2015•2016
FACULTEIT INDUSTRIËLE INGENIEURSWETENSCHAPPEN
master in de industriële wetenschappen: energie

Masterproef

Kaartopbouw en lokalisatie voor automatisch geleide voertuigen met natuurlijke oriëntatiepunten

Promotor :
Prof. dr. ir. Eric DEMEESTER

Promotor :
ing. JAN KEMPENEERS

Stijn Vandenboer , Miggel Wellens

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

Gezamenlijke opleiding Universiteit Hasselt en KU Leuven

2015•2016

Faculteit Industriële ingenieurswetenschappen

master in de industriële wetenschappen: energie

Masterproef

Kaartopbouw en lokalisatie voor automatisch geleide voertuigen met natuurlijke oriëntatiepunten

Promotor :
Prof. dr. ir. Eric DEMEESTER

Promotor :
ing. JAN KEMPENEERS

Stijn Vandenboer , Miggel Wellens

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

universiteit hasselt | KU LEUVEN

Woord vooraf

Deze masterproef werd volbracht in ons laatste jaar als studenten Industrieel ingenieur-Energie met als focus automatisering. Hiervoor hebben we in september als onderzoekstopic “AGV-navigatie op basis van natuurlijke oriëntatiepunten” gekozen. Dit vooral omdat we hier nog geen ervaring in hadden en dit onderwerp ons interesseert. Ook zien we dat AGV-systemen de laatste jaren een sterke groei kennen waardoor deze een steeds belangrijkere rol krijgen in industriële automatisering.

Ook de manier van programmering zou nieuw voor ons zijn, in voorgaande jaren in de labo’s werd gewerkt met programma’s die weinig rekenkracht nodig hebben. Door de grote hoeveelheid data die in dit project wordt uitgelezen en verwerkt, en de berekeningen die hierop volgen was dit zeker een uitdaging met een steile leercurve voor ons.

Graag willen we nog het onderzoekscentrum Sirris te Diepenbeek bedanken waar we onze masterproef hebben kunnen realiseren. Alsook alle personen die ons geholpen en/of ondersteund hebben in deze periode, in het bijzonder: Ing. Jan Kempeneers onze externe promotor voor zijn tijd, ondersteuning en zijn inzicht; prof. dr. ir. Eric Demeester onze interne promotor voor zijn hulp tijdens de programmatie en ondersteuning tijdens de gehele masterproef. Mabo nv voor het ter beschikking stellen van de test-AGV en hulp tijdens de programmatie en testen; en tenslotte Pepperl+Fuchs met in het bijzonder ing. Luc Bervoets voor het ter beschikking te stellen van hun 2D-laserscanner.

Ook danken wij onze ouders voor hun morele en financiële steun tijdens deze masterproef maar ook om ons de mogelijkheid hebben gegeven om onze studies te doen.

Verder zijn wij alle medewerkers van Sirris en Acro dankbaar voor ons altijd met alle vriendelijkheid te ontvangen.

Vandenboer Stijn
Wellens Miggel

Inhoudsopgave

Lijst van tabellen

Lijst van figuren

Verklarende woordenlijst

AGV	Automatisch geleid voertuig
AMCL	Adaptive monte carlo localization
CNC	Computer numerical control
COM	Communication port
CPU	Central processing unit
EKF	Extended kalman filter
KLD	Kullback-Leibler divergence
MCL	Monte carlo localization
RAM	Random-access memory
ROS	Robot operating system
Rviz	ROS visualisation
SLAM	Simultaneous localization and mapping
Tf	Transformatie
UDP	User datagram protocol

Abstract

Zelfrijdende robots worden steeds meer gebruikt in fabrieksomgevingen; hierbij ontlasten ze magazijnmedewerkers door repetitieve taken op zich te nemen. Tot dusver positioneert een industriële AGV zich met een 2D-laserscanner, die zich oriënteert op kunstmatige oriëntatiepunten, met name reflectoren op een gekende positie. Maar in de praktijk worden deze reflectoren soms geblokkeerd door objecten. Bovendien kost het inmeten van deze kunstmatige oriëntatiepunten veel tijd, en kan de eindgebruiker dit meestal niet zelf. Een mogelijke oplossing is het gebruik van de aanwezige omgevingskenmerken, de zgn. “natuurlijke oriëntatiepunten”.

In deze masterproef werd volgende procedure uitgewerkt. Een 2D- laserscanner meet een puntenwolk van de omgeving, die gebruikt wordt om de positie van de laserscanner te bepalen door deze puntenwolk te vergelijken met een op voorhand ingemeten kaart van de omgeving. De kaart wordt opgebouwd m.b.v. het Gmapping algoritme. Dit gebeurt door handmatig rond te rijden met de AGV, in Rviz, de visualisatieomgeving van ROS, kan de mapopbouw gevolgd worden. Eenmaal de kaart gekend is wordt het AMCL-algoritme gebruikt om lokalisatie in die kaart te doen.

Uit de testen blijkt dat navigatie met natuurlijke oriëntatiepunten mogelijk is en een mogelijke vervanger is voor kunstmatige oriëntatiepunten. Beide systemen hebben hun eigen voor- en nadelen. Een keuze tussen de twee zal van verschillende factoren afhangen.

Abstract in English

Nowadays, self-propelled robots are increasingly used in factory environments; hereby they relieve warehouse employees by carrying out repetitive tasks. So far, industrial AGVs position themselves by means of a 2D laser scanner which is oriented based on artificial landmarks, in particular reflectors at a known position. However, in practice these reflectors are sometimes blocked by objects. Moreover, it takes a lot of time to survey these artificial landmarks, and the end user is often not able to do this himself. One possible solution is to use the existing environmental characteristics, the so-called "natural landmarks."

In this master thesis following procedure was developed. A 2D laser scanner records a point cloud of the environment and determines its position by comparing it with a pre-built map of the environment. The map is constructed with the Gmapping algorithm. In Rviz, the visualization area of ROS, the build-up of the map can be monitored while the AGV is driven around by hand. Once the map is known, the AMCL algorithm is used for localization based on that map.

Tests show that navigation with natural landmarks is possible and could be a potential replacement for artificial landmarks. Both systems have their own advantages and disadvantages. A choice between the two will depend on several factors.

1 Inleiding

1.1 Situering

AGV's (Automated Guided Vehicles) worden de laatste jaren meer en meer ingezet in industriële toepassingen om automatisch allerlei goederen te vervoeren in productie-, magazijn- en distributieomgevingen. Deze masterproef verricht onderzoek naar navigatie op basis van natuurlijke omgevingskenmerken voor Mabo Engineering & Automation nv. via de organisatie Sirris. Sirris is het collectief centrum van de Belgische technologische industrie en helpt bedrijven bij het invoeren van technologische innovaties [1]. Hiermee versterken Belgische bedrijven hun concurrentiepositie op een duurzame manier. Sirris beschikt over een demoruimte die representatief is voor industriële omgevingen en die volledig autonoom kan functioneren. Het hoofddoel van deze werkomgeving is demonstreren en onderzoeken hoe automatisering ook voor kleine series efficiënt gerealiseerd kan worden. Vorig jaar werd de werkomgeving uitgebreid met een AGV-systeem. Mabo heeft een heftruckgamma en verkoopt ook AGV-systemen die werken op basis van kunstmatige oriëntatiepunten. Maar momenteel willen ze weten of een systeem op basis van natuurlijke oriëntatiepunten interessanter is. Hiervoor heeft Mabo nv. de vraag gesteld aan Sirris, die dit in de demoruimte kan onderzoeken. Mabo heeft een test AGV ter beschikking die al uitgerust is met encoders op de wielen en een bestaande laserscanner. Momenteel kan de AGV navigeren door middel van *kunstmatige* oriëntatiepunten (in dit geval zijn dit reflectoren) en een 2D-laserscanner. De posities van deze kunstmatige oriëntatiepunten moeten vooraf op de kaart van het AGV-systeem bepaald zijn. De laserscanner meet voortdurend de afstand en de hoek t.o.v. deze oriëntatiepunten. De software kan hieruit de positie en oriëntatie van de AGV bepalen.

Figuur 1: Test AGV Mabo

1.2 Probleemstelling

Kunstmatige oriëntatiepunten maken het systeem vanwege 3 redenen minder flexibel. Ten eerste kunnen oriëntatiepunten verstoord worden wanneer machines veranderen van plaats, en dat maakt dat deze oriëntatiepunten opnieuw geplaatst moeten worden. Ten tweede is de plaatsing van nieuwe oriëntatiepunten vrij omslachtig en tijdrovend, omdat de reflectoren met een theodoliet ingemeten worden. Ten derde beschikt de eindgebruiker meestal niet over het nodig materiaal en kennis waardoor zelf aanpassingen maken moeilijk is.

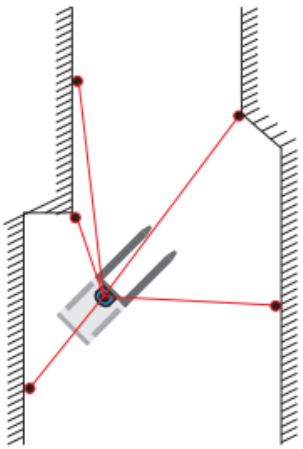
1.3 Doelstellingen en onderzoeksvragen

De masterproef heeft als doel een navigatiesysteem te implementeren dat *natuurlijke* oriëntatiepunten gebruikt voor lokalisatie van de AGV (Figuur 2). Natuurlijke oriëntatiepunten zijn stabiele omgevingskenmerken zoals muren of andere geometrische en visuele informatie die van nature aanwezig zijn in de omgeving.

