak met een transportband uitgerust zodat ze de last automatisch kunnen laden en lossen [5]. Dit geeft een voordeel ten opzichte van een tow vehicle waarbij dit nog manueel moet gebeuren. Een nadeel ten opzichte van een tow vehicle is dat deze minder grote volumes kan verplaatsen. Figuur 2-2 toont een unit load carrier voor goederentransport. Figuur 2-3 toont een Unit load carrier voor personenvervoer gebruikt in de luchthaven van Heathrow [6].

Figuur 2-2: Unit load carrier voor goederenvervoer [4]

Figuur 2-3: Unit load carrier voor personenvervoer [6]

Vorkheftruk AGV

De vorkheftruk AGV kan autonoom paletten optillen en verplaatsen (zie Figuur 2-4). Het laden en lossen gebeurt met een vork, dit in tegenstelling tot de unit load carrier die gebruikt maakt van een transportband. Er bestaat een groot aantal subtypes vorkheftruk AGV zoals de “Narrow aisle” voor smalle gangen en “counterbalanced” voor zwaardere lasten [5].

Figuur 2-4: Forklift AGV [4]

Automated guided carts

De automated guided cart is een kleiner model van AGV. Deze positioneert zichzelf onder een rek met goederen. Indien het rek wielen heeft bevestigd de kar zichzelf aan het rek om deze te kunnen transporteren (zie Figuur 2-5) indien er geen wielen zijn kan de kar het rek optillen.

Figuur 2-5: Automated guided cart [7]

Uit de beschrijvingen van deze verschillende soorten AGV's valt af te leiden dat enkel tow vehicles en unit load carriers in staat zijn om mensen te vervoeren.

2.1.2 Navigatiemethoden AGV

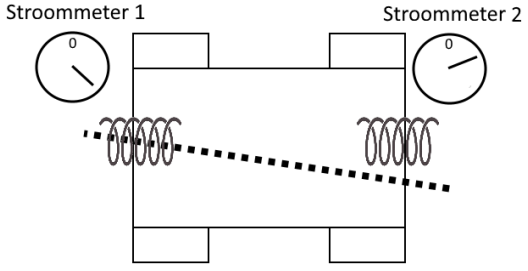
Onafhankelijk van het type AGV zal deze zich in zijn omgeving moeten navigeren. Hiervoor moet de AGV eerst weten wat zijn positie en oriëntatie in deze omgeving zijn. Een AGV gebruikt hiervoor twee groepen sensoren. De eerste groep bestaat uit sensoren die de toestand van het voertuig over de tijd opmeten, deze krijgen de naam proprioceptieve sensoren. Denk hierbij aan een encoder die de stand van de wielen meet of een IMU die de versnelling en oriëntatie bepaalt. Iedere fabrikant zal een aantal van dit soort sensoren gebruiken, maar dit is niet genoeg. De metingen van de proprioceptieve sensoren zijn namelijk niet foutloos, deze fouten zullen met de tijd op elkaar stapelen waardoor de verwachte positie niet meer met de werkelijke overeenkomt. Meer uitleg over de werking van de encoder en IMU volgt in deel 3.4.1.

Om deze reden gebruikt elke fabrikant ook een tweede groep sensoren. Deze zullen aan de hand van elementen in de omgeving de positie en oriëntatie van de AGV bepalen, enkele voorbeelden hiervan zijn LIDAR, camera's en GPS. Meer informatie over deze sensoren gebeurt respectievelijk in deel 3.4.2, 3.4.3 en 3.4.4.

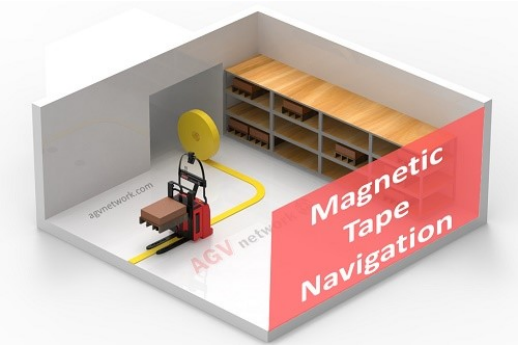
Ook enkel sensoren uit deze tweede groep gebruiken valt sterk af te raden. De metingen houden namelijk geen rekening met de voorgaande toestand van de AGV, hierdoor kan de verwachte positie van de AGV door een foutieve meting sterk verspringen ten opzichte van de werkelijke. Afhankelijk van de gekozen navigatiemethode zullen de benodigde sensoren van de tweede groep veranderen. De rest van dit deel gaat dieper in op de meest toegepaste methoden en welke sensoren bij deze methoden horen.

Lint navigatie

Bij lint navigatie zal een AGV een vooropgestelde lijn volgen om zich te navigeren. Dit kan op drie manieren gebeuren, actief inductief, passief inductief of visueel. Bij de actieve methode is het lint een draad die onder het wegdek ligt, door deze draad vloeit een wisselstroom. Deze wisselstroom induceert een stroom in twee spoelen gemonteerd op de voor- en achterkant van de AGV. Het verschil tussen de twee geïnduceerde stromen geeft aan hoe goed de AGV de draad volgt [3]. Indien het voertuig afwijkt zal de voorste spoel (2) minder overlap hebben met de draad en dus ook minder stroom induceren (zie Figuur 2-6). De passieve methode zal gebruik maken van een magnetische strip of schijven als lint (zie Figuur 2-7), dit lint is op of onder het wegdek aangebracht [3]. Er zijn enkele magnetische sensoren op de AGV bevestigd. Deze zullen, gelijkaardig aan de actieve methode, hun meetwaarden vergelijken om te zien of de AGV het lint nog volgt. De laatste methode is de visuele methode, het lint is hierbij verf die over het wegdek is aangebracht. Een camera gemonteerd op het voertuig zal controleren dat de AGV de streep blijft volgen [3]. Voor een goede werking is een kleur die duidelijk verschilt van het wegdek noodzakelijk.



Figuur 2-6: Afwijken actieve lint navigatie



Figuur 2-7: Passieve lint navigatie [8]

De voordelen van deze methoden zijn hun grote betrouwbaarheid en nauwkeurigheid [9], de benodigde hardware en software is ook vrij simplistisch wat de ontwikkelingskost van de AGV zelf zal verlagen. Er zijn echter belangrijke nadelen. De AGV kan enkel vaste paden volgen, als er een uitbreiding of verandering van het pad noodzakelijk is zal ook de positie van het lint moeten veranderen. Dit zorgt voor een grote uitdaging bij de inductieve methoden waar de draad of magneten onder het wegdek liggen. Hiernaast is er ook de grote installatiekost voor het plaatsen van het lint over het hele terrein. Verder zijn de passieve en visuele methode die op het wegdek aangebracht zijn onderhevig aan slijtage. Tenslotte is het buiten gebruiken van de visuele methode nog eens extra moeilijk omdat de camerabeelden afhankelijk zijn van de weersomstandigheden zoals regen en lichtinval.

Stip navigatie

Deze methode is gelijkaardig aan de passieve lint methode. Bij deze aanpak ligt er geen volledige strip op of onder het wegdek, maar zijn er magnetische schijven op het wegdek geplaatst. Elke magnetische schijf kan hierbij beschouwd worden als een stip. De AGV zal voornamelijk op basis van zijn proprioceptieve sensoren rijden, de stippen dienen om de opbouw van dead reckoning fouten te vermijden. Wanneer een AGV erover rijdt is hij terug zeker van zijn locatie. De plaatsing van de stippen kan in een lijn, zoals te zien is in Figuur 2-8, of in een rooster [9].

Figuur 2-8: Stip navigatie geplaatst in een lijn [8]

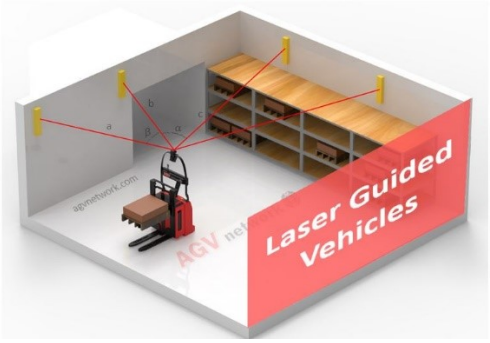
De voordelen van de methode zijn in het algemeen hetzelfde als die van lint navigatie. De nadelen zijn ook gelijkaardig aan lint navigatie maar minder uitgesproken omdat het nu niet nodig is om een volledige lijn te leggen. De installatietijd is wel groter, dit omdat de afstand van de stippen, ten opzichte van elkaar, nauwkeurig genoeg geweten moet zijn om een terug een goede lokalisatie te bekomen [5], [9]. Het gebruiken van een rooster zal de installatiekost en -tijd verder verhogen, maar zal wel voor een grotere flexibiliteit zorgen bij het bepalen van een route.

Baken navigatie

Deze methode is afhankelijk van trilateratie. Trilateratie is het bepalen van de coördinaten van een onbekend punt (mobiel baken) door gebruik te maken van minstens drie afstanden tot minimaal drie gekende punten (stationaire bakens). Het bepalen van de afstand gebeurd door een signaal uit te zenden en te ontvangen. Hoelang het signaal hierover doet, gecombineerd met de snelheid van het signaal, bepaalt de afstand. Een verdere toelichting gebeurt in deel 3.4.4. Mogelijke signalen zijn laser, ultrasoon, radiogolven en zichtbaar licht [5], [10], [11]. Afhankelijk van de producent zal de functie van het mobiel baken en de stationaire bakens anders zijn, er zijn drie verschillende situaties te onderscheiden:

- het mobiel baken treedt op als zowel zender als ontvanger van het signaal, de stationaire bakens reflecteren het signaal;
- het mobiel baken is zender en het stationair baken is ontvanger;
- het mobiel baken is ontvanger en het stationair baken is zender.

Figuur 2-9 toont een voorbeeld van baken navigatie op basis van een laser, dit is de meest populaire methode in de navigatie industrie [9]. Een ander voorbeeld van baken navigatie is GPS.

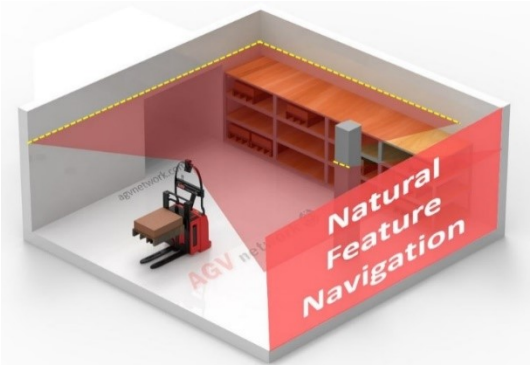


Figuur 2-9: Laser baken navigatie [8]

Er zijn grote verschillen tussen de mogelijke signalen, elk signaal heeft dan ook zijn eigen specifieke voor- en nadelen. De voordelen van deze methode zijn de nauwkeurigheid en flexibiliteit. Verder is de installatiekost relatief lager dan de vorige methoden en vergt het weinig onderhoud. Een nadeel van deze methode is dat het plaatsen van de bakens meer tijd kost, dit is nog eens extra moeilijk in omgevingen met veel obstakels. De gebruikte hardware en software zijn ook complexer wat leidt tot een hogere ontwikkelingskost voor de AGV [3], [5].

Natuurlijke baken navigatie

De laatste methode is natuurlijke baken navigatie, deze methode werkt op basis van structuren die van natura in de omgeving aanwezig zijn. Voorbeelden hiervan zijn de hoeken van een kamer of vaste objecten zoals pilaren. De AGV maakt een 2D of 3D scan van de omgeving en vergelijkt deze met een vooraf opgestelde kaart van de omgeving (zie Figuur 2-10). De scan gebeurt met een camera op basis van de inval van zichtbaar licht of via LIDAR op basis van het verzenden en ontvangen van een infrarood laser [12]. Het is bij 2D scans belangrijk dat de weg ongeveer vlak is, de scans gebeuren namelijk op een bepaalde hoogte. Wanneer de weg niet vlak is, zal de scanner niet meer op dezelfde hoogte meten en het beeld dus niet meer overeenkomen [5]. Dit leidt tot een vervorming van de kaart. Indien deze vervorming licht is, is navigatie doorheen dit gebied mogelijk.



Figuur 2-10: Natuurlijke 2D navigatie [8]

Deze methode heeft als voordeel dat het aanbrengen van wijzigingen aan de omgeving niet nodig is. Natuurlijke baken navigatie is ook zeer flexibel omdat deze enkel van de omgeving afhankelijk is. Nadelen zijn dat de nauwkeurigheid sterk afhankelijk is van de gekozen hard- en software, dit kan deze methode zeer prijzig maken [5]. Verder is natuurlijke baken navigatie zeer afhankelijk van de vooraf opgebouwde kaart. Indien de omgeving vaak verandert zal dit de werking dus bemoeilijken. Er zijn ook belangrijke verschillen tussen het gebruik van LIDAR of camera. LIDAR werkt zowel in lichtere als donkerdere omgevingen en kan heel goed de afstand tot obstakels bepalen. LIDAR is echter duurder en heeft een beperkte resolutie [13]. Camera's zijn relatief goedkoop, zien kleur en hebben een hogere resolutie. De werking van camera's is echter afhankelijk van de hoeveelheid licht in de omgeving, ook is er meer rekenkracht nodig om de beelden te verwerken [12], [13] .

2.1.3 Veiligheid

De AGV zal zich verplaatsen in omgevingen waar goederen staan en mensen komen. Om botsingen met zowel goederen als mensen te vermijden is het belangrijk dat de verplaatsing op een veilige manier verloopt. Hiervoor moet de AGV weten wat er zich rondom hem afspeelt. Afhankelijk van de gekozen navigatiemethode bevat het voertuig reeds sensoren die de omgeving waarnemen, denk hierbij aan natuurlijke baken navigatie. Echter beschouwen navigatiemethoden onregelmatigheden in de sensormetingen, zoals een persoon die voorbij rent, vaak als ruis. Daarom moet iedere AGV in het bezit zijn van extra elementen die specifiek bedoeld zijn om de veiligheid te garanderen.

Veiligheidslaserscanner

Deze scanners zullen bepalen of er zich een obstakel rond de AGV bevindt. Dit doen ze door een infrarood laserstraal over een bepaalde hoogte uit te zenden, wanneer er een obstakel is zal het licht terug reflecteren in de laserscanner. Omdat de laserscanners de laserstraal op een bepaalde hoogte uitzenden, is het nodig dat deze laag op de AGV gemonteerd zijn. Dit om ook de lager gelegen obstakels te detecteren. Een veiligheidslaserscanner onderscheidt zich van een gewone laserscanner door zijn grotere betrouwbaarheid. De beoordeling van de betrouwbaarheid gebeurt door een erkende keuringsinstantie.

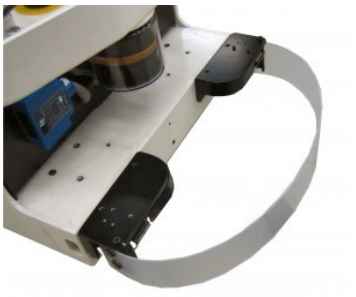
De AGV's die gebruik maken van een veiligheidslaser kunnen op twee manieren reageren. De eerste manier is door te stoppen wanneer de laser een obstakel in een vaste zone rondom hem detecteert en pas terug te vertrekken wanneer dit obstakel uit de zone weg is. Deze manier is simpel om te implementeren maar niet optimaal omdat de AGV zal stoppen op momenten wanneer dit niet nodig is. Dit omdat de vaste zone rekening moet houden met de maximumsnelheid van het voertuig en niet de huidige. De tweede manier is efficiënter, de veiligheidslaser deelt hierbij het gebied rondom de AGV op in verschillende zones [5]. Wanneer er zich een obstakel in de buitenste zone bevindt zal de AGV vertragen maar niet stoppen. Wanneer er echter een obstakel in de binnenste zone aanwezig is zal de AGV meteen stoppen. Afhankelijk van de fabrikant kunnen verschillen optreden, zo kan een fabrikant

er ook voor kiezen om drie zones te gebruiken (zie Figuur 2-11) [12]. De grootte van deze zones is bepaald op basis van de remafstand [5], deze remafstand is op zijn beurt weer afhankelijk van snelheid, gewicht van de wagen, weersomstandigheden, etc. Omdat de zones continu veranderen tijdens de werking van het voertuig is er meer rekenkracht nodig dan bij de eerste manier.

Figuur 2-11: Zones van Veiligheidslaserscanners [14]

Contact bumpers

Een ander veiligheidselement is de contact bumper. Wanneer er iets in fysiek contact komt met de bumper zal de AGV stoppen met rijden. De bumpers zijn zodanig ontworpen dat de schade aan zowel zichzelf als hetgeen waarmee ze botsten minimaal is [3]. Contact bumpers hebben als voordeel dat ze makkelijker te implementeren en goedkoper zijn. Nadelig is echter dat ze pas opmerken dat er een obstakel is wanneer er fysiek contact is. Omwille van hun voor- en nadelen komt de combinatie van contact bumpers en laserscanners frequent voor. Hierbij controleren de laserscanners enkel de rijrichting van de AGV. De plaatsing van contact bumpers gebeurt in de zones die de laserscanner niet ziet. Figuur 2-12 geeft een voorbeeld weer van een contact bumper.



Figuur 2-12: Contact bumper [14]

Noodstop

Het moet altijd mogelijk zijn om de AGV te laten stoppen, hiervoor moeten er één of meerdere noodstoppen op de AGV aanwezig zijn. Door het induwen van de noodstop zal de AGV veilig tot stilstand komen.

Waarschuwingen

Naast de vorige actieve veiligheidselementen is er ook nood aan passieve elementen, deze waarschuwingen de omgeving dat er een AGV in de buurt is. Dit om botsingen te vermijden. Enkele voorbeelden van deze waarschuwingen zijn een lichttoren, een sirene en signalisatie die de aanwezigheid van de AGV benadrukt. Voor het implementeren van signalisaties of sirenes dient een evaluatie plaats te vinden. Deze passieve elementen zouden namelijk nadelig kunnen zijn wanneer er mensen met bepaalde beperkingen aanwezig zijn. Bijvoorbeeld mensen met epilepsie.

Veiligheidsnormen

Naast de gebruikte veiligheidselementen moet ook het voertuig zelf veilig zijn. Om dit te controleren zijn er veiligheidsnormen waaraan het gebruikte voertuig moet voldoen. De veiligheidsnorm voor AGV's heet DIN EN ISO 3691-4 "*Werkvoertuigen - eisen voor veiligheidsuitrusting en verificatie - Deel 4: Automatisch geleide werkvoertuigen en hun systemen*".

2.1.4 Haalbaarheid

Het aankopen van een AGV bij een erkende firma zal ervoor zorgen dat Sint Oda verzekerd is dat deze aan de nodige veiligheidsnormen voldoet, verder is het ook mogelijk om deze firma om hulp te vragen indien er iets misgaat. Om de kost in kaart te brengen is een offerte aangevraagd bij enkele van de grote bedrijven op de AGV markt waaronder Savant, Dematic, STILL en EK-Automation. Deze bedrijven gingen echter niet in op de offerte aanvraag. Er is daarom online naar mogelijke prijzen gezocht [15]. Tabel 2-1 toont de gevonden prijzen van verschillende types AGV. Er zijn geen prijzen van unit load carriers gevonden.

Tabel 2-1: Prijzen verschillende soorten AGV's

Type AGV	Prijs (€)
Automated guided cart	14.000 - 35. 000
Tow vehicle	20.000 - 54.000
Forklift AGV	60.000 - 150.000

Naast de kost van de AGV komen er ook nog infrastructuurkosten, installatiekosten, servicekosten en bijkomende kosten bij. Deze kosten zullen afhankelijk van het gekozen type AGV, de navigatiemethode en het bedrijf verschillen waardoor ze moeilijk te bepalen zijn. De prijzen van de AGV op zich overstijgen het gegeven budget van Sint Oda waardoor deze oplossing niet haalbaar is. Voor een exacte prijs moet een grondige financiële analyse gebeuren.

2.2 Elektrische trekker

2.2.1 Beschrijving

Om meer inzicht te krijgen in het onderwerp van vervoer voor rolstoelgebruikers is er besloten om ook naar niet autonome oplossingen te kijken en de mogelijkheden tot automatisatie van deze te evalueren. Het principe van de elektrische trekker berust op een manueel gestuurde machine die een reeks rolstoelen of een met rolstoelen gevulde kar kan voorttrekken. Het besturen van de machine gebeurd zoals het besturen van een grasmaaier. De bestuurder loopt achter de machine terwijl de machine de reeks rolstoelgebruikers, of een kar met rolstoelgebruikers, kan voorttrekken (zie Figuur 2-13).

Figuur 2-13: De E-Tug-20 van Vestil [16]

2.2.2 Haalbaarheid

Sint Oda heeft dit idee al eens voorgesteld, veel van het personeel zag dit idee niet zitten door de kans op een menselijke fout. Hierdoor is deze oplossing voor Sint Oda niet interessant. De conceptuele studie kijkt naar de mogelijkheid om de oplossing om te vormen naar een (semi-)autonome oplossing. Dit verkleint de kans op een menselijke fout al aanzienlijk. Ook zijn alle veiligheidsaspecten die de norm voor deze machine vereist, zoals een noodstop, al ingebouwd door de fabrikant.

2.3 Huifkar

2.3.1 Beschrijving

Naast de elektrische trekker is een huifkar ook een mogelijke oplossing om de bewoners te vervoeren. Sint Oda heeft in het verleden al naar dergelijke huifkarren gezocht bij het bedrijf Jecam. De huifkarren van Jecam staan laag bij de grond of bevatten een knielsysteem waardoor ze zich tijdelijk kunnen verlagen. Hierdoor is het makkelijker om rolstoelen in- en uit te rijden. Verder bevat elke huifkar een bevestigingssysteem voor de rolstoelen op basis van bevestigingshaken, de werking van deze bevestigingsmethode staat beschreven in deel 2.4.2. Tenslotte bestaat er ook de mogelijkheid om een huifkar aan te kopen die slechts drie wielen heeft, deze is wendbaarder dan één met vier wielen. Alle huifkarren die Jecam levert zijn bedoeld voor transport op eigen terrein en zullen dus geen hoge snelheden halen. Figuur 2-14 toont een mogelijke huifkar. Deze huifkar is een driewieler voor het vervoer van vier personen in een rolstoel.

Figuur 2-14: Huifkar Jecam [17]

Er is ook naar andere bedrijven zoals Jecam gezocht. Volgende voorwaarden zijn gesteld:

- de huifkar moet minstens vier personen in rolstoel tegelijk kunnen vervoeren;
- de huifkar moet een bevestigingsmethode voor de rolstoelen bevatten;
- er moet een lift of oprijplaat aanwezig zijn voor het in- en uitrijden van de rolstoelen.

Er zijn in België en omstreken geen bedrijven gevonden die hetzelfde soort wagens verkopen als Jecam. Wel zijn er automerken als Mercedes-benz, Renault en Volkswagen die bedrijfswagens verkopen die uitgerust zijn voor rolstoelvervoer. Deze wagens kunnen ook op de openbare weg rijden (zie Figuur 2-15).

Figuur 2-15: Exterieur (links) en interieur (rechts) van Mercedes-benz voor rolstoelvervoer [18]

2.3.2 Haalbaarheid

Het aankopen van een huifkar heeft als voordeel dat de fabrikant de veiligheid van de bewoners verzekert tijdens transport. Sint Oda heeft echter aangegeven dat zij geen extra personeel ter beschikking willen stellen om de bewoners rond te rijden. Dit zorgt ervoor dat de oplossing niet haalbaar is. Om deze reden beschrijft de conceptuele studie de mogelijkheid om de huifkar autonoom te maken. Er ontstaat hierdoor een soort unit load carrier. Een gevolg van deze wijziging is dat de huifkar ook moet voldoen aan de veiligheidsnorm voor AGV's.

2.4 Bevestigingsmethoden

De vorige paragrafen, met uitzondering van deel 2.3, bespreken enkel de verschillende manieren om de bewoners te verplaatsen. Om deze verplaatsing veilig te laten gebeuren moet er echter een goede bevestigingsmethode zijn. De oplossing moet hierdoor ook voldoen aan ISO 10542-1:2012 "*Technische systemen en hulpmiddelen voor mensen met beperkingen of/en handicap - Rolstoelvastzetsystemen en veiligheidssystemen voor inzittenden*". Volgende paragrafen geven een opsomming van de bestaande en bruikbare bevestigingsmethoden.

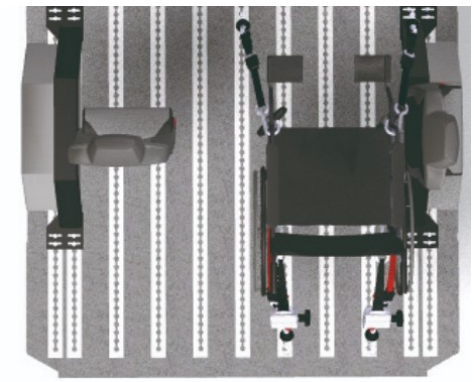
2.4.1 Stangenverbinding

Een stangenverbinding is de oplossing die Sint Oda momenteel gebruikt om meerdere bewoners tegelijkertijd te vervoeren (zie Figuur 1-1). Het is mogelijk om deze oplossing zodanig uit te breiden dat er vier rolstoelen aan elkaar bevestigd zijn, waarbij de voorste aan het gekozen voertuig vast zit.

2.4.2 Bevestigingshaken

Een andere manier om de rolstoelen te bevestigen is door gebruik te maken van haken die aan de vloer van het voertuig vastzitten. Hierbij zijn twee haken aan de voorkant van de rolstoel bevestigd en twee haken aan de achterkant. De haken aan de vloer vastzetten kan via een rail (zie Figuur 2-15) of een raster dat aan de vloer gemonteerd is. De rail of het raster hoeven niet op het voertuig zelf te zitten. Het is ook mogelijk om deze methode in een kar te installeren en de kar aan het voertuig te bevestigen. Een bedrijf die deze producten levert is Smartfloor [19], de catalogus van Smartfloor bevat rails en

haken maar ook andere onderdelen zoals oprijplaat en liften. Figuur 2-16 toont een bovenaanzicht van het haaksysteem van Smartfloor.



Figuur 2-16: Bovenaanzicht van het haaksysteem van Smartfloor [19]

2.4.3 Docking station

Deze methode is gelijkaardig aan deel 2.4.2, nu zal de rolstoel echter via een docking station bevestigd zijn aan de vloer. Dit docking station is een klem die zich onder de rolstoel bevindt (zie Figuur 2-17). Het is ook mogelijk om deze met de haken te combineren.

Figuur 2-17: Docking station van Advance Mobility [20]

2.4.4 Veiligheidsriem

Naast de bevestiging van de rolstoelen aan het voertuig moeten ook de personen zelf vastzitten. Dit zodat ze niet uit de rolstoel vallen bij een abrupte stop. Indien de oplossing gebruik maakt van de stangenverbinding zullen de rolstoelen zelf uitgerust moeten zijn met een veiligheidsriem. Bij haken en het docking station is dit ook een optie, een andere optie is om de riem aan het voertuig of kar te verbinden. Dit kan enerzijds door de rails op de vloer te gebruiken om de riem over het middel van de bewoner te gespen. Anderzijds is het ook mogelijk om rails op de wanden te installeren. Het bevestigen van de riem kan nu zoals in een auto gebeurt. Smartfloor levert naast de producten aangehaald in deel 2.4.2 ook veiligheidsriemen voor rolstoelgebruikers.

2.5 Conclusie

Er zijn in *hoofdstuk 2 Transportmogelijkheden* drie verschillende voertuigen, die op de markt beschikbaar zijn, onderzocht. Geen enkele van deze methoden voldoet aan de eisen gesteld door Sint Oda. Dit omdat ze niet autonoom zijn of omdat deze nog niet over de nodige veiligheids- en bevestigingsvereisten beschikken. In *hoofdstuk 5 Conceptuele studie* komen daarom enkele oplossingen aan bod alsook een vergelijking van deze oplossingen. Er is voor het bedenken van deze oplossingen onder andere uitgegaan van de ideeën die in dit deel zijn opgedaan.

3 Navigatie-algoritmen en -hardware

3.1 Lokalisatie

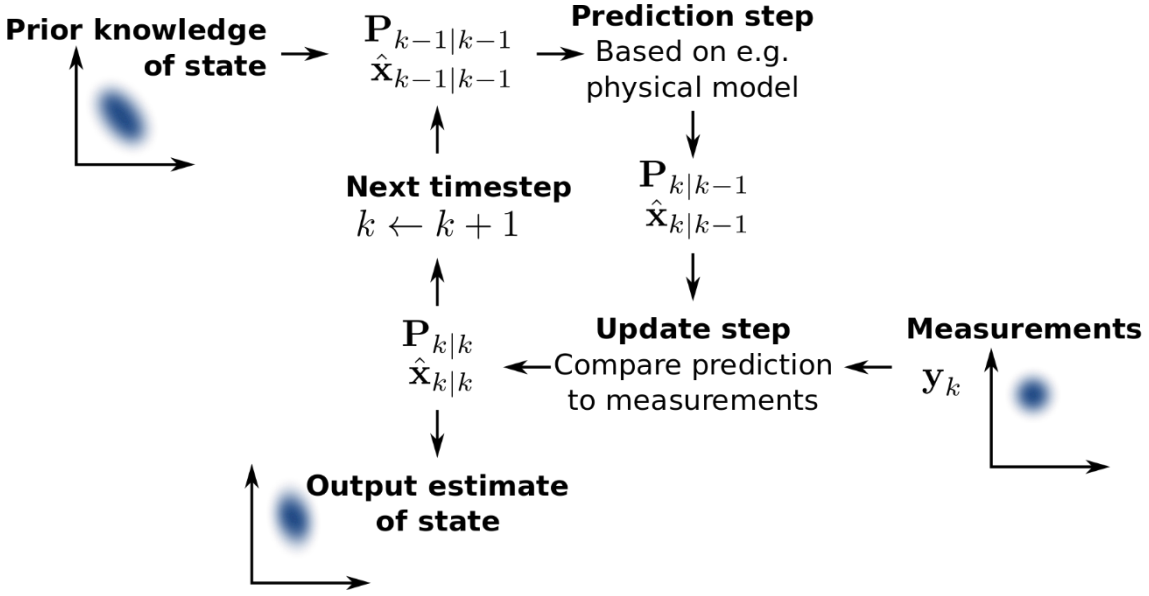
Voor autonome navigatie is het noodzakelijk dat de AGV zijn positie in de omgeving kan vinden. Vaak beschikt een AGV over een kaart van de omgeving. Het zoeken van de huidige positie op de kaart heet lokalisatie. Lokalisatie algoritmes werken op basis van probabiliteit, ze maken dus gebruik van statistische technieken en kansrekenen.

3.1.1 Regel van Bayes en de Kalman filter

Aan de basis van meeste lokalisatie algoritmes ligt de Bayes filter, deze werkt o.b.v. de regel van Bayes.

$$\boldsymbol{P(B|A) = \frac{P(A|B) * P(B)}{P(A)}} \tag{1}$$

De regel van Bayes berekent de kans op situatie A in het geval van omstandigheid B. In deze formule zijn de voorwaardelijke kans dat omstandigheid B optreedt in het geval van situatie A en de kansen op B en A gekend [21]. Een Bayes filter is het recursief oproepen van de regel van Bayes om gegevens te verwerken. Als de voorwaardelijke kans variabelen van het algoritme normaal verdeeld en lineair zijn, is dit een Kalman filter [21]. De Kalman filter helpt dus het schatten van een variabele die indirect meetbaar is. In Figuur 3-1 is het principe weergeven. De Kalman filter schat de huidige toestand van het systeem aan de hand van de vorige toestand, de geschatte stap die het systeem gaat maken en metingen. Door het in rekening brengen van de covarianties (kans op afwijkingen) kan de Kalman filter de toestand van het systeem nauwkeuriger schatten.

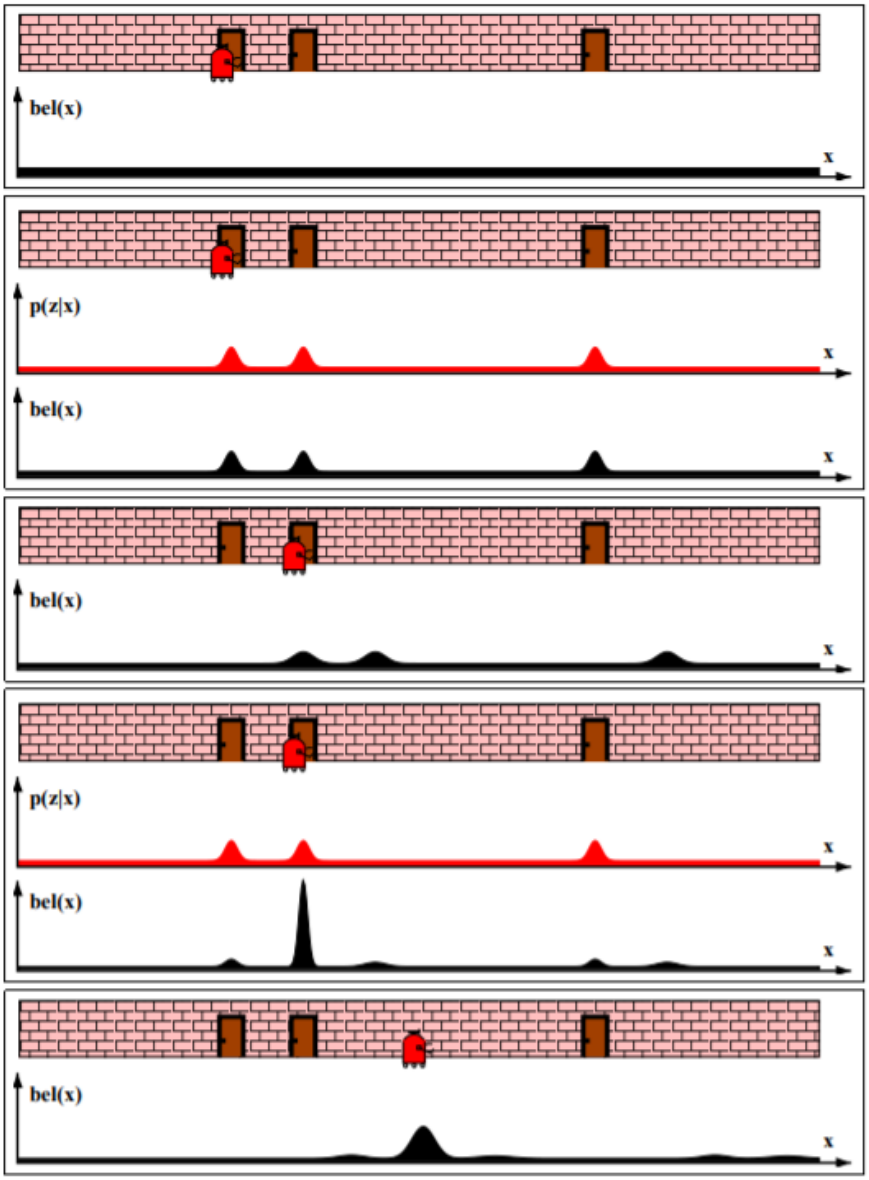


Figuur 3-1: Principe Kalman filter [22]

3.1.2 Markov lokalisatie

In het geval van lokalisatie bij een AGV gebruikmakend van een Bayes filter zal A de werkelijke positie zijn van de AGV en B de positie die de sensoren denken te meten. De AGV heeft een kaart met alle obstakels en grenzen waarmee hij de metingen van zijn sensoren vergelijkt (stap 1 in Figuur 3-2).