Ten eerste moet het systeem via een teach-in modus de werkomgeving van de AGV inscannen. Deze kaartopbouw is in de literatuur gekend als SLAM: Simultaneous Localisation And Mapping. Eenmaal de kaart gekend is, moet de AGV in staat zijn zijn positie te bepalen met een nauwkeurigheid van ±1 cm en zijn oriëntatie met een nauwkeurigheid van ±0,5°.

Ten tweede moeten verschillende 2D-laserscanners gemonteerd worden op de AGV (Pepperl+Fuchs, Sick, Hokuyo), zodat we een scanner met de beste prijs/kwaliteit verhouding kunnen selecteren. De kwaliteit zal bepaald worden door de nauwkeurigheid van de positionering zowel statisch als dynamisch.

Tenslotte wordt het systeem op basis van natuurlijke oriëntatiepunten vergeleken met het systeem op basis van kunstmatige oriëntatiepunten op basis van kostprijs, gebruiksgemak, nauwkeurigheid en betrouwbaarheid. Uit deze vergelijking zal blijken of het interessant is voor Mabo om al dan niet over te schakelen.



Figuur 2: Navigatie op basis van natuurlijke oriëntatiepunten [2]

1.4 Methode

Als eerste zal er een literatuurstudie gedaan worden naar de huidige navigatiesystemen die men in de industrie gebruikt: hoe deze systemen in het algemeen werken en hoe de scanner- en encoderdata samen gebruikt worden om een zo juist mogelijke positionering te bereiken. Vervolgens zal er een vergelijking van al de 2D-laserscanners die er momenteel op de markt zijn en die in aanmerking komen voor deze toepassing, met als resultaat een lijst met hun eigenschappen en prijs. Daarnaast is een gedetailleerde studie naar de algoritmes voor de kaartopbouw en lokalisatie essentieel voor het verwezenlijken van dit project.

Om de kaartopbouw te programmeren wordt "Gmapping" gebruikt, dit is een SLAM algoritme dat aanwezig is in de ROS bibliotheek (zie 4.1 ROS Robot Operating System). In rviz, de visualisatieomgeving van ROS, kan de map opbouw gevolgd worden wanneer handmatig rondgereden word met de AGV. Eenmaal de kaart gekend is wordt er gebruik gemaakt van het "AMCL" algoritme om de lokalisatie in die map te doen. Het "AMCL" algoritme is tevens ook aanwezig in de ROS bibliotheek.

Vervolgens is er contact opgenomen met fabrikanten, voor de vraag om eventueel mee te werken aan ons eindwerk. Dankzij het bedrijf Pepperl+Fuchs is het mogelijk om een 2D-laserscanner te lenen over de volledige masterproef. Verder worden ook de "Sick NAV350" die al op de test AGV staat en de "Hokuyo URG-04LX-UG01" die we van Acro kunnen lenen getest en vergeleken. De vergelijking gebeurt zowel statisch als dynamisch. Statisch wordt de scanner getest door de AGV te laten stoppen op zijn eindpositie, vervolgens kan men zijn x- en y-positie aanduiden. Uit de afwijking van deze posities kan men de statische nauwkeurigheid bepalen. Om het systeem dynamisch te testen krijgt de AGV een vast pad om te volgen. Er kan dan gecontroleerd worden hoe nauwkeurig de AGV zijn pad volgt, eveneens kan zijn herhaalbaarheid zo getest worden.

Als laatste zal het volledig systeem getest worden in de demoruimte van Sirris die o.a. een draaifreescentrum en een CNC-meetsysteem bevat. De AGV transporteert hierbij afgewerkte producten van het draaifreescentrum naar het CNC-meetsysteem. Vervolgens worden deze verder getransporteerd naar een klein magazijn.

Figuur 3: Demoruimte

1.5 Belangrijkste bijdrage

In deze masterproef is een AGV uitgerust met een positionering o.b.v. natuurlijke oriëntatiepunten. Vervolgens is dit systeem vergeleken met de conventionele positioneermethode o.b.v. kunstmatige oriëntatiepunten. De 2 systemen zijn vergeleken o.b.v. nauwkeurigheid, gebruiksgemak, kostprijs en betrouwbaarheid. Ook is er een overzicht gecreëerd van commercieel beschikbare 2D-laserscanners voor AGV-navigatie.

1.6 Overzicht structuur scriptie

Gezien de snelle evolutie in 2D scanners voor navigatie van mobiele platformen (AGV's, robots, …) was er geen duidelijk en up-to-date overzicht beschikbaar van commercieel beschikbare 2D scanners. Omdat het selecteren van een alternatieve 2D scanner deel uitmaakte van de masterproef hebben we via een literatuurstudie een dergelijk overzicht gecreëerd. Dit wordt samen met een aantal algemene begrippen die in de rest van de scriptie gebruikt worden uitgelegd in hoofdstuk 2. Als je nog nooit met ROS hebt gewerkt is het zeer moeilijk om u hier in het begin wegwijs uit te maken. Daarom wordt in hoofdstuk 3 het concept en enkele begrippen van ROS uitgelegd. Verder wordt in dit hoofdstuk ook nog onze programma opbouw uitgelegd. Nadat alles geprogrammeerd en werkend was, zijn er enkele testen uitgevoerd op zijn nauwkeurigheid. Die testen kan men vinden in Hoofdstuk 4. Tenslotte is in hoofdstuk 5 een vergelijking gemaakt met de positioneermethode o.b.v. kunstmatige oriëntatiepunten.

2 Literatuurstudie

2.1 Inleiding

Het principe van probabilistische robotica is het inschatten van een toestand met behulp van data die via een sensor bekomen worden. Bijvoorbeeld, het bewegen van een mobiele robot is relatief gemakkelijk als men de exacte locatie van de robot en alle nabijgelegen obstakels kent. Helaas zijn deze variabelen niet direct meetbaar. In plaats daarvan is een robot aangewezen op zijn sensoren om deze informatie te verzamelen. Sensoren verzamelen alleen maar stukjes informatie, bovendien bestaat de data, door ruis, gedeeltelijke uit foute informatie [3].

2.1.1 Verklaring begrip toestand

De toestand wordt gezien als de verzameling van alle aspecten van de robot en zijn omgeving die invloed kunnen hebben op de toekomst. Het bevat variabelen met betrekking tot de robot zelf, zoals positie, snelheid, de correcte werking van zijn sensor enzovoort. Verder wordt de toestand aangeduid met een x. De toestand op tijdstip t wordt in het algemeen aangeduid met $\boldsymbol{x_t}$.

Een toestand x_t zal compleet zijn als de kennis van de vorige toestanden (metingen of controles) geen aanvullende informatie meer dragen die de toekomst nauwkeuriger zou voorspellen [3].

2.1.2 Interactie met de omgeving

Er zijn twee fundamentele vormen van interactie tussen een robot en zijn omgeving. De robot kan de toestand van zijn omgeving beïnvloeden door middel van haar actuatoren. En het kan informatie verzamelen over zijn toestand door middel van haar sensoren.

- **Sensormetingen**: waarnemen is het proces waarmee de robot zijn sensorens gebruikt om informatie over de toestand van zijn omgeving te verkrijgen. Typisch arriveren sensormetingen met enige vertraging, vandaar dat ze informatie geven over de toestand van een paar tijdstippen geleden.

  De meetgegevens geven informatie over een tijdelijke toestand van de omgeving. Voorbeelden van meetgegevens zijn onder meer camerabeelden, range scanners enzovoort. Voor het grootste deel zullen we kleine timing effecten negeren. Bijvoorbeeld 2D-laserscanners scannen omgevingen achtereenvolgens bij zeer hoge snelheden, maar we zullen eenvoudig aannemen dat de meting overeenkomt met een bepaald tijdstip. Een sensormeting op tijdstip t wordt in het algemeen aangeduid met $\boldsymbol{z_t}$ [3].

- **Controleacties:** veranderen de toestand van de wereld. Robots doen dit door actief kracht uit te oefenen op de omgeving. Voorbeelden van controleacties omvatten robot beweging en de manipulatie van objecten. Zelfs als de robot geen acties uitvoert zal zijn staat toch nog veranderen. Dus voor de consistentie zullen we aannemen dat de robot altijd een controleactie uitvoert. In werkelijkheid zal de robot constant en tegelijkertijd controles en metingen uitvoeren. Een controleactie op tijdstip t wordt in het algemeen aangeduid met $\boldsymbol{u_t}$ [3].

2.2 AGV navigatie methodes

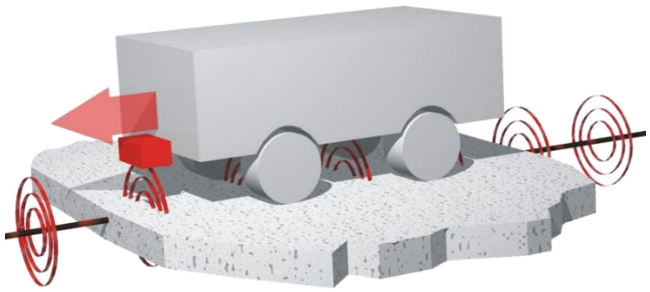
De navigatie van de AGV kan op verschillende manieren gebeuren, hieronder worden deze verschillende manieren beschreven.

2.2.1 Draadgeleide

Een sleuf wordt gesneden in de vloer en een draad wordt op ongeveer 2,5cm onder het oppervlak geplaatst. Deze draad zendt een radiosignaal uit, dat door een sensor aan de onderkant van de AGV wordt gedetecteerd. De sensorinformatie wordt gebruikt om het stuurcircuit van de AGV te regelen, waardoor de AGV de draad kan volgen [4] [5].

Voordelen:
- hoge betrouwbaarheid;
- hoge nauwkeurigheid;
- geen invloed van vuile en stoffige omgevingen;
- lage kosten.

Nadelen:
- draad in de grond kan andere werken beïnvloeden;
- heeft nog encoders nodig om zijn positie op de draad te vinden;
- methode is niet flexibel, het pad wijzigen is tijdrovend en kostelijk.