De AGV zal de kans voor elke mogelijke positie uitrekenen, dit aan de hand van de positie gemeten door de sensoren (stap 2 in Figuur 3-2). De positie met de hoogste kans is dan een benadering van de werkelijke positie waarop de AGV zich bevindt. De AGV zal dit algoritme recursief blijven oproepen tijdens het bewegen en de huidige sensor metingen vergelijken met de vorige geschatte positie, Het is ook mogelijk om de verwachte beweging van de robot mee te implementeren als extra variabele. Dit noemt men Markov lokalisatie [21], Figuur 3-2 weergeeft een voorbeeld hiervan.



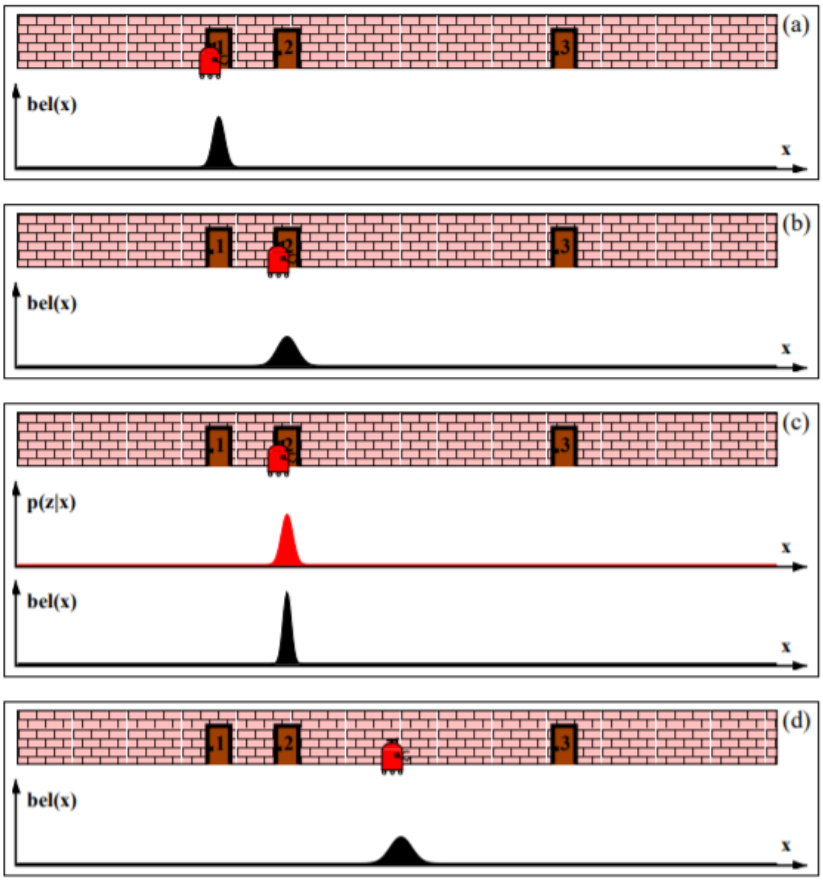
Figuur 3-2: Voorbeeld van Markov lokalisatie [21, p165]

De AGV kent zijn omgeving dankzij zijn kaart en weet dat er 3 deuren aanwezig zijn op de gang. Na het detecteren van één deur zijn er dus 3 mogelijke posities. Wanneer de AGV verder rijdt en de tweede deur op die positie detecteert kan er nog maar één positie mogelijk zijn waarop de AGV zich bevindt omdat de robot de afstand tussen de deuren kent.

3.1.3 Extendend Kalman Filter en Unscented Kalman Filter

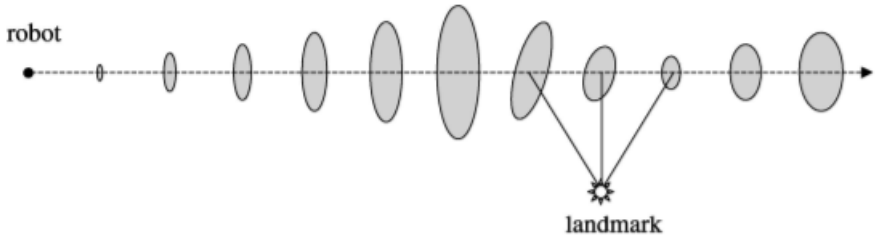
Extended Kalman Filter (EKF) lokalisatie is een vorm van Markov lokalisatie waarbij de kaart bestaat uit oriëntatiepunten [21], het grote verschil is dat de EKF kan werken met niet lineaire dynamische systemen. Het algoritme zal op basis van de detectie van deze oriëntatiepunten (zoals bijvoorbeeld

bakens uit deel 2.1.2 of de nummers in de deuren op Figuur 3-3) zijn positie kunnen bepalen, Figuur 3-3 weergeeft een voorbeeld hiervan.



Figuur 3-3: Lokalisatie door EKF [21, p. 167]

De AGV detecteert een deur en weet dat dit de eerste deur is door de "1" die erop afgebeeld is. De zekerheid van de positie neemt af naarmate de tijd of afstand dat er geen oriëntatiepunt is gedetecteerd. Wanneer de AGV een oriëntatiepunt detecteert zal de onzekerheid (weergeven door de grootte van de ellips) weer afnemen (zie Figuur 3-4).



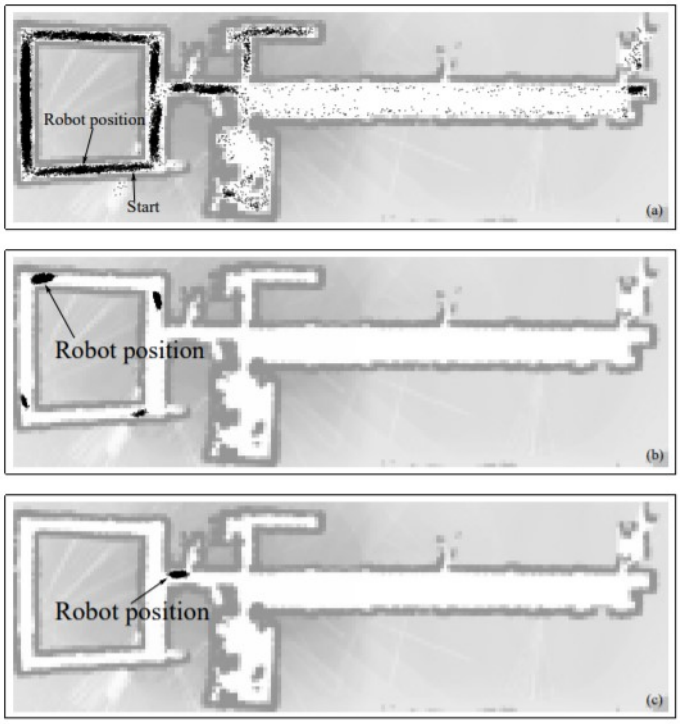
Figuur 3-4: De onzekerheid van de positie uitgedrukt als ellips [21, p. 170]

In het geval van een zeer niet lineair dynamisch systeem is het toepassen van een EKF niet precies genoeg. Een Unscented Kalman Filter (UKF) is dan een mogelijkheid. Deze bemonstert zijn inkomende data om deze zo veel mogelijk te lineariseren om vervolgens een Kalman filter toe te passen.

3.1.4 Monte Carlo lokalisatie

Een ander principe van lokalisatie is Monte Carlo lokalisatie. Zoals al af te leiden valt uit de naam gebruikt dit principe een Monte Carlo simulatie. Dit is een simulatie die een proces meerdere keren simuleert, telkens met andere initiële parameters [21]. Een kansverdelingsfunctie geeft de resultaten van alle simulaties weer. In het geval van lokalisatie zijn de beginpositie en de sensormetingen de

startparameters. De AGV is in het bezit van een volledige kaart, aan de hand van een sensormeting kan de AGV de mogelijke werkelijke posities aanduiden op de kaart en aan elke positie een “gewicht” toekennen. Hoe groter de kans dat de AGV zich op die positie bevindt, hoe groter het gewicht. Wanneer de AGV beweegt en nog een sensormeting uitvoert kan hij al meerdere punten uitsluiten aan de hand van het gewicht, dit doet hij tot de werkelijke positie bepaald is. Het principe van het uitsluiten van de punten staat bekend als “particle filtering”. Figuur 3-5 weergeeft hoe de AGV meer en meer punten uitsluit naarmate hij in beweging is.



Figuur 3-5: Monte Carlo lokalisatie voorbeeld [21, p. 203]

De donkerste stippen op de kaart van Figuur 3-5 geven de posities aan waar de kans het grootst is dat de AGV zich bevindt [21]. Naarmate dat de AGV een grotere afstand aflegt zal het algoritme meerdere punten kunnen uitsluiten. Uiteindelijk zal de AGV zijn positie kunnen bepalen.

3.2 Kaartopbouw

Kaartopbouw is essentieel voor navigatie van een AGV. Aan de hand van een kaart kan de AGV zichzelf lokaliseren en routes plannen naar een volgende positie. Kaartopbouw is zowel in 2D als 3D mogelijk. Het terrein van Sint Oda bevat geen significante hoogteverschillen waardoor het gebruik van een 2D kaart mogelijk is. Bovendien is het gebruik van een 2D kaart minder processor intensief. Daarom is er gekozen om in de literatuurstudie enkel 2D kaartopbouw te onderzoeken.

3.2.1 Opbouw

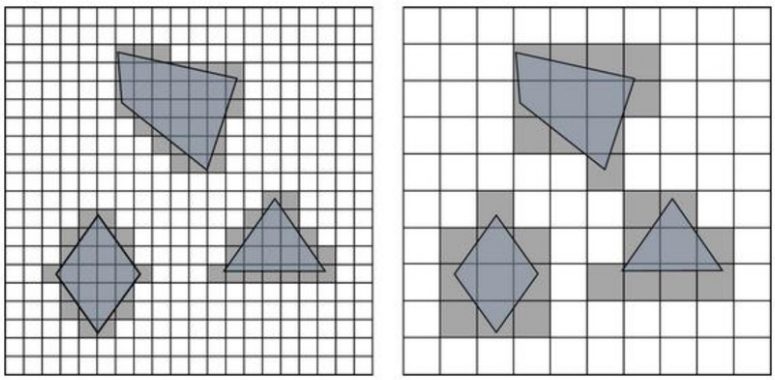
Het opbouwen van een kaart kan op veel verschillende manieren gebeuren. Eén van de manieren is de kaart zelf uittekenen met een CAD programma. Een andere manier is door gebruik te maken van SLAM (Simultaneous Localisation And Mapping), dit zorgt ervoor dat een sensorplatform een kaart kan maken van een omgeving. Tijdens het bouwen van de kaart houdt het sensorplatform zijn positie op de kaart bij [23]. Deze methode is sneller dan de handmatige methode omdat hier het sensorplatform de kaart zelf maakt terwijl hij rondrijdt. SLAM is ook zeer interessant voor omgevingen die vaak veranderen, dit omdat SLAM telkens een nieuwe kaart maakt en de elementen die er niet meer zijn vergeet. SLAM staat zeer dicht bij natuurlijke navigatie. Dit is logisch aangezien het toestel de omgeving moet “zien” vooraleer hij er een kaart van kan maken, dit principe gebruikt

natuurlijke navigatie ook voor zijn lokalisatie. Hiernaast zijn de proprioceptieve sensoren uit deel 3.4.1 ook nog altijd nodig om fouten van de omgevingssensoren te minimaliseren en andersom.

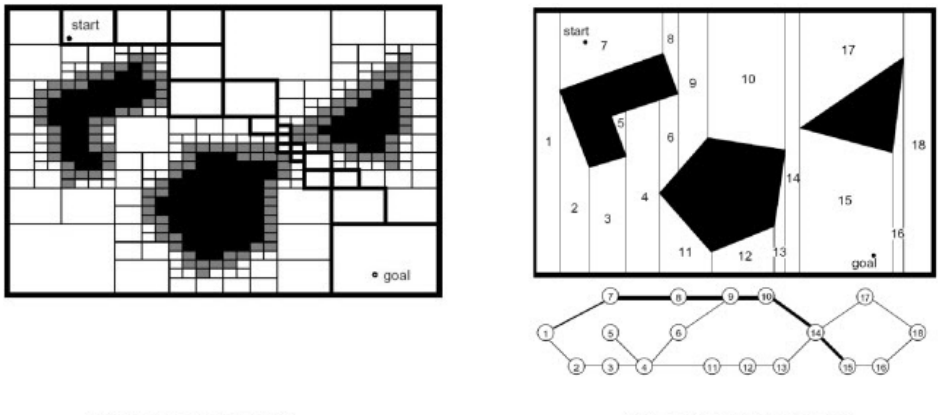
3.2.2 Cel decompositie

Dit proces verdeeld de kaart in meerdere cellen rondom de obstakels. Dit zorgt voor een vlotte werking van de pad plan algoritmes. Deze verdeling kan op drie manieren gebeuren.

- Geschatte cel decompositie, de kaart is verdeeld in meerdere cellen die een perfect raster vormen. De grootte van de cellen is vast en vooraf bepaald. Elke cel waarin zich ook maar een gedeelte van een obstakel bevindt is beschouwd als obstakel. Deze methode vergt weinig rekenkracht maar beschouwt vaak grote obstakelvrije stukken als obstakels, wat zeer nadelig is (zie Figuur 3-6).
- Adaptieve cel decompositie, dit proces begint met één grote cel. Vervolgens zal elke cel waarin zich een obstakel bevindt zich opdelen in vier nieuwe cellen van gelijke grootte. Dit blijft zich herhalen tot een kaart met "obstakel bevattende cellen" en "obstakelvrije cellen" is bereikt. Deze methode vergt iets meer rekenkracht maar verdeelt de kaart beter op dan geschatte cel decompositie (zie Figuur 3-7).
- Exacte cel decompositie, de cellen passen zich aan de vorm van de kaart en de vorm van de obstakels aan. Deze methode deelt de kaart perfect op in "obstakel bevattende cellen" en "obstakelvrije cellen" maar vergt erg veel geheugen en rekenkracht (zie Figuur 3-7).



Figuur 3-6: Twee voorbeelden van geschatte cell decompositie [24, p. 6]



(a) Exact Cell Decomposition (b) Adaptive Cell Decomposition

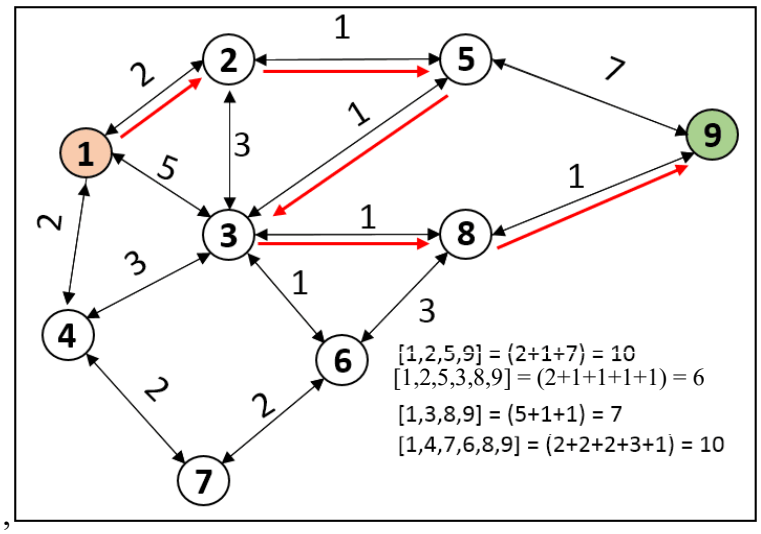
Figuur 3-7: Exacte cell decompositie (links) adaptieve cell decompositie (rechts) [25, p. 281]

3.3 Pad planning

Het doel van pad planning is zo efficiënt mogelijke routes plannen. Hiervoor gebruiken de meeste algoritmes altijd een "kost". Deze kost bepaald het aspect waarin de pad planning zo efficiënt mogelijk zal zijn. Typische voorbeelden zijn: tijd (kortste reistijd), afstand (kortste route) en verbruik (zuinigste pad) [25] of een combinatie van deze.

3.3.1 Dijkstra's algoritme

Dit algoritme heeft als doel de route met de minste kost te zoeken door de kaart op te delen in verschillende knooppunten en tussen elk knooppunt een kost te schatten. Het algoritme analyseert de verschillende mogelijke paden en kiest het pad met de minste kost. In Figuur 3-8 zoekt het algoritme een pad tussen de knooppunten 1 en 9. Het algoritme neemt de eerste stap naar knooppunt twee en zal nu alle eerst alle mogelijkheden bekijken die beginnen met een eerste stap naar knooppunt 2. Het eerste gevonden pad is [1,2,5,9] met een kost van tien, dit is momenteel het meest efficiënte pad dat het algoritme bepaald heeft dus slaat hij deze op. Het volgende gevonden pad is [1,2,5,3,8,9] met een kost van 6, dit is efficiënter dan het vorig pad dus verwerpt het algoritme het vorige pad en behoud dit nieuwe pad. Het volgende bepaalde pad is [1,3,8,9] met een kost van 7, dit is minder efficiënt dan pad [1,2,5,3,8,9] dus heeft het geen zin om pad [1,3,8,9] op te slaan. Het algoritme blijft de paden doorlopen tot alle mogelijkheden doorlopen zijn. Het pad [1,2,5,3,8,9] blijft het efficiëntst en is dus het gekozen pad.



Figuur 3-8: Dijkstra algoritme [26]

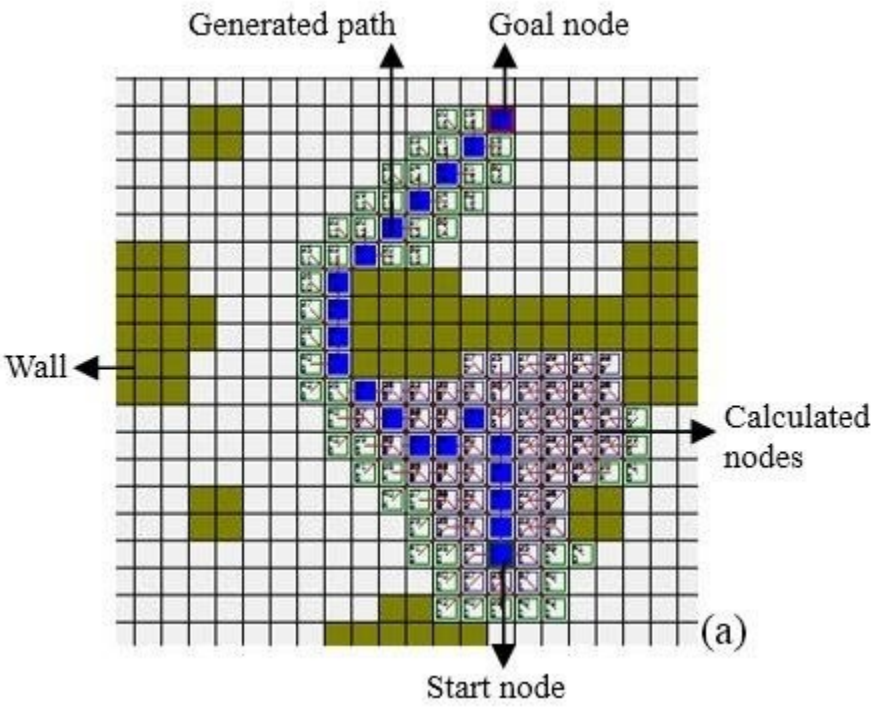
Hoewel het Dijkstra algoritme het meest efficiënte pad bepaald, vraagt het een grote computationele kost omdat het algoritme elke mogelijke combinatie evalueert.

3.3.2 A* algoritme

Dit algoritme verdeelt, net zoals het Dijkstra algoritme, de kaart in verschillende knooppunten en schat de kost tussen deze. Het grote verschil is dat deze methode altijd het eerstvolgende knooppunt zal kiezen dat zich het meest in de richting van het eindpunt bevindt. Wanneer het algoritme botst op een obstakel zal het algoritme opnieuw beginnen zoeken vanaf het begin knooppunt tot de optimale route gevonden is. Deze methode is sneller en vergt minder computationele kost dan het algoritme van Dijkstra. Het is dus geschikt voor pad planning met traditionele wegenkaarten [27].

3.3.3 D* Lite algoritme

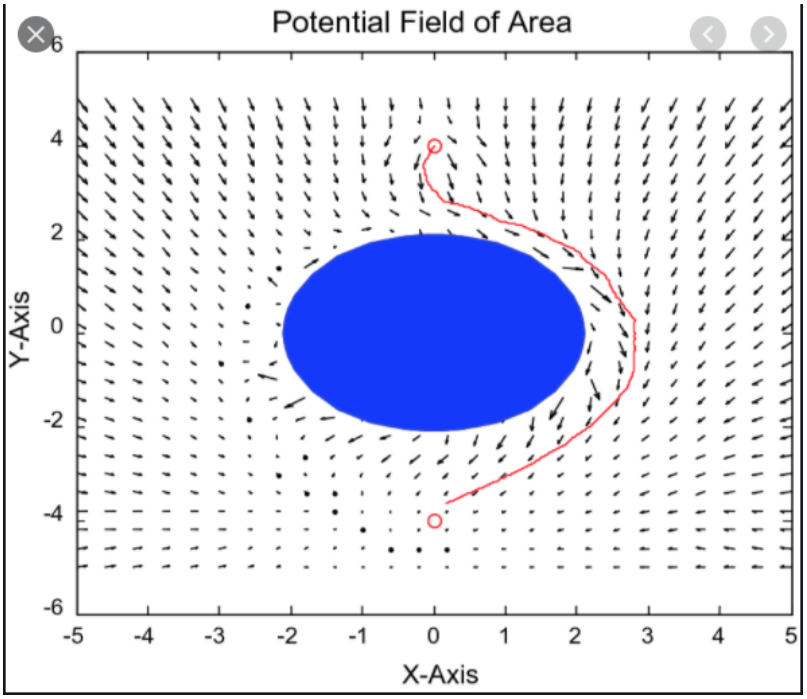
Dit algoritme is vergelijkbaar met het A* algoritme, enkel vertrekt dit algoritme vanaf het eindpunt en dus niet het beginpunt (zie Figuur 3-9). Ook onderzoekt het algoritme de knooppunten expansief en zal het na botsing met een obstakel niet opnieuw beginnen vanaf het eindpunt maar het laatst gevonden knooppunt voor de botsing. D* Lite is sneller dan het A* algoritme en veel effectiever bij toepassing in complexere omgevingen [27]. De computationele kost van D*Lite of A* is nagenoeg gelijk.



Figuur 3-9: D* Lite algoritme [27, p. 594]

3.3.4 Artificial potential fields

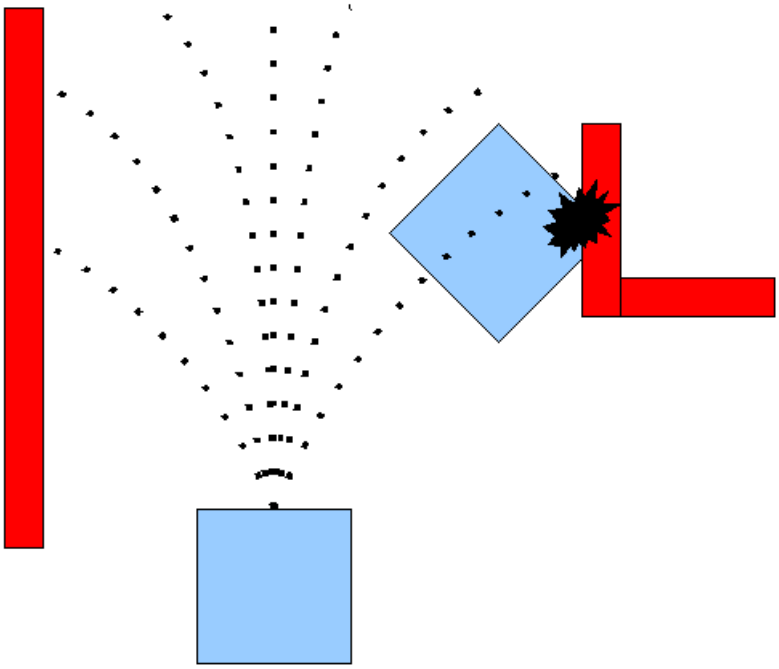
Deze methode bekijkt de obstakels als voorwerpen met een afstotend potentiaalveld rondom de voorwerpen. Hoe korter bij het obstakel hoe groter de afstotende kracht. Het eindpunt heeft een aantrekkend potentiaalveld. De combinatie van alle velden geeft een pad (zie Figuur 3-10). Het bekomen pad is soms niet de meest optimale route. Een ander nadeel is dat de robot in bepaalde situaties vast kan komen te zitten in een lokaal minimum [28].



Figuur 3-10: Pad en potential field rondom een voorwerp [28, p. 188]

3.3.5 Dynamic window approach

Dynamic window approach (DWA) is vooral een algoritme dat botsingen voorkomt. Het algoritme simuleert van tevoren de mogelijke bewegingen die de AGV kan maken op basis van zijn dynamische eigenschappen en restricties. Vervolgens analyseert DWA de simulaties en evalueert deze [29]. Ten slotte kiest het algoritme het best scorende traject (zie Figuur 3-11). Door vele korte collisievrije trajecten achter elkaar te generen kan de AGV zonder te botsen van begin naar eindpunt bewegen.



Figuur 3-11: DWA simuleert collisie route [29]

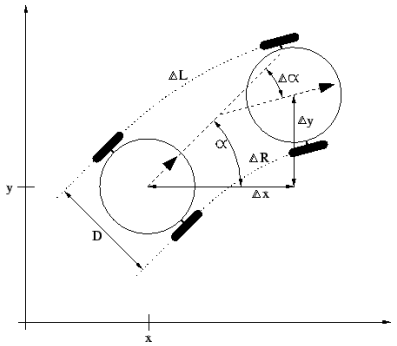
3.3.6 Anderen

Zoals besproken in 2.1.2 Navigatiemethoden zijn er ook systemen die werken met visuele, magnetische of inductieve linten of stippen. Er bestaan algoritmes om deze te volgen maar deze methoden vallen niet onder pad planning omdat het pad al bepaald is en de algoritmes ervoor zorgen dat de AGV het pad volgt.

3.4 Sensoren

3.4.1 Proprioceptieve sensoren

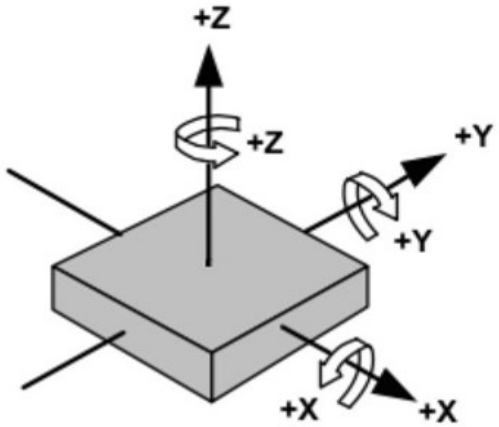
Aan de hand van proprioceptieve sensoren zoals encoders of IMU's is het mogelijk de positieverandering i.f.v. de tijd van een AGV te bepalen, dit staat bekend als odometrie (zie Figuur 3-12).



Figuur 3-12: Voorbeeld odometrie op basis van de verdraaiing van de wielen [29]

Het opmeten van verdraaiingen van de wielen gebeurt met encoders. Dit door de encoders vast te maken aan de assen van de wielen en bij te houden hoeveel graden of radialen het wiel gedraaid heeft. Encoders bestaan in twee varianten namelijk incrementeel en absoluut. Een incrementele encoder telt hoeveel graden het wiel gedraaid heeft vanaf de beginpositie. Een absolute encoder geeft de absolute hoekpositie van het wiel aan en weet dus altijd de absolute positie van het wiel en niet de relatieve positie ten opzichte van de beginpositie. Aan de hand van deze gegevens is het mogelijk de oriëntatie en positie van de AGV te schatten.

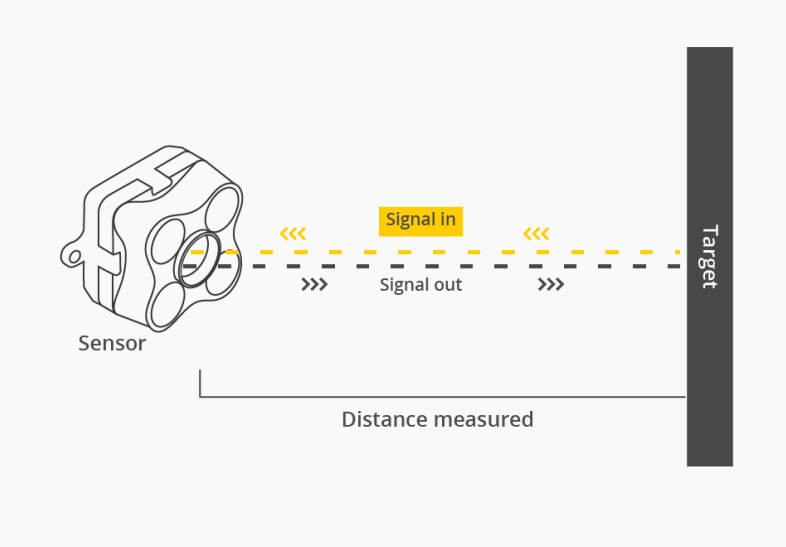
Een andere sensor is een IMU. Deze bevat gyroscopen, accelerometers en vaak magnetometers. De gyroscopen meten de hoeksnelheden rond de assen en de accelerometers meten de versnellingen volgens de assen (zie Figuur 3 13). De eventuele magnetometers meten verandering in het magnetisch veld van de aarde. Door de IMU te bevestigen aan de AGV is het berekenen van de positie en oriëntatie van de AGV mogelijk [30].



Figuur 3-13: Een IMU en de assen waarover de metingen plaatsvinden [30]

3.4.2 Afstandssensoren

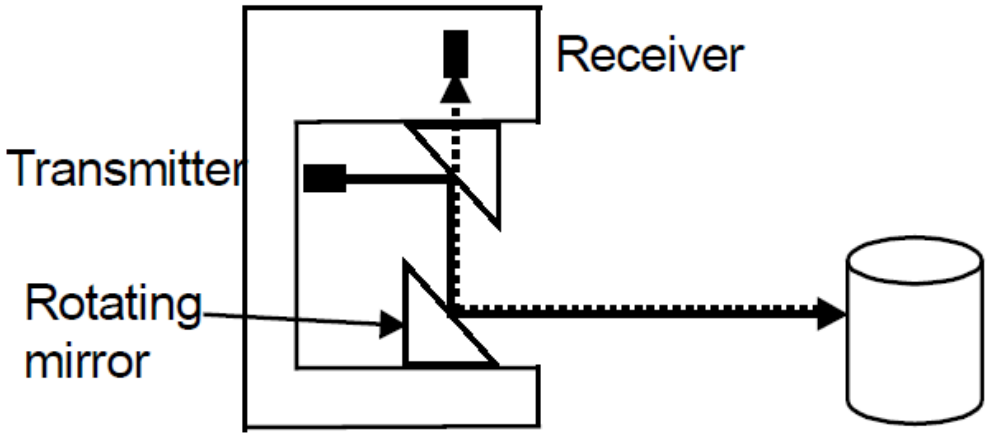
Afstandssensoren werken volgens meerdere principes. Eén van deze principes is het Time-of-Flight (ToF). De sensor stuurt een puls uit die vervolgens na reflectie terug invalt op de sensor, de tijd tussen het uitsturen en ontvangen is evenredig met de afstand (zie Figuur 3-14).