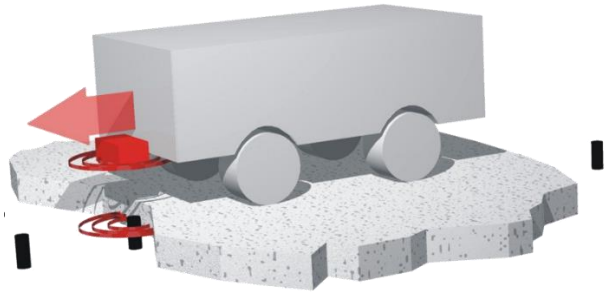
Figuur 4: Draadgeleide navigatie [6]

2.2.2 Inertie

Een andere vorm van AGV navigatie is inertiële navigatie. Hierbij worden er transponders geplaatst in de vloer van de werkplaats. Elke transponder heeft een unieke code zodat de AGV bij elke transponder zijn positie weet. Hierna weet de AGV waar de volgende transponder zich bevindt. Een gyroscoop zal de richting van het voertuig detecteren en met behulp van zijn odometrie kan de AGV zijn pad blijven volgen [4].

Voordelen:
- hoge betrouwbaarheid;
- geen invloed van vuile en stoffige omgevingen;
- lage kosten.

Nadeel:
- methode is niet flexibel, het pad wijzigen is tijdrovend en kostelijk.



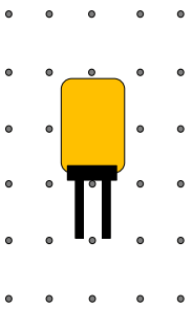
Figuur 5: Inertiële navigatie [6]

2.2.3 Grid

Deze methode is sterk gelijkend aan de inertie navigatie alleen worden hier transponders over heel de werkomgeving geplaatst. De X- en Y-coördinaten zijn van elke transponder gekend. De lokalisatie gebeurt o.b.v. de transponders in de vloer, de encoders en de gyroscoop voor het bepalen van de hoek [4].

Voordelen:
- deze methode kan wel eender welk pad nemen;
- hoge betrouwbaarheid;
- geen invloed van vuile en stoffige omgevingen.

Nadeel:
- Duurder dan inertie navigatie.



Figuur 6: Grid navigatie [7]

2.2.4 Optisch

Bij deze methode wordt gekleurde tape op de grond van de werkplaats aangebracht. De AGV zal via een optische sensor deze tape volgen. De tape wijst het pad aan dat de AGV moet volgen. De kleur van de tape moet uiteraard onderscheidbaar zijn van de ondergrond [4].

Voordeel:
- kan gemakkelijk worden verwijderd en verplaatst;
- lage kosten.

Nadeel:
- tape kan beschadigd of vuil geraken.

Figuur 7: Optische navigatie [8]

2.2.5 Magnetische tape

Deze methode is sterk gelijkend aan de optische navigatie. Hierbij wordt er magnetische in plaats van gekleurde tape op de grond van de werkplaats aangebracht. Hierdoor zullen er andere voor- en nadelen zijn [4].

Voordelen:
- de paden zijn vast maar makkelijk te wijzigen;
- tape mag vuil worden.

Nadeel:
- duurder dan optische navigatie.
- tape kan beschadigd worden.

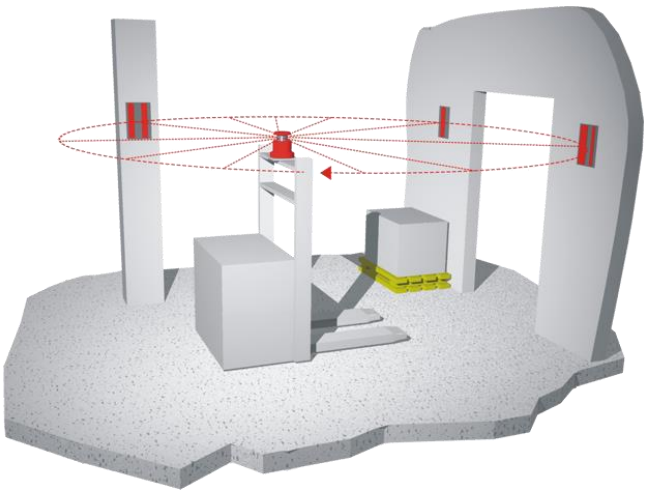
Figuur 8: Magnetische tape [9]

2.2.6 Via kunstmatige oriëntatiepunten

Dit is de methode die in een vorige masterproef van Bart Evens en Mitras Van Aken getiteld "Haalbaarheidsstudie naar het inzetten van AGV's in een flexibele werkvloeromgeving" gebruikt werd [10], en waar het huidige systeem met natuurlijke oriëntatiepunten mee vergeleken wordt. Voor deze methode met kunstmatige oriëntatiepunten wordt een 2D-laserscanner gemonteerd op de AGV en worden reflectoren geplaatst op de muren of andere obstakels. Ook bevat het geheugen van de AGV een kaart met daarop de locatie van al de reflectoren in de omgeving. De 2D-laserscanner bestaat uit een zender en ontvanger, die geplaatst zijn op een roterende eenheid. De scanner meet de hoek en afstand tot een zichtbare reflector en vergelijkt deze met zijn vooropgestelde kaart. Deze afstanden worden vervolgens gebruikt om via driehoeksmeetkunde zijn positie te bepalen. Deze positie wordt vergeleken met de route die de AGV zou moeten afleggen en indien nodig kan er nog worden bijgestuurd. Dit proces wordt op hoge snelheid herhaald, waardoor de AGV nauwkeurig de gewenste route kan volgen [4].

Voordelen:
- flexibeler dan al de voorgaande systemen;
- er moeten geen kabels of stroken worden aangelegd;
- nauwkeurigheid van ± 10mm.

Nadelen:
- oriëntatiepunten kunnen verstoord geraken;
- zelf aanpassingen maken is moeilijk, eindgebruiker beschikt meestal niet over het nodig materiaal en kennis;
- nieuwe reflectoren inmeten kost veel tijd;
- 2D-laserscanner is duur.



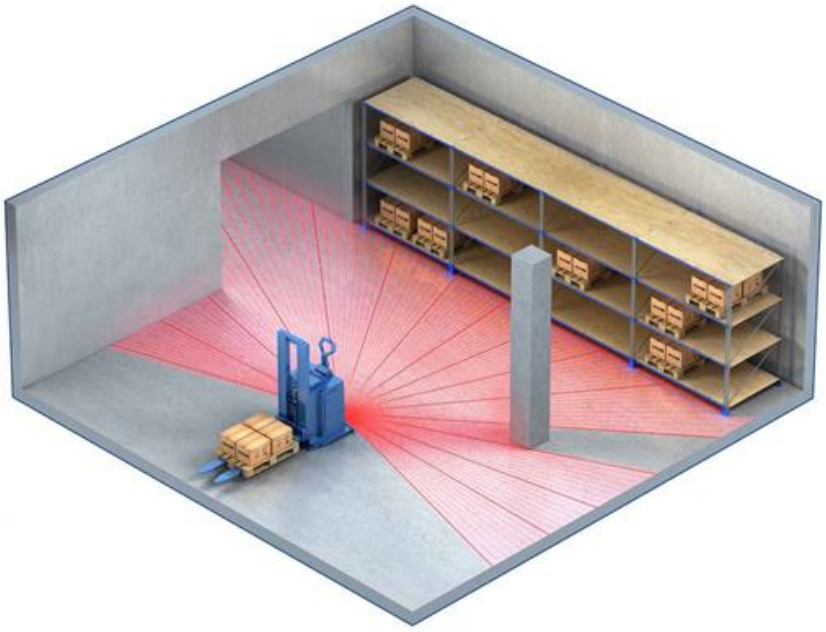
Figuur 9: Navigatie op kunstmatige oriëntatiepunten [6]

2.2.7 Via natuurlijke oriëntatiepunten

De AGV die momenteel gebruikt wordt in deze masterproef werd in het kader van deze masterproef uitgerust met deze methode. De AGV bevat een kaart van zijn omgeving, een 2D-laserscanner en odometrie data. De 2D-laserscanner scant zijn omgeving in zoals muren of pilaren en vergelijkt die met een vooropgestelde kaart via een lokalisatie algoritme, zodat de AGV zich in een omgeving kan lokaliseren. De odometrie wordt gebruikt om een ruwe schatting te geven van zijn positie. Het algoritme verfijnt deze positie dan a.d.h.v. de ingemeten natuurlijke oriëntatiepunten [4].

Voordelen:
- geen kosten aan de infrastructuur;
- zeer flexibel;
- snel geïnstalleerd.

Nadelen:
- 2D-laserscanner is duur;
- algoritmes vereisen veel rekenkracht.



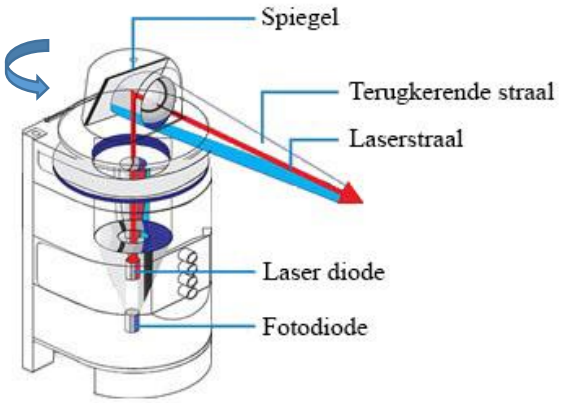
Figuur 10: Navigatie op natuurlijke oriëntatiepunten [11]

2.3 2D-laserscanners

Laserscanners produceren per meetpunt 2 tot 3 gegevens: een afstand, een hoekpositie en vaak ook de reflectie factor. De reflectie factor geeft aan hoe goed het gemeten object reflecteert. Deze reflectie word voornamelijk bepaald door de kleur en matheid van het obstakel. Factoren als stof of stoom hebben ook in vloed op de reflectie factor. Deze reflectie factor is nodig om reflectoren te detecteren als er gebruikt gemaakt wordt van kunstmatige oriëntatie punten. Metingen met een reflectie factor boven een bepaalde waarde geven aan dat er een reflector gedetecteerd is. Dit is ook de reden dat er zoveel mogelijk reflecterende oppervlakken moeten verwijdert worden op de hoogte waar de scanner meet. Ramen en roestvrijstalen buizen kunnen problemen geven. Wanneer er natuurlijke oriëntatiepunten gebruikt worden voor de positionering wordt de reflectiefactor niet in beschouwing genomen.