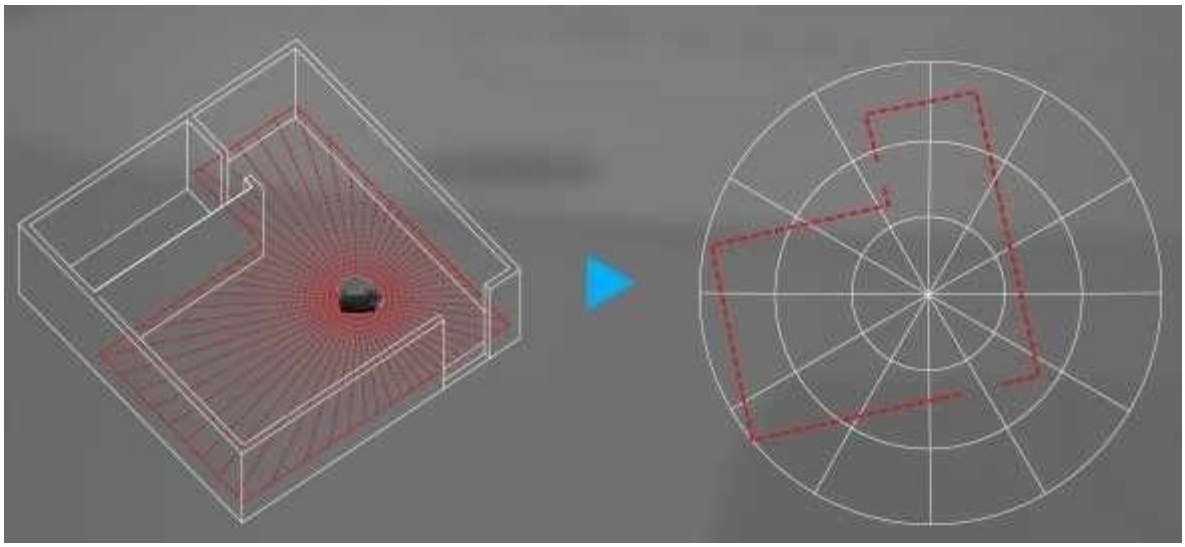
Figuur 3-14: ToF principe [31]

De puls kan een lichtpuls (laser) of een geluidspuls (ultrasoon) zijn. In de praktijk is bewezen dat een lasersensor veel nauwkeuriger is dan een ultrasoon sensor omdat een lichtpuls minder vatbaar is voor weerkaatsingen. Een 2D LIDAR, zoals in Figuur 3-15 werkt volgens hetzelfde principe als een

lasersensor, enkel zendt de sensor de puls uit op een roterende spiegel waarvan de rotatie hoek gekend is. Voor elke rotatie hoek is een afstand gemeten [32], dit geeft een 2D beeld zoals in Figuur 3-16. De hoeken waartussen het spiegeltje kan roteren bepalen de field of view (FOV) van de LIDAR. Sommige LIDAR's gebruiken geen spiegeltje maar laten de lasersensor zelf ronddraaien. Sommige LIDAR's kunnen volledig ronddraaien en hebben dus een field of view van 360°.

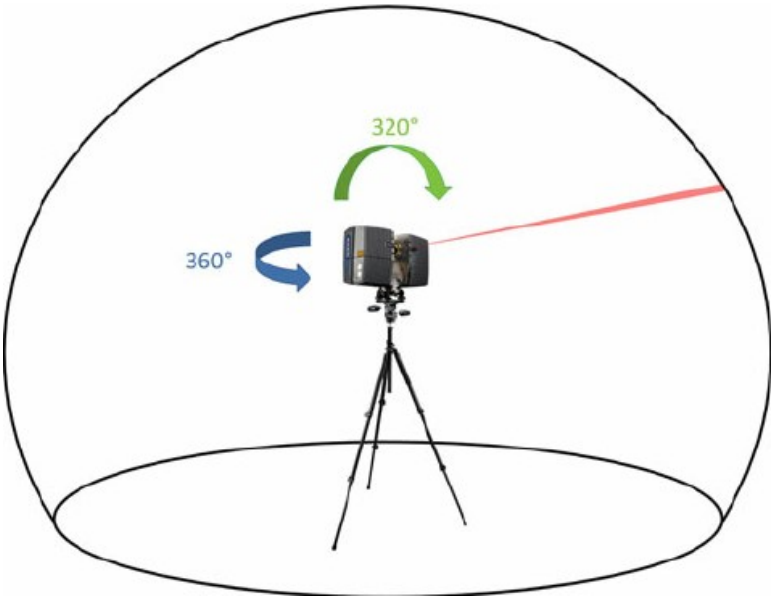


Figuur 3-15: 2D LIDAR principe [32, p. 2]

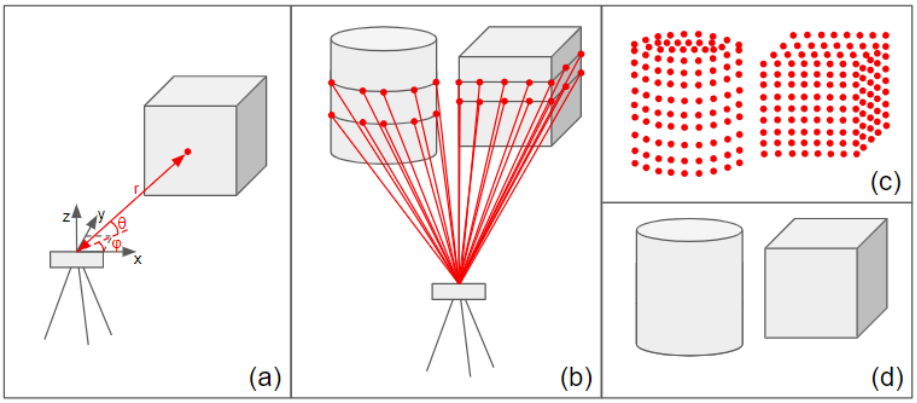


Figuur 3-16: Laserscan van ruimte door een 2D LIDAR met een field of view van 360° [33]

Een 3D LIDAR roteert over 2 assen en geeft dus een 3D beeld. De FOV beschrijft de hoeken waarover LIDAR meet, in Figuur 3-17 is de FOV dus 360°x320°. Het 3D beeld is weergeven in de vorm van een puntenwolk zoals in Figuur 3-18.



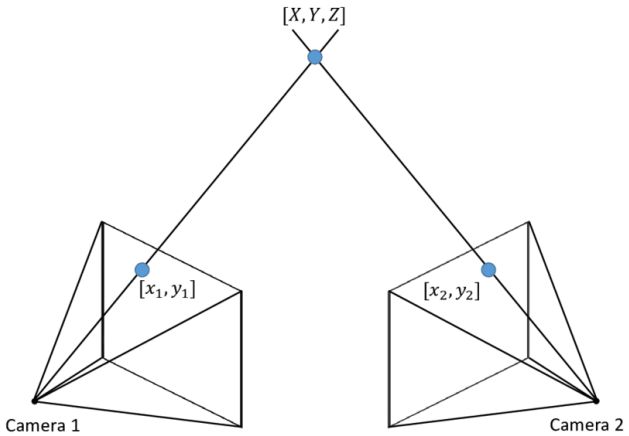
Figuur 3-17: 3D LIDAR [34, p. 961]



Figuur 3-18: Maken van een puntenwolk [35]

3.4.3 Camera's

Het is mogelijk om te werken met camera's en visie software. Een veelzijdigere optie is een 3D camera met visie software. Deze kan volgens meerdere principes werken. Eén principe is hetzelfde als bij menselijke ogen, omdat de afstand tussen de camera's gekend is het mogelijk om op basis van de beide beelden de afstand tot bepaalde punten te bepalen (zie Figuur 3-19) [36]. Dit heet stereo visie.

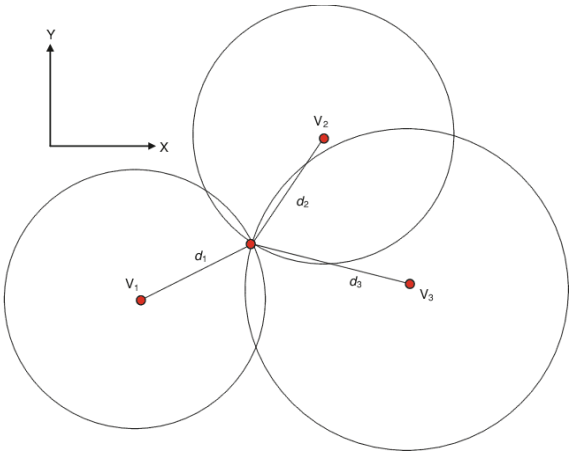


Figuur 3-19: 3D camera principe [36, p. 6]

Het is mogelijk om ToF en een 2D camera te combineren om een 3D beeld te bekomen. De 3D camera bestaat dus uit een 3D scanner zoals in Figuur 3-18 en een gewone camera. De gewone camera neemt zijn beeld op als een collectie van pixels die elk een RGB (rood, groen en blauw) waarde hebben. De 3D scanner geeft de diepte waarop elke pixel zich bevind. Combinatie van de twee metingen kan dus een 3D beeld in kleur geven. Deze methode bevat één groot nadeel en dat is dat de 3D scanner vaak op basis van infrarood werkt en dus niet bruikbaar is buiten. Dit omdat de zon veel infrarood stralen uitzendt en de metingen zwaar verstoort.

3.4.4 Bakensensoren

Bakensensoren werken op basis van vaste bakens waarvan de posities bekend zijn, een bewegende module en een trilaterale berekening zoals vermeld in 2.1.2. In Figuur 3-20 zijn V1, V2 en V3 de vaste bakens die elk een signaal met andere frequentie uitsturen. De bewegende module voert een intensiteitsmeting uit. Met behulp van die intensiteitsmeting op de frequentie van V1 is het mogelijk om de afstand d1 te bepalen. Deze afstandsbepaling zal voor elk baken plaats vinden. Wanneer alle afstanden ten opzichte van de vaste bakens berekend zijn is de positie van de bewegende module ook gekend.



Figuur 3-20: Trilaterale berekening [37, p. 10]

GPS

Een mogelijkheid van positiebepaling is simpelweg GPS. GPS werkt door signalen van satellieten (bakens) die zich op een vaste plek boven de aarde bevinden op te vangen. De resolutie van GPS bedraagt enkele meters. GPS is voor een AGV dus enkel interessant in combinatie met andere lokalisatiemethoden. Een oplossing hiervoor is Real Time Kinematic (RTK), dit is een vorm van GPS die op enkele centimeters nauwkeurig is. GPS en GPS-RTK zijn enkel te gebruiken in outdoor omgevingen aangezien het signaal in indoor omstandigheden sterk verzwakt [38].

Sonar

Een andere mogelijkheid is het gebruik van sonar. Plaatsing van sonarbakens in de te navigeren omgeving is dan noodzakelijk. Elk baken zendt een signaal uit met een verschillende frequentie, de bewegende module meet de intensiteit van elke frequentie. Om op elke plek in de omgeving de positie te kunnen bepalen moet de bewegende module ten alle tijden drie bakens kunnen ontvangen.

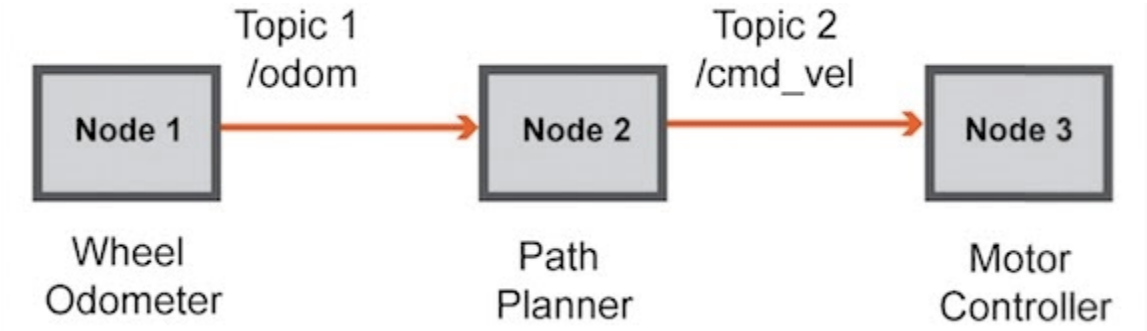
3.5 Middleware framework

Om de bovenstaande algoritmes vlot en robuust toe te passen is het gebruik van een bestaand framework aangeraden. Er bestaan meerdere middleware frameworks voor robotica bv. Urbi, Orca, Player Project, et cetera. In dit onderzoek is gekozen om te werken met ROS.

3.5.1 Robot operating system

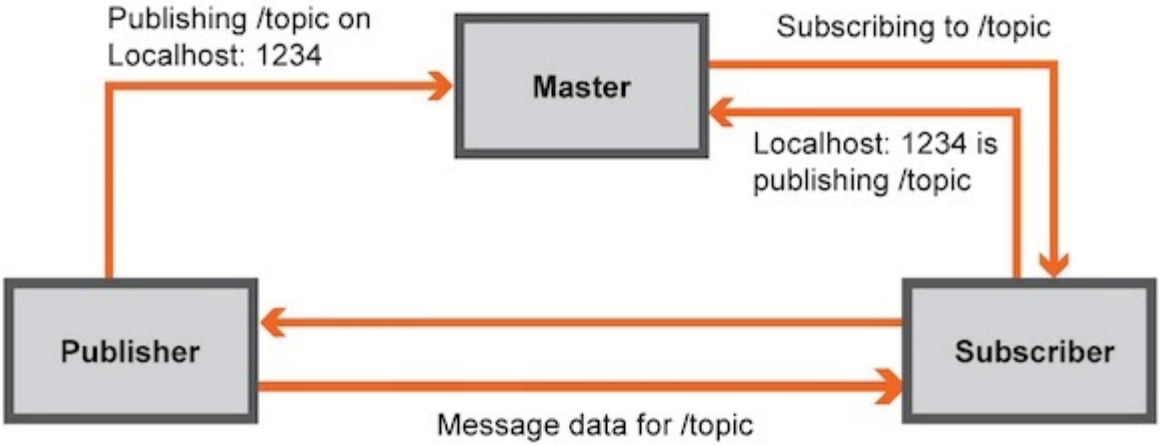
ROS staat voor Robot Operating System, dit is een flexibel open source framework voor het implementeren van software voor toepassingen binnen de robotica. Het is in feite een collectie van hulpmiddelen en softwarebibliotheken die het mogelijk maken om de ontwikkeling van complexe robot toepassingen te versimpelen [29]. ROS draait op Linux gebaseerde systemen. Voor dit onderzoek is gebruik gemaakt van Ubuntu 16.04 lts. ROS is ontstaan door het aanmoedigen van samenwerking naar het ontwikkelen van robotica software. Verschillende onderzoekers met verschillende specialisaties werken samen om dit framework te blijven ontplooien. Ook beschikt ROS over een enthousiaste community die altijd klaarstaat om mede programmeurs te helpen en hun vragen te beantwoorden.

ROS is opgebouwd uit een aantal “nodes” die met elkaar communiceren via “topics”. De onderling verstuurde gegevens heten “messages” en bevatten data die de nodes kunnen verwerken. De data bestaat uit een vaste structuur afgesproken structuur. Elke node stelt een proces voor die berekeningen uitvoert. De nodes ondersteunen onder andere Python, C++, Java of Common Lisp [29]. Een node kan messages publiceren in een topic of messages ontvangen van een topic (zie Figuur 3-21).



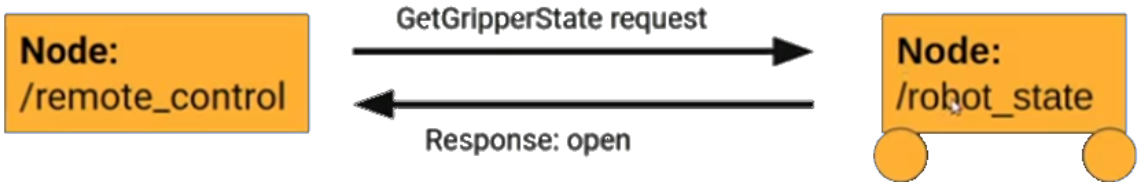
Figuur 3-21: Opeenvolgende nodes die data via topics doorgeven [39]

Nodes die messages in topics publiceren heten “publishers” en de nodes die messages ontvangen uit topics heten “subscribers”, dit heet het publisher/subscriber model (zie Figuur 3-22). Een publisher kan meerdere topics aanmaken en een subscriber kan meerdere topics inlezen.



Figuur 3-22: ROS master verbindt publisher en subscriber [39]

De ROS master houdt bij welke nodes op welke topics publiceren en inschrijven. Via de master kunnen de nodes de juiste data poorten aanspreken. Wanneer de nodes elkaar gevonden hebben zullen ze direct onderling communiceren via een topic. Bij het publisher/subscriber model is er echter geen garantie dat de data aankomt bij de subscriber, dit is wel het geval bij het gebruik van “services” (zie Figuur 3-23) [29].



Figuur 3-23: Services via request reply model [39]

Services zijn vergelijkbaar met topics, alleen werken topics via het publisher/subscriber model en services via het request/reply model. In het request/reply model stuurt de node die data wilt ontvangen een request naar de node die de data bezit. Vervolgens stuurt de node die de data bezit de data door naar de node die om de data vroeg. Ten slotte stuurt de data vragende node een bevestiging dat de data ontvangen is. Hier is dus wel en garantie dat de data aankomt bij de vragende node. Ook hier loopt het verbinden van de nodes via de ROS master.

4 Systeeminfrastructuur

4.1 Systeemcommunicatie

Om een goede en veilige werking te bekomen moet het voertuig met alle belanghebbende actoren kunnen communiceren. Zo moet het voertuig bijvoorbeeld kunnen ontvangen waar het naartoe moet gaan. De communicatiestructuur van systemen met AGV's is hiervoor geanalyseerd. Dit omdat AGV's een analoge functie hebben aan het door Sint Oda gewenste voertuig. De volgende deelpunten geven de algemene actoren van een systeem, een meer uitgebreide beschrijving van één van de actoren en welke communicatiemethoden er bestaan.

4.1.1 Actoren

In het algemeen zijn er drie groepen in het systeem aanwezig. De eerste groep zijn de voertuigen met als doel het verplaatsen van personen of goederen. De tweede groep zijn de gebruikers. De groep gebruikers bestaat uit operatoren, onderhoudstechniekers en management, deze beschikken over een Human Machine Interface (HMI), tablet of computer. Deze hebben als doel de voertuigen aan te sturen of te controleren. De laatste groep is het controlesysteem. Dit is een centrale computer met als primair doel de communicatie tussen de twee vorige groepen mogelijk te maken [3], [12]. Dit gebeurt door berichten van de ene groep te ontvangen, te verwerken en deze te verzenden als een bericht dat de andere groep zal begrijpen.

Een simpel voorbeeld hiervan is het oproepen van een wagen. Een operator geeft aan dat hij een voertuig op een bepaalde locatie nodig heeft. Dit door deze locatie op zijn touchpad aan te klikken. Dit verzendt de informatie naar de centrale computer. Deze zal de locatie omzetten naar een coördinaat op de kaart en vervolgens versturen naar een beschikbaar voertuig. Tenslotte zal dit voertuig het commando ontvangen en naar deze locatie rijden.

4.1.2 Centrale computer

Zoals in deel 4.1.1 aangehaald moet een centrale computer berichten ontvangen, verwerken en naar de andere groep verzenden. Om dit op een correcte manier te doen moet de centrale computer twee onderdelen bevatten. Ten eerste een script dat ervoor zorgt dat de centrale computer op een gepaste wijze reageert op een ontvangen bericht. Het tweede onderdeel is een database, deze zal enerzijds ontvangen en verzonden berichten opslaan. Anderzijds kan de database extra informatie bevatten om het inkomende bericht op een correcte manier te verwerken. Naast het zelf maken van een script en database is het ook mogelijk om deze aan te kopen. Een voorbeeld van software die de twee onderdelen combineert is een Warehouse Management System (WMS) [5]. Deze zal onder andere een inventaris bijhouden met de locatie van elk product en een lijst bevatten die zegt wanneer een voertuig een product ergens moet leveren. Het gebruik van dit soort systemen is omwille van de kost enkel bij grotere bedrijven aan te raden.

Een database gebruiken kan een bijkomend voordeel opleveren. Het geeft de mogelijkheid om het systeem te optimaliseren. Dit gebeurt door de ontvangen en verzonden berichten, samen met extra informatie, te analyseren. Zo kan de centrale computer bijhouden hoeveel keer een gebruiker een bepaald product opvraagt en hoelang het leveren naar de gebruiker duurt. Op basis hiervan kan het bedrijf de opslag optimaliseren. Dit analyseren kan manueel gebeuren, het is echter bij grotere databases aangewezen zelf een applicatie te implementeren of er een aan te kopen. Een ander voorbeeld is dat de centrale computer na een bepaalde periode de locatie van het voertuig ontvangt, indien er iets misgaat weet de centrale computer bij benadering waar dit gebeurt en kan het de bevoegde gebruikers inlichten.

4.1.3 Communicatiemethoden

Er zijn verschillende manieren waarop de communicatie tussen de drie groepen kan gebeuren. Afhankelijk van de situatie en omgeving zijn bepaalde methoden echter niet mogelijk. Zo is het onmogelijk om het voertuig via een kabel met het systeem te verbinden of mag er geen communicatie zijn met een al bestaand netwerk omwille van veiligheidsoverwegingen. De volgende paragrafen delen communicatiemethoden op, op basis van het medium waarmee ze de informatie verzenden. Er zijn drie grote groepen te onderscheiden: radiogolven, infrarood en kabel. Het is mogelijk om verschillende methoden te combineren.

Radiogolven

Het transporteren van de data gebeurt met behulp van radiogolven, deze golven bevinden zich tussen 20 kHz en 300 GHz. Er zijn een aantal protocollen die op deze manier communiceren. Enkele voorbeelden zijn WIFI (2,4 of 5 GHz), bluetooth (2,4 GHz) en Zigbee (voornamelijk 2,4 GHz) [40]. Afhankelijk van het gekozen protocol zal onder andere verbruik, afstand en kost van de verzonden informatie anders zijn. Tabel 4-1 vergelijkt een aantal methodes van dataverzending.

Tabel 4-1: Vergelijking van radiogolf verzendmethodes [40]

Eig.	Zigbee	Z-wave	WIFI	Bluetooth
Verbruik	100 mW	1 mW	Hoog	10 mW
Afstand	100 m	30 m	1000 m	10 mW
Kost	Laag	Hoog	Middel	Zeer laag
Compat.	Zelfde maker	Verschillende makers	WIFI compatibele toestellen	Bluetooth compatibele toestellen

Infrarood

Een andere vorm van draadloos data verzenden maakt gebruik van infrarood golven, deze golven komen na de radiogolven en bevinden zich tussen 300 GHz en 430 THz. Er zijn grote verschillen tussen infrarood en radiogolven, zo kan infrarood niet door muren en dient infrarood technologie voor communicatie tussen slechts twee toestellen op korte afstand. Een voorbeeld uit de industrie is het openen van een poort wanneer er een voertuig voor staat.

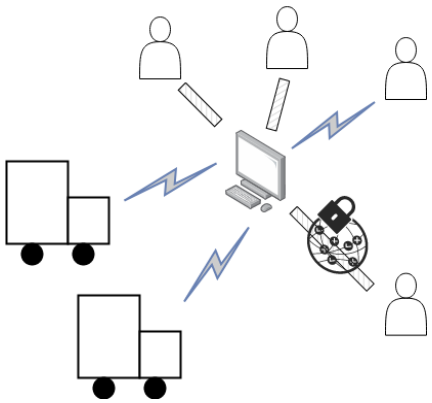
Kabel

Door elektrische pulsen over een kabel te sturen is het mogelijk om informatie te verzenden. Er zijn een aantal protocollen die op deze manier communiceren. Drie voorbeelden zijn ethernet, profibus en USB. Net zoals bij de radiogolven zullen de eigenschappen van de verzonden informatie verschillen naargelang het gebruikte protocol. Omdat de doeleinden van deze protocollen zeer verschillend zijn, was het niet mogelijk om een goede vergelijking te vinden. Zoals eerder aangehaald kunnen voertuig en systeem niet via kabel communiceren. De gebruikers en het controlesysteem kunnen dit echter wel. Het gebruik van kabels is in dit geval interessant omdat deze sneller, betrouwbaarder en veiliger zijn dan draadloos.

Voorbeeld

De meest voorkomende oplossing uit de industrie is een combinatie van ethernet en WIFI [3]. Deze zijn makkelijk te combineren omdat hun protocol analoog is opgesteld. Onderstaande afbeelding (zie Figuur 4-1) geeft een voorbeeld van een eenvoudig netwerk dat van deze oplossing gebruikmaakt. De voertuigen zullen via WIFI communiceren met de centrale computer. Deze computer is op zijn beurt verbonden met gebruikers via ethernet en WIFI. Dit afhankelijk van de functie van de gebruiker. Omwille van beveiliging mogen enkel mensen die deel uitmaken van het lokaal netwerk data sturen

naar en ontvangen van de centrale computer. Door gebruik te maken van een Virtual Private Network (VPN) kan een gebruiker alsnog via het internet deelnemen aan het lokaal netwerk. Zo kan de fabrikant van het voertuig op afstand een diagnose uitvoeren.



Figuur 4-1: Simpel netwerk op basis van ethernet en WIFI

De meeste netwerken zijn veel uitgebreider dan het hierboven beschreven voorbeeld, ook bestaan ze uit verschillende niveaus. Zo zou de centrale computer ook verbonden kunnen zijn met een in de communicatiestructuur hoger gelegen centrale computer. Deze hoger gelegen computer heeft als doel om verschillende onderliggende computers te coördineren.

4.2 Gedragsalgoritmen

Afhankelijk van het tijdstip zal het voertuig andere berichten naar het systeem sturen en zal het anders met de ontvangen berichten moeten omgaan. Zo moet het voertuig zijn locatie niet naar de centrale computer sturen wanneer het voor langere tijd stilstaat of zal het voertuig een bepaald nieuw bevel negeren als het nog bezig is met een vorige. Om de taaksequentiëring van het voertuig op een eenvoudige manier te implementeren zijn er echter hulpmiddelen nodig.

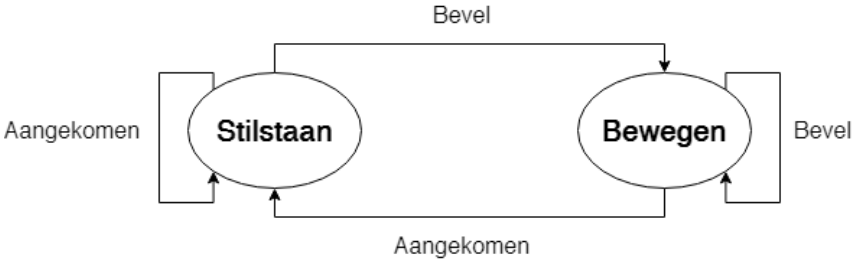
Volgend voorbeeld maakt deze noodzaak duidelijk. Het voertuig is naar een bestemming aan het rijden. Tijdens het rijden moet het voertuig met verschillende variabelen rekening houden zoals batterijstatus, toestand van de weg, etc. Elk van deze condities zal naar een ander deel van het gedrag leiden. Waardoor de complexiteit snel stijgt. Indien een conditie niet geïmplementeerd is, bijvoorbeeld het induwen van een noodstop, vereist het toevoegen ervan het zoeken naar en aanpassen van de delen van het gedrag die deze conditie nodig hebben. Dit proces is traag en gevoelig voor fouten.

Er zijn gedragsalgoritmen ontwikkeld om deze problemen aan te pakken. Deze algoritmen zijn subsumption architecture, teleo-reactive programs, decision trees, finite state machines (FSM's) en behavior trees (BT's). Wat volgt is het basisidee van de twee meest prominente hulpmiddelen, FSM's en BT's. Hierna volgt een vergelijking van deze twee om het beste hulpmiddel te kiezen voor het implementeren van het gedrag van het voertuig. Tenslotte is er naar tools in ROS gezocht om het gekozen gedragsalgoritme te implementeren.

4.2.1 Finite state machine

Een FSM zal de werking van het voertuig beschrijven als een verzameling van toestanden, de overgang tussen de verschillende toestanden heten transities. Er kan op elk moment maar één toestand actief zijn [41]. Wanneer er een transitie is, leidt dit ook tot een bepaalde actie. Figuur 4-2 toont een simpel voorbeeld van de werking. Tabel 4-2 geeft uitleg bij de verschillende onderdelen. In dit voorbeeld zal het voertuig tussen de toestanden "Stilstaan" en "Bewegen" wisselen door gebruik te maken van de transities "Bevel" en 'Aangekomen". Afhankelijk van met welke transitie een nieuwe

toestand bereikt is zal er een andere actie optreden. Zo zal het voertuig geen actie ondernemen indien het in de toestand “Stilstaan” de transitie “Aangekomen” ontvangt. Ontvangt het voertuig echter de transitie “Bevel” dan zal de actie “Start navigatie” optreden.



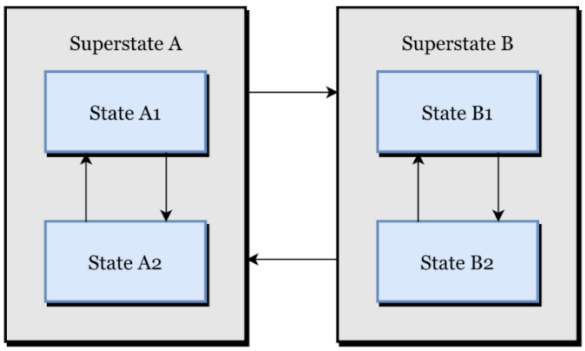
Figuur 4-2: Voorbeeld van simpele FSM

Tabel 4-2: Uitleg bij Figuur 4-2

Huidige toestand	Transitie	Nieuwe toestand	Actie
Stilstaan	Bevel	Bewegen	Start navigatie
Stilstaan	Aangekomen	Stilstaan	/
Bewegen	Bevel	Bewegen	/
Bewegen	Aangekomen	Stilstaan	Zet motor uit

Hierarchical finite state machine

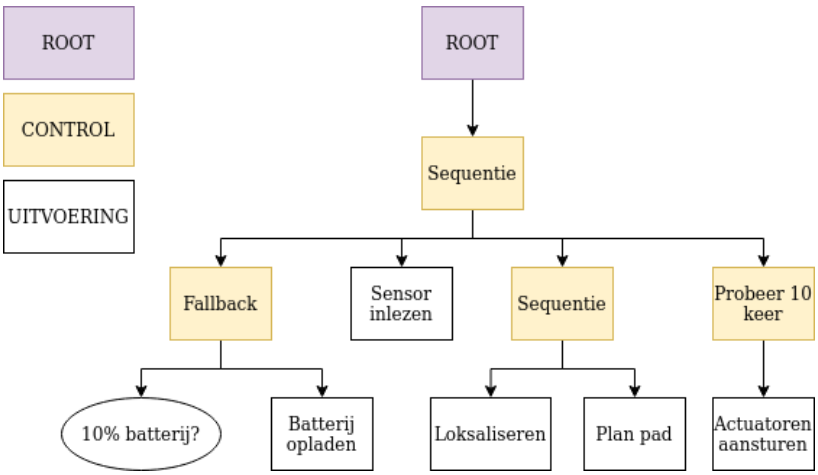
Een variant van de FSM is de hierarchical finite state machine (HFSM). Hier kan iedere toestand een aantal subtoestanden hebben. Toestanden die over subtoestanden beschikken krijgen de naam supertoestanden. Deze methode verlaagt het totaal aantal benodigde transities aangezien er enkel verbindingen nodig zijn tussen subtoestanden van dezelfde groep en de verschillende supertoestanden [42]. Figuur 4-3 geeft hiervan een voorbeeld. Bij het gebruik van een FSM zouden de toestanden A1, A2, B1 en B2 allemaal aan elkaar verbonden zijn. HFSM laat ons echter toe de gelijkaardige toestanden in de supertoestanden A en B te implementeren en enkel deze met elkaar te verbinden. Het is hiernaast ook mogelijk om subtoestanden de eigenschappen van hun supertoestand mee te geven.



Figuur 4-3: Voorbeeld van de HFSM [41, p. 8]

4.2.2 Behavior tree

Een BT zal de werking van het voertuig beschrijven aan de hand van nodes in een boomstructuur. Figuur 4-4 is een simpel voorbeeld.

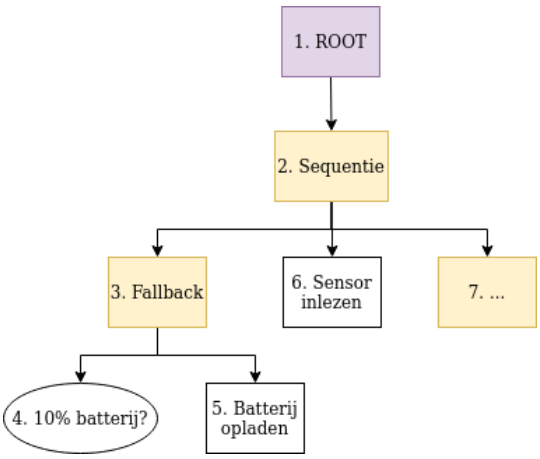


Figuur 4-4: Voorbeeld van simpele BT

Uit Figuur 4-4 vallen volgende zaken af te leiden. Er zijn drie verschillende soorten nodes, de "root", controle nodes en uitvoeringsnodes. Wanneer twee nodes verbonden zijn in de boomstructuur is de bovenste node de ouder en de onderste het kind. Elke node kan ouder van meerdere onderliggende nodes zijn, maar is slecht het kind van één bovenliggende node. Er zijn twee uitzondering. De eerste uitzondering is de root, deze heeft slechts één kind en geen ouder. De tweede uitzondering zijn de uitvoeringsnodes, deze hebben geen kinderen.

Root

De root is altijd de bovenliggende node. Hier begint ook de BT. Het stuurt met een bepaalde frequentie een "tick" naar zijn kind, deze frequentie is afhankelijk van hoe snel de computer de BT kan doorlopen. Deze tick start de taak van het kind. Een kind zal na uitvoering van zijn taak één van drie mogelijke statussen terug naar zijn ouder sturen, "succes" als de taak goed uitgevoerd is, "fout" als de taak gefaald is of "bezig" als de taak nog niet volledig uitgevoerd is [41]. Het activeringssignaal zal altijd van links naar rechts de nodes aflopen doorheen de structuur. Het is dan ook belangrijk dat de cruciale onderdelen, zoals de obstakelherkenning, als eerste aan bod komen. Figuur 4-5 geeft een voorbeeld van de volgorde van activering.



Figuur 4-5: Volgorde van activeringssignaal

Controle nodes

Afhankelijk van de situatie mogen niet alle nodes een activeringssignaal krijgen. Zo mag het voertuig in figuur 4-5 niet de opdracht krijgen om de batterij op te laden zolang deze meer dan 10% opgeladen is. Om te controleren welk kind het activeringssignaal ontvangt zijn controle nodes nodig. Wanneer een controle node een activeringssignaal ontvangt zal het van links naar rechts zijn kinderen activeren.

Op basis van de status van de kinderen bepaalt de controle node welke status het naar zijn ouder stuurt. De basis controle nodes zijn sequentie nodes, fallback nodes en decorator nodes [42].

Sequentie nodes zullen hun kinderen afgaan, indien alle kinderen succesvol zijn zal het ook succes teruggeven. Vanaf dat één van zijn kinderen fout is slaat de sequentie node al zijn overige kinderen over en geeft het fout terug aan zijn ouder. Als één van zijn kinderen zegt dat het met de actie bezig is zal de sequentie node ook bezig teruggeven aan zijn ouder. De sequentie node is bedoeld om een structuur af te lopen waarbij iedere node afhankelijk is van de succesvolle beëindiging van de vorige.

Fallback nodes zullen ook hun kinderen afgaan, indien één kind succesvol is zal het al de overige kinderen overslaan en succes teruggeven. Als alle kinderen fout zijn geeft de fallback node een fout terug aan zijn ouder. Vanaf dat er één kind bezig is zal het de overige kinderen overslaan en ook bezig teruggeven aan zijn ouder. De fallback node krijgt ook wel de naam selector dit omdat het als doel heeft om een aantal verschillende strategieën af te lopen tot er één succesvol is.

Decorator nodes hebben maar één kind, ze kunnen een aantal verschillende dingen doen. Zoals het inverteren van de status van een kind, deze wisselt succes naar fout en andersom. Andere functionaliteiten zijn de mogelijkheid om een node meerdere keren uit te voeren tot deze een bepaalde status stuurt of het forceren van een succes of fout. De decorator node is bedoeld om op een simpele manier een bepaalde functionaliteit aan één node toe te voegen.

Uitvoeringsnodes

De uitvoeringsnodes zijn de laagste nodes van elke tak, om deze reden hebben ze zelf geen kinderen. Deze uitvoeringsnodes zijn enerzijds de acties die het systeem moet uitvoeren [42]. Denk aan het plannen van een pad. Anderzijds kunnen deze uitvoeringsnodes ook vragen zijn. Bijvoorbeeld “is de batterij bijna leeg?”. Deze nodes zullen dus niets veranderen aan de werking van het systeem maar enkel een ja (succes) of nee (failure) antwoorden op de gestelde vraag [42].

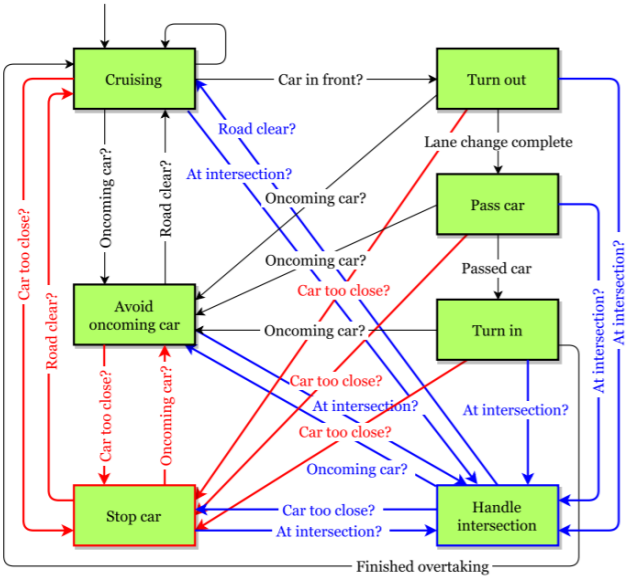
4.2.3 Vergelijking

Volgende parameters zijn gebruikt om het hulpmiddel van de taaksequentië te bepalen:

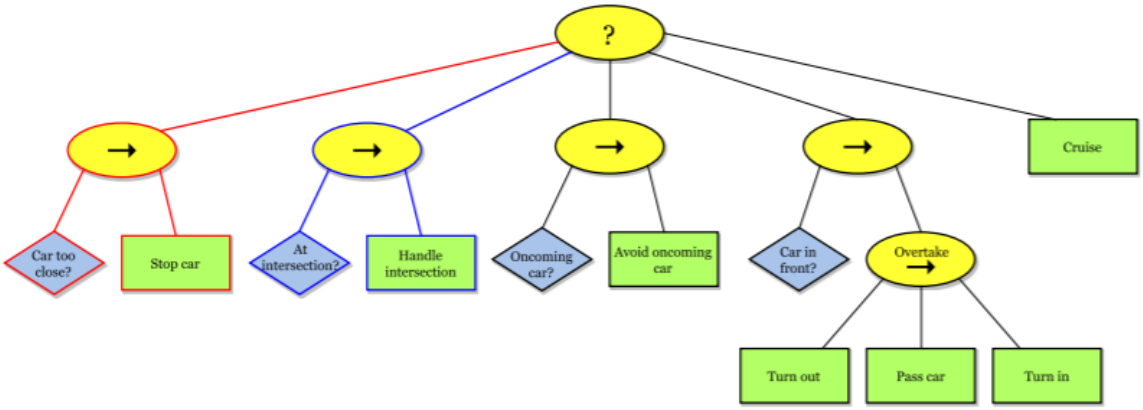
- overzichtelijkheid;
- aanpasbaarheid;
- implementatietijd;
- Overige opmerkingen.

Overzichtelijkheid

De overzichtelijkheid van een FSM daalt snel in functie van de grootte van het aantal uit te voeren acties. Bij complexere taaksequentiëringen ontstaan er “spaghettistructuren” omdat veel verschillende toestanden via transities verbonden zijn [42]. Dit is onder meer het geval voor het gedrag van een zelfrijdende wagen. Deze moet, omwille van de veiligheid, zeer gemakkelijk tussen verschillende toestanden kunnen wisselen. Dit vraagt een groot aantal transities. Dankzij de supertoestanden bij een HFSM zijn er minder transities, dit verbetert de overzichtelijkheid ten opzichte van de traditionele FSM. Toch zal bij de HFSM de overzichtelijkheid ook afnemen bij grotere programma’s [42]. De BT zal zijn overzichtelijkheid zeer goed bewaren, zelfs bij complexere problemen. Dit komt omwille van de ouder-kind structuur die het aantal mogelijke verbindingen beperkt. Volgende figuren maken dit zeer duidelijk. Figuur 4-6 geeft het rijgedrag van een gesimuleerde auto weer met een FSM. Figuur 4-7 toont hetzelfde gedrag geïmplementeerd met een BT. Bij figuur 4-7 is het vraagteken een fallback node en de pijl naar rechts een sequentie node.



Figuur 4-6: Rijgedrag in FSM [41, p. 31]



Figuur 4-7: Rijgedrag in BT [41, p. 32]

Aanpasbaarheid

Het aanpassen van een FSM is omslachtig. Bij het wijzigen van een FSM is het nodig om alle verbonden toestanden na te kijken. Dit geeft opnieuw extra problemen bij autonome wagens omwille van de grote verbondenheid van de toestanden. Een HFSM gebruiken zal deze aanpasbaarheid licht verbeteren omwille van het kleinere aantal transities. Toch zal de modulariteit ook bij een HFSM een uitdaging vormen [42]. De BT is, dankzij de ouder-kind structuur, veel gemakkelijker aan te passen.

Implementatietijd

Een FSM is een ouder hulpmiddel. Gevolg is dat er meer kennis is en de beschikbare tools ook gemakkelijker en meer compleet zijn. De werking is ook zeer intuïtief en makkelijk te begrijpen [42]. De HFSM deelt de eigenschappen van de FSM, maar omwille van de super-sub structuur moet de gebruiker beter nadenken over de opbouw van de verschillende lagen. Hierdoor ligt de implementatietijd iets hoger. De BT is een recentere techniek dan de FSM, verder zijn er veel meer verschillende symbolen en begrippen om rekening mee te houden. Dit zorgt ervoor dat de implementatietijd van een BT, vooral voor beginners, groter is dan die bij het opstellen van een gelijkaardige FSM of HFSM [41].

Overige opmerkingen
De BT bezit nog enkele kleinere voordelen ten opzichte van de FSM. Zo is het mogelijk om code te hergebruiken [42]. Bij een BT is het ook mogelijk om verschillende nodes tegelijkertijd een activeringssignaal te sturen door zogenaamde parallelle nodes [41]. Meerdere toestanden tegelijk in een FSM activeren is niet mogelijk.

Besluit
Op basis van de vergelijking is geconcludeerd dat bij eenvoudige problemen de FSM de meer gangbare oplossing is, naarmate de complexiteit stijgt is het beter om over te schakelen naar een HFSM. Tenslotte is de BT een betere oplossing voor zeer complexe taaksequenties. Er is op basis van deze bevinding besloten om een BT te gebruiken, het volledige gedrag van een autonoom voertuig bestaat namelijk uit veel verschillende onderdelen die nauw met elkaar verbonden zijn om een veilige werking te garanderen.

4.2.4 Tools
Er zijn twee tools gevonden om een BT in ROS te implementeren, behavior_tree en BehaviorTree.cpp. Er zijn drie grote verschillen tussen de gevonden tools. De eerste is dat behavior_tree implementeren mogelijk is in zowel C++ als python, BehaviorTree.cpp maakt enkel gebruik van C++. Het tweede verschil is dat BehaviorTree.cpp beschikt over meer functionaliteiten. Zo beschikt de tool over een aantal variaties op de standaard sequentie en fallback nodes. Ten slotte is het ook mogelijk om de grafische tool Groot te gebruiken in combinatie met BehaviorTree.cpp. Deze grafische tool vergemakkelijkt het ontwikkelen en controleren van een BT. Het opslaan van de boomstructuren in BehaviorTree.cpp gebeurt normaal in een .xml bestand, met Groot is het mogelijk om dit bestand grafisch aan te maken. Verder is het mogelijk om te zien welke nodes actief zijn terwijl het programma actief is. Er is omwille van de tweede en derde reden besloten om gebruik te maken van BehaviorTree.cpp.

4.3 Behavior tree tool
4.3.1 Controle nodes
BehaviorTree.cpp onderscheidt drie verschillende types sequentie nodes. Deze zijn sequentie, sequentieSter en reactieveSequentie. De algemene werking van de sequentie nodes is hetzelfde als die beschreven in deel 4.2.2. Ze zullen dus allemaal hun kinderen doorlopen tot ze een kind tegenkomen die bezig of fout als status geeft. Wanneer dit gebeurt zal elke sequentie, wanneer de volgende tick binnenkomt, iets anders doen. Zo kan de sequentie bij het eerste kind beginnen (Herstarten) of begint de sequentie bij het kind dat de vorige tick bezig of fout als status gaf (tick opnieuw), alle voorgaande kinderen komen in dit geval niet meer aan bod [43]. Een overzicht van het gedrag van elke sequentie is in Tabel 4-3 te vinden.

Tabel 4-3: Gedrag van sequenties bij volgende tick

Type sequentie	Kind geeft fout	Kind geeft bezig
sequentie	Herstarten	Tick opnieuw
sequentieSter	Tick opnieuw	Tick opnieuw
reactieveSequentie	Herstarten	Herstarten

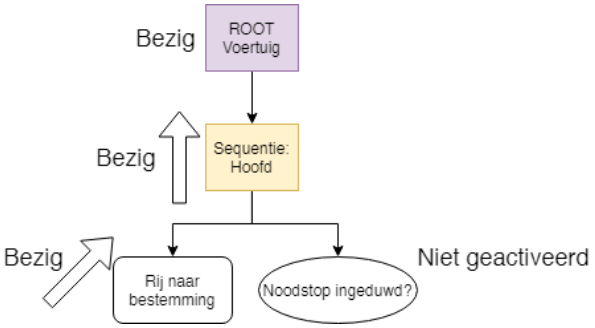
Er zijn in BehaviorTree.cpp ook twee verschillende types fallback nodes. Deze zijn fallback en reactieveFallback. De algemene werking van de fallback nodes is hetzelfde als die beschreven in deel 4.2.2. Ze zullen dus beide al hun kinderen afgaan tot er één succes als status geeft, ze allemaal falen of er één bezig is. Indien een kind bezig als status geeft zal het gedrag de volgende tick afwijken.

Fallback zal direct naar het kind met de bezig status gaan en dus alle voorgaande kinderen overslaan. reactieveFallback zal terug van het begin beginnen.

4.3.2 Coroutines

De reactiviteit van het voertuig is afhankelijk van de executietijd van de BT. Dit wilt zeggen dat geen enkele node zo lang mag duren dat deze het real-time uitvoeren van de taaksequentie in gedrang brengt. Echter zal dit niet altijd mogelijk zijn. Zo zou het uitvoeren van de pad planning veel tijd in beslag kunnen nemen. In deze situatie is een coroutine noodzakelijk. Hierdoor is het mogelijk om de taak na een bepaalde tijd een halt toe te roepen. De node geeft dan bezig als status aan zijn ouder. De volgende tick zal de node verdergaan waar het de vorige tick halt hield. Er zijn twee manieren om van een node een coroutine te maken in BehaviorTree.cpp, AsyncActionNode of CoroActionNode [43].

Het gebruiken van coroutines kan echter gevaarlijk zijn voor de BT. Dit omwille van de werking van sequentie en fallback nodes. Zowel sequentie als fallback nodes zullen op dezelfde manier reageren als hun kind bezig als status geeft. Ze slaan hun overige kinderen over en geven zelf bezig als status terug. Dus voor een bepaalde tick, stopt het aflopen van de boom wanneer één kind bezig als status geeft. Figuur 4-8 toont in een simpel voorbeeld waarom dit gevaarlijk is.



Figuur 4-8: Fout gebruik van coroutine

Er zijn twee uitvoeringsnodes. De eerste is de node “Rij naar bestemming”, deze geeft succes wanneer het voertuig is aangekomen en fout indien het voertuig er niet naartoe kan gaan. Aangezien deze node, zolang het voertuig naar de bestemming aan het rijden is, de rest van de BT ophoudt is er een coroutine van gemaakt die na een bepaalde tijd zijn werking beëindigd en bezig als status geeft. Dit geeft tot gevolg dat de node “Rij naar bestemming” altijd bezig is zolang het voertuig rijdt. Hierdoor komt de tweede node “Noodstop ingeduwd?” tijdens het rijden nog altijd niet aan bod, dit mag niet gebeuren. Om deze reden moet de opbouw van de BT zodanig zijn dat coroutines de veiligheid niet in het gedrang brengen. In dit voorbeeld kan dit simpelweg door de twee uitvoeringsnodes van positie te wisselen. Deze twee nodes wisselen is ook logisch aangezien de vraag of de noodstop is ingeduwd belangrijker is dan het rijden naar de bestemming zoals beschreven in deel 4.2.2.

4.3.3 Blackboard

Het is in BehaviorTree.cpp ook mogelijk om data van een node beschikbaar te maken voor andere nodes. Dit gebeurt met een systeem genaamd Blackboard [43]. Een node kan informatie op dit Blackboard schrijven door gebruik te maken van een output poort, deze poort bevat de informatie die de node publiek wil maken en een sleutel om deze informatie te identificeren. Nodes die deze informatie willen gebruiken kunnen in een input poort deze sleutel ingeven. Zij kunnen hierna gebruik maken van de informatie verbonden met deze sleutel. Andere systemen, zoals Unity, maken ook gebruik van dit systeem om in een BT data uit te wisselen. Naast nodes kan de gebruiker zelf ook data beschikbaar maken op Blackboard.

4.3.4 Subtree

In deel 4.2.3 is aangegeven dat het mogelijk is om delen van de BT te hergebruiken. Dit is nuttig wanneer dezelfde grote delen vaker in de BT voorkomen. In BehaviorTree.cpp gebeurd dit met behulp van subtrees. De subtree is een boom die losstaat van de hoofdboom, dit wil zeggen dat deze in plaats van een root een subtree ID als bovenste node heeft. Het oproepen van de subtree gebeurt aan de hand van dit ID. De blackboards van verschillende bomen zijn niet met elkaar verbonden, dit om naamconflicten te voorkomen. Dit wil echter zeggen dat het niet mogelijk is om een poort die in de hoofdboom gemaakt is in een subtree te gebruiken en omgekeerd. Het is mogelijk om poorten van verschillende bomen te verbinden door gebruik te maken van “setBlackboard”. Een voorbeeld hiervan is terug te vinden in Figuur 7-17.

4.3.5 Groot

Het gebruik van Groot om de BT van het voertuig grafisch op te stellen bleek niet mogelijk. Om dit pakket naar een executable om te zetten is namelijk een applicatie nodig die niet werkt voor versies van Ubuntu na de versie 16.04 (Xenial). Dit was een probleem omdat in onze masterproef Ubuntu versie 18.04 (Bionic) gebruikt is. Ook Ubuntu Xenial gebruiken gaf problemen bij de installatie. Er is daarom besloten om Groot achterwege te laten en de BT met draw.io op te stellen. Dit omdat het met draw.io makkelijk is om grafisch structuren op te stellen.

5 Conceptuele studie

5.1 Van den Kroonenberg

In deze studie is de “van den Kroonenberg” ontwerpstructuur toegepast. Deze structuur berust op eisen waar de oplossing aan moet voldoen. Voor deze masterproef zijn de eisen opgesteld in samenwerking met de opdrachtgever Sint Oda. Er zijn schalingsfactoren toegevoegd aan de eisen, deze beïnvloeden het belang van bepaalde eisen. Functies zijn handelingen of zaken om te voldoen aan de gestelde eisen, een reeks goede functies zal leiden tot een werkende oplossing. Het onderzoek vertrekt van een zeer algemene idee. Elke volgende stap definieert de functies specifieker.

5.2 Voorwaarden en eisen

Zoals verteld in 5.1 zijn er schalingsfactoren toegekend aan de eisen. Deze liggen tussen één en vier. Eén is minst belangrijk en vier is meest belangrijk. Een opsomming van de eisen, samen met hun schalingsfactor en beschrijving, is te vinden in Tabel 5-1.

Tabel 5-1: Eisentabel met schalingsfactor

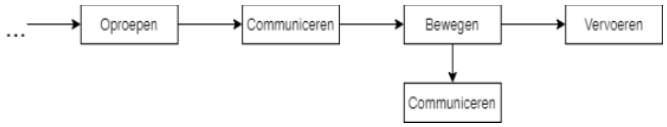
EISEN	Factor	Beschrijving
Beschikbaarheid	1	De oplossing moet tussen 8u30-12u30 en 13u30-19u beschikbaar zijn
Kostprijs	1	Het budget van de oplossing is maximaal €10.000
Kans op fout	2	De oplossing mag slechts 1% van zijn werktijd stilstaan omwille van een fout (ongeveer een halfuur per week)
Levensduur	2	De oplossing heeft een levensduur van maximaal 20 jaar
Onderhoud	2	De oplossing mag maximaal drie dagen per jaar niet beschikbaar zijn door gepland onderhoud
Snelheid	3	De snelheid van de oplossing moet tussen de 5 km/u en 12 km/u liggen
Weersvereisten	3	Het percentage van storingen, veroorzaakt door weersomstandigheden, op de metingen moet onder 20% blijven
Netwerkbelasting	3	De oplossing mag maximaal 1% van de bestaande bandbreedte van het netwerk in beslag nemen
Aanwezigheid begeleider	4	Er moet vanaf één bewoner minstens één begeleider aanwezig zijn
Wegdek	4	Verplaatsing moet zowel op beklinkerde als asfaltweg mogelijk zijn
Rolstoel capaciteit	4	De oplossing moet minstens 4 rolstoelen tegelijkertijd verplaatsen van punt A naar B
Wegbreedte	4	De oplossing moet over wegen van minimaal twee meter breed kunnen rijden
Opstapmogelijkheden	4	Er moeten tussen de 20 en 30 “bushaltes” voorzien zijn waar de oplossing de bewoners kan ophalen
Belasting begeleider	4	Fysieke belasting moet niet zwaarder zijn dan nu en er moet een reductie van belastingstijd zijn van minimaal 30%
Gemak zorgbehoevenden	4	Comfort, veiligheid en aantal handelingen van de zorgbehoevenden mogen niet erger zijn dan in de huidige situatie
Veiligheidsnormen	4	Het aantal veiligheidsnormen die Sint Oda moet implementeren zodat het voertuig een CE-keuring zou verdienen

5.3 Technische installatie

Dit is de meest algemene stap in de conceptuele studie. Om elke mogelijke (deel)oplossing te evalueren is het belangrijk om van een algemeen idee te vertrekken. Dit voorkomt het vergeten van potentieel interessante (deel)oplossingen.

5.3.1 Functies en functieblokschema

Op een abstract niveau moet de oplossing het functieblokschema van Figuur 5-1 volgen.



Figuur 5-1: Functieblokschema van de technische installatie

Wanneer de cyclus tweemaal doorlopen is zijn de zorgbehoevenden van de beginlocatie naar de eindbestemming gebracht. De eerste cyclus is om de zorgbehoevenden op te halen en de tweede om ze naar hun bestemming te brengen. Onder "Oproepen" behoort het doorgeven van een bestemming aan de oplossing. Onder "Communiceren" valt relevante data zoals huidige locatie, reistijd en route doorsturen naar belanghebbende organen binnen het systeem. Onder "Bewegen" valt het type voertuig. Onder "Vervoeren" valt de methode van het transporteren van de zorgbehoevenden.

5.3.2 Bespreking mogelijke opties per functie

De verschillende opties voor de functies bevinden zich in Tabel 5-2.

Tabel 5-2: Opties voor functies voor de technische installatie

	Optie 1	Optie 2	Optie 3	Optie 4	Optie 5
Oproepen	Vast schema	Bellen/sms	HMI	Knop	
Communiceren	Internet	Manueel ingeven	Bluetooth		
Bewegen	Bestaande wagen	AGV	Mensbekrachtigd	Semi-aut. elektr. trekker	Huifkar
Communiceren	Internet	Manueel ingeven	Bluetooth		
Vervoeren	Rolstoelen koppelen	Kar	Frame		

De opties voor de functie "Oproepen" zijn.

- Vast schema, de oplossing heeft een vast schema met tijdstippen en corresponderende locaties.
- Bellen/sms, door middel van een sms of telefoongesprek de oplossing kunnen aansturen.
- HMI, Human Machine Interface is in feite een soort tablet. Eén HMI is verbonden aan de oplossing zelf, de anderen bevinden zich in de leefgroepen of andere haltes. Het is mogelijk de bestaande tablets van de leefgroepen te gebruiken hiervoor. De HMI kan naast signalen sturen ook informatie tonen zoals wachttijd en huidige locatie van de oplossing.
- Knop, elke halte bevat een knop. Op de oplossing bevinden zich ook knoppen om de gewenste locatie in te geven.

De opties voor de functie "Communiceren" zijn.

- Internet, communicatie met alle organen via het internet.
- Manueel ingeven, dit werkt enkel bij het vaste schema. De begeleiders schrijven in een planning op wanneer ze het voertuig willen gebruiken.
- Bluetooth, communicatie kan over bluetooth. We gebruiken deze optie echter niet omdat bluetooth moeilijk te gebruiken is in het geval van meerzijdige communicatie en grote afstanden.

De opties voor de functie “Bewegen” zijn.

- Bestaande wagen, de oplossing gebruikt de bestaande wagen, die al aanwezig is op Sint Oda, die autonoom gemaakt is.
- AGV, de oplossing gebruikt een nieuw aangekochte AGV van een gespecialiseerde fabrikant.
- Mensbekrachtigd, de oplossing berust enkel op de fysieke inspanning van mensen.
- Semi-autonome trekker, De oplossing gebruikt de elektrische trekker uit deel 2.2 waarop nog enkele veiligheidssensoren zijn aangebracht om een veilige werking te garanderen. Zo kunnen ze stoppen of vertragen wanneer de veiligheidssensoren een aankomende botsing detecteren. Deze oplossing is echter niet volledig autonoom.
- Huifkar, de oplossing gebruikt een huifkar uit deel 2.3 die autonoom is gemaakt. De huifkar beschikt al over een transportruimte om zorgbehoevenden te vervoeren.

De opties voor de functie “Vervoeren” zijn.

- Rolstoelen koppelen, de rolstoelen achter elkaar koppelen en vervolgens vast maken aan de oplossing.
- Kar, de zorgbehoevenden samen met hun rolstoelen vervoeren in een speciale kar.
- Frame, het gebruik van een frame met mogelijkheden om rolstoelen te bevestigen.

De opties uit Tabel 5-2 opties zijn samengebracht in verschillende structuren in Tabel 5-3. Hierna volgt een beschrijving en evaluatie van de verschillende structuren.

Tabel 5-3: Structuren voor de technische installatie

	Struct. 1	Struct. 2	Struct. 3	Struct. 4	Struct. 5
Oproepen	HMI	Knop	Bellen/sms	Vast schema	HMI
Communiceren	Internet	Internet	/	/	Internet
Bewegen	Bestaande wagen	AGV	Mensbekrachtigd	Semi-aut. elektr. trekker	Huifkar
Communiceren	Internet	Internet	/	/	Internet
Vervoeren	Kar	Kar	Rolstoelen koppelen	Frame	/

5.3.3 Bespreking structuren

Bespreking structuur 1

Een begeleider roept de autonoom omgebouwde bestaande wagen op via de HMI in de leefgroep. De wagen communiceert zijn route, reistijd en locatie aan de belanghebbende organen van het systeem via het internet. Bij aankomst aan de leefgroep plaatst de begeleider de zorgbehoevenden in de speciale kar die achter de bestaande wagen is bevestigd. De cyclus begint weer opnieuw. De begeleider kan de gewenste bestemming ingegeven op de HMI die aan de bestaande wagen bevestigd is, de andere HMI’s zullen dan aangeven dat de bestaande wagen in gebruik is. De bestaande wagen zal dan weer de route, reistijd en locatie door communiceren en zich naar de gewenste bestemming bewegen. Eenmaal aangekomen op de gewenste bestemming begeleidt de begeleider de zorgbehoevenden uit de speciale kar. Indien de begeleider niet meer naar een andere locatie moet kan hij/zij aangeven dat de opdracht voltooid is. De bestaande wagen is terug beschikbaar voor de volgende oproep.

Bespreking structuur 2

Deze structuur komt sterk overeen met structuur 1. Het grootste verschil is het gebruik van een

aangekochte AGV. Ook gebruikt deze structuur knoppen in plaats van HMI's. Voor de rest blijft het principe hetzelfde.

Bespreking structuur 3
Een begeleider belt of sms't naar een persoon wiens taak het is om de fysieke inspanning te leveren. Deze persoon zal naar de leefgroep komen, de rolstoelen aan elkaar koppelen en door middel van fysieke inspanning de zorgbehoevenden van beginlocatie naar eindbestemming brengen.

Bespreking structuur 4
De begeleiders van de leefgroepen beschikken elk over een semi-autonome elektrische trekker zoals in deel 2.2. Ze kennen de tijdstippen waarop ze de zorgbehoevenden moeten vervoeren dankzij een vast schema. De begeleider bevestigd de rolstoelen aan het frame en transporteert de zorgbehoevenden met behulp van de elektrische trekker van beginlocatie naar eindbestemming.

Bespreking structuur 5
Deze structuur komt overeen met structuur 1. Hier is het verschil het gebruik van de huifkar uit deel 2.3 die autonoom is gemaakt.

5.3.4 Keuze en conclusie
Om te bepalen welke structuur het meest geschikt is voor de oplossing is een score toegekend aan hoe goed de structuur voldoet aan elke eis. Hiervoor is er eerst een selectie gemaakt van relevante eisen voor dit gedeelte van het onderzoek. De punten gaan van één (heel slecht) tot vier (heel goed). Een structuur die een 0 scoort op één van de eisen is in het verdere onderzoek verworpen. Het totaal resultaat brengt ook de schalingsfactoren uit Tabel 5-1 in rekening. De resultaten zijn te vinden in Tabel 5-4. Het is hierbij belangrijk om te vermelden dat het zeer waarschijnlijk is dat alle structuren het opgegeven budget zullen overschrijden. De score van iedere structuur bij de eis "Kostprijs" is daarom bepaald door de structuren enkel met elkaar te vergelijken en niet met de eis op zich.

Tabel 5-4: Keuze van de structuur voor de technische installatie

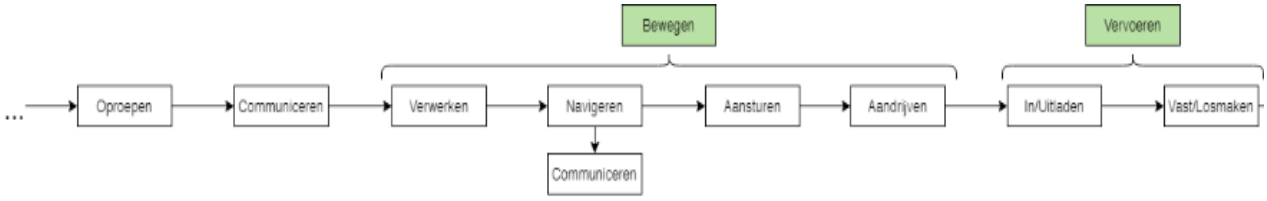
	Struct. 1	Struct. 2	Struct. 3	Struct. 4	Struct. 5	Factor
Beschikbaarheid	3	3	4	3	3	1
Kostprijs	2	0	4	3	1	1
Kans op fout	3	4	2	2	3	2
Levensduur	1	2	3	2	2	2
Onderhoud	1	2	4	3	1	2
Snelheid	4	4	1	2	4	3
Weersvereisten	2	2	3	3	2	3
Netwerkbelasting	3	3	4	4	3	3
Wegdek	4	4	2	4	4	4
Rolstoel capaciteit	4	4	2	4	4	4
Belasting begeleider	4	4	0	3	4	4
Gemak zorgbehoevenden	4	4	2	2	4	4
Veiligheidsnormen	2	4	4	3	2	4
Totaal	114	126	90	111	115	

De hoogst scorende structuur is structuur 2. Een aangekochte AGV beschikt over de juiste keuringen voor veilig gebruik. Ook is het onderhoud inbegrepen hierdoor is de levensduur lang. Aangezien een aangekochte AGV onbetaalbaar is, is er gekozen om deze structuur te verwerpen. Op de tweede en derde plaats bevinden zich respectievelijk structuur vijf en structuur één. Aangezien het verschil van de resultaten zeer klein is, en beide oplossingen veel op elkaar lijken zijn beide structuren evenwaardig. In de volgende stappen is dus uitgegaan van het gebruik van een huifkar of de bestaande wagen met speciale kar, communicatie via internet en het gebruik van HMI's.

5.4 Samengesteld werktuig

5.4.1 Functies en functieblokschema

De eerste stap naar een concretere uitwerking is het onderverdelen van de functie "Bewegen" naar "Verwerken", "Navigeren", "Aansturen" en "Aandrijven". Ook "Vervoeren" is onderverdeeld in "In/Uitladen" van de zorgbehoevenden en "Vast/Losmaken" van de rolstoelen. Het nieuwe schema is te zien op Figuur 5-2.



Figuur 5-2: Functieblokschema van het samengesteld werktuig

Onder "Verwerken" behoort het toestel dat instaat voor de berekeningen die de AGV moet maken. "Navigeren" is de methode die het voertuig gebruikt om te bewegen van beginlocatie naar eindbestemming. Onder "Aansturen" valt het toestel die de berekeningen uit de verwerking omzet in stuursignalen voor de aandrijvingen. De aandrijvingen manipuleren het stuur en de pedalen van de bestaande wagen of huifkar. "Vervoeren" is opgedeeld in de functies "In/uitladen" en "Vast/losmaken" van de zorgbehoevenden in hun rolstoel.

Tijdens het doorlopen van de eerste cyclus (het ophalen van de zorgbehoevenden) loopt de cyclus precies zoals in Figuur 5-2. Bij de tweede cyclus (het brengen van de zorgbehoevenden naar de eindbestemming) zijn de functies "In/uitladen" en "Vast/Losmaken" omgedraaid.

5.4.2 Bespreking mogelijke opties per functie

De opties voor het "Oproepen" en "Communiceren" zijn vastgelegd in 5.3. De nieuwe functies en hun opties zijn weergeven in Tabel 5-5.

Tabel 5-5: Opties voor functies voor het samengesteld werktuig

	Optie 1	Optie 2	Optie 3	Optie 4
Oproepen	HMI			
Communiceren	WIFI			
Verwerken	Ind. computer (NUC)	Centraal system	Raspberry pi	Laptop
Navigeren	Strips	Bakens + natuurlijk	Bakens	Natuurlijk (SLAM)
Communiceren	WIFI			
Aansturen	PLC	Ethernet/WIFI verbinding met PC	Arduino	I/O kaarten op PC
Aandrijven	Hydraulisch	Elektrisch	Pneumatisch	Mensbekrachtigd
In/uitladen	Mensbekrachtigd	Mensbekrachtigd + Elektr.	Mensbekrachtigd + Hydr.	Mensbekrachtigd + Pneum.
Vast/Losmaken	Klem	Haken	Op stoel zetten	

De mogelijke opties voor “Verwerking” zijn.