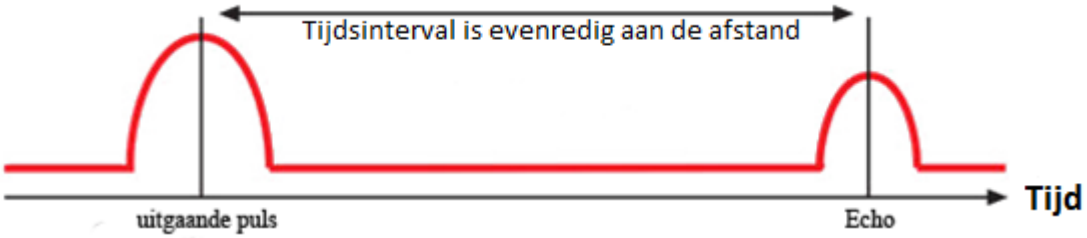
2.3.1 Time of flight principe

Zoals men in figuur 11 kan zien bestaat de 2D-laserscanner uit een laser diode en fotodiode. Een spiegel die geplaatst is op een roterende eenheid zorgt ervoor dat de laserstraal een hoek van 360° kan maken rond de scanner. Dit principe is van toepassing op laserscanner die werken op natuurlijke- en kunstmatige oriëntatiepunten [12].



Figuur 11: Werking SICK NAV350 [12]

De uitgezonden laserstraal zal reflecteren op een voorwerp zoals bijvoorbeeld een muur en vervolgens via de spiegel gemeten worden in de fotodiode. Zoals men in figuur 12 kan zien is de tijd tussen de uitgaande puls en de ontvangen echo afhankelijk van de afstand. Deze afstand kan dus berekend worden door het gemeten tijdinterval (vandaar "time-of-flight") te vermenigvuldigen met de snelheid van het licht en dit de delen door twee, omdat de laserstraal deze afstand tweemaal moet afleggen [12].



Figuur 12: Afstand berekening voor time of flight [12]

2.3.2 Belangrijkste parameters

Er is veel verschil tussen verschillende 2D-laserscanner modellen, afhankelijk van de kostprijs. De vier belangrijkste parameters voor een goede scanner zijn:

- **Hoekresolutie**: dit is de afstand tussen de laserstralen. Hoe kleiner men de afstand kan maken, hoe vollediger de gegevens kunnen zijn die we naar de robot sturen [12].
- **Metingen per seconde**: het aantal afstandsmetingen per omwenteling dat de scanner maximaal kan halen.
- **Nauwkeurigheid:** de nauwkeurigheid van de gemeten afstand tot het obstakel.
- **Werkbereik:** de maximale afstand die de scanner kan waarnemen.

Tabel 1: Overzicht 2D-laserscanners

	Merk	Prijs	metingen per sec.	Scan speed [Hz]	Maximale afstand [mm]	Hoek	Nauwkeurig-heid	Hoek resolutie
UBG-04LX_F01	Hokuyo	€ 1 995	30000	35	5600	240°	1% of Distance	0.36°
UBG-04LX	Hokuyo	€ 1 775	8500	10	4095	240°	1% of Distance	0.36°
URG-04LX_UG01	Hokuyo	€ 995	8500	10	5600	240°	3% of Distance	0.36°
UST-10LX	Hokuyo	€ 1 365	43240	40	10000	270°	±40mm	0.25°
UST_20LX	Hokuyo	€ 2 280	43240	40	20000	270°	±40mm	0.25°
UTM-30LX	Hokuyo	€ 3 985	28800	20	30000	270°	±50mm	0.25°
UTM-30LX_EW	Hokuyo	€ 4 500	43240	40	30000	270°	±50mm	0.25°
UXM-30LX_EW	Hokuyo	€ 4 295	28800	20	30000	190°	±50mm	0.25°
UXM-30LXH-EWA	Hokuyo	€ 4 500	30400	20	80000	190°	±50mm	0.125°
OMD10M-R2000-B23-V1V1D	Pepperl-Fuchs	€ 5 403	250000	50	10000	360°	± 9 mm	0.014°
OMD30M-R2000-B23-V1V1D-1L	Pepperl-Fuchs	€ 5 670	250000	50	30000	360°	± 10 mm	0.014°
OMD30M-R2000-B23-V1V1D-T-1L	Pepperl-Fuchs	€ 5 880	250000	50	30000	360°	± 10 mm	0.014°
VUX-1HA	RIEGL	€ 94 500	300000	250	420000	360°	± 5 mm	0.001°
NAV350-3232	SICK	€ 6 790	11500	8	35000	360°	± 15 mm	0.25°
LMS511-20190	SICK	€ 6 063	85000	75	65000	190°	± 35 mm	0,1667°
LMS111-10100	SICK	€ 3 311	54000	50	20000	270°	± 12 mm	0.25°
TIM551-2050001	SICK	€ 1 688	4000	15	10000	270°	± 40 mm	1°

2.3.3 Vergelijking

In industriële omgevingen kunnen de herkenningspunten, zoals muren, ver van de AGV liggen waardoor de scanner een voldoende groot werkbereik moet hebben. Ook zullen sommige herkenningspunten een lage reflectie hebben. Hierdoor zal de keuze van een 2D-laserscanner van cruciaal belang zijn voor een correcte kaartopbouw en lokalisatie. Er is een vergelijking gemaakt van de meest courante scanners die momenteel op de markt verkrijgbaar zijn:

Oorspronkelijk werd de AGV gebruikt in combinatie met de NAV350 van Sick, deze scanner is dan ook beschikbaar om verder testen op uit te voeren. De scanner van Pepperl+Fuchs is gedurende het 2de semester van de masterproef in ons bezit, zodat ook hierop getest kan worden.

Tabel 2: Gekozen scanners

Merk	Type	Prijs	Werkbereik
SICK	NAV350	€ 6 790	35m
Pepperl+Fuchs	OMD30M-R2000	€ 5 670	30m

2.4 Lokalisatie

2.4.1 Robotlokalisatie in het algemeen

In robotlokalisatie willen we de positie van de robot kennen op huidig tijdstip *k*, en willen dit doen met de initiële positie en alle metingen $Z^t=\{z_t,i=1\ldots k\}$ tot de huidige positie. Er wordt hier met een driedimensionale toestandsvector gewerkt $\mathbf{x}=[x,y,\theta]^T$ welke de positie en oriëntatie van de robot bevat (ook wel "pose" genoemd). Dit schattingsprobleem word opgelost met de Bayesiaanse filter waar we naar de laatste berekening van de positie $p(\mathbf{x}_t|Z^t)$ gaan kijken. Deze berekening geeft de warschijnlijkheidsverdeling p(.) over robotposes. Deze waarschijnlijkheidverdeling kan op verschilende manieren worden gemodelleerd, bijv. via normale verdelingen (Kalman filter) of met deeltjes of partikels (particle filter). In deze masterproef maken we gebruik van de particle filter: elk deeltje stelt een min of meer waarschijnlijke robotpose voor. Wanneer de particles zich op een kleine oppervlakte bevinden, is er een hoge dichtheid, m.a.w. dan is het schattingsalgoritme redelijk zeker dat de robotpose zich op die plaats bevindt. Het doel is om een zo groot mogelijke dichtheid te bekomen, waardoor de AGV een hogere zekerheid heeft van zijn lokalisatie. In de Bayesiaanse aanpak gebruiken we alle kennis om zo een schatting te maken van de huidige positie. Samengevat moeten we, om de robot te lokaliseren, bij elke stap de dichtheid $p(\mathbf{x}_t|Z^t)$ van de positie van de robot berekenen [13].