- Een industriële computer of NUC, de verwerking gebeurd door een NUC die bevestigd is aan het voertuig. het voertuig verwerkt dus zelf de sensordata en data van andere organen. Een NUC draait zeer stabiel maar heeft geen bevestigd scherm, muis of toetsenbord, al zijn deze wel makkelijk extern toe te voegen. Wel beschikt de NUC over een groot processorvermogen en betrouwbaarheid.

- Centraal systeem, het voertuig stuurt sensordata en andere gegevens door naar een centrale computer die de berekeningen maakt en vervolgens de uitkomst terug naar het voertuig stuurt. het voertuig zelf bevat dus geen computer die aan verwerking kan doen. Of het gebruik van een centraal systeem snel en betrouwbaar genoeg is, is niet bewezen.

- Raspberry Pi, de verwerking gebeurd door een kleine Linux gebaseerde microcomputer. Het voertuig verwerkt dus zelf de sensordata en data van andere organen. Een Raspberry Pi heeft een beperkt processorvermogen en is minder betrouwbaar. Het grootste voordeel is de kostprijs.

- Laptop, hier is het principe hetzelfde als bij de NUC. Een laptop beschikt echter over een ingebouwd scherm, muis en toetsenbord. Een laptop is meestal minder betrouwbaar dan een NUC. Ook hier verwerkt het voertuig dus zelf de sensordata en data van andere organen.

De mogelijke opties voor “Navigeren” zijn terug te vinden in deel 2.1.2.

De mogelijke opties voor “Aansturen” zijn.

- PLC, Process Logic Controllers zijn industriële computers speciaal gemaakt voor het aansturen van hardware zoals motoren, servo’s et cetera. Ze zijn in de industrie onmisbaar en dus zeer betrouwbaar. De PLC ontvangt gegevens van “Navigeren”, op basis van deze stuurt de PLC stuursignalen naar de hardware.

- Ethernet/WiFi verbinding met PC, de verwerkingseenheid kan de verschillende onderdelen rechtstreeks aansturen via een ethernet of wifi verbindingen. Omdat deze onderdelen moeilijk te vinden zijn is er besloten deze niet in de structuur te gebruiken.

- Arduino, een Arduino stuurt signalen naar de aandrijvingen. Arduino’s zijn minder betrouwbaar en fragile ten opzichte van de andere opties. Het grootste voordeel is de kostprijs.

Er bestaan ook industriële versies van Arduino betrouwbaarder zijn, deze worden niet besproken in deze studie.

- I/O kaarten op PC, het is mogelijk om in/uitgangskaarten aan te sluiten aan een PC (bijvoorbeeld NUC of laptop). I/O kaarten zijn over het algemeen ook betrouwbaar.

De mogelijke opties voor “Aandrijven” zijn.

- Elektrisch, het gebruik van elektrische componenten om het stuur en de pedalen te bewegen.
- Hydraulisch, het gebruik van hydraulische componenten om het stuur en de pedalen te bewegen.
- Pneumatisch, het gebruik van pneumatische componenten om het stuur en de pedalen te bewegen.
- Mensbekrachtigd, een mens beweegt het stuur en de pedalen. Deze optie is echter niet opgenomen in de structuren omdat dit niet een autonome optie is.

De mogelijke opties voor “In/uitladen” zijn.

- Mensbekrachtigd, de begeleiders laden de zorgbehoevenden op eigen kracht in en uit.
- Mensbekrachtigd + elektrisch, de begeleiders laden de zorgbehoevenden in en uit met behulp van een elektrisch systeem.
- Mensbekrachtigd + hydraulisch, de begeleiders laden de zorgbehoevenden in en uit met behulp van een hydraulisch systeem.
- Mensbekrachtigd + pneumatisch, de begeleiders laden de zorgbehoevenden in en uit met behulp van een pneumatisch systeem.

De mogelijke opties voor “Vast/losmaken” zijn.

- Klem, het docking station uit deel 2.4.3.
- Haken, uit deel 2.4.2.
- Op stoel zetten, de begeleiders helpen de zorgbehoevenden uit hun rolstoel en begeleiden ze in het voertuig waar ze plaats zullen nemen op een stoel. Deze optie is niet geschikt voor zorgbehoevenden die hun rolstoel moeilijk kunnen verlaten.

De verschillende opties zijn gecombineerd in structuren die te zien zijn in Tabel 5-6.

Tabel 5-6: Structuren voor het samengesteld werktuig

	Struct. 1	Struct. 2	Struct. 3	Struct. 4
Oproepen	HMI	HMI	HMI	HMI
Communiceren	WIFI	WIFI	WIFI	WIFI
Verwerken	Raspberry pi	Ind. computer (NUC)	Ind. computer (NUC)	Laptop
Navigeren	Strips	Natuurlijk (SLAM)	Bakens + natuurlijk	Spots
Communiceren	WIFI	WIFI	WIFI	WIFI
Aansturen	Arduino	I/O kaarten op PC	PLC	I/O kaarten op PC
Aandrijven	Elektrisch	Hydraulisch	Elektrisch	Pneumatisch
In/uitladen	Mensbekrachtigd + Elektr.	Mensbekrachtigd + Hydr.	Mensbekrachtigd	Mensbekrachtigd + Pneum.
Vast/Losmaken	Klem	Haken	Klem	Op stoel zetten

5.4.3 Bespreking structuren

Bespreking structuur 1

Deze structuur gebruikt een Raspberry Pi als verwerkingstoestel en een Arduino om de elektrische componenten aan te sturen die het stuur zullen draaien en de pedalen zullen indrukken. De navigatievorm is stripnavigatie. De begeleiders laden met behulp van een elektrisch systeem de zorgbehoevenden in en uit. Een klemsysteem zorgt ervoor dat de rolstoelen gedurende de rit vast blijven staan.

Bespreking structuur 2

Deze structuur gebruikt een NUC als verwerkingsorgaan en I/O kaarten verbonden aan de NUC om de hydraulische componenten aan te sturen die het stuur zullen draaien en de pedalen zullen indrukken. De navigatievorm is SLAM. De begeleiders laden met behulp van een hydraulisch systeem de zorgbehoevenden in en uit. Een hakensysteem uit deel 2.4.2 zorgt ervoor dat de rolstoelen gedurende de rit vast blijven staan.

Bespreking structuur 3

Deze structuur gebruikt een NUC als verwerkingsorgaan en een PLC om de elektrische componenten aan te sturen die het stuur zullen draaien en de pedalen zullen indrukken. De navigatievorm is bakennavigatie gecombineerd met natuurlijke navigatie. De begeleiders laden op eigen kracht de zorgbehoevenden in en uit. Een klemsysteem zorgt ervoor dat de rolstoelen gedurende de rit vast blijven staan.

Bespreking structuur 4

Deze structuur gebruikt een laptop als verwerkingsorgaan en I/O kaarten verbonden aan de laptop om de pneumatische componenten aan te sturen die het stuur zullen draaien en de pedalen zullen indrukken. De navigatievorm is spotnavigatie. De begeleiders laden met behulp van een pneumatisch systeem de zorgbehoevenden in en uit. In deze structuur zijn de rolstoelen en de zorgbehoevenden apart vervoerd. De zorgbehoevenden zitten op stoelen of banken en de rolstoelen liggen in een apart gedeelte. Dit zorgt voor een grotere vervoerscapaciteit. De pneumatische componenten vereisen een toevoeging van een compressor, filters, et cetera.

5.4.4 Keuze en conclusie

De resultaten zijn te vinden in Tabel 5-7.

Tabel 5-7: Keuze van de structuur voor het samengesteld werktuig

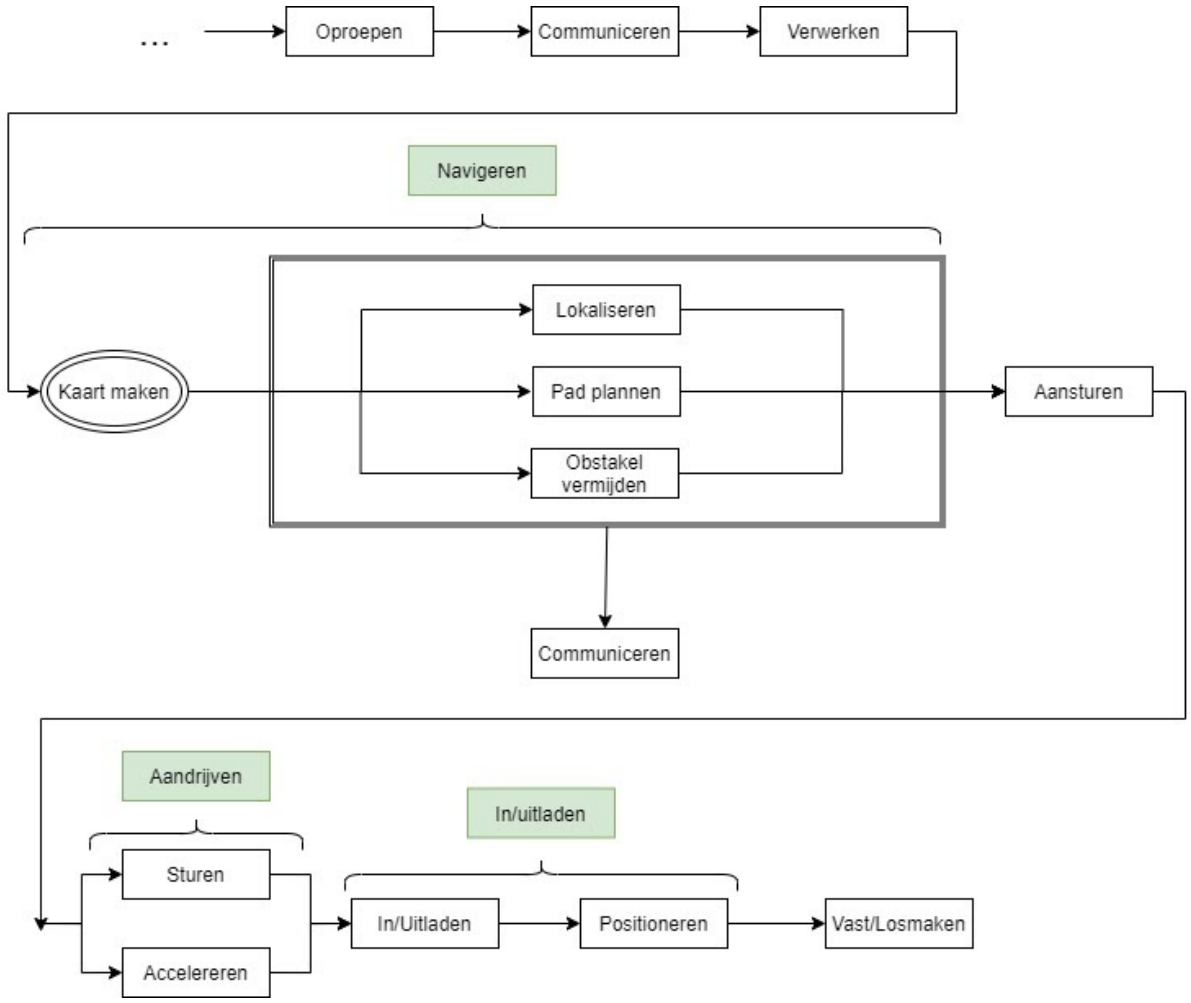
	Struct. 1	Struct. 2	Struct. 3	Struct. 4	Factor
Kostprijs	4	2	3	3	1
Kans op fout	1	3	4	1	2
Levensduur	1	3	4	2	2
Onderhoud	3	2	4	1	2
Weersvereisten	4	1	3	4	3
Rolstoel capaciteit	3	3	3	4	4
Wegbreedte	3	4	4	3	4
Belasting begeleider	4	4	2	3	4
Gemak zorgbehoevenden	4	4	4	1	4
Veiligheidsnormen	0	4	4	4	4
Totaal	82	97	104	83	

Structuur 3 is de meest belovende structuur. Het aanbrengen van extra systemen, zoals hydraulische of pneumatische, is niet nodig. Bovendien beschikt de verwerking over genoeg processorvermogen en gebeurt het aansturen van de aandrijvingscomponenten op een betrouwbare manier met een PLC. De gecombineerde navigatievorm haalt de voordelen van beide methodes naar boven. Ook kunnen de rolstoelgebruikers in hun rolstoel blijven zitten.

5.5 Werktuig

5.5.1 Functies en functieblokschema

De tweede stap naar een concretere uitwerking is het verdelen van de functie “Navigeren” in “Kaart maken”, “Lokaliseren”, “Pad plannen” en “Obstakels vermijden”. De laatste 3 vormen een parallelle uitvoering. Ook “Aandrijven” is onderverdeeld in “Sturen” en “Accelereren”. Ten slotte is er na “In/uitladen” een hulp functie “Positioneren” toegevoegd. De volgorde van de laatste twee functies is afhankelijk van de cyclus. Bij de eerste cyclus is de volgorde hetzelfde zoals het schema in Figuur 5-3 namelijk “Inladen”, “Positioneren” en “Vastmaken”. Bij de tweede cyclus zijn er al zorgbehoevenden aan boord. De begeleiders moeten dan eerst de rolstoelen “Losmaken”, “Positioneren” en “Uitladen”.



Figuur 5-3: Functieblokschema van het werktuig

Als het voertuig zich van A naar B wilt verplaatsen moet deze in het bezit zijn van een kaart. De manier waarop de kaart is gemaakt valt onder “Kaart maken”, deze functie krijgt een speciaal teken omdat het maken van de kaart al op voorhand is gebeurd. Het voertuig moet zichzelf ook constant kunnen lokaliseren op deze kaart, dit valt onder “Lokaliseren”. Het bepalen van de route tussen A en B is afhankelijk van de “Pad planning”. Het vermijden van botsingen met mensen of voorwerpen valt onder “Obstakel vermijden”.

De gebruikte component om het stuur te draaien valt onder “Sturen”. De gebruikte componenten voor het induwen van het gas- of rempedaal valt onder “Accelereren”. Het positioneren van alle zorgbehoevenden op de juiste plaats in het voertuig valt onder “Positioneren”.

5.5.2 Bespreking mogelijke opties per functie

De opties voor “Verwerken”, “Aansturen” en “Vast/losmaken” zijn in 5.4 vastgelegd. De nieuwe functies en hun opties zijn weergeven in Tabel 5-8.

Tabel 5-8: Opties voor functies voor het werktuig

	Optie 1	Optie 2	Optie 3	Optie 4
Oproepen	HMI			
Communiceren	WIFI			
Verwerken	Ind. computer (NUC)			
Kaart maken	Tekenen	CAD-model	SLAM wagen huren	Aerial imagery
Lokal. (Sensor)	Sonar + Prop	Camera + Prop	2D LIDAR + GPS + Prop	2D LIDAR + GPS-RTK + Camera + Prop
Lokal. (Algor.)	Markov lokal.	EKF (UKF) lokal.	Monte Carlo lokal.	
Pad plannen	D* Lite	A*	DWA	Potential fields
Obstakel vermijden	laser + contact	sonar + contact	laser + sonar + contact	
Communiceren	WIFI			
Aansturen	PLC			
uren	Servomotor	Stappenmotor	Elektr. lin. actuator	DC-motor + motor drive
Accelereren	Servomotor	Elektr. lineaire actuator	DC-motor + motor drive	
In/uitladen	Helling (bevestigd aan wagen)	Helling (los van wagen)	Handmatig inhijsen	
Positioneren	Visueel (Tape)	Rails	Contactsensor	
Vast/Losmaken	Klem			

De mogelijke opties voor “Kaart maken” zijn.

- Tekenen, zelf een 2D kaart opstellen met behulp van een tekenprogramma.
- CAD-model, aan de hand van het 2D CAD model van Sint Oda een kaart opstellen. CAD modellen zijn enkel bruikbaar wanneer deze accuraat genoeg zijn.
- SLAM wagen huren, een derde gespecialiseerde instantie inschakelen om een 2D op te stellen.
- Aerial Imagery, aan de hand van lucht/satellietfoto’s een 2D kaart opstellen.

De methode van “Lokaliseren” is afhankelijk van de gebruikte sensoren en lokalisatie algoritmen. De mogelijke bruikbare sensoren zijn.

- Alle functies maken gebruik van proprioceptieve (Prop) sensoren, zie deel 3.4.1.
- Sonar, het gebruik van sonar bakens uit deel 3.4.4. De zenders van de signalen zijn de vaste bakens die verspreid zijn over het zorgcentrum. De ontvanger bevindt zich op de AGV.
- 2D LIDAR, het gebruik van een 2D LIDAR uit deel 3.4.2 en laserbakens uit deel 2.1.2 om zijn locatie te bepalen.
- Camera, het gebruik van een camera uit deel 3.4.3 die aan de hand van herkenningspunten zijn locatie weet te vinden.
- 2D LIDAR + GPS, het gebruik van 2D LIDAR en laserbakens in combinatie met een GPS uit deel 3.4.4 om zijn locatie te bepalen.
- 2D LIDAR + Camera + GPS-RTK, het gebruik van 2D LIDAR in combinatie met een GPS-RTK uit deel 3.4.4 en een camera uit deel 3.4.3 om zijn locatie te bepalen.

De algoritmen voor het “Lokaliseren” zijn terug te vinden in deel 3.1.

De mogelijke opties voor “Pad plannen” zijn terug te vinden in deel 3.3.

De mogelijke opties voor “Obstakel vermijden” zijn.

- Laser safety scanner uit deel 2.1.3 gebruiken. Wanneer een object te dichtbij komt moet het voertuig stoppen of vertragen.
- Sonar afstandssensoren zoals in deel 3.4.4 gebruiken. Wanneer een object te dichtbij komt moet het voertuig stoppen of vertragen.
- Contactsensoren gebruiken zoals in deel 2.1.3. Wanneer deze een botsing detecteren moet het voertuig stoppen.

De mogelijke opties voor “Sturen” zijn.

- Servomotor, het gebruik van een servomotor om het stuur te draaien. De servomotor kan bewegen tussen een beginhoek en een eindhoek. Het stuursignaal beslist welke hoek de servomotor aanneemt.
- Stappenmotor, het gebruik van een stappenmotor om het stuur te draaien. De stappenmotor kan door stuursignalen in kleine stappen linksom of rechtsom draaien.
- Elektrische lineaire motor, het gebruik van een elektrische lineaire motor en een lineair naar rotationele overbrenging om het stuur te draaien. De lineaire motor kan door stuursignalen vooruit en achteruit bewegen. Het gebruik van een encoder op het stuur of op de slede van de lineaire motor is hier vereist.
- DC motor + motor drive, het gebruik van een DC motor om het stuur te draaien. Het gebruik van een encoder op het stuur of motor is hier ook vereist. Ook regeltechnisch moet er veel gebeuren om deze methode succesvol te laten werken.

De mogelijke opties voor “Accelereren” zijn.

- Servomotor, het gebruik van een servomotor om het pedaal in te duwen. De servomotor kan bewegen tussen een beginhoek en een eindhoek. Het stuursignaal beslist welke hoek de servomotor aanneemt.
- Stappenmotor, het gebruik van een stappenmotor om het pedaal in te duwen. De stappenmotor kan door stuursignalen in kleine stappen linksom of rechtsom draaien.
- Elektrische lineaire motor, het gebruik van een elektrische lineaire motor om het pedaal in te duwen. De lineaire motor kan door stuursignalen vooruit en achteruit bewegen. Het gebruik van een encoder op de slede van de lineaire motor is hier vereist.

De mogelijke opties voor “Positioneren” zijn.

- Visueel, met tape of een andere visuele tool voorstellen waar de begeleider de rolstoel dient te plaatsen zodat er een goede klemming plaatsvindt.
- Rails maken zodat de rolstoel een traject moet volgen voor het bij de klemmen aankomt.

- Contactsensor, het voertuig uitrusten met contactsensoren die enkel ingedrukt zijn als de rolstoel correct geplaatst is. Het voertuig kan dan pas vertrekken als iedereen correct geplaatst is.

De mogelijke opties voor “In/uitladen” zijn.

- Helling bevestigd aan wagen, het voertuig beschikt over een helling waarmee de begeleiders de zorgbehoevenden het voertuig in kunnen duwen.
- Helling los van wagen, de hellingen bevinden zich bij de haltes.
- Handmatig in hijsen, de begeleiders gebruiken puur hun fysieke kracht om de zorgbehoevenden in hun rolstoel het voertuig in te tillen.

De verschillende opties zijn gecombineerd in structuren die te zien zijn in Tabel 5-9.

Tabel 5-9: Structuren voor het werktuig

	Struct. 1	Struct. 2	Struct. 3	Struct. 4
Oproepen	HMI	HMI	HMI	HMI
Communiceren	WIFI	WIFI	WIFI	WIFI
Verwerken	Ind. computer (NUC)	Ind. computer (NUC)	Ind. computer (NUC)	Ind. computer (NUC)
Kaart maken	CAD-model	SLAM wagen huren	Aerial imagery	Tekenen
Lokal. (Sensor)	2D LIDAR + GPS + Prop	2D LIDAR + GPS-RTK + Camera + Prop	Camera + Prop	Sonar + Prop
Lokal. (Algor.)	Monte Carlo lokal.	Monte Carlo lokal.	EKF (UKF) lokal.	EKF (UKF) lokal.
Pad plannen	A*	D* Lite	Potential fields	DWA
Obstakel vermijden	laser + sonar + contact	laser + contact	sonar + contact	laser
Communiceren	WIFI	WIFI	WIFI	WIFI
Aansturen	PLC	PLC	PLC	PLC
Sturen	Stappenmotor	Servomotor	DC-motor + motor drive	Elektr. lin. actuator
Accelereren	Elektr. lin. actuator	Servomotor	DC-motor + motor drive	Elektr. lin. actuator
Positioneren	Rails	Visueel (Tape)	Contactsensor	Visueel (Tape)
In/Uitladen	Helling (los van wagen)	Helling (bevestigd aan wagen)	Helling (los van wagen)	Handmatig inhijsen
Vast/Losmaken	Klem	Klem	Klem	Klem

5.5.3 Bespreking structuren

Bespreking structuur 1

De kaart die het voertuig gebruikt stamt uit een CAD-model. Monte Carlo lokalisatie lokaliseert het voertuig in de kaart met behulp van een LIDAR en een GPS. De route die het voertuig dient af te leggen is bepaald door het A* algoritme. Een combinatie van laser, sonar en contact veiligheidssensoren zorgt voor het vermijden van obstakels. Het besturen van het voertuig gebeurd met een stappenmotor op het stuur en elektrische lineaire actuatoren op de pedalen. Een helling die bij de halte staat helpt de begeleiders om de zorgbehoevenden in het voertuig te rijden. De positionering in het voertuig gebeurd door het gebruik van rails.

Bespreking structuur 2

De kaart van het voertuig is gemaakt door een externe instantie. Monte Carlo lokalisatie lokaliseert het voertuig op de kaart met behulp van een LIDAR, een GPS-RTK en camera. De route die het voertuig dient af te leggen is bepaald door het D* Lite algoritme. Een combinatie van laser en contact veiligheidssensoren zorgt voor het vermijden van obstakels. Het besturen van het voertuig gebeurd

met servomotoren. Een aan het voertuig bevestigde helling helpt de begeleiders met het inrijden van de zorgbehoevenden in hun rolstoelen. De positionering in het voertuig gebeurd visueel door het gebruik van gekleurde tape.

Bespreking structuur 3

De kaart van het voertuig is gemaakt door aerial imagery. Een EKF lokaliseert aan de hand van camerabeelden het voertuig op de kaart. De route die het voertuig dient af te leggen is bepaald door potential fields. Een combinatie van sonar en contact veiligheidssensoren zorgt voor het vermijden van obstakels. Het besturen van het voertuig gebeurd met DC motoren. Een aan het voertuig bevestigde helling helpt de begeleiders met het inrijden van de zorgbehoevenden in hun rolstoelen. De positionering in het voertuig gebeurd met contactsensoren die een signaal geven als de positionering correct is.

Bespreking structuur 4

De kaart van het voertuig is getekend op een tekenprogramma. De lokalisatie gebeurd met proprioceptieve sensoren, sonarbakens en de trilaterale berekening. De route die het voertuig dient af te leggen is bepaald door DWA. Een laser veiligheidssensor zorgt voor het vermijden van obstakels. Het besturen van het voertuig gebeurd met elektrische lineaire actuatoren. De begeleiders tillen de zorgbehoevenden vanuit hun rolstoel in het voertuig. De positionering in het voertuig gebeurt visueel door het gebruik van gekleurde tape.

5.5.4 Keuze en conclusie

De resultaten zijn te vinden in Tabel 5-10.

Tabel 5-10: Keuze van de structuur voor het werktuig

	Struct. 1	Struct. 2	Struct. 3	Struct. 4	Factor
Kostprijs	4	2	3	3	1
Kans op fout	4	4	2	2	2
Levensduur	4	3	3	3	2
Onderhoud	4	3	3	3	2
Weersvereisten	2	4	2	3	3
Belasting begeleider	2	4	3	0	4
Gemak zorgbehoevenden	4	4	4	2	4
Veiligheidsnormen	3	4	3	4	4
Totaal	70	82	65	52	

Structuur 2 is de meest belovende optie. De GPS-RTK in combinatie met de LIDAR en camera zorgt voor een robuuste Monte Carlo lokalisatie. Het D* Lite algoritme is zeer geschikt voor de wegenstructuur van Sint Oda. De obstakelvermijding op basis van lasers en contactsensoren werkt onder alle weersomstandigheden. De servomotoren kunnen het voertuig besturen zonder het aanbrengen van extra mechanische overbrengingen. Het feit dat de helling vastzit aan het voertuig zorgt ervoor dat het aanbrengen van hellingen bij elke halte niet nodig is.

6 Evaluatie outdoor kaartopbouw en lokalisatie

6.1 Wijzigingen aan conceptuele studie

In Hoofdstuk 5 Conceptuele studie is bepaald dat het voertuig, uitgerust met 2D LIDAR, GPS-RTK, camera en proprioceptieve sensoren (IMU en encoders), de beste keuze is.

Door dezelfde sensoren tijdens het evalueren te gebruiken is het mogelijk om na te gaan of de keuze van deze oplossing correct is. Dit is echter niet mogelijk omdat de testopstelling zo gemaakt moet zijn dat deze makkelijk vast en los te maken is van het voertuig. Encoders zullen de hoekstand van de wielen meten, hiervoor moeten ze op de wielas bevestigd zijn waardoor het bevestigen en losmaken niet eenvoudig is. Er is uiteindelijk gekozen om de camera ook de rol van de encoders in te laten vullen, het is namelijk mogelijk om visuele odometrie uit te voeren door de verschillende camerabeelden in de tijd met elkaar te vergelijken. In combinatie met de IMU geeft dit de odometrie. Ook is er geen GPS-RTK beschikbaar, er is daarom gebruik gemaakt van een standaard GPS.

De gekozen oplossing maakt gebruik van een kaart die door een gespecialiseerde instantie is opgesteld. Dit is omwille van de hieraan verbonden kosten niet mogelijk. Daarom wordt deze kaartopbouw zelf uitgevoerd. Omdat zowel lokalisatie als kaartopbouw noodzakelijk zijn, is er gekeken naar SLAM-methodes geïmplementeerd in ROS, die compatibel zijn met de gebruikte sensoren (zie Tabel 6-1).

Tabel 6-1: ROS geïmplementeerde SLAM-methodes met inputs en outputs [43, p. 420]

	Inputs							Online Outputs			
	Camera				Lidar		Odom	Pose	Occupancy		Point
	Stereo	RGB-D	Multi	IMU	2D	3D			2D	3D	Cloud
GMapping					✓		✓	✓	✓		
TinySLAM					✓		✓	✓	✓		
Hector SLAM					✓			✓	✓		
ETHZASL-ICP					✓	✓	✓	✓	✓		Dense
Karto SLAM					✓		✓	✓	✓		
Lago SLAM					✓		✓	✓	✓		
Cartographer					✓	✓	✓	✓	✓		Dense
BLAM						✓		✓			Dense
SegMatch						✓					Dense
VINS-Mono				✓				✓			
ORB-SLAM2	✓	✓									
S-PTAM	✓							✓			Sparse
DVO-SLAM		✓						✓			
RGBiD-SLAM		✓									
MCPTAM	✓		✓					✓			Sparse
RGBDSLAMv2		✓					✓	✓		✓	Dense
RTAB-Map	✓	✓	✓		✓	✓	✓	✓	✓	✓	Dense

Uit deze tabel valt af te leiden dat er slechts één SLAM-methode is die in staat is om een camera en 2D LIDAR data te combineren, namelijk RTAB-Map. Deze geeft als outputs de pose (positie + oriëntatie), een 2D en 3D kaart waarop de obstakels staan en een puntenwolk van de opgestelde kaart.

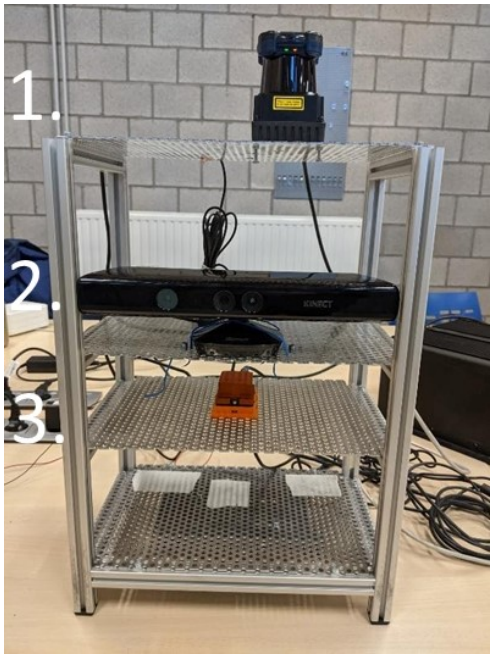
6.2 Hardware

6.2.1 Sensoren

Er zijn drie sensoren gebruikt voor het opbouwen van de kaart en de lokalisatie:

1. de HOKUYO UTM-30LX-EW 360° 2D LIDAR;
2. de Microsoft Kinect V1 RGBD camera;
3. de Xsens MTI-G-710 IMU uitgerust met GPS.

Zoals vermeld in deel 3.4.3 is de RGBD camera niet ideaal voor outdoor omstandigheden en zou een stereo camera een betere optie zijn. Dit was door omstandigheden echter niet mogelijk. Verder is in de conceptuele studie een RTK-GPS aangeraden in plaats van de standaard GPS geleverd met de Xsens IMU. Figuur 6-1 toont alle sensoren gemonteerd op een sensorplatform. Dit sensorplatform is gemaakt van aluminium extrusies waarop geperforeerde metaalplaten gemonteerd zijn. De ontvanger van de GPS is naast de LIDAR bevestigd, dit is echter op de onderstaande figuur niet te zien.



Figuur 6-1: Sensorplatform

6.2.2 Computer

De computer waarmee het sensor platform verbonden is, is een Moxa MC-7000 NUC (zie Figuur 6-2). Deze is robuust gebouwd en waterdicht, met uitzondering van de poorten. De computer is uitgerust met het Ubuntu 16.04 (Xenial) besturingssysteem en gebruikt ROS Kinetic.

Figuur 6-2: Moxa MC-7000 NUC

6.2.3 Mobiel platform

Om grotere gebieden in kaart te brengen is een mobiel testplatform gemaakt. Dit door een frame te maken waarop het sensorplatform en de computer bevestigd zijn. Vervolgens is dit frame op een karretje van de ACRO gezet (zie Figuur 6-3). De overige onderdelen die op het mobiel platform staan zijn een monitor, muis, toetsenbord, lithium-ion batterij en een 24V naar 12V spanningsomvormer.

Figuur 6-3: Mobiel platform

6.3 Software

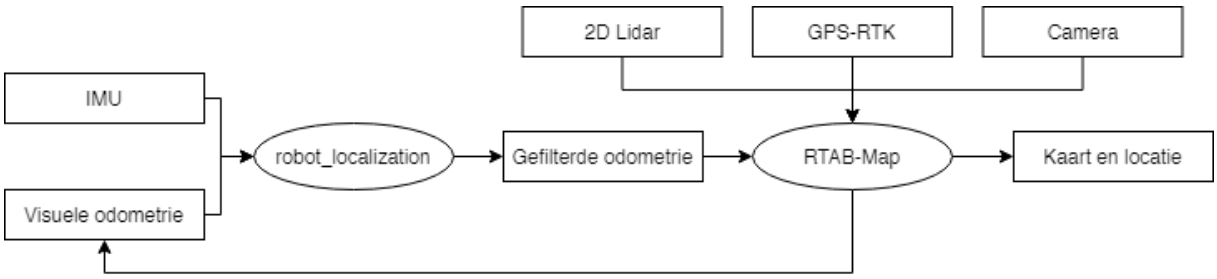
6.3.1 Drivers

Om de data uit te lezen uit de sensoren en te verwerken in ROS zijn drivers nodig. De gekozen driver is hierbij afhankelijk van het type sensor en het merk van deze sensor. Om de 2D LIDAR data van een Hokuyo uit te lezen in ROS is het pakket urg_node noodzakelijk [45]. Het gebruiken van een IMU met GPS van XSENS MTI-G-710 vereist het xsens_node pakket [46]. Tenslotte vereist de Kinect V1 RGBD camera van Microsoft het pakket Freenect_stack [47].

6.3.2 RTAB-map en robot_localization

RTAB-map is een SLAM-methode die de features van een nieuwe afbeelding, zoals randen van objecten, vergelijkt met de voorheen gemaakte afbeeldingen [44]. Door op deze manier verschillende afbeeldingen naast en/of over elkaar te leggen is het mogelijk om een kaart op te stellen en zijn locatie te bepalen. Op gelijkaardige wijze kan RTAB-map ook visuele odometrie genereren. Het gebruiken van 2D LIDAR zal het proces van RTAB-map verbeteren omdat deze nu beschikt over een afstand tot de afbeeldingen op een bepaalde hoogte. Ondanks dat dit niet in Tabel 6-1 vermeld is kan RTAB-map ook GPS(-RTK) data gebruiken om de lokalisatie te verfijnen. Tenslotte is er geen rechtstreekse input voor IMU data, Er is daarom gekozen om de IMU data met de visuele odometrie te combineren en deze vervolgens als gefilterde odometrie aan te bieden aan RTAB-map. De fusie van IMU en visuele odometrie is gebeurt met behulp van robot_localization [48]. Robot_localization baseert zich hiervoor

op EKF of UKF die in deel 3.1.3 besproken zijn, in dit onderzoek is gebruik gemaakt van een UKF. Figuur 6-4 geeft het hele proces schematisch weer, hierbij stellen vierkanten topics voor en ovalen de gebruikte pakketten.



Figuur 6-4: Schematisch overzicht van implementatie RTAB-map

6.3.3 Uitwerking opstartparameters

Er zijn drie bestanden uitgewerkt om de nodige parameters op te stellen, deze bestanden maken allemaal gebruik van het xml-formaat. Het eerste bestand is een urdf bestand, deze bevat informatie over de positie en oriëntatie van de verschillende sensoren tot een bepaald punt. Door gebruikt te maken van deze informatie is het mogelijk om de verkregen sensordata op een correcte manier te transformeren en combineren. Het tweede bestand is een launch file die de gebruikte sensoren activeert en de nodige parameters inlaadt. Tenslotte bevat nog een launch file de gebruikte parameters van RTAB-map en robot_localization. Deze drie bestanden zijn terug te vinden in bijlage A onder voertuig.urdf, sensor_launch.launch en rtabRGBD_launch.launch.

6.4 Indoor testen

6.4.1 Testlocatie

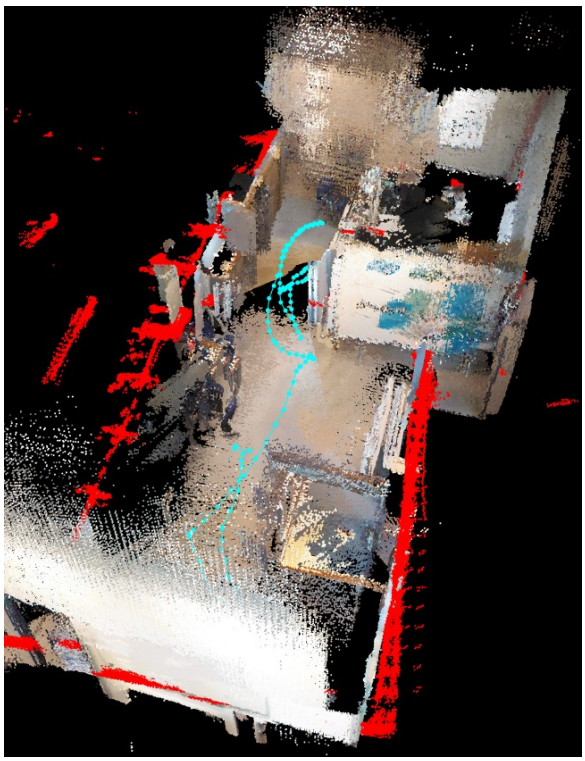
De testlocatie voor de indoor testen is het onderzoekscentrum, met name de lokalen van ACRO. Tijdens de indoor testen is geen gebruik gemaakt van de GPS.

6.4.2 Resultaat en evaluatie

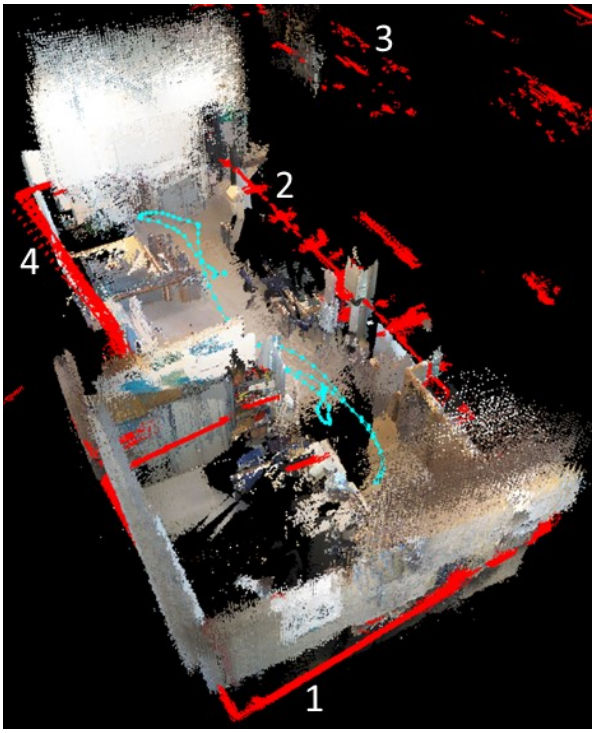
Er zijn verschillende indoor testen uitgevoerd, hierbij stelt de blauwe streep-punt lijn de verschillende posities van de opstelling voor terwijl deze doorheen de kamer beweegt en de rode lijnen de punten van de LIDAR.

Test 1

Figuur 6-5 en Figuur 6-6 tonen twee hoeken van een 3D kaart van één van de ruimtes van ACRO. Globaal komt de opgestelde kaart goed overeen met de werkelijke kamer. Ook de gesimuleerde positie van de opstelling kwam goed overeen met de werkelijke positie. Enkele zaken vallen op, deze zijn op Figuur 6-6 aangeduid.



Figuur 6-5: Kaart indoor test 1 (1/2)



Figuur 6-6: Kaart indoor test 1 (2/2)

Ten eerste valt bij cijfer 1 op dat de scans van de LIDAR en de beelden van de camera niet compleet overeenkomen. Er zijn hiervoor twee mogelijke redenen. De eerste mogelijke reden is dat er een afwijking is tussen het werkelijke sensorplatform en het sensorplatform dat in het URDF-bestand opgesteld is. Het gevolg hiervan is een foutieve transformatie van de camera en LIDAR gegevens. De tweede mogelijke reden zou aan RTAB-map kunnen liggen. Deze zal namelijk zijn 3D kaart opstellen op basis van een aantal verschillende sensoren. Het is mogelijk dat de gegevens van de verschillende sensoren moeilijk of niet met elkaar overeenkomen. RTAB-map moet dus de verschillende gegevens

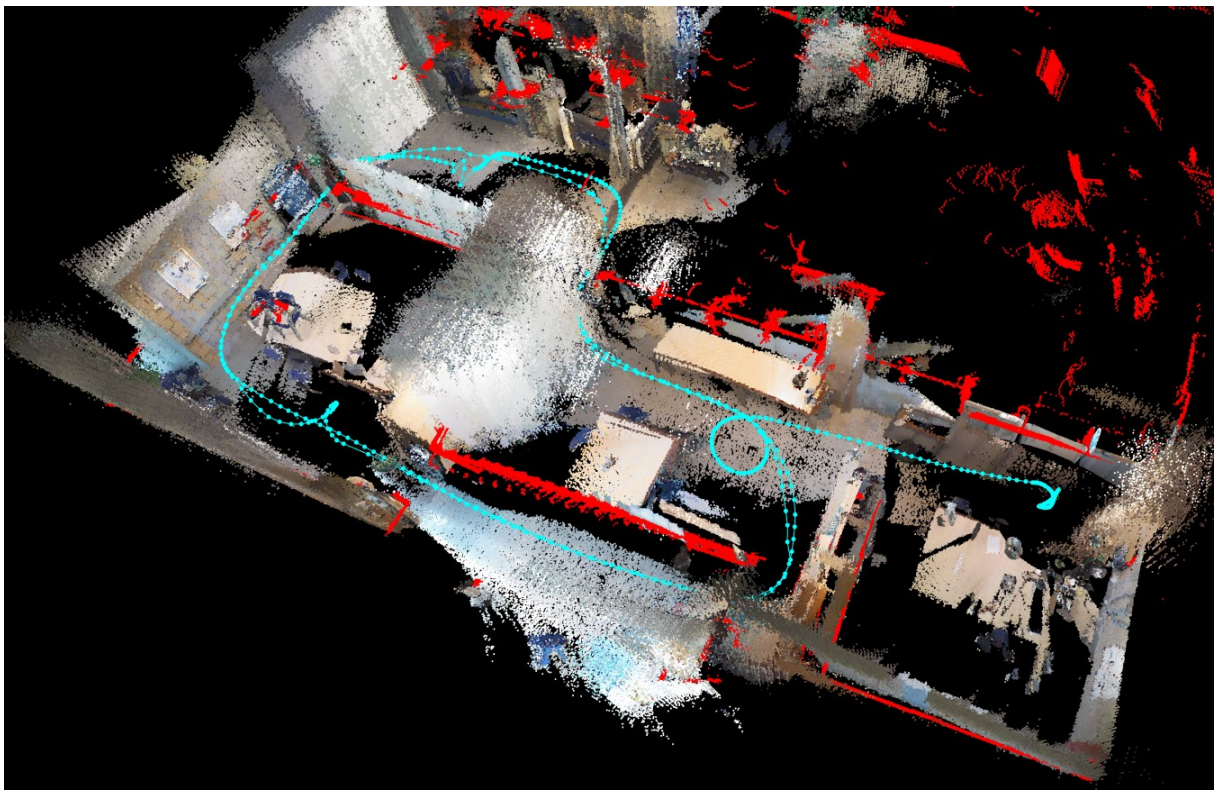
afwegen en op basis hiervan de meest waarschijnlijke 3D kaart opstellen. Ook moet RTAB-map de lokalisatie bepalen met als gevolg dat het resultaat niet altijd overeenkomt met wat de sensoren aangeven. De eerste reden weegt het meeste door, dit omdat er een vaste hoek is tussen de scan en 3D kaart.

Ondanks dat er scans zijn gemaakt is er bij cijfer 2 geen kaart opgesteld. Dit komt doordat dit gedeelte van de muur uit glas bestaat. Een deel van de LIDAR infraroodscan zal tegen dit glas reflecteren zoals bij 2 te zien is. Het overige deel zal door het glas heen gaan en pas op de achterliggende muur terug reflecteren (zie cijfer 3). De Kinect vangt zichtbaar licht op, zichtbaar licht reflecteert minder tegen het glas waardoor er te weinig data is om dit glas in de kaart op te nemen. De reden dat er geen kaart is opgesteld van de omgeving achter het glas komt door het feit dat de dieptemeting van de Kinect niet over deze grotere afstand bruikbaar is en de Kinect deze automatisch verwerpt.

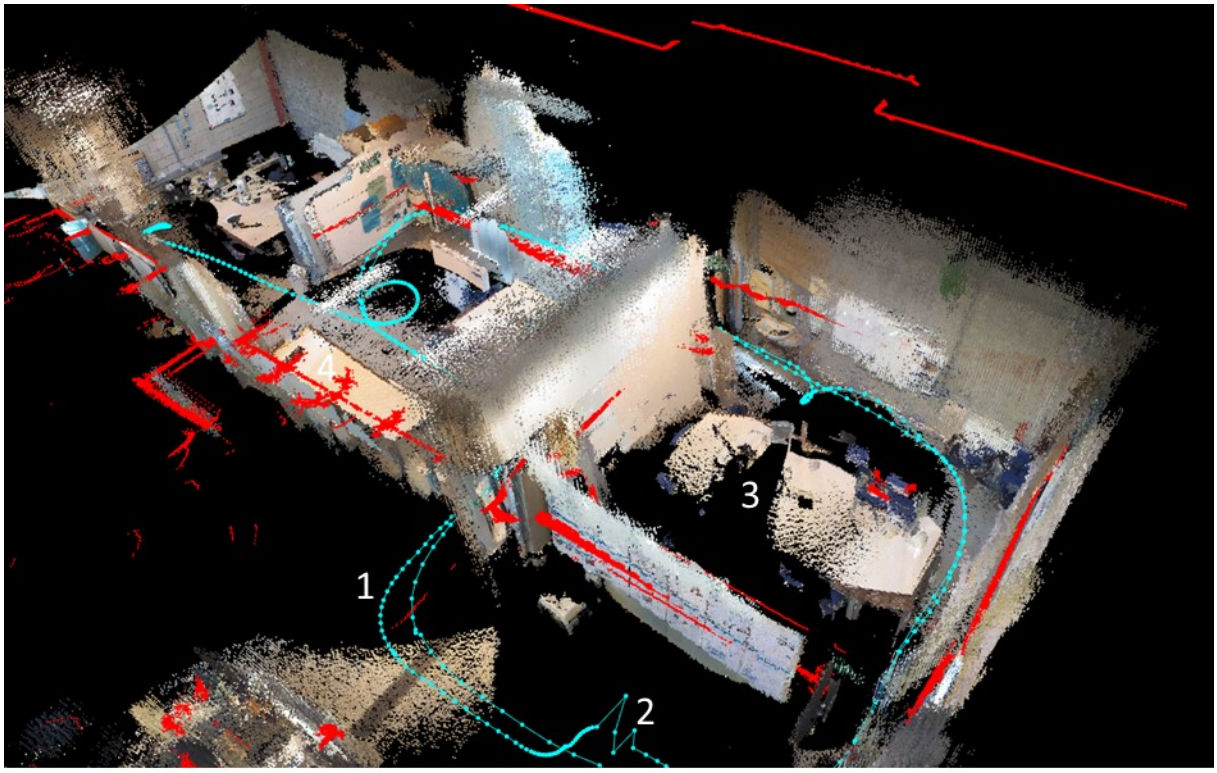
Tenslotte valt bij cijfer 4 op dat de bredere LIDAR streep in werkelijkheid bestaat uit verschillende stippen die elkaar overlappen. In het ideale geval duiden de stippen de locaties van obstakels aan met een dunne lijn. De brede lijn wijst er echter op dat de opstelling niet zeker is waar het obstakel daadwerkelijk staat. Dit komt omwille van twee redenen. De eerste is een afwijking in de LIDAR meting. De tweede reden is dat de opstelling zijn huidige locatie gebruikt om te bepalen op welke afstand het obstakel ligt. Deze tweede reden betekent dat de bepaalde locatie nog niet compleet overeenkomt met de werkelijke locatie.

Test 2

Na een verbetering van het URDF- en rtabRGBD-bestand is gekozen om tweemaal hetzelfde traject rond enkele kamers van de ACRO in kaart te brengen. Dit enerzijds om de lokalisatie beter te evalueren maar anderzijds om te kijken hoe RTAB-map de beelden van de tweede ronde verwerkt. Het resultaat is te zien in Figuur 6-7 en Figuur 6-8. Globaal komt de opgestelde kaart en positie zoals in test 1 goed overeen met de werkelijkheid. Ook komen de scans van de LIDAR beter overeen met de camera en zijn de lijnen van de LIDAR over het algemeen minder breed. Het verbeteren van de bestanden heeft de kaart geoptimaliseerd. Er zijn een aantal cijfers op Figuur 6-8 aangebracht.



Figuur 6-7: Kaart indoor test 2 (1/2)



Figuur 6-8: Kaart indoor test 2 (2/2)

Er is in de tweede ronde door de ruimte niet exact hetzelfde pad gevolgd als in de eerste ronde (zie cijfer 1). De twee paden in deuropeningen en smalle gangen komen goed overeen. Hieruit valt te besluiten dat de er weinig drift is op de lokalisatie na verschillende bochten. Het valt wel op dat de opstelling bij 2 enkele grote sprongen maakt. Deze lokalisatiefout gebeurde in de eerste ronde en is te wijten aan een tekort aan features in de afbeeldingen. In voorgaande testen gebeurde gelijkaardige

fouten ook in de vergaderzaal (zie cijfer 3). Om dit op te lossen zijn er een aantal figuren op het whiteboard getekend en zijn er een aantal objecten op de tafel gezet.

Schijnbaar lijkt het alsof RTAB-map de afbeeldingen van de tweede ronde gebruikt om de kaart van de eerste ronde te verbeteren. Het valt bij een aantal objecten op dat deze een lichte afwijking hebben, dit is te zien aan de tafel aangeduid met cijfer 4. Deze tafel is uitvergroot te zien op Figuur 6-9. Op deze figuur is zichtbaar dat de tafel onder een lichte verdraaiing tweemaal in de kaart is opgenomen. RTAB-map zal de nieuwe beelden dus op de kaart toevoegen zonder deze met de oude beelden te combineren. Indien de lokalisatie nauwkeurig genoeg blijkt, zijn meerdere rondes nog altijd nuttig. Dit omdat de extra rondes de bruikbare dataset voor latere lokalisatie vergroot.

Figuur 6-9: Tafel indoor test 2

Test 3

Als laatste indoor test is een lokalisatie van de opstelling gebeurd aan de hand van de kaart uit test 2. Voor deze test is de opstelling begonnen dicht bij een willekeurige plek van het opgestelde pad. Na deze kort rond te bewegen slaagde de opstelling er elke keer in zijn locatie te vinden.

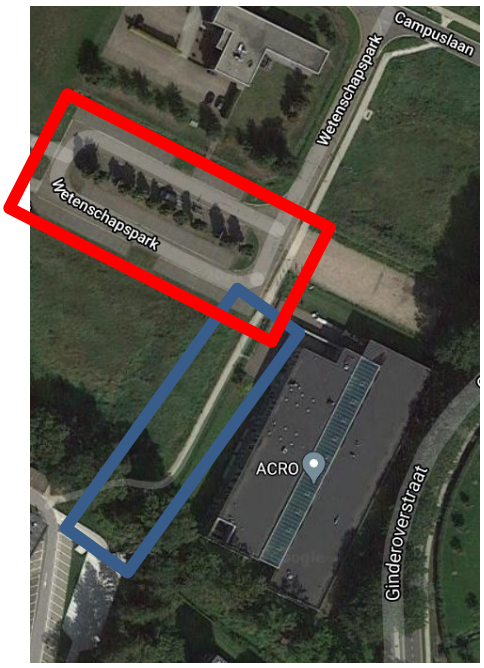
6.5 Outdoor testen

6.5.1 Beperkingen opstelling

De opstelling heeft twee grote beperkingen. De eerste is dat de opstelling in zonnige omstandigheden nauwelijks bruikbaar is. Dit komt doordat de Kinect RGBD camera diepte bepaald door gebruikt te maken van een infrarood signaal. Zoals in deel 3.4.3 al aangehaald is zal de zon dit signaal verstoren. Het gebruik van een stereo camera bleek echter niet mogelijk. Dit omdat de MOXA MC-7000 niet beschikt over een grafische kaart die krachtig genoeg is om de stereo beelden te verwerken. Het gevolg is dat de buitentesten enkel bij schemering uitgevoerd zijn. Dit zodat de infraroodstraling van de zon minimaal is terwijl er nog genoeg zichtbaar licht is zodanig dat de camera kan functioneren. De tweede beperking is dat het sensorplatform niet waterbestendig is. Beide beperkingen hebben als gevolg dat het niet mogelijk is om aan kaartopbouw en lokalisatie te doen bij wisselende weersomstandigheden.

6.5.2 Testlocatie

Er is besloten de outdoor testen niet te laten plaatsvinden op Sint Oda vanwege de Covid-19 pandemie, de testen zijn daarom buiten het onderzoekscentrum ACRO (Figuur 6-10) uitgevoerd. Langs het onderzoekscentrum loopt een fietspad (blauw) wat sterk lijkt op de paden van Sint Oda . Ook ligt er een parkeerplaats (rood) waar het mogelijk is om uitgebreid te testen. Het is belangrijk dat de testlocaties gelijkenissen vertonen aan de omgeving van Sint Oda aangezien de lokalisatie en odometrie grotendeels afhankelijk zijn van camerabeelden.



Figuur 6-10: Satellietbeeld outdoor testlocaties (Map data ©2020 Aerodata International Surveys, GeoContent, Maxar Technologies)

6.5.1 Resultaat en evaluatie

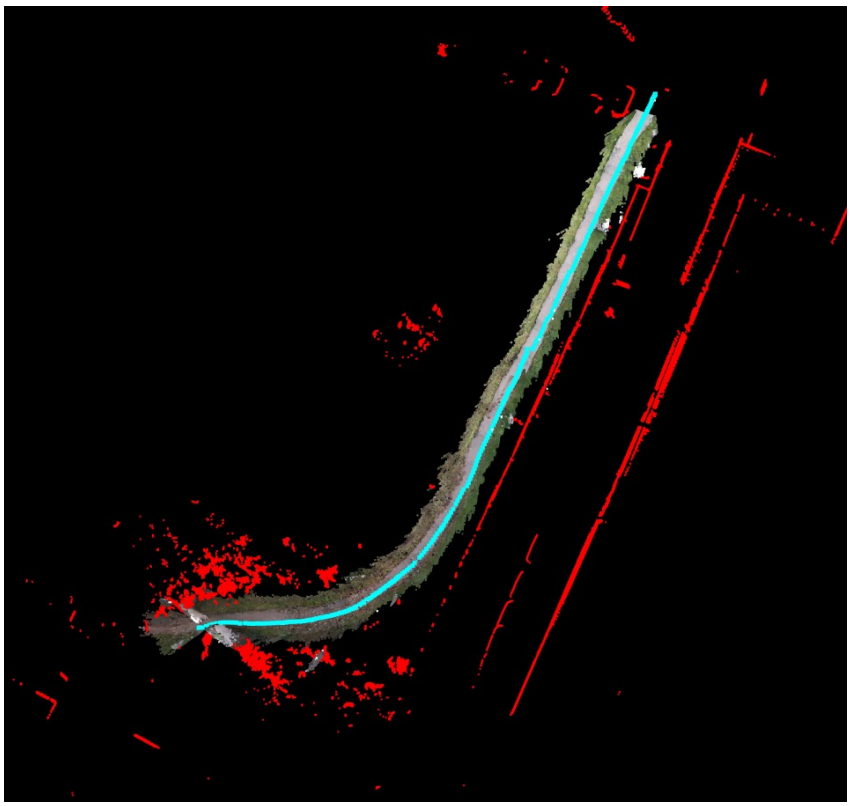
Test 1

De eerste test heeft net voor de ingang van het onderzoekscentrum plaatsgevonden. Dit om te evalueren of de camera in staat is te werken in de toenmalige belichting. Deze eerste test gaf de puntenwolk in Figuur 6-11. De camera is in staat te werken bij de toenmalige schemering.

Figuur 6-11: Puntenwolk technologiecentrum

Test 2

Na de conclusie dat de belichting bij schemering gunstig is voor outdoor testen is een kaart gemaakt van het fietspad. Er is gekozen om ter hoogte van de parking te vertrekken en vervolgens in zuidwestelijke richting het pad te volgen omdat het pad gelijkenissen vertoont aan de paden bij Sint Oda. Het resultaat is de kaart in Figuur 6-12. De blauwe lijn stelt de odometrie voor, dit is de route die volgens de IMU en de camera is afgelegd. De rode lijnen zijn metingen van de LIDAR, zo is het hek langs het fietspad en de muur van het onderzoekscentrum zichtbaar. Bij vergelijking met het satellietbeeld uit Figuur 6-10 is te concluderen dat de kaart op één minimale verspringing na vrij nauwkeurig overeenkomt met dit satellietbeeld. Het algoritme had gedurende de hele weg genoeg features om te gebruiken. Een puntenwolk van het fietspad is te zien op Figuur 6-13.



Figuur 6-12: Bovenaanzicht kaart fietspad

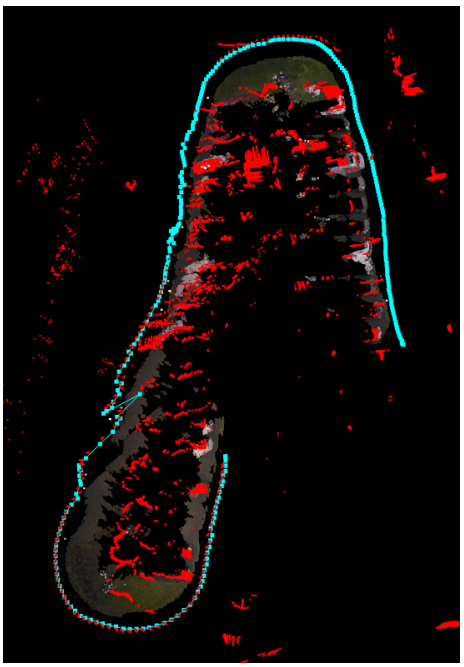
Figuur 6-13: Puntenwolk fietspad

Vervolgens is besloten om de lokalisatie in de kaart te testen. Dit door de opstelling uit te zetten, te verplaatsen naar een ander punt op het pad en weer aan te zetten. Na een kleine voorwaartse beweging wist de opstelling zich correct te lokaliseren op de kaart.

Test 3

De volgende stap in het testproces was het maken van een kaart tijdens het afleggen van een lus rond de parkeerplaats. Als de software ook aangeeft dat het begin- en eindpunt van de lus op dezelfde plek liggen, is dit een indicatie van een goede werking van de opstelling. Dit was echter niet het geval.

Deze test is meerdere keren herhaald maar geen enkele test is succesvol afgerond. De blauwe lijn in Figuur 6-14 stelt de odometrie voor, het begin- en eindpunt liggen niet op dezelfde plek terwijl dit fysiek wel zo is. Tijdens deze test gaf RTAB-Map aan dat het niet in staat was genoeg features te verzamelen.



Figuur 6-14: Mislukte lus op parkeerplaats

De conclusie uit de outdoor testen is dat de huidige opstelling niet geschikt is voor outdoor kaartopbouw en lokalisatie. Ten eerste is de gebruikte Kinect V1 RGBD camera in outdoor omgevingen enkel bruikbaar bij schemering, ook kan de camera maar ongeveer vijf meter ver afstand scannen. Ten tweede is de gebruikte GPS niet nauwkeurig genoeg. De gebruikte algoritmes zijn wel veelbelovend voor kaartopbouw en lokalisatie in outdoor omgevingen. Deze conclusie kan getrokken worden door de resultaten behaald tijdens de indoor testen. Het gebruik van een stereo camera zou kunnen zorgen voor de detectie van meer bruikbare features. Ook is een stereo camera in veel meer belichtingsomstandigheden te gebruiken. Ten slotte zou het gebruik van GPS-RTK de algoritmes nauwkeuriger maken.

7 Taaksequentiëring

7.1 Dagoverzicht

Het dagoverzicht bestaat uit de verschillende activiteiten die het voertuig iedere dag kan uitvoeren. Het dagoverzicht bevat twee geplande activiteiten. De eerste geplande activiteit is het stilstaan van het voertuig bij de werkplaats. De tweede geplande activiteit is het transporteren van de bewoners van hun leefgroep naar Sens-City en terug. Omdat er slechts twee geplande activiteiten zijn is het makkelijk om een algemeen dagoverzicht op te stellen. Dit op basis van wanneer de bewoners Sens-City bezoeken. De bewoners maken typisch van 9u-11u30 en 14u-18u gebruik van Sens-City. Aangezien het voertuig de bewoners heen en terug moet brengen zal het voertuig het transporteren van bewoners van 8u30-12u30 en 13u30-19u uitvoeren. De overige tijd zal het voertuig stilstaan. Omdat het transporteren van de bewoners een algemene activiteit is, is het mogelijk om deze in een aantal deelactiviteiten op te splitsen waaronder ontvangen opdracht en pad planning.

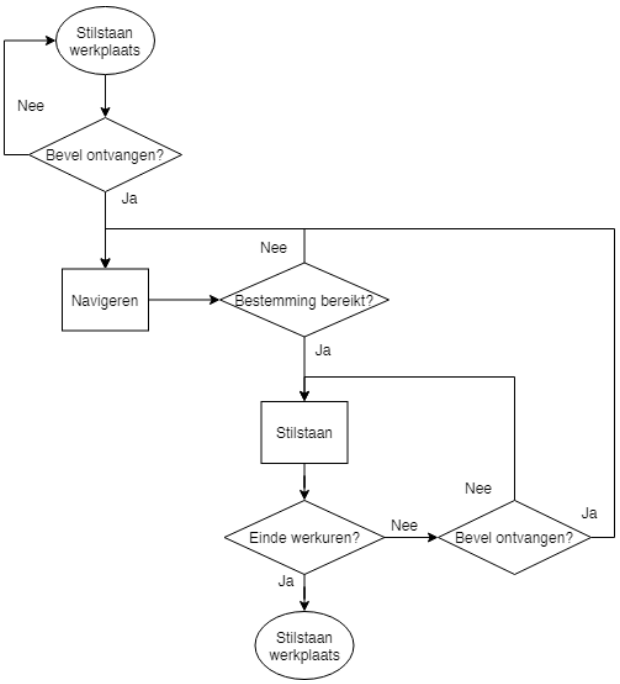
Naast deze geplande activiteiten zijn er ook niet-geplande activiteiten, dit soort activiteiten zijn fouten of noodgevallen die gebeuren tijdens het uitvoeren van de geplande activiteiten. De meeste niet-geplande activiteiten zullen plaatsvinden tijdens het transporteren van de bewoners. Tabel 7-1 geeft een overzicht van de belangrijkste mogelijke activiteiten. De flowchart en BT moeten met deze activiteiten rekening houden.

Tabel 7-1: Overzicht belangrijkste activiteiten

Geplande activiteiten	Niet-geplande activiteiten
Stilstaan bij werkplaats	Batterij bijna leeg
Ontvangen opdracht	Verbinding met systeem kwijt
Bereiken bestemming	Obstakel gedetecteerd
Pad planning	Botsing
Lokalisatie	Noodstop
Bewegen	Vindt halte niet
	Rijden tijdens in-/uitladen bewoners
	Pad planning faalt
	Lokalisatiefout
	Ruis op data

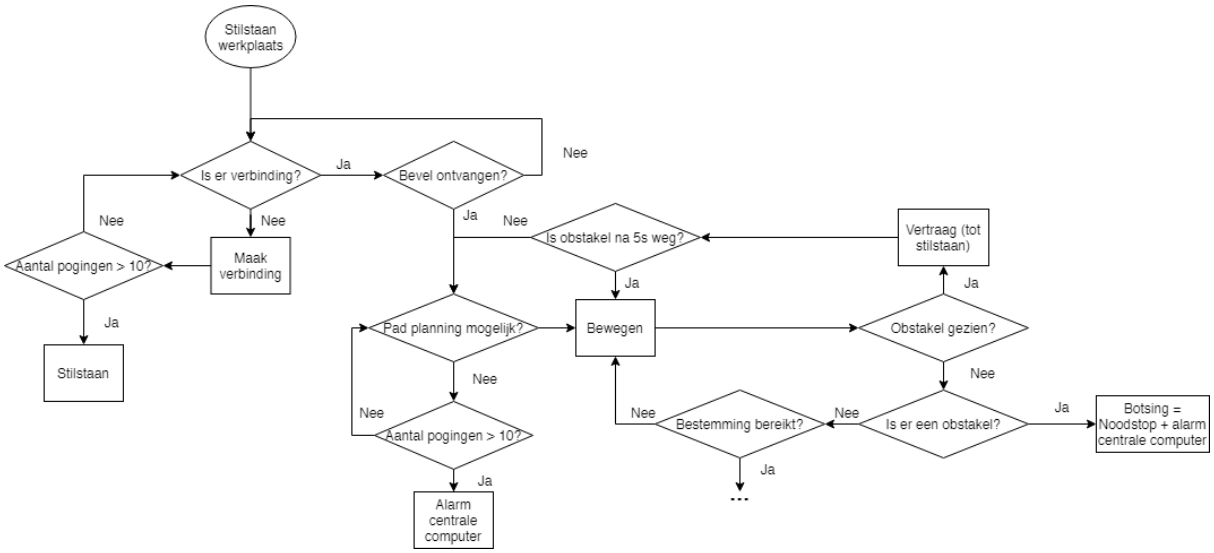
7.2 Flowchart

Figuur 7-1 geeft een flowchart van de werking indien enkel geplande activiteiten plaatsvinden. Hierbij is het transporteren van de bewoners opgesplitst in zijn verschillende deelactiviteiten. Het blok met de tekst navigeren in deze figuur bevat de activiteiten pad planning, lokalisatie en bewegen. Omdat er in deze flowchart niets mis kan gaan is deze opdeling niet gemaakt om het overzicht te bewaren. Ook is er bij deze en de volgende flowchart vanuit gegaan dat het voertuig enkel opdrachten van de centrale computer zal ontvangen tussen 8u30-12u30 en 13u30-19u.



Figuur 7-1: Flowchart geplande activiteiten

Een flowchart maken van alle activiteiten geeft dezelfde problemen als aangehaald bij de FSM. Wanneer de complexiteit stijgt zal de overzichtelijkheid snel dalen. Figuur 7-2 toont een deel van deze flowchart om te laten zien dat er geen goede overzichtelijkheid is. Deze flowchart houdt er geen rekening mee dat sommige niet-geplande activiteiten ten alle tijden kunnen gebeuren. Dit zou de overzichtelijkheid verder verslechteren. Een voorbeeld is de verbinding. De controle hiervan gebeurt enkel in het begin. Echter kan het voertuig op ieder moment de verbinding met de rest van het systeem verliezen. Omwille van de onoverzichtelijkheid is besloten om de flowchart niet verder uit te werken en de BT enkel op basis van de activiteiten op te stellen.