Dit gebeurt in twee stappen, die elkaar voortduren afwisselen::

1) Voorspellingsfase

In de voorspellings fase wordt er gekeken naar het bewegingsmodel om de huidige positie van de robot te bepalen. We veronderstellen dat de huidige positie $\mathbf{x}_t$ enkel afhankelijk is van de vorige positie $\mathbf{x}_{t-1}$ (Markov) en de gekende input van de encoders $\mathbf{u}_{t-1}$. Via het bewegingsmodel $p(\mathbf{x}_t|\mathbf{x}_{t-1}, \mathbf{u}_{t-1})$ en de vorige waarschijnlijkheidsverdeling over robotposes $p(\boldsymbol{x}_{t-1}|Z^{t-1})$ kan dan de voorspelde dichtheid van positie $\mathbf{x}_t$ bepaald worden via integratie [14]:

$$p(\boldsymbol{x}_t|Z^{t-1}) = \int p(\boldsymbol{x}_t|\boldsymbol{x}_{t-1}, \boldsymbol{u}_{t-1})\, p(\boldsymbol{x}_{t-1}|Z^{t-1})\, d\boldsymbol{x}_{t-1}$$

2) Updatefase

In de update- of correctiefase wordt gebruik gemaakt van het meetmodel om de data van de scanner samen te voegen met de geschatte pose uit de voorspellingsfase. We veronderstellen dat de metingen $\mathbf{z}_t$ onafhankelijk zijn van andere en eerdere metingen Z^{t-1}. Het meetmodel is gegeven in de vorm van een waarschijnlijkheid $p(\mathbf{z}_t|\mathbf{x}_t)$. Deze term geeft de waarschijnlijkheid dat de robot zich op positie $\mathbf{x}_t$ bevindt wanneer hij de meetgegevens $\mathbf{z}_t$ gebruikt. De waarschijnlijkheid van $\mathbf{X}_t$ kan dan bepaald worden via de regel van Bayes [14]:

$$p(\boldsymbol{x}_t|Z^t) = \frac{p(\boldsymbol{z}_t|\boldsymbol{x}_t)\, p(\boldsymbol{x}_t|Z^{t-1})}{p(\boldsymbol{z}_t|Z^{t-1})}$$

2.4.2 Monte Carlo Lokalisatie (MCL)

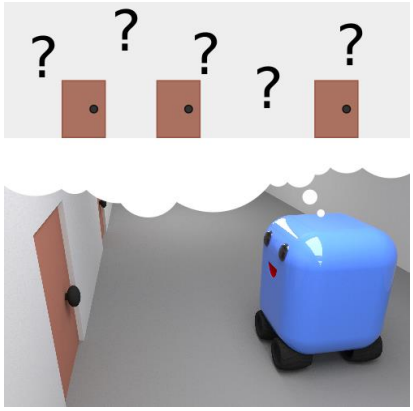
Het MCL algoritme is uitgegroeid tot één van de meest populaire lokalisatie algoritmen in de robotica. Wanneer een kaart van de omgeving wordt gegeven, kan het algoritme de positie en oriëntatie van de robot bepalen terwijl deze beweegt en de omgeving via zijn sensoren waarneemt. Het algoritme maakt gebruik van een particlefilter, elke particle stelt een mogelijke toestand van de robot voor. Een particle heeft hierin de volgende waardes:

$$\ll x, y, \theta >, \mathrm{p} >$$

Waarin <x, y, θ> de positie weergeeft en $p \geq 0$ de kans aanduidt dat de robot zich op deze positie bevindt.

Het algoritme begint typisch met een uniforme willekeurige verdeling van particles over zijn volledig geconfigureerde ruimte. Dit betekent dat de robot geen informatie heeft over waar hij is, en dat de waarschijnlijkheid over zijn positie in elk punt in de ruimte even groot is. Wanneer de robot beweegt, verschuiven de particles, zodat ze na een beweging een nieuwe toestand voorspellen. Wanneer de robot iets detecteert, worden de particles geresampled door middel van voorspellingen van de Bayes filter. Deze filter maakt gebruik van de overeenstemming tussen de werkelijk waargenomen data en de voorspelde toestand. Uiteindelijk zullen de particles convergeren naar de actuele positie van de robot [14].

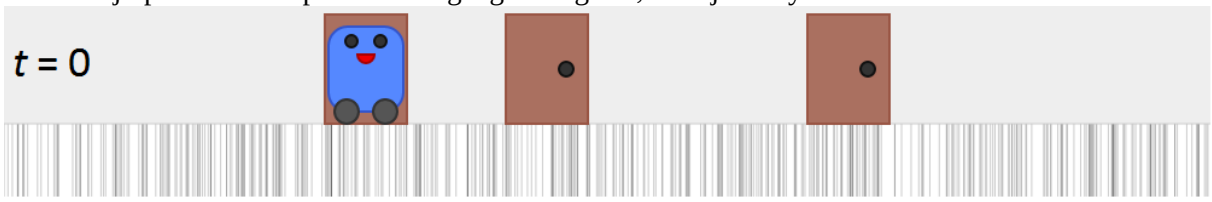
2.4.1.1 Voorbeeld Monte Carlo Lokalisatie



Figuur 13: Robot in een 1-dimensionale omgeving met onbekende positie [15]

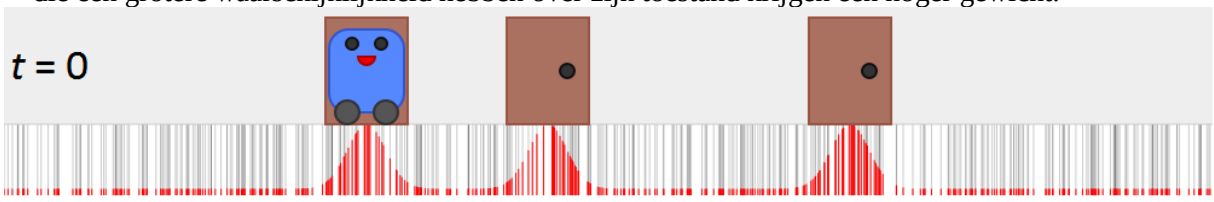
Veronderstel een robot in een eendimensionale gang met drie identieke duren, gewapend met een sensor die alleen kan zien of er een deur is of niet.

1) Het algoritme initialiseert met een uniforme verdeling van de particles. De robot beschouwt de kans over zijn positie in elk punt van de gang even groot, terwijl het fysiek aan de eerste deur staat.



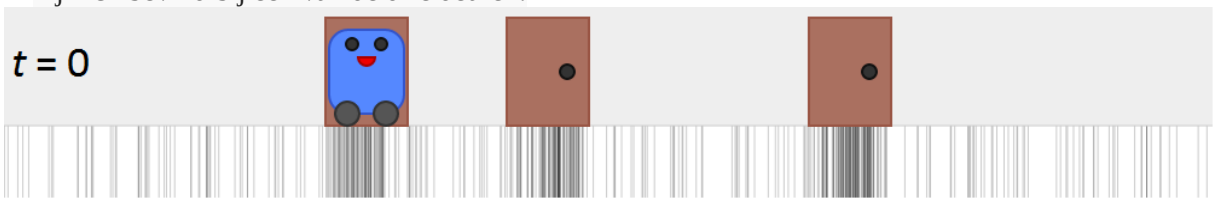
Figuur 14: t=0, beginpositie [15]

2) **Sensor update**: De robot detecteert een deur en het wijst een gewicht toe aan elke particle. De particles die een grotere waarschijnlijkheid hebben over zijn toestand krijgen een hoger gewicht.



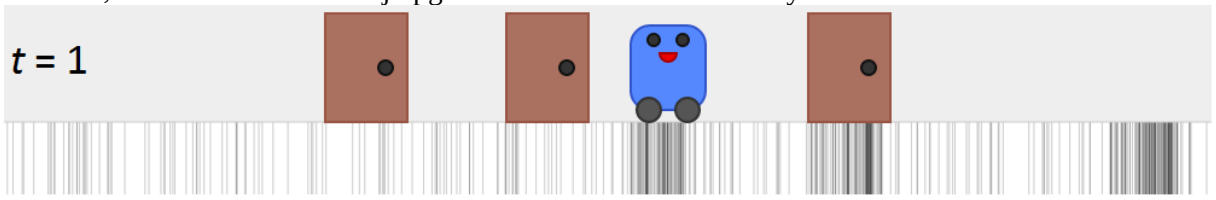
Figuur 15: t=0, sensor update [15]

3) **Resampling:** De robot genereert een reeks van nieuwe particles, waarvan de meeste gegenereerd worden rond de vorige particles die een hoger gewicht kregen toegewezen. Hij is nu van mening dat hij zich bevind bij één van de drie deuren.



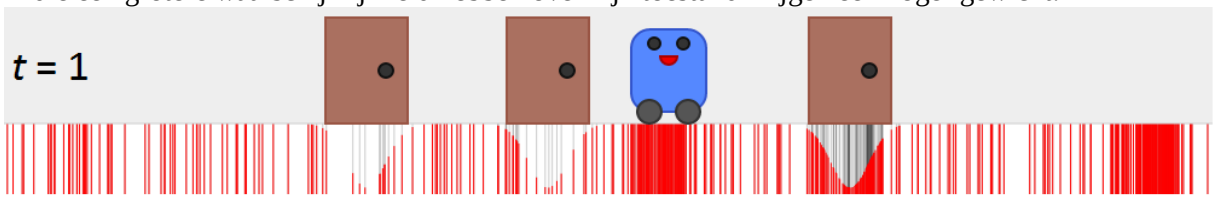
Figuur 16: t=0, Resampling [15]

4) **Motion update:** De robot beweegt een bepaalde afstand naar rechts. Alle particles bewegen mee naar rechts, en er wordt wat ruis bij opgeteld. De robot bevindt zich fysiek tussen de tweede en derde deur.



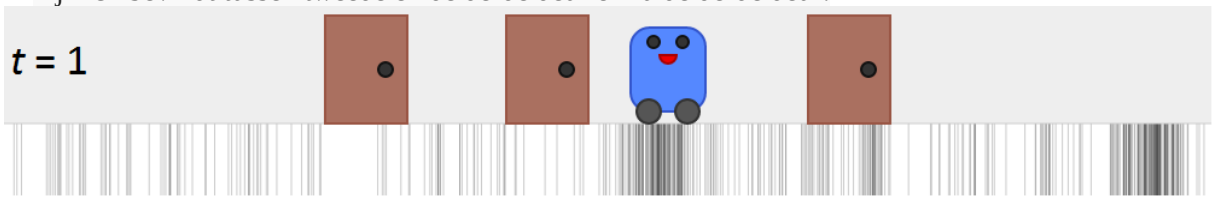
Figuur 17: t=1, motion update [15]

5) **Sensor update**: De robot detecteert geen deur en het wijst een gewicht toe aan elke particle. De particles die een grotere waarschijnlijkheid hebben over zijn toestand krijgen een hoger gewicht.



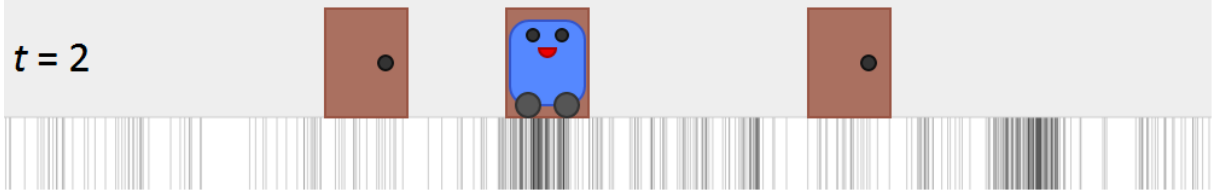
Figuur 18: t=1, sensor update [15]

6) **Resampling:** De robot genereert een reeks van nieuwe particles, waarvan de meeste gegenereerd worden rond de vorige particles die een hoger gewicht kregen toegewezen. Hij is nu van mening dat hij zich bevindt tussen tweede en de derde deur of na de derde deur.



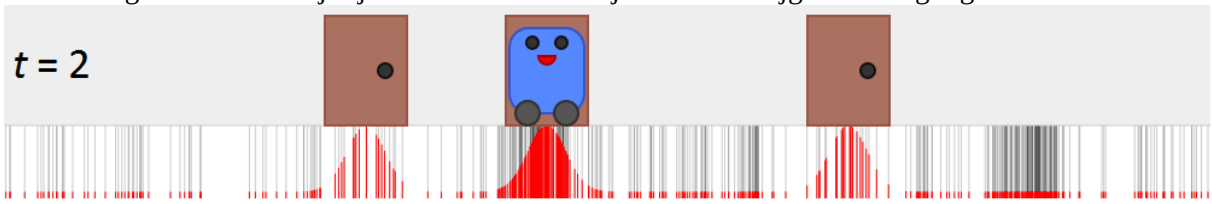
Figuur 19: t=1, resampling [15]

7) **Motion update:** De robot beweegt een bepaalde afstand naar links. Alle particles bewegen mee naar links, en er wordt wat ruis aan opgeteld. De robot bevindt zich fysiek aan de tweede deur.



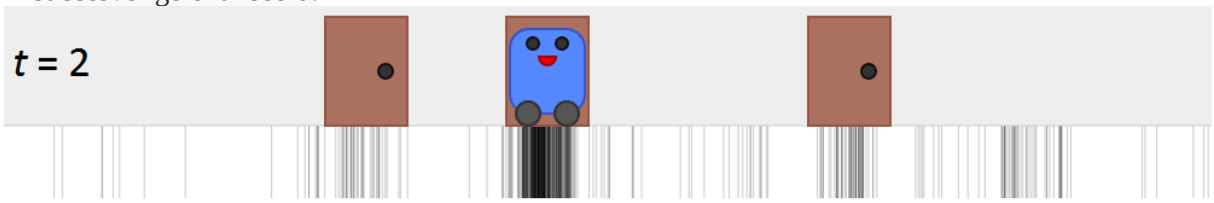
Figuur 20: t=2 motion update [15]

8) **Sensor update**: De robot detecteert een deur en het wijst een gewicht toe aan elke particle. De particles die een grotere waarschijnlijkheid hebben over zijn toestand krijgen een hoger gewicht.



Figuur 21: t=2, sensor update [15]

9) **Resampling:** De robot genereert een reeks van nieuwe particles, waarvan de meeste gegenereerd worden rond de vorige particles die een hoger gewicht kregen toegewezen. De robot heeft zich nu succesvol gelokaliseerd.



Figuur 22: t=2, resampling [15]

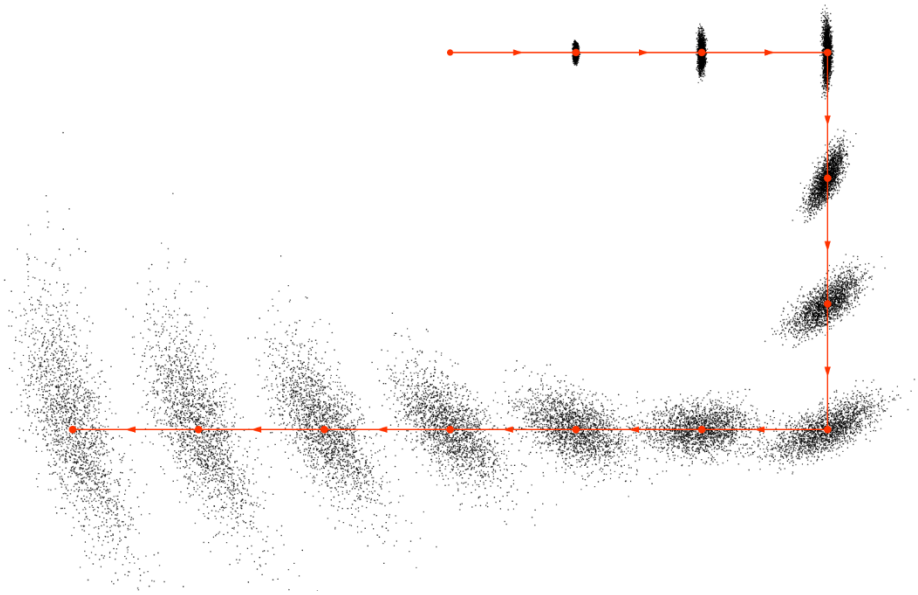
2.4.1.2 Algoritme

1: **Algorithm MCL**($\mathcal{X}_{t-1}, u_t, z_t, m$)**:**
2: $\bar{\mathcal{X}}_t = \mathcal{X}_t = \emptyset$
3: for $m = 1$ to M do
4: $x_t^{[m]} =$ **Motion_update** $(u_t, x_{t-1}^{[m]})$
5: $w_t^{[m]} =$ **Sensor update** $(z_t, x_t^{[m]}, m)$
6: $\bar{\mathcal{X}}_t = \bar{\mathcal{X}}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
7: endfor
8: for $m = 1$ to M do
9: draw i with probability $\propto w_t^{[i]}$
10: add $x_t^{[i]}$ to $\mathcal{X}_t$
11: endfor
12: return $\mathcal{X}_t$

Figuur 23: Monte Carlo Lokalisatie algoritme [15]

Het algoritme neemt als input de vorige geschatte positie $X_{t-1} = \{x_t{}^{[1]}, x_t{}^{[2]}, \dots, x_t{}^{[M]}\}$, een controle actie u_t, een sensormeting z_t en een kaart van de omgeving m. En het algoritme geeft het nieuwe geschatte positie X_t terug.

Tijdens de "**Motion_update**" in lijn 4 voorspelt de robot zijn nieuwe locatie op basis van zijn gegeven controle actie, door het toepassen van een gesimuleerde beweging aan elke particle. Als bijvoorbeeld een robot naar voren beweegt, zullen alle particles ook naar voren bewegen in hun eigen richting, ongeacht hoe ze gedraaid zijn. Als een robot 90° met de klok mee roteert, zullen ook alle deeltjes ook 90° met de klok meedraaien, ongeacht waar ze zijn. Maar in de praktijk is geen enkele aandrijving perfect: er kan overshoot of undershoot zijn op de gewenste hoeveelheid beweging. Wanneer een robot probeert te rijden in een rechte lijn, zal het onvermijdelijk afwijken naar de ene of de andere richting door minieme verschillen in wiel omtrek, slip, spelling en andere invloeden. Vandaar dat het motion model moet ontworpen zijn zodat er ruis wordt bij opgeteld. Na een lange controle actie zullen de particles zich terug meer verspreiden, door deze ruis. Dit is logisch omdat de robot minder zeker wordt van zijn positie als het blindelings beweegt zonder zijn sensormetingen te gebruiken, dit wilt zeggen dat hij enkel op basis van zijn encoder gegevens (odometrie) rijdt [13].



Figuur 24: Dichtheid van de geschatte positie na enkele stappen [15]

Tijdens de "**Sensor update**" in lijn 5 zal de robot zijn omgeving waarnemen, de particles worden geüpdatet om een nauwkeurigere positionering te doen. Voor elke particle berekent de robot de waarschijnlijkheid dat deze de toestand van het particle had. Het geeft een gewicht aan $w_t{}^{[i]}$ voor elke particle, evenredig met de genoemde waarschijnlijkheid.

Vervolgens zal het in lijn 8 willekeurig nieuwe particles nemen uit zijn vorige geschatte posities, met een waarschijnlijkheid evenredig aan $w_t{}^{[i]}$. Particles die overeenkomen met de sensormeting maken meer kans om te worden gekozen en particles die in strijd zijn met de sensormeting worden zelden gekozen. Bijgevolg zullen particles convergeren naar een betere schatting van de werkelijke toestand (pose) van de robot. Dit is logisch omdat een robot steeds zekerder wordt van zijn positie als het de omgeving waarneemt [13].

2.4.2 Adaptive Monte Carlo Localization (AMCL)

In al de MCL algoritmes, zijn het aantal particles vast en deze moeten meestal voldoende groot zijn. Echter wanneer het algoritme gewoon de positie van de robot wilt volgen, kan het aantal particles veel lager zijn. Het aanpassen van het aantal particles kan ervoor zorgen dat het algoritme efficiënter werkt. Eén aanpak die het aantal particles kan aanpassen is de Kullback-Leibler divergence (KLD) methode. Deze methode wordt ook gebruikt in het AMCL algoritme dat in de ROS bibliotheek staat.