Figuur 7-2: Gedeeltelijke flowchart van alle activiteiten

7.3 Behavior tree

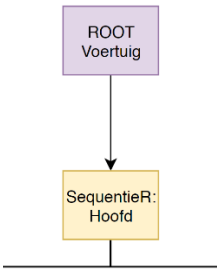
7.3.1 Structuur behavior tree

Uitleg over de werking van de BT en de tool BehaviorTree.cpp is respectievelijk in deel 4.2.2 en 4.3 te vinden. Omwille van de grote van de BT zal de uitleg gebeuren door zijn verschillende componenten te bespreken. De BT is in zijn geheel in bijlage B terug te vinden. Er is bij het maken van deze BT vanuit gegaan dat errors, waarschuwingen en meldingen altijd succes zullen geven. Volgende delen van de BT komen aan bod:

- root- en hoofdcontrole;
- lokaliseren;
- noodstop;
- botsing;
- tijdelijk obstakel;
- data controle
- verbinding verloren;
- reistijd;
- stoppen;
- permanent obstakel;
- opdracht;
- bestemming bereikt;
- rijden.

Root- en hoofdcontrole

Figuur 7-3 toont de root en hoofdcontrole, de hoofdcontrole is een reactieveSequentie die rechtstreeks verbonden is met alle overige delen van de BT. Deze delen zijn fallback nodes die de verschillende geplande en niet-geplande activiteiten van het voertuig bevatten, de geplande activiteit *Transporteren* is hierbij opnieuw onderverdeeld in zijn verschillende deelactiviteiten. Volgende punten zullen deze delen van links naar rechts uitleggen. Tabel 7-2 toont welke activiteit bij welk deel hoort, sommige activiteiten komen bij verschillende delen voor. Deze activiteiten staan cursief in de tekst ter verduidelijking. Omwille van het gebruik van een reactieveSequentie is het zeer belangrijk dat geen enkel deel faalt. Indien dit gebeurt zal de sequentie namelijk stoppen en zal er geen controle van de overige delen gebeuren in deze tick.



Figuur 7-3: Root- en hoofdcontrole

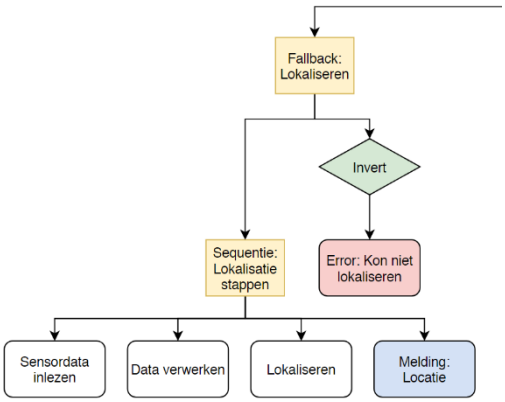
Tabel 7-2: Overzicht activiteiten per fallback

Deel	Activiteit
Root en hoofdcontrole	/
Lokaliseren	Lokalisatie(fout)
Noodstop	Noodstop
Botsing	Botsing
Tijdelijk obstakel	Obstakel gedetecteerd
Data controle	Ruis op data
Verbinding verloren	Verbinding met systeem kwijt
Reistijd	Vindt halte niet

Deel	Activiteit
Stoppen	Niet-geplande activiteiten
Permanent obstakel	Pad planning (faalt)
Opdracht	Ontvangen van opdracht
	Pad planning (faalt)
Reset error	Niet-geplande activiteiten
Bestemming bereikt	Bereiken bestemming
	Pad planning (faalt)
Rijden	Batterij bijna leeg
	Bewegen

Lokaliseren

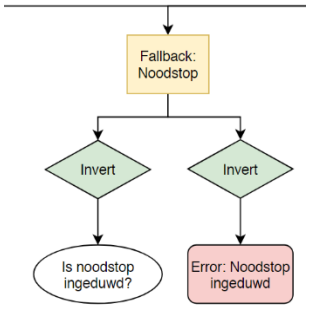
Als eerst komt de fallback Lokaliseren. Deze is hier geplaatst zodat de volgende delen, indien nodig, altijd de meest recente locatie en sensordata ter beschikking hebben op het Blackboard. De fallback lokaliseren omvat twee activiteiten. Ten eerste de geplande activiteit *Lokalisatie*, dit is simpelweg een sequentie van de verschillende stappen om de lokalisatie te bekomen. Hierna volgt een melding naar de centrale computer om deze de locatie van het voertuig te laten weten. Deze melding moet niet iedere tick gebeuren, dit om niet onnodig het netwerk te belasten. De tweede activiteit is de *niet-geplande lokalisatiefout*. Dit deel activeert wanneer één stap uit de sequentie lokalisatie een fout teruggeeft. Dit leidt tot het genereren van een error die het voertuig naar de centrale computer en het Blackboard kan sturen. Figuur 7-4 toont de fallback Lokaliseren.



Figuur 7-4: Fallback Lokaliseren

Noodstop

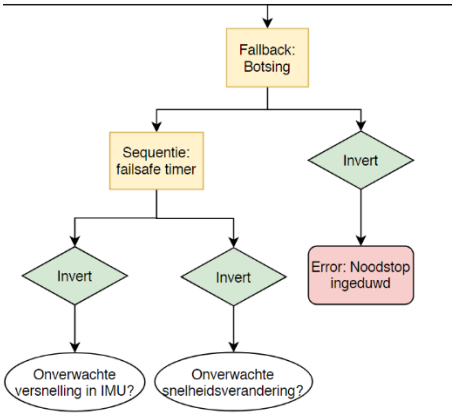
Figuur 7-5 toont de fallback Noodstop. Deze implementeert de activiteit *Noodstop* Dit deel zal simpelweg vragen of de noodstop ingeduwd is. Indien dit zo is zal de BT een error genereren.



Figuur 7-5: Fallback Noodstop

Botsing

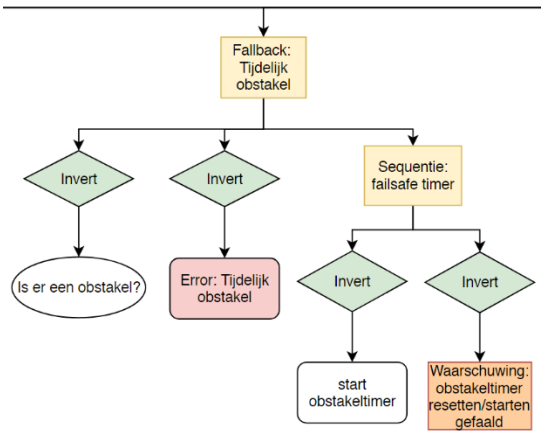
Indien de activiteit *Botsing* plaatsvindt wil dit zeggen dat de obstakelherkenning gefaald heeft. Vertrouwen op de sensoren die de omgeving waarnemen, zoals camera of LIDAR, is daarom niet aangeraden. Om deze reden zal de fallback Botsing een error genereren wanneer de IMU een onverwachte versnelling voelt of wanneer de snelheid abrupt verandert (zie Figuur 7-6).



Figuur 7-6: Fallback Botsing

Tijdelijk obstakel

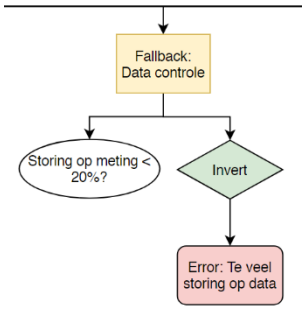
Indien het voertuig een *obstakel herkent* zal de BT een error genereren vanuit de fallback Tijdelijk obstakel. Vervolgens zal de BT de sequentie Failsafe timer activeren. Deze sequentie start een obstakeltimer, het nut hiervan volgt in het deel Permanent obstakel, het is mogelijk dat het activeren van de timer faalt. In dit geval genereert de BT een waarschuwing die het naar het systeem kan versturen. Er is voor een waarschuwing gekozen omdat het niet kunnen starten van de timer het voertuig en zijn inzittenden niet meteen in gevaar brengt. Figuur 7-7 toont de fallback Tijdelijk obstakel.



Figuur 7-7: Fallback Tijdelijk obstakel

Data controle

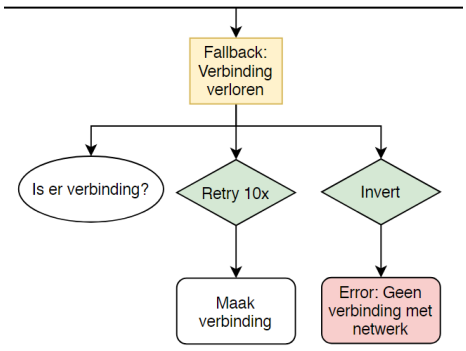
Figuur 7-8 toont de fallback Data controle. Indien er uit de verwerking van data blijkt dat er te veel *Ruis op de data* zit zal dit deel activeren om het voertuig te stoppen.



Figuur 7-8: Fallback Data controle

Verbinding verloren

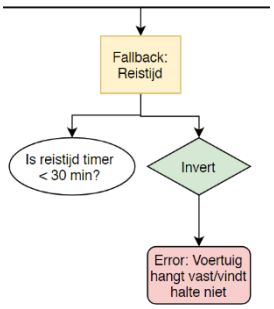
Als het voertuig de *Verbinding met systeem kwijt* is zal de BT in de fallback Verbinding verloren een aantal keer proberen om verbinding te maken. Indien het onmogelijk is om verbinding te maken zal het voertuig stoppen. Indien er geen verbinding is kan de BT geen errors, waarschuwingen en meldingen naar het systeem sturen. Het zal daarom enerzijds nodig zijn om de gemaakte logs in het voertuig zelf te bewaren en anderzijds om aan de systeemkant een timer in te stellen die een error geeft als de verbinding te lang verloren is. Figuur 7-9 toont de fallback Verbinding verloren.



Figuur 7-9: Fallback Verbinding verloren

Reistijd

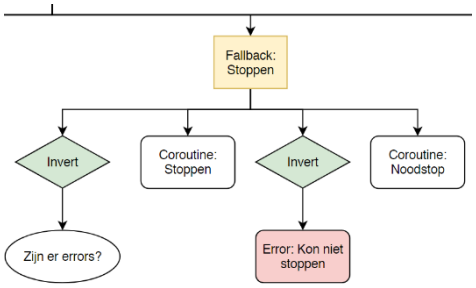
Indien het te lang duurt vooraleer het voertuig zijn bestemming bereikt heeft zal de fallback Reistijd alarm slaan (zie Figuur 7-10). Mogelijke redenen voor het overschrijden van de reistimer zijn dat het niet mogelijk was om de obstakeltimer te stoppen of omdat het voertuig de *Halte niet vindt*. De BT genereert hiervoor een error. Het starten en stoppen/resetten van de reistimer gebeurt respectievelijk in het deel Opdracht en Bestemming bereikt.



Figuur 7-10: Fallback Reistijd

Stoppen

Figuur 7-11 toont de fallback Stoppen. Wanneer een *Niet-geplande activiteit* plaatsvindt stuurt de BT een error naar het Blackboard. Deze fallback controleert of er errors op het Blackboard staan en stopt het voertuig indien dit het geval is. Deze fallback zal eerst op een normale manier proberen stoppen. Indien dit faalt zal de BT een error genereren en vervolgens naar een noodstop overgaan. Voor een veilige werking te garanderen mag deze noodstop niet falen.



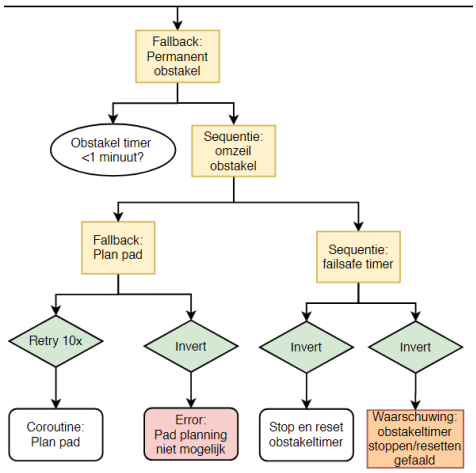
Figuur 7-11: Fallback Stoppen

De nodes Stoppen en Noodstop zullen coroutines zijn die bezig als status geven zolang het voertuig niet stilstaat, zo is het mogelijk om de voorgaande gedeeltes van de BT opnieuw te bekijken terwijl het voertuig stopt. Indien de error vanzelf weg is kan, in sommige gevallen, het voertuig gewoon doorrijden. Hierover volgt meer in deel Reset error. Het gebruiken van een coroutine zorgt er echter voor dat de volgende gedeelten van de BT geen ticks meer ontvangen tot het voertuig stilstaat. De uitleg van de volgende paragrafen zal uitwijzen dat dit geen probleem is.

Permanent obstakel

Wanneer het tijdelijk obstakel te lang aanwezig is, wordt het als een permanent obstakel beschouwd en zal de BT via de obstakeltimer de fallback Permanent obstakel activeren (zie Figuur 7-12). Indien dit gebeurt zal dit deel in de sequentie Omzeil obstakel een nieuw *Pad plannen*, het kan dit een aantal keren proberen. Het kan mogelijks lang duren om een pad te plannen, daarom staat deze voor de zekerheid in een coroutine. Indien de planning succesvol is zal de BT de obstakeltimer proberen te stoppen en resetten, vervolgens zal het voertuig beginnen rijden volgens het nieuw pad. Indien de *Pad planning faalt* zal de BT een nieuwe error genereren. Het is geen probleem dat dit deel na de fallback Stoppen komt. Dit deel zal, omwille van de timer, pas activeren nadat het voertuig tot stilstand

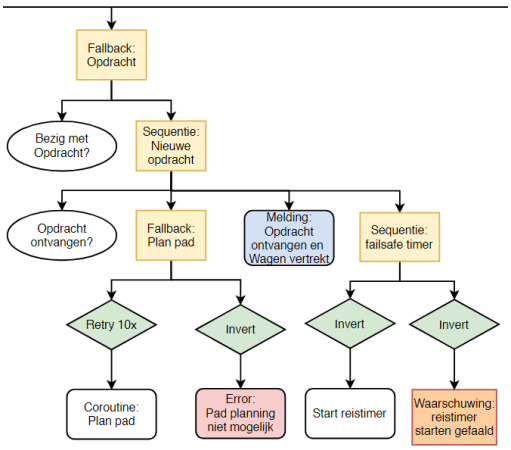
gekomen is. Ook de coroutine Plan pad zal, omdat het voertuig stilstaat wanneer de BT deze node uitvoert, de volgende delen niet in de weg staan.



Figuur 7-12: Fallback Permanent obstakel

Opdracht

De fallback Opdracht zal instaan voor het plannen van de eerste route naar de locatie. Deze fallback kijkt eerst of het voertuig niet al met een opdracht bezig is. Als dit niet zo is wacht de BT tot het voertuig een *Opdracht ontvangt*. Deze opdracht bevat een locatie waarnaar de fallback een *Pad plant*. Indien dit lukt geeft de BT een melding en probeert het de reistimer te starten. Indien de *Pad planning faalt* genereert de BT een error. Het voertuig zal tijdens dit deel al stilstaan, het is dus geen probleem dat dit deel na de fallback Stoppen komt en een coroutine bevat. Figuur 7-13 toont de fallback Opdracht.

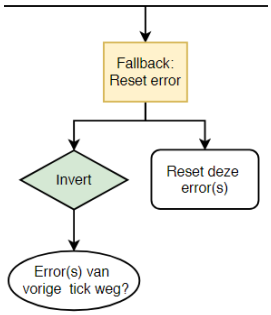


Figuur 7-13: Fallback Opdracht

De fallback Opdracht is zo gemaakt dat slechts één begeleider tegelijk het voertuig kan “besturen”. Initieel stuurt een begeleider zijn locatie met behulp van een HMI. Indien het voertuig niet bezet is zal de BT deze locatie gebruiken om een pad te plannen en start het de opdracht. Zolang het voertuig met een opdracht van een begeleider bezig is, zal het voertuig deze fallback overslaan. Dit wil zeggen dat het onder controle blijft van een begeleider tot deze begeleider zijn opdracht afsluit, dit gebeurt in de fallback Bestemming bereikt. De enige manier waarop het voertuig, terwijl het met een opdracht bezig is, een nieuwe locatie kan ontvangen is door deze locatie bij de HMI van het voertuig in te geven. Dit gebeurt opnieuw in de fallback Bestemming bereikt.

Reset error

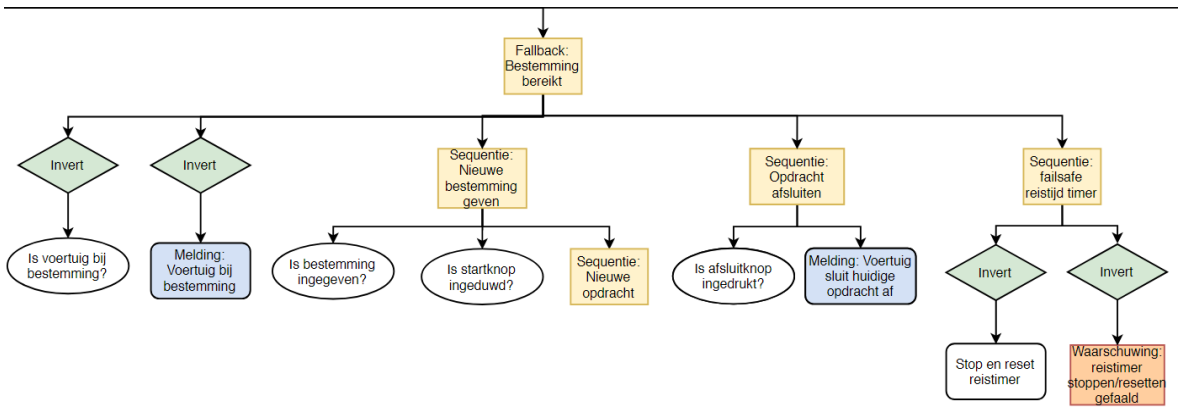
De fallback Reset error zal controleren of een errornode die de vorige tick geactiveerd is, omwille van het plaatsvinden van een *Niet-geplande activiteit*, nu opnieuw geactiveerd is (zie Figuur 7-14). Indien dit niet het geval is kan de BT deze error resetten. Het onmiddellijk resetten van een error omdat de *Niet-geplande activiteit* zich niet meer voordoet mag niet in iedere situatie. Zo zal bij de noodstop error ook eerst een bevestiging moeten gebeuren dat alles veilig is.



Figuur 7-14: Fallback Reset error

Bestemming bereikt

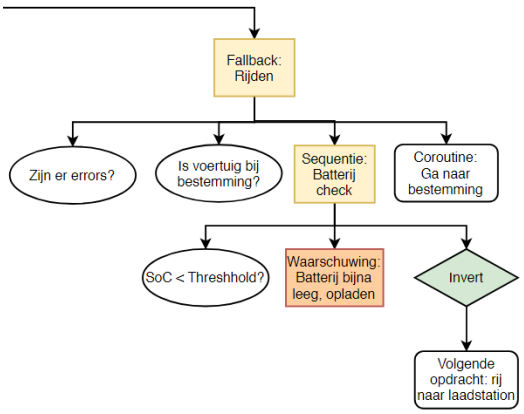
Wanneer het voertuig zijn *bestemming bereikt* heeft activeert de fallback Bestemming bereikt (zie Figuur 7-15). De BT zal hiervan een melding genereren. Vervolgens krijgt de begeleider twee keuzes. De eerste keuze is om een nieuwe bestemming in te geven. Dit zal een *Pad plannen* door de sequentie uit de fallback Opdracht te activeren. De tweede keuze is het stoppen van de opdracht. Wanneer het voertuig aankomt zal de BT de reistimer proberen te resetten en stoppen, lukt dit niet dan genereert de BT een waarschuwing. De sequentie Nieuwe opdracht is identiek aan deze uit fallback Opdracht en is omwille van de grootte niet volledig uitgeschreven.



Figuur 7-15: Fallback Bestemming bereikt

Rijden

Tenslotte komt de BT bij de fallback Rijden. De BT laat het voertuig *Bewegen* wanneer er geen errors zijn en het voertuig nog niet bij zijn bestemming is. De fallback zal vervolgens de actuatoren volgens het geplande pad aansturen in de coroutine “Ga naar bestemming”. Hiervoor controleert de fallback eerst of de *batterij bijna leeg* is. Als dit het geval is genereert de BT een waarschuwing en zal het zelf een volgende opdracht inplannen. Wanneer de begeleider de huidige opdracht stopt zal het voertuig hierdoor meteen naar het laadstation rijden. Figuur 7-16 toont de fallback Rijden.



Figuur 7-16: Fallback Rijden

7.3.2 Opmerkingen bij gemaakte behavior tree

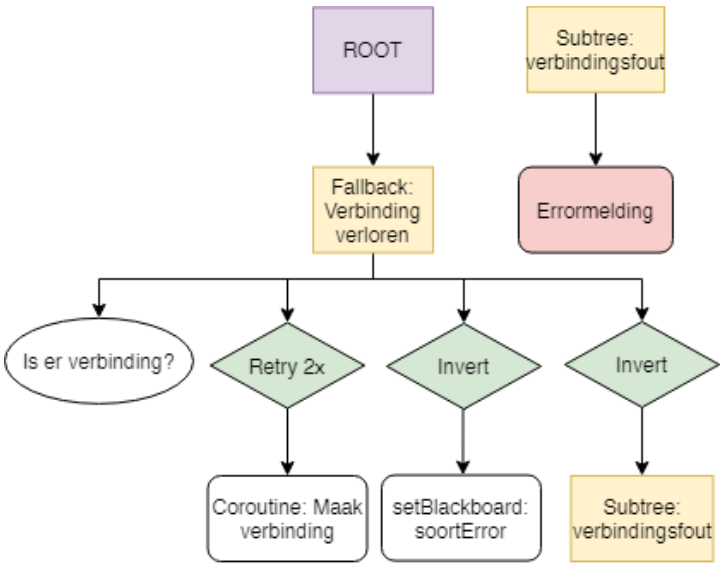
De BT bevat twee grote vereenvoudigingen. Ten eerste kunnen errors, waarschuwingen en meldingen enkel succes als status geven. Ten tweede zijn enkele uitvoeringsnodes nog te algemeen omschreven. Denk hierbij aan de coroutine Stoppen, het is mogelijk om van deze node een controle node te maken met eronder uitvoeringsnodes zoals bepalen remafstand, aansturen actuatoren etc. Dit gebeurt niet waardoor de node Stoppen alle code zal bevatten. Indien een node zeer algemeen omschreven is zal deze veel code bevatten, wat tot onoverzichtelijkheid leidt. Iets wat de BT net moet voorkomen. Andere voorbeelden van te algemene nodes zijn Reset deze error(s) en Ga naar bestemming.

De voorgestelde BT is niet gecontroleerd in een simulatie, dit wil zeggen dat het op andere manieren kan reageren dan voorzien is. Om deze simulatie uit te voeren moet elke uitvoeringsnode code bevatten zodat deze zijn functie kan uitvoeren. Het schrijven van deze code neemt echter veel tijd in beslag. Omwille van de grootte van deze masterproef en het gebrek aan gepast testmateriaal is besloten om dit niet te doen. In plaats hiervan is de fallback Verbinding verloren theoretisch uitgewerkt. Dit om een idee te geven van de effectieve opbouw van een BT in BehaviorTree.cpp.

7.4 Uitwerking fallback Verbinding verloren

7.4.1 Boomstructuur

Om het deel Verbinding verloren te kunnen implementeren is de boomstructuur aangepast. Enerzijds zodat de structuur een root heeft, anderzijds zodat meer functionaliteiten van BehaviorTree.cpp aanwezig zijn (zie Figuur 7-17).



Figuur 7-17: Aangepaste boomstructuur Verbinding verloren

Er zijn de volgende wijzigingen aangebracht:

- het proberen van de node Maak verbinding gebeurt slechts tweemaal;
- er is een coroutine gemaakt van de node Maak verbinding;
- de node setBlackboard en subtree verbindingsfout zijn geïmplementeerd.

7.4.2 Gebruikte bestanden en files

Om deze BT te implementeren is gebruik gemaakt van BehaviorTree.cpp V3 [49]. Verder zijn er vier files aangemaakt: verbindingVerloren.cpp bevat de main-functie, verbindingVerloren.xml de gebruikte boomstructuur in xml-formaat, nodes.h is de header file van de gebruikte nodes en nodes.cpp is de cpp file van de gebruikte nodes. De gemaakte BT is getest met ROS Melodic, het uitvoeren van de code is mogelijk met iedere versie van ROS die gebruik maakt van catkin om pakketten te bouwen. Bijlage C bevat de vier zelfgemaakte files.

8 Besluit

Deze masterproef onderzoekt het integreren van autonoom transport voor zorgbehoevenden in zorgcentrum Sint Oda. Eerst is er informatie verzameld over voertuigen die in aanmerking komen voor autonoom transport (Hfst 2), de onderdelen die nodig zijn voor autonome navigatie (Hfst 3) en de beschikbare communicatiemethoden tussen het voertuig en andere delen van Sint Oda (Hfst 4). Vervolgens is deze kennis toegepast om drie doelstellingen uit te werken. Ten eerste het uitvoeren van een conceptuele studie om de mogelijke oplossingen te vergelijken (Hfst 5). Ten tweede data van Sint Oda verzamelen om lokalisatie en kaartopbouw te evalueren (Hfst 6). Ten slotte het opstellen van een taaksequentiëring om het dagelijks gedrag van het autonoom voertuig te implementeren (Hfst 7).

8.1 Conceptuele studie

In de conceptuele studie zijn verschillende oplossingen bepaald en vergeleken door de "van den Kroonenberg" ontwerpstructuur toe te passen. Er is vastgesteld dat iedere oplossing meer kost dan het budget van Sint Oda. Om deze reden is besloten om de kostprijzen van de oplossingen enkel met elkaar te vergelijken en niet met het gegeven budget.

Het automatiseren van één van de elektrische wagens van Sint Oda of een huifkar is aangeraden. Deze oplossing biedt enerzijds een grotere betrouwbaarheid en gebruiksgemak dan de autonome elektrische trekker. Anderzijds is deze minder duur dan het aankopen van een AGV. Bij het autonoom maken van de oplossing moet Sint Oda wel zelf garanderen dat deze zich aan de benodigde normen voor autonoom vervoer houdt.

8.2 Evaluatie outdoor kaartopbouw en lokalisatie

Er is besloten om de testen niet in Sint Oda te laten plaatsvinden vanwege de Covid-19 pandemie, de testen zijn daarom buiten het onderzoekscentrum uitgevoerd. De huidige testopstelling blijkt ongeschikt voor outdoor kaartopbouw en lokalisatie. Dit is vooral te wijten aan het gebruik van een Kinect V1 RGBD camera, die niet in staat is genoeg features te detecteren. Ook werkt deze enkel in specifieke belichtingsomstandigheden. De gebruikte algoritmes zijn wel veelbelovend. Dit door de behaalde resultaten van de indoor testen. Aan te raden wijzigingen zijn het gebruik van een stereo camera en GPS-RTK.

8.3 Taaksequentiëring

Er is gezocht naar een geschikt algoritme om de taaksequentiëring van het voertuig zo eenvoudig mogelijk te implementeren. De behavior tree (BT) is gekozen omdat deze bij grote en complexe structuren overzichtelijk blijft. De BT is enkel op basis van de geplande en niet-geplande activiteiten opgesteld. Dit omdat de flowchart van deze activiteiten te onoverzichtelijk was om te gebruiken. De opgestelde BT bevat nog enkele vereenvoudigingen. Zo zijn enkele uitvoeringsnodes nog te algemeen omschreven. Verder is er gekozen om de BT niet te simuleren, dit omdat er niet genoeg tijd was om de code van alle uitvoeringsnodes te schrijven. De fallback Verbinding verloren is wel uitgewerkt, dit om een idee te krijgen van de effectieve opbouw van een BT.

8.4 Vooruitblik

Deze masterproef is een eerste studie om autonoom vervoer van zorgbehoevenden op Sint Oda te bekomen. Er is hierbij voornamelijk gekeken naar de verschillende voertuigen, algoritmes en sensoren die nodig zijn om autonoom vervoer te implementeren. Volgende masterproeven kunnen deze

masterproef gebruiken als fundering om meer specifieke onderdelen van het autonoom vervoer uit te werken. Wij raden aan de volgende onderwerpen te behandelen in volgende masterproeven:

- een haalbaarheidsstudie van de verschillende elementen die nodig zijn om autonoom vervoer te bekomen;
- een uitgebreide veiligheidsanalyse;
- wijzigingen aanbrengen aan de testopstelling voor betere outdoor resultaten;
- aanbrengen gewijzigde testopstelling op elektrische wagen (of ander vervoersmiddel) om datasets van Sint Oda op te stellen;
- het uitwerken en testen van de taaksequentie.

Bibliografie

[1] vzw Stijn, "Dienstencentrum - Sint oda," *2018*. https://www.sintoda.be/nl_BE/ (accessed Sep. 30, 2020).

[2] F. J. Siers and H. H. van den (Henricus H. Kroonenberg, *Methodisch ontwerpen : volgens H.H. van den Kroonenberg*. 2004.

[3] G. Ullrich, *Automated Guided Vehicle Systems - A Primer with Practical Applications*, vol. 5, no. 1. Springer Berlin Heidelberg, Berlin, Heidelberg, 2015.

[4] Ellis Systems, "AGV - Automated Guided Vehicles," *1 Februari 2018*. http://www.ellis-systems.com/ellis-systems-pages/industrial-solutions/automated-guided-vehicles/ (accessed Oct. 29, 2020).

[5] L. Lu and I. Verlinde, "Haalbaarheidsstudie naar het gebruik van AGV ' s [eindwerk]," *Gent Fac. Ind. Ingenieurswetenschappen*, 2019.

[6] WordlessTech, "Heathrow driverless Pod | wordlessTech," *20 September 2011*. https://wordlesstech.com/heathrow-driverless-pod/ (accessed Oct. 29, 2020).

[7] Material handling 247, "Tite-Space BST automatic guided vehicle - Material Handling 24/7," *14 May 2020*. https://www.materialhandling247.com/product/tite_space_bst_automatic_guided_vehicle (accessed Nov. 22, 2020).

[8] AGV Network, "Types of AGV Navigation Systems - Comparison, Pros and Cons - Illustrated Guide," *8 Mei 2020*. https://www.agvnetwork.com/types-of-navigation-systems-automated-guided-vehicles (accessed Oct. 29, 2020).

[9] M. De Ryck et. al., "Automated guided vehicle systems, state-of-the-art control algorithms and techniques," *J. Manuf. Syst.*, vol. 54, pp. 152–173, 2020, doi: 10.1016/j.jmsy.2019.12.002.

[10] T. Discart and P. Aerts, "Haalbaarheidsstudie voor implementatie van een AGV te Sabic [eindwerk]," *Diepenbeek Gezam. Opleid. Ind. Ingenieurswetenschappen UHasselt KU Leuven*, 2016.

[11] H. Makela and T. Von Numers, "Development of a navigation and control system for an autonomous outdoor vehicle in a steel plant," *Control Eng. Pract.*, vol. 9, no. 5, pp. 573–583, 2001.

[12] A. J. Moshayedi et al., "AGV (automated guided vehicle) robot: Mission and obstacles in design and performance," *J. Simul. Anal. Nov. Technol. Mech. Eng.*, vol. 12, no. 4, pp. 5–18, 2019.

[13] B. Templeton, "Cameras or Lasers," *21 May 2020*. https://www.templetons.com/brad/robocars/cameras-lasers.html (accessed Oct. 29, 2020).

[14] Resbot, “Understanding AGV safety systems,” *24 Augustus 2018*. http://www.resbot.cn/newsinfo/1538991.html (accessed Oct. 29, 2020).

[15] AGV Network, “AGV cost estimation. How much does an automated guided vehicle cost?,” *20 October 2019*. https://www.agvnetwork.com/agv-cost-estimation-how-much-does-an-automated-guided-vehicle-cost (accessed Oct. 31, 2020).

[16] Vestil, “Electric Powered Tuggers,” *4 February 2019*. https://www.vestil.com/product.php?FID=1019 (accessed Oct. 29, 2020).

[17] Jecam, “Elektrische huifwagen,” *14 October 2018*. https://www.jecam.nl/huifwagen-elektrisch.html (accessed Oct. 29, 2020).

[18] Trucks, “Rolstoevervoer Mercedes,” *27 April 2019*. https://www.trucks.nl/mercedes-benz-sprinter-311-cdi-80-kw-9-sitze-klima-rollstuhllift-euro-4-6597309-vd (accessed Oct. 29, 2020).

[19] Smartfloor, “Floor plans catalog,” *9 August 2020*. https://www.smartfloor.nl/catalog/floor-plans/ (accessed Oct. 29, 2020).

[20] Advance Mobility, “Wheelchair Docking Station,” *30 March 2020*. https://www.advancemobility.com.au/shop/products/wheelchair-docking-station/ (accessed Oct. 29, 2020).

[21] S. Thrun, “Probabilistic robotics,” *Commun. ACM*, vol. 45, no. 3, pp. 52–57, 2002, doi: 10.1145/504729.504754.

[22] Wikiwand, “Kalman filter,” 2018. https://www.wikiwand.com/en/Kalman_filter.

[23] J. M. Santos et al., “An evaluation of 2D SLAM techniques available in Robot Operating System,” *2013 IEEE Int. Symp. Safety, Secur. Rescue Robot. SSRR 2013*, 2013, doi: 10.1109/SSRR.2013.6719348.

[24] M. Mekni and P. Graniero, “A multiagent geosimulation approach for intelligent sensor web management,” *Int. J. Distrib. Sens. Networks*, vol. 2010, no. March 2016, pp. 1–16, 2010, doi: 10.1155/2010/846820.

[25] S. Aggarwal and N. Kumar, “Path planning techniques for unmanned aerial vehicles: A review, solutions, and challenges,” *Comput. Commun.*, vol. 149, no. October 2019, pp. 270–299, 2020, doi: 10.1016/j.comcom.2019.10.014.

[26] D. B. Atta Ur Rehman, Kwame Awuah-Offei, D. A. Baker, “Illustration of Dijkstra’s algorithm,” *Researchgate.net*, 2012. https://www.researchgate.net/figure/Illustration-of-Dijkstras-algorithm_fig1_331484960.