Het belangrijkste idee van de KLD methode kan omschreven worden als volgt:

"Bij elke iteratie van de particle filter, wordt het aantal samples bepaald, zodat met waarschijnlijkheid $1 - \delta$*, de fout tussen de echte toestand en de sample-gebaseerde kleiner is dan een vaste waarde* ε*."* [16]

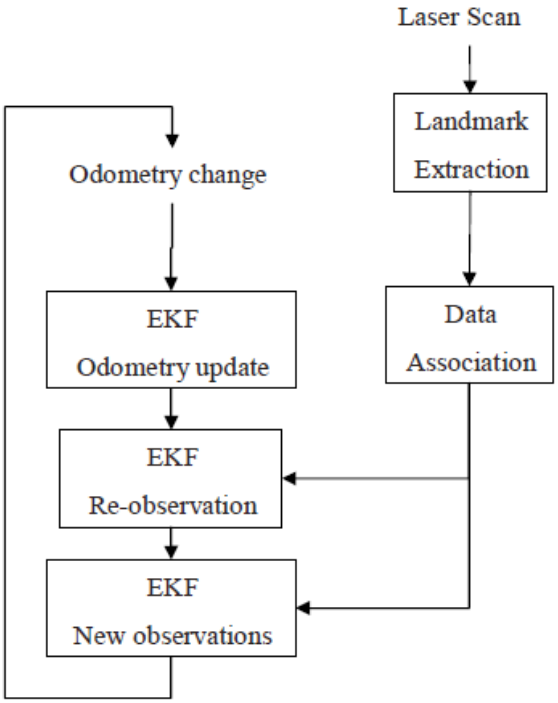
2.5 Kaartopbouw

2.5.1 SLAM

Wanneer de robot geen toegang heeft tot een kaart van zijn omgeving, noch toegang heeft tot zijn eigen werkelijke poses kan SLAM een oplossing bieden. Via dit principe kan een mobiele robot geplaatst in een onbekende omgeving op een onbekende locatie stapsgewijs een kaart opbouwen terwijl hij zich lokaliseert in deze kaart [17] [18].

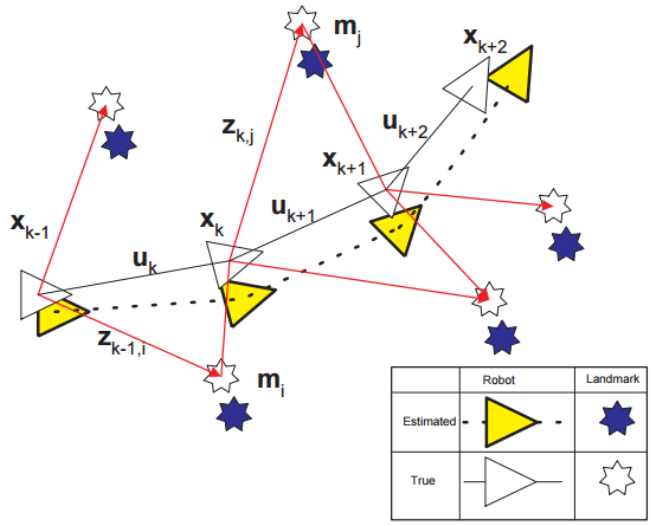
SLAM bestaat uit een aantal stappen. Het doel van het proces is om de omgeving te gebruiken om zo telkens opnieuw de positie van de robot en van de omgeving zelf (de oriëntatiepunten) te schatten. Om een correcte schatting te bekomen kunnen we niet alleen op de odometrie afgaan. Laserscanners zijn noodzakelijk om de omgeving waar te nemen en kunnen gebruikt worden om de positie van de robot in de omgeving te corrigeren. Dit wordt verwezenlijkt door kenmerken uit de omgeving te extraheren en herhaaldelijk te observeren wanneer de robot beweegt. Een EKF (Extended Kalman Filter) is het hart van een SLAM proces. Het is verantwoordelijk voor het updaten van de positie waar de robot denkt te zijn, op basis van oriëntatiepunten. De EKF houdt een schatting van onzekerheid in de robots positie en ook de onzekerheid in deze oriëntatiepunten bij.

Wanneer de robot beweegt zal de poseschatting en de onzekerheid erop in de EKF bijgewerkt worden met "*Odometry update*". Oriëntatiepunten worden vervolgens geëxtraheerd uit de omgeving van de robot zijn nieuwe positie. De robot zal vervolgens proberen om deze oriëntatiepunten te koppelen aan eerdere observaties. Opnieuw waargenomen oriëntatiepunten worden dan gebruikt om de positie van de robot te updaten. Oriëntatiepunten die hij nog niet eerder heeft gezien worden toegevoegd aan het EKF als *"New observations"* zodat ze later opnieuw waargenomen kunnen worden [19].



Figuur 25: Overzicht van het SLAM proces [19]

Dit is schematisch voorgesteld in het onderstaande diagram:



Figuur 26: Schatting van de robot positie en natuurlijke oriëntatie punten [17]

De robot is voorgesteld als een driehoek, de sterren stellen oriëntatiepunten voor. Initieel meet de robot met behulp van een laserscanner de locatie van de oriëntatiepunten (Lasermetingen geïllustreerd met een rode lijn). Door deze meting kan de robot zijn eigen positie schatten, deze positie komt niet volledig overeen met zijn werkelijk positie door fouten op zijn lasermeting. Vervolgens beweegt de robot een bepaalde afstand, via EKF zal hij zijn odometrie- en scannerdata samen nemen om vervolgens een nieuwe geschatte positie te krijgen. Ook de odometrie is niet volledig correct waardoor er altijd naar zijn positie wordt geschat [19].

2.5.2 GMapping

GMapping gebruikt een “Rao-Blackwellized particle filter” om gelijktijdig te lokaliseren en een map op te bouwen. Deze techniek past een particle filter toe waarbij elk particle een individuele kaart van de omgeving bevat. De particle waarvan de map het best overeenstemt met de laserscan zal de positionering bepalen. Het is dus belangrijk om het aantal particles te verminderen. GMapping maakt gebruik van adaptieve technieken om het aantal particles te minimaliseren. Het gebruikt ook de odometrie samen met zijn scannerdata, dit zal de onzekerheid over de toestand van de robot drastisch verlagen [20] [21].

3 Software

3.1 ROS Robot Operating System

Om de kaartopbouw- en lokalisatie- algoritmes te implementeren is het Robot Operating System (ROS) gebruikt. ROS is een gratis en opensource systeem voor robots. ROS wordt voornamelijk gebruikt door universiteiten, onderzoeksinstellingen en hobbyisten maar ook verschillende bedrijven maken hiervan gebruik. Voor bedrijven die producten op de markt brengen is ROS interessant omdat het makkelijk is om prototypes te bouwen waardoor de kans groter wordt dat hun product in het uiteindelijk resultaat terecht komt. Het is een Linux gebaseerd softwaresysteem, voornamelijk Ubuntu, dat bestaat uit:

- hardware abstractie
- drivers
- uitvoeren van veel gebruikte functies
- berichten doorsturen tussen verschillende processen
- package manager
- ….

De bibliotheek op de ROS webpagina bevat drivers voor hardware, algoritmes om met de data afkomstig van de hardware om te gaan en informatie om over de beschikbare pakketten. Verder zijn op de webpagina verschillende tutorials te volgen om bekend te raken met ROS en zijn uitgebreide mogelijkheden. Verder kan men ook zelf code toe voegen voor een specifieke applicatie of om zelf een driver voor een apparaat te schrijven en deze dan te uploaden zodat anderen deze kunnen gebruiken. ROS kan gebruik maken van volgende programmeertalen: C++, Python, Octave en Lisp [22].

3.2 ROS concepten

Om ROS te verduidelijken kan best eerst gekeken worden naar de opbouw van ROS. De opbouw bestaat uit 3 niveaus: het ROS filesystem niveau, het ROS computation graph niveau en het ROS community niveau [23].

3.3 ROS filesystem level

Dit gaat voornamelijk over de bestanden die op je harde schijf staan en je zelf kan toevoegen en aanpassen. “packages” worden in je “Catkin workspace” geplaatst. Wanneer we de code compileren wordt heel deze map gecompileerd [23].

3.3.1 Packages:

Packages worden in ROS gebruikt om de software georganiseerd te houden. Een package bevat de Code (nodes) om de hardware aan te sturen/uit te lezen. Eventuele datasets, configuratie bestanden of andere bestanden die gebruikt worden door de code staan ook in de package. Bestanden die niets met effectieve code te maken hebben kunnen hier ook aan toegevoegd worden. Een package kan verschillende nodes bevatten. Een package kan bijvoorbeeld een driver zijn voor een camera van een robot [23].

3.3.2 Metapackages:

Metapackages zijn speciale packages waarin zich meerdere packages bevinden, bijvoorbeeld om in grote programma's bepaalde delen van een robot samen te voegen; dit wordt gedaan om structuur in het programma te houden [23].

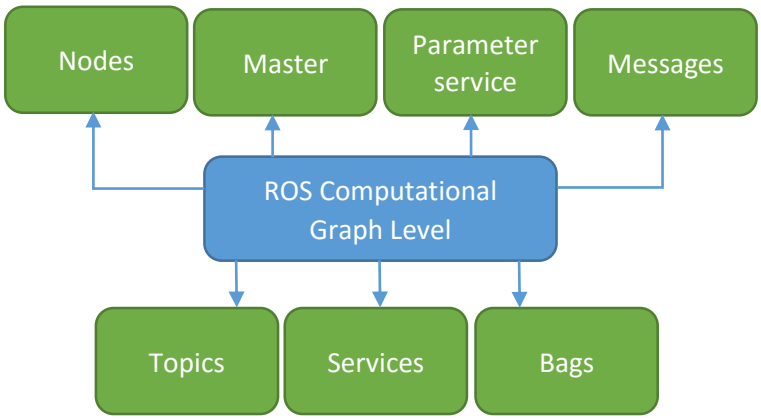
3.3.3 Package Manifest:

Dit bestand bevat bepaalde eigenschappen zoals: naam van de package, versie, schrijvers, onderhouders en afhankelijkheden van ander packages. De package manifest is een XML bestand namelijk package.xml dat elke package moet bevatten [23].