[27] D. H. Kim et al., “A Guide to Selecting Path Planning Algorithm for Automated Guided Vehicle (AGV),” *Lect. Notes Electr. Eng.*, vol. 465, no. January, pp. 587–596, 2018, doi: 10.1007/978-3-319-69814-4_56.

[28] H. Ahmad et al., “Analysis of Mobile Robot Path Planning with Artificial

Potential Fields," *Lect. Notes Electr. Eng.*, vol. 538, pp. 181–196, 2019, doi: 10.1007/978-981-13-3708-6_16.

[29] Garage Willow, Stanford Artificial Intelligence Laboratory, and Open Robotics, "https://wiki.ros.org/," *Open Robotics*, 2020. .

[30] R. Prinsen, "IMU: De toekomst voor bewegingsanalyse?," *ProCare*, 2018. https://www.procarebv.nl/imu-de-toekomst-voor-bewegingsanalyse/.

[31] Terabee, "Time-of-Flight principle," *Terabee*, 2019. https://www.terabee.com/time-of-flight-principle/.

[32] E. Canciani et al., "Characterization of a 2-D laser scanner for outdoor wide range measurement," *J. Phys. Conf. Ser.*, vol. 658, no. 1, 2015, doi: 10.1088/1742-6596/658/1/012008.

[33] Slamtec, "Lidar A1," *Slamtec*, 2018. https://www.slamtec.com/en/Lidar/A1.

[34] M. Dassot et al., "The use of terrestrial LiDAR technology in forest science: Application fields, benefits and challenges," *Ann. For. Sci.*, vol. 68, no. 5, pp. 959–974, 2011, doi: 10.1007/s13595-011-0102-2.

[35] F. Rooms, "What's the point?," 2018. https://blog.bricsys.com/point-clouds-whats-the-point/.

[36] A. Chatterjee, "Geometric Calibration and Shape Refinement for 3D Reconstruction [eindwerk]," *Bengaluru Dep. Electr. Eng. INDIAN Inst. Sci.*, no. September 2015, 2016, [Online]. Available: https://www.researchgate.net/publication/307600956.

[37] F. Franceschini et al., *Distributed Large-Scale Dimensional Metrology*, vol. M, no. January 2018. 2011.

[38] H. Landau et al. "Trimble's Rtk And Dgps Solutions In Comparison With Precise Point Positioning," *Int. Assoc. Geod. Symp.*, vol. 133, pp. 709–718, 2009, doi: 10.1007/978-3-540-85426-5_81.

[39] J. Huang, "ROS tutorial #2: Publishers and subscribers," *YouTube*, 2016. https://www.youtube.com/watch?v=bJB9tv4ThV4.

[40] S. J. Danbatta, "Bluetooth Wireless Technologies Used in Home Automation," *2019 7th Int. Symp. Digit. Forensics Secur.*, pp. 1–5, 2019.

[41] M. Olsson, "Behavior Trees for decision-making in Autonomous Driving [eindwerk]," *Stock. R. Inst. Comput. Sci. communcication*, 2016, [Online]. Available: http://www.diva-portal.org/smash/get/diva2:907048/FULLTEXT01.pdf.

[42] M. Colledanchise and P. Ögren, "Behavior Trees in Robotics and AI: An Introduction," *Chapman Hall/CRC Press*, 2017, doi: 10.1201/9780429489105.

[43] D. Faconti, "BehaviorTree.CPP," *2018*. https://www.behaviortree.dev/ (accessed Nov. 16, 2020).

[44] M. Labbé and F. Michaud, “RTAB-Map as an open-source lidar and visual simultaneous localization and mapping library for large-scale and long-term online operation,” *J. F. Robot.*, vol. 36, no. 2, pp. 416–446, 2019, doi: 10.1002/rob.21831.

[45] T. Baltovski, “ros-drivers/urg_node: ROS wrapper for the Hokuyo urg_c library.,” *27 October 2020*. https://github.com/ros-drivers/urg_node (accessed Dec. 04, 2020).

[46] Xsens, “xsens/xsens_mti_ros_node: ROS node driver for Xsens devices.,” *27 February 2019*. https://github.com/xsens/xsens_mti_ros_node (accessed Dec. 04, 2020).

[47] P. Khandelwal and J. O’Quin, “ros-drivers/freenect_stack: libfreenect based ROS driver,” *18 March 2020*. https://github.com/ros-drivers/freenect_stack (accessed Dec. 11, 2020).

[48] T. Moore, “robot_localization wiki — robot_localization 2.6.8 documentation,” *7 Juli 2018*. http://docs.ros.org/en/melodic/api/robot_localization/html/index.html (accessed Nov. 25, 2020).

[49] D. Faconti, “BehaviorTree/BehaviorTree.CPP: Behavior Trees Library in C++.,” *10 October 2020*. https://github.com/BehaviorTree/BehaviorTree.CPP (accessed Dec. 11, 2020).

Bijlage A: ROS bestanden

```
<robot name="test_robot">
    <link name="base_link" />
    <link name="frame_base_link" />
    <link name="horizontal_laser_link" />
    <link name="imu" />
    <link name="camera_link" />

    <joint name="base_joint" type="fixed">
      <parent link="base_link"/>
      <child link="frame_base_link"/>
      <origin xyz="0 0 0" rpy="0 0 0" />
    </joint>

    <joint name="joint1" type="fixed">
      <parent link="frame_base_link"/>
      <child link="horizontal_laser_link"/>
      <origin xyz="0.042 0.04 0.45" rpy="0 0 0" />
    </joint>

    <joint name="joint2" type="fixed">
      <parent link="frame_base_link"/>
      <child link="camera_link"/>
      <origin xyz="0.09 0 0.265" rpy="0 0 0" />
    </joint>

    <joint name="joint3" type="fixed">
      <parent link="frame_base_link"/>
      <child link="imu"/>
      <origin xyz="0.0035 -0.005 0.155" rpy="0 0 0" />
    </joint>

</robot>
```

Figuur A-1: voertuig.urdf

```
<launch>
    <!-- Kinect start (dit of Ensenso activeren) -->
    <include file="$(find freenect_launch)/launch/freenect.launch">
        <arg name="depth_registration" value="True" />
    </include>

    <!-- Xsens IMU data -->
    <arg name="device_imu" default="auto" />
    <arg name="baudrate" default="0" />
    <arg name="timeout" default="0.002" />
    <arg name="frame_id_imu" default="imu" />
    <arg name="frame_local" default="ENU" />
    <arg name="no_rotation_duration" default="0" />
    <arg name="angular_velocity_covariance_diagonal" default="[0.0004, 0.0004, 0.0004]" />
    <arg name="linear_acceleration_covariance_diagonal" default="[0.0004, 0.0004, 0.0004]" />
    <arg name="orientation_covariance_diagonal" default="[0.01745, 0.01745, 0.15708]" />
    <arg name="position_covariance_diagonal" default="[0, 0, 0]" /> <!-- neem standaardfout van 10m op GPS data -->
    <arg name="position_covariance_type" default="2" />

    <!-- Xsens IMU start -->
    <node pkg="xsens_driver" type="mtnode.py" name="xsens_driver" output="screen" >
        <param name="device" value="$(arg device_imu)"/>
        <param name="baudrate" value="$(arg baudrate)"/>
        <param name="timeout" value="$(arg timeout)"/>
        <param name="frame_id" value="$(arg frame_id_imu)"/>
        <param name="frame_local" value="$(arg frame_local)"/>
        <param name="no_rotation_duration" value="$(arg no_rotation_duration)"/>
        <rosparam param="angular_velocity_covariance_diagonal" subst_value="True">$(arg angular_velocity_covariance_diagonal)</rosparam>
        <rosparam param="linear_acceleration_covariance_diagonal" subst_value="True">$(arg linear_acceleration_covariance_diagonal)</rosparam>
        <rosparam param="orientation_covariance_diagonal" subst_value="True">$(arg orientation_covariance_diagonal)</rosparam>
        <rosparam param="position_covariance_diagonal" subst_value="True">$(arg position_covariance_diagonal)</rosparam>
        <rosparam param="position_covariance_type" subst_value="True">$(arg position_covariance_type)</rosparam>
    </node>

    <!-- Hokuyo start -->
    <node pkg="urg_node" type="urg_node" name="lidar" output="screen">
        <param name="ip_address" value="192.168.0.11"/>
        <param name="serial_port" value=""/>
        <param name="frame_id" value="horizontal_laser_link"/>
        <param name="calibrate_time" type="bool" value="false"/>
        <param name="intensity" type="bool" value="false"/>
        <param name="publish_multiecho" value="true"/>
        <param name="cluster" value="1"/>
    </node>

    <!-- URDF model inladen -->
    <arg name="gui" default="False" />
    <param name="robot_description" textfile="/home/aun/catkin_ws_n00b/src/urdf/sensor_platform.urdf" />
    <param name="use_gui" value="$(arg gui)"/>
    <node name="joint_state_publisher" pkg="joint_state_publisher" type="joint_state_publisher" />
    <node name="robot_state_publisher" pkg="robot_state_publisher" type="state_publisher" />
</launch>
```

Figuur A-2: sensors_launch.launch

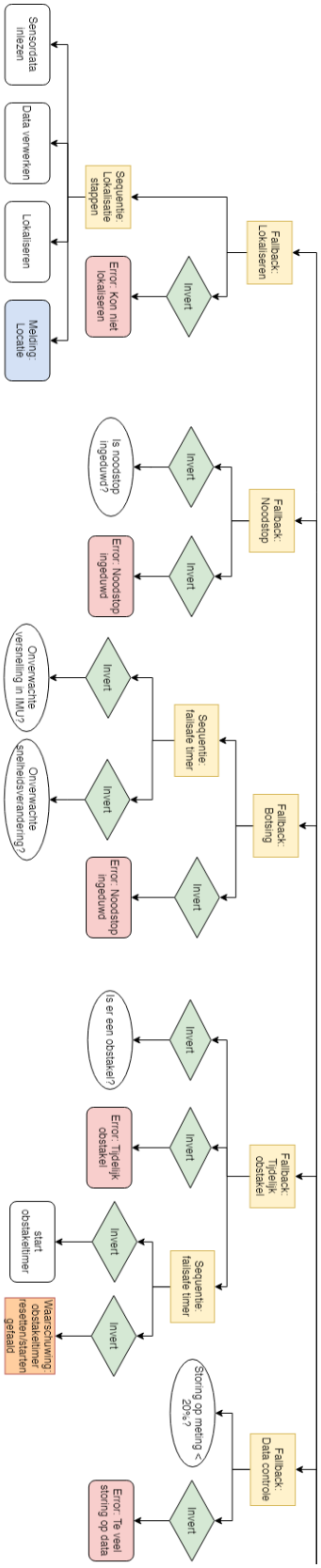
```
<launch>

  <!-- De sensoren en URDF moeten geactiveerd zijn vooraleer dit bestand te launchen.
       (gebruik hiervoor sensors_launch.launch) -->

  <!-- vaak voorkomende argumenten definiëren -->
  <arg name="frame_id"                default="base_link" />
  <arg name="rgb_topic"               default="/camera/rgb/image_rect_color" />
  <arg name="depth_topic"             default="/camera/depth_registered/image_raw" />
  <arg name="camera_info_topic"       default="/camera/rgb/camera_info" />
  <arg name="imu_topic"               default="/imu/data" />
  <arg name="odom_in"                 default="/odometry/filtered" />
  <arg name="scan"                    default="/first" />
  <arg name="gps"                     default="/fix" />

  <arg name="imu_ignore_acc"          default="true" />
  <arg name="imu_remove_gravitational_acceleration" default="true" />
  <arg name="queue_size"              default="20"/>

  <!-- Lokalisatie mode activeren -->
  <arg name="localization"      default="true"/>
  <arg     if="$(arg localization)" name="rtabmap_args"  default=""/>
  <arg unless="$(arg localization)" name="rtabmap_args"  default="--delete_db_on_start"/>

  <group ns="rtabmap">
    <!-- RGBD-sync -->
    <node pkg="nodelet" type="nodelet" name="rgbd_sync" args="standalone rtabmap_ros/rgbd_sync" output="screen">
      <remap from="rgb/image"       to="$(arg rgb_topic)"/> <!-- input -->
      <remap from="depth/image"     to="$(arg depth_topic)"/>
      <remap from="rgb/camera_info" to="$(arg camera_info_topic)"/>
      <remap from="rgbd_image"      to="rgbd_image"/>  <!-- output -->

      <!-- Zet argument op true voor niet-gesynchroniseerde camera's
           (bvb. false voor kinectv2, zed, realsense, true for xtion, kinect360)-->
      <param name="approx_sync"     value="true"/>
    </node>

    <!-- Visual Odometry -->
    <node pkg="rtabmap_ros" type="rgbd_odometry" name="visual_odometry" output="screen" args="$(arg rtabmap_args)">

      <param name="subscribe_depth" type="bool" value="false"/>
      <param name="subscribe_rgbd"  type="bool" value="true"/>

      <remap from="rgbd_image"      to="rgbd_image"/>
      <remap from="odom"            to="/vo"/>

      <param name="frame_id"              type="string" value="$(arg frame_id)"/>
      <param name="publish_tf"            type="bool"   value="false"/>
      <param name="publish_null_when_lost" type="bool"  value="true"/>
      <param name="Odom/FillInfoData"     type="string" value="true"/>
      <param name="Odom/ResetCountdown"   type="string" value="1"/>
      <param name="Vis/FeatureType"       type="string" value="6"/>
      <param name="OdomF2M/MaxSize"       type="string" value="1000"/>
      <param name="RGBD/OptimizeMaxError" type="string" value="10"/>
      <param name="queue_size"            type="int"    value="$(arg queue_size)"/>
    </node>

    <!-- SLAM -->
    <node name="rtabmap" pkg="rtabmap_ros" type="rtabmap" output="screen" args="$(arg rtabmap_args)">

      <param name="subscribe_depth" type="bool" value="false"/>
      <param name="subscribe_rgbd"  type="bool" value="true"/>
      <param name="subscribe_scan"  type="bool" value="true"/>

      <remap from="odom"            to="$(arg odom_in)"/>
      <remap from="scan"            to="$(arg scan)"   />
      <remap from="gps/fix"         to="$(arg fix)"    />
      <remap from="rgbd_image"      to="rgbd_image"    />

      <param name="frame_id"                 type="string" value="$(arg frame_id)"/>
      <param name="queue_size"               type="int"    value="$(arg queue_size)"/>
      <param name="Kp/DetectorStrategy"      type="string" value="6"/>
      <param name="RGBD/NeighborLinkRefining" type="string" value="true"/>
      <param name="RGBD/ProximityBySpace"    type="string" value="true"/>
      <param name="RGBD/AngularUpdate"       type="string" value="0.01"/>
      <param name="RGBD/LinearUpdate"        type="string" value="0.01"/>
      <param name="RGBD/OptimizeFromGraphEnd" type="string" value="false"/>
      <param name="Grid/FromDepth"           type="string" value="false"/>
      <param name="Reg/Strategy"             type="string" value="2"/>
      <param name="Reg/Force3DoF"            type="string" value="true"/>
      <param name="Rtabmap/WithGPS"          type="string" value="true"/>

      <!-- lokalisatie mode -->
      <param name="Mem/InitWMWithAllNodes" type="string" value="$(arg localization)"/>
      <param     if="$(arg localization)" name="Mem/IncrementalMemory" type="string" value="false"/>
      <param unless="$(arg localization)" name="Mem/IncrementalMemory" type="string" value="true"/>
    </node>
```

Figuur A-3: rtabRGBD.launch (1/2)

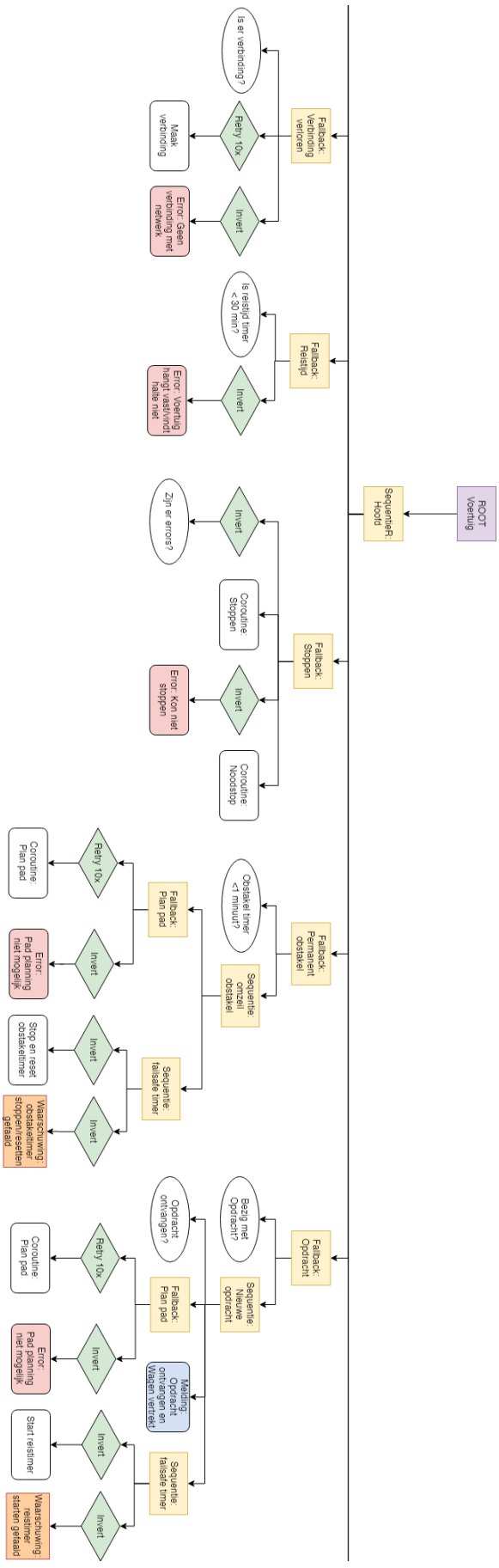
```
        <!-- Visualisation RTAB-Map -->
    <node pkg="rtabmap_ros" type="rtabmapviz" name="rtabmapviz" args="$(arg rtabmap_args)" output="screen" launch-prefix="">
      <param name="subscribe_depth"            type="bool"   value="false"/>
      <param name="subscribe_rgbd"             type="bool"   value="false"/>
      <param name="subscribe_scan"             type="bool"   value="false"/>
      <param name="subscribe_odom"             type="bool"   value="false"/>
      <param name="subscribe_scan_cloud"       type="bool"   value="false"/>
      <param name="subscribe_scan_descriptor" type="bool"   value="false"/>
      <param name="subscribe_odom_info"        type="bool"   value="false"/>
      <param name="frame_id"                   type="string" value="$(arg frame_id)"/>
      <param name="queue_size"                 type="int"    value="$(arg queue_size)"/>

      <remap from="rgbd_image"   to="rgbd_image"/>
      <remap from="odom"         to="$(arg odom_in)"/>
      <remap from="scan"         to="$(arg scan)"/>
    </node>
  </group>

  <!-- Odometrie fusion (UKF) -->
  <node pkg="robot_localization" type="ukf_localization_node" name="ukf_localization" clear_params="true" output="screen">

      <param name="frequency"             value="10"/>
      <param name="sensor_timeout"        value="0.1"/>
      <param name="two_d_mode"            value="false"/>
      <param name="odom_frame"            value="odom"/>
      <param name="base_link_frame"       value="$(arg frame_id)"/>
      <param name="world_frame"           value="odom"/>
      <param name="transform_time_offset" value="0.0"/>
      <param name="odom0"                 value="/vo"/>
      <param name="imu0"                  value="$(arg imu_topic)"/>

      <!-- De volgorde is x,y,z, roll,pitch,yaw, vx,vy,vz, vroll,vpitch,vyaw, ax,ay,az. -->
      <rosparam param="odom0_config">[false, false, false,
                                      false, false, false,
                                      true, true, false,
                                      false, false, true,
                                      false, false, false]</rosparam>

      <rosparam     if="$(arg imu_ignore_acc)" param="imu0_config">[
                                     false, false, false,
                                     true,  true,  true,
                                     false, false, false,
                                     true,  true,  true,
                                     false,  false,  false] </rosparam>
      <rosparam unless="$(arg imu_ignore_acc)" param="imu0_config">[
                                     false, false, false,
                                     true,  true,  true,
                                     false, false, false,
                                     true,  true,  true,
                                     true,  true,  false] </rosparam>

      <param name="odom0_differential" value="false"/>
      <param name="imu0_differential" value="false"/>
      <param name="odom0_relative" value="true"/>
      <param name="imu0_relative" value="true"/>
      <param name="imu0_remove_gravitational_acceleration" value="$(arg imu_remove_gravitational_acceleration)"/>
      <param name="print_diagnostics" value="true"/>

      <!-- Geavanceerde parameters -->
      <param name="odom0_queue_size" value="5"/>
      <param name="imu0_queue_size" value="5"/>

      <!-- De volgorde is x,y,z, roll,pitch,yaw, vx,vy,vz, vroll,vpitch,vyaw, ax,ay,az. -->
      <rosparam param="process_noise_covariance">[0.005, 0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                                                  0,     0.005, 0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                                                  0,     0,     0.006, 0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                                                  0,     0,     0,     0.003, 0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                                                  0,     0,     0,     0,     0.003, 0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                                                  0,     0,     0,     0,     0,     0.003,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                                                  0,     0,     0,     0,     0,     0,     0.0025, 0,     0,     0,     0,     0,     0,     0,     0,
                                                  0,     0,     0,     0,     0,     0,     0,     0.0025, 0,     0,     0,     0,     0,     0,     0,
                                                  0,     0,     0,     0,     0,     0,     0,     0,     0.0000004, 0,     0,     0,     0,     0,     0,
                                                  0,     0,     0,     0,     0,     0,     0,     0,     0,     0.001, 0,     0,     0,     0,     0,
                                                  0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0.001, 0,     0,     0,     0,
                                                  0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0.002, 0,     0,     0,
                                                  0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0.001, 0,     0,
                                                  0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0.001, 0,
                                                  0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0.0015]</rosparam>

          <!-- De volgorde is x,y,z, roll,pitch,yaw, vx,vy,vz, vroll,vpitch,vyaw, ax,ay,az. -->
          <rosparam param="initial_estimate_covariance">[1e-9, 0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                                                         0,     1e-9, 0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                                                         0,     0,     1e-9, 0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                                                         0,     0,     0,     1e-9, 0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                                                         0,     0,     0,     0,     1e-9, 0,     0,     0,     0,     0,     0,     0,     0,     0,     0,
                                                         0,     0,     0,     0,     0,     1e-9, 0,     0,     0,     0,     0,     0,     0,     0,     0,
                                                         0,     0,     0,     0,     0,     0,     1e-9, 0,     0,     0,     0,     0,     0,     0,     0,
                                                         0,     0,     0,     0,     0,     0,     0,     1e-9, 0,     0,     0,     0,     0,     0,     0,
                                                         0,     0,     0,     0,     0,     0,     0,     0,     1e-9, 0,     0,     0,     0,     0,     0,
                                                         0,     0,     0,     0,     0,     0,     0,     0,     0,     1e-9,  0,     0,     0,     0,     0,
                                                         0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     1e-9,  0,     0,     0,     0,
                                                         0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     1e-9, 0,     0,     0,
                                                         0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     1e-9, 0,     0,
                                                         0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     1e-9, 0,
                                                         0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     0,     1e-9]</rosparam>
    </node>
</launch>
```

Figuur A-4: rtabRGBD (2/2)

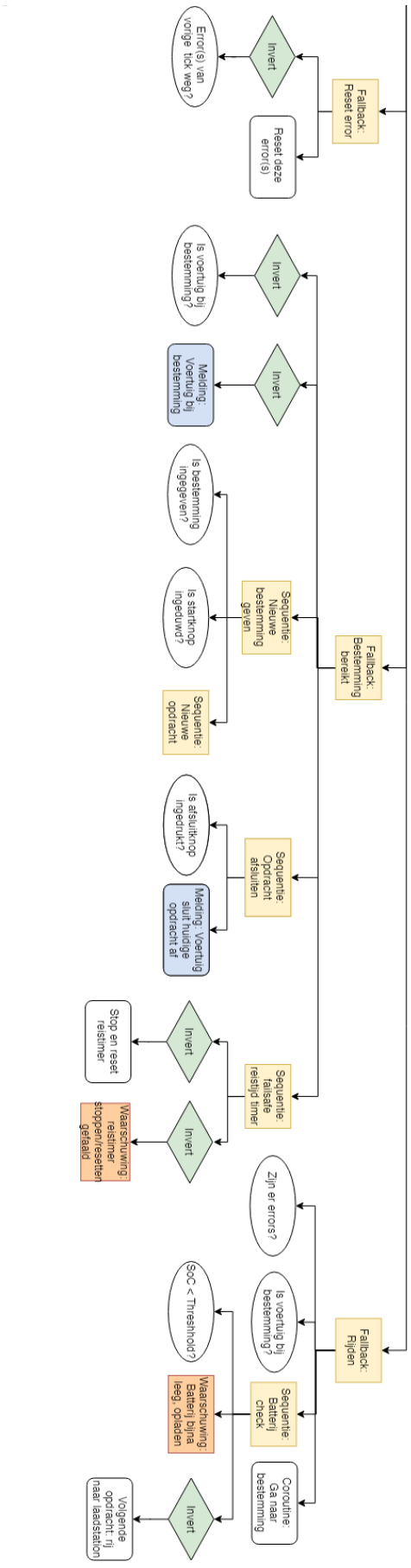
Bijlage B: Behavior tree



Figuur B-1 : Volledige BT (1/3)



Figuur B-2 : Volledige BT (2/3)



Figuur B-3 : Volledige BT (3/3)

Bijlage C: Verbinding verloren

```cpp
#include <chrono>

#include "behaviortree_cpp_v3/bt_factory.h"
#include "nodes.h"

/*
 * Main-functie van voorbeeld verbindingVerloren
 * Maakt gebruik van nodes.h, nodes.cpp, verbindingVerloren.xml en BehaviorTree.cpp
*/
int main() {
    using namespace nodes;
    using namespace BT;

     // Gebruik BehaviorTreeFactory om nodes te registreren
    BehaviorTreeFactory factory;

    // Opgelet: Naam om node te registreren moet zelfde zijn als ID gebruikt in .XML

    /*
     * Aangeraden manier om nodes te maken door erving
     * Geeft meer functionaliteiten dan andere methodes zoals het direct kunnen gebruiken van poorten
    */
    factory.registerNodeType<maakVerbinding>("maakVerbinding");
    factory.registerNodeType<genereerError>("genereerError");

    /*
     * Alternatieve manier om nodes te maken door functie pointer
     * Enkel aangeraden voor simpele nodes aangezien het gebruik van poorten niet meteen mogelijk is.
    */
    factory.registerSimpleCondition("isErVerbinding", std::bind(isErVerbinding));

    /*
     * Boom maken op basis van .XML (op het moment is enkel .XML ondersteund)
     * Volledige pad van home naar .xml moet opgegeven worden
    */
    auto tree = factory.createTreeFromFile("/home/ubuntu/catkin_ws_BT/src/BehaviorTree.CPP-master/examples/verbindingVerloren.xml");

    /*
     * Blijf ticken tot root "SUCCESS" of "FAILURE geeft
     * In het geval van "RUNNING" zal boom een tijd pauzeren vooraleer verder te werken OBV gemaakte boom
    */
    while( tree.tickRoot() == NodeStatus::RUNNING)
    {
        std::this_thread::sleep_for( std::chrono::milliseconds(1500) );
    }


    return 0;
}
```

Figuur C-1 : verbindingVerloren.cpp

```xml

<!-- Xml file van het voorbeeld verbindingVerloren, pas de WIFI aan om de werking van de BT te veranderen -->
<!-- Het is hierna niet nodig om opnieuw een catkin_make uit te voeren -->
<root main_tree_to_execute = "verbindingVerloren">
    <BehaviorTree ID="errorMelding">
       <Action ID="genereerError" name="genereer_error" errorMelding="{error}" />
    </BehaviorTree>

    <BehaviorTree ID="verbindingVerloren">
       <Fallback>
          <Condition ID="isErVerbinding"   name="is_er_verbinding"/>
          <RetryUntilSuccesful num_attempts="2">
         <Action ID="maakVerbinding" name="maak_verbinding" WIFI="192.168.0.1" />
        </RetryUntilSuccesful>
          <Inverter>
         <SetBlackboard output_key="soortError" value="verbindingsfout" />
          </Inverter>
          <Inverter>
         <SubTree ID="errorMelding" error="soortError" />
        </Inverter>
        </Fallback>
    </BehaviorTree>
</root>
```

Figuur C-2 : verbindingVerloren.xml

```cpp
#ifndef BT_NODES_H
#define BT_NODES_H

#include "../include/behaviortree_cpp_v3/behavior_tree.h"
#include "../include/behaviortree_cpp_v3/bt_factory.h"

/*
 * Header file van de gebruikte nodes
 * isErVerbinding, maakVerbinding en genereerError
 * Maakt gebruik van BehaviorTree.cpp
*/

namespace nodes
{

/*
 * Conditie isErVerbinding, opgesteld als functie met "registerSimpleCondition" in verbindingVerloren.cpp
*/
BT::NodeStatus isErVerbinding();

/*
 * Actie maakVerbinding, opgesteld als klasse met "registerNodeType" in verbindingVerloren.cpp
 * Gebruikt inputpoort WIFI
*/
class maakVerbinding : public BT::CoroActionNode
{
  public:
    // Constructor van klasse maakVerbinding
    maakVerbinding(const std::string& name, const BT::NodeConfiguration& config):
        CoroActionNode(name, config)
    {
    }

    // De functie tick() moet override worden
    BT::NodeStatus tick() override;

    // Verplicht om de gebruikte poorten aan te duiden
    static BT::PortsList providedPorts()
    {
        return{BT::InputPort<std::string>("WIFI") };
    }
};

/*
 * Actie geneerError, opgesteld als klasse met "registerNodeType" in verbindingVerloren.cpp
 * Gebruikt inputpoort errorMelding
*/
class genereerError : public BT::SyncActionNode
{
  public:
    // Constructor van klasse maakVerbinding
    genereerError(const std::string& name, const BT::NodeConfiguration& config)
      : SyncActionNode(name, config)
    {
    }

    // De functie tick() moet override worden
    BT::NodeStatus tick() override;

    // Verplicht om de gebruikte poorten aan te duiden
    static BT::PortsList providedPorts()
    {
        return{BT::InputPort<std::string>("errorMelding") };
    }
};
}

#endif   // BT_NODES_H
```

Figuur C-3 : nodes.h

```cpp
#include <iostream>
#include <string>

#include "nodes.h"

/*
 * .cpp file van de gebruikte nodes
 * isErVerbinding, maakVerbinding en genereerError
 * Maakt gebruik van nodes.h
*/
namespace nodes
{

/*
 * Functie isErVerbinding
*/
BT::NodeStatus isErVerbinding()
{
    std::string verbonden;

    std::cout << "Is er verbinding? (y/n)" << std::endl;
    std::cin >> verbonden;

    if (verbonden == "y") {
        std::cout << " " << std::endl;
        std::cout << "Er is verbinding met netwerk" << std::endl;
        return BT::NodeStatus::SUCCESS;
    }
    else if (verbonden == "n") {
        std::cout << " " << std::endl;
        std::cout << "geen verbinding met netwerk, proberen verbinding te maken ..." << std::endl;
        std::cout << " " << std::endl;
        std::cout << " -------------------- " << std::endl;
        std::cout << " " << std::endl;
        return BT::NodeStatus::FAILURE;
    }
    else {
        std::cout << " " << std::endl;
        std::cout << "verkeerde input, test stoppen" << std::endl;
        return BT::NodeStatus::SUCCESS;
    }

}

//--------------------------------------------------------------------------


/*
 * Functie tick() van klasse maakverbinding
*/
BT::NodeStatus maakVerbinding::tick() {

    auto msg = getInput<std::string>("WIFI");
    int counter = 0;

    if (!msg)
    {
        throw BT::RuntimeError( "missing required input [message]: ", msg.error() );
    }

    while (true ){
        if (msg.value() == "192.168.0.1" && counter == 4) {
            std::cout << "verbinding is gemaakt" << std::endl;
            return BT::NodeStatus::SUCCESS;
        }
        else if  (counter == 2 && !(msg.value() == "192.168.0.1")) {
            std::cout << "kon geen verbinding maken met netwerk, controleer WIFI-gegevens:" << std::endl;
            std::cout << msg.value() << std::endl;
            std::cout << " " << std::endl;
            std::cout << " -------------------- " << std::endl;
            std::cout << " " << std::endl;
            return BT::NodeStatus::FAILURE;
        }

        std::cout << "uitvoering " << name() << " duurt te lang, node geeft bezig: " << counter << std::endl;
        std::cout << " " << std::endl;
        std::cout << " -------------------- " << std::endl;
        std::cout << " " << std::endl;
        counter++;

        /*
         * Return nodestatus "RUNNING"
         * Na pauze zal het programma in while-lus verder gaan (zie verbindingVerloren.cpp)
        */
        setStatusRunningAndYield();
    }
}

//--------------------------------------------------------------------------

/*
 * Functie tick() van klasse genereerError
*/
BT::NodeStatus genereerError::tick()
{
```

Figuur C-4 : nodes.cpp (1/2)

```
 96      auto msg = getInput<std::string>("errorMelding");
 97
 98      if (!msg)
 99      {
100          throw BT::RuntimeError( "missing required input [message]: ", msg.error() );
101      }
102
103      std::cout << "twee keer geprobeerd verbinding te maken, maar mislukt." << std::endl;
104      std::cout << " " << std::endl;
105      std::cout << "error melding: " << msg.value() << std::endl;
106      return BT::NodeStatus::SUCCESS;
107  }
108
109  }
110
```

Figuur C-5 : nodes.cpp (2/2)