3.4 ROS Computation Graph Level

De berekeningen binnen ROS worden gedaan door een netwerk van ROS nodes. Dit netwerk, Computation Graph, maakt voornamelijk gebruik van volgende onderdelen**: nodes, Master, parameter service, messages, topics, services** en **bags**. Elk onderdeel heeft zijn eigen bijdrage.

Het ROS_comm pakket zorgt voor de communicatie tussen de verschillende onderdelen. Ook voorziet het pakket tools voor het eventueel debuggen van de code zoals: rostopic, rosparam, rosservice en rosnode [23].



Figuur 27: Overzicht

3.4.1 Nodes

Nodes bevatten de werkelijke code van het programma; hierin worden alle berekeningen gedaan. Hierin worden de berichten (messages) ook verstuurd naar ROS om zo bij een ander ROS onderdeel terecht te komen bijvoorbeeld: een andere node of Rviz (de visualisatieomgeving van ROS). De bedoeling van deze nodes is het opsplitsen van de code, elk onderdeel van een robot zal zijn eigen node hebben. Zo is het makkelijk onderdelen te vervangen, aan te passen of te debuggen zonder heel de code te moeten begrijpen. Deze nodes kunnen in verschillende programmeertalen geschreven worden zoals: C++, Python, Octave en Lisp [23].

3.4.2 Master

De rol van de ROS master is ervoor te zorgen dat verschillende nodes met elkaar kunnen communiceren. De ROS master registreert en benoemt de verschillende nodes in het ROS systeem en volgt alle berichten tussen de nodes. In een systeem met meerdere computers mag er maar één ROS master zijn de andere modules communiceren via deze master [23].

3.4.3 Parameter service

De parameter service zorgt ervoor dat alle data op één plaats staan. Nodes kunnen deze data uitlezen en aanpassen. De parameter service is een onderdeel van de ROS master [23].

3.4.4 Messages

Nodes communiceren via berichten (messages). Deze berichten bevatten constanten en variabelen. De variabelen bevatten de echte data en de constanten bevatten bijvoorbeeld de naam van variabelen. Er zijn de standaard types (integer, floating point, boolean enz.) maar binnen ROS zijn er ook complexere types zoals de geometry_msg/pose.msg die een 3-dimensionale positie en rotatie bevat. Ook is het mogelijk zelf een formaat te creëren [23].

3.4.5 Topics

Wanneer er een bericht van de ene node naar de andere moet verstuurd worden moet dit gebeuren via een topic. De node die het bericht verzendt zal het bericht versturen (publishen) naar een topic, de node die het bericht wil ontvangen zal deze topic uitlezen (subscriben). Dit maakt het mogelijk om binnen ROS real-time deze topics te monitoren wat debuggen vergemakkelijkt. Meerdere nodes kunnen naar hetzelfde topic publishen en meerdere nodes kunnen subscriben op hetzelfde topic [23].

3.4.6 Services

In sommige applicaties kunnen de topics niet gebruikt worden. Bij topics worden berichten gepublished ongeacht wat de subscriber vraagt. Soms is er een request/respons systeem nodig. Hier kan ROS services gebruikt worden. Hier wordt een server node en een cliënt node gebruikt. Wanneer de cliënt node een bericht stuurt naar de server node zal deze een antwoord sturen met het gevraagde bericht [23].

3.4.7 Bags

Hierin kunnen topics in functie van de tijd opgenomen worden en kan deze bag file achteraf opnieuw afgespeeld worden. Dit maakt het mogelijk om bijvoorbeeld sensormetingen achteraf met verschillende algoritmes te testen [23].

3.5 Ros community level

ROS is mogelijk gemaakt door wereldwijde samenwerking tussen programmeurs, onderzoeksinstellingen, bedrijven en hobbyisten. Op de ROS wiki webpagina staat veel documentatie, tutorials en info over packages (die voornamelijk te vinden zijn GitHub). Als gebruiker heb je de mogelijkheid om er zelf toe te voegen zodat deze door anderen gebruikt kunnen worden. Op ROS Answers (answers.ros.org) kan je vragen stellen over ROS en de problemen die je tegen komt [23].

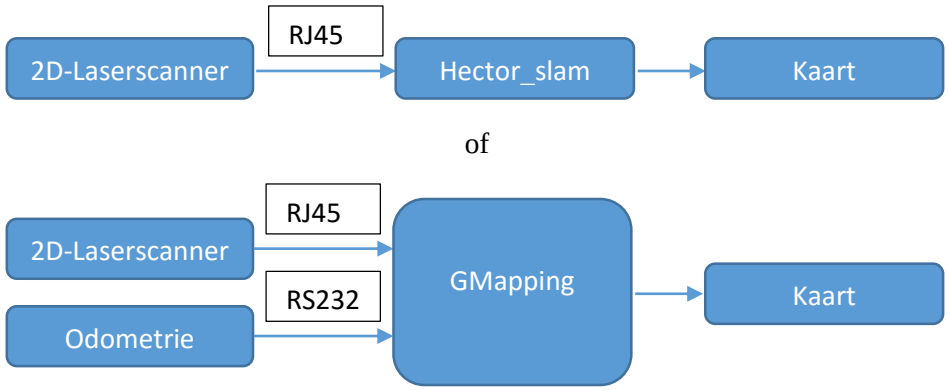
3.6 Problemen ROS

Om met een nieuw programma bekend te raken is het best om kleine stappen te nemen. Dit wordt in ROS mogelijk gemaakt door de vele tutorials. Deze leggen stap voor stap uit wat je moet doen en waarom. Zonder de tutorials is het bijna onmogelijk om met ROS te leren werken o.b.v. zelfstudie. Vooral de stap van de tutorials naar de echte hardware kan problematisch zijn. In de tutorials wordt meestal gebruik gemaakt van virtuele machines die specifiek gemaakt zijn voor de tutorials. Hierdoor zijn er veel parameters die niet ingesteld moeten worden tijdens de tutorials. De stap van de tutorials naar het echte programeer werk is dus groot. De zeer uitgebreide bibliotheek, mogelijkheden en commando's kunnen voor nieuwe gebruikers overdonderend zijn. Door de specifieke hardware en bijhorende instellingen ontstaan er vaak kleine problemen waar soms lang achter een oplossing gezocht moet worden maar die dan uiteindelijk makkelijk op te lossen blijkt. Wanneer we ROS vergelijken met bijvoorbeeld programmeren in C++ zien we dat in ROS vaak maar één juiste oplossing is, in C++ kunnen we vaak via verschillende wegen hetzelfde resultaat bereiken wat het makkelijker maakt je eigen gedachtegang toe te passen.

3.7 Programma opbouw

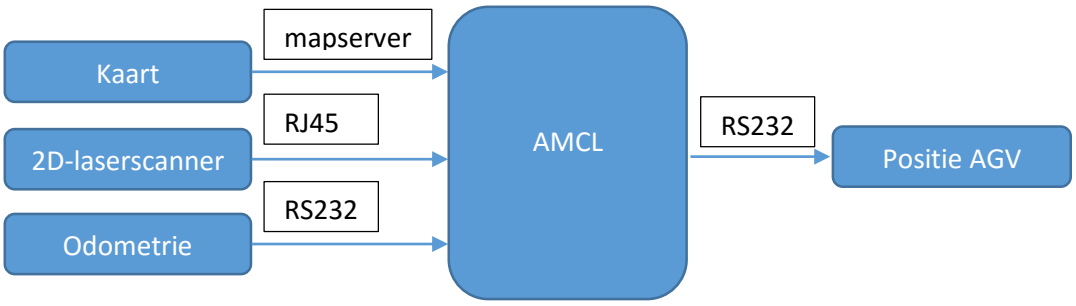
Om lokalisatie m.b.v. natuurlijke oriëntatiepunten mogelijk te maken zal de software bestaan uit twee delen. Één deel zal zorgen voor de mapping via SLAM, hiervoor is HectorSLAM en GMapping onderzocht. GMapping heeft als voordeel dat het ook gebruik maakt van odometrie data en dus aan de hand hiervan al kan voorspellen waar de AGV zich ongeveer bevindt en dan met de sensor data deze positie corrigeert. Het tweede deel zal, nadat de map van de omgeving is opgebouwd, de positie in de map bepalen. Dit zal door AMCL (Adaptive Monte Carlo Localization) gebeuren, dit algoritme maakt ook gebruik van de odometry data. Zowel HectorSLAM, GMapping en AMCL zijn opensource pakketten die binnen ROS werken en te vinden zijn op GitHub.

Kaartopbouw:



Figuur 28: Programma opbouw, kaartopbouw

Lokalisatie:



Figuur 29: Programmaopbouw, lokalisatie

3.7.1 Pepperl_fuchs_r2000

Deze package bevat de driver voor de R2000 laser range finder van Pepperl+Fuchs. De data worden ontvangen via het UDP protocol langs een ethernetpoort. Elk bericht bevat een header en een set afstandsmetingen. De header bevat gegevens zoals: scanfrequentie, samples per scan, starthoek van de metingen en eindhoek. Deze gegevens worden in de header gezet in plaats van met iedere meting mee te sturen om de berichten zo klein mogelijk te houden. De algoritmes kunnen aan de hand van deze gegevens bij elke afstandsmeting een hoekpositie berekenen. Eenmaal een bericht binnen is wordt het omgevormd naar een standaardformaat dat binnen ROS door de algoritmes gebruikt kan worden namelijk het sensor_msgs/LaserScan.msg formaat. Via de driver node kunnen ook parameters van de scanner aangepast worden. De in te stellen parameters zijn:

- frame_id: topic waar de berichten worden gepublished
- scanner_ip: ip adress waar de driver verbinding mee moet maken
- scan_frequency: draaifrequentie van de scanner
- samples_per_scan: aantal metingen per rotatie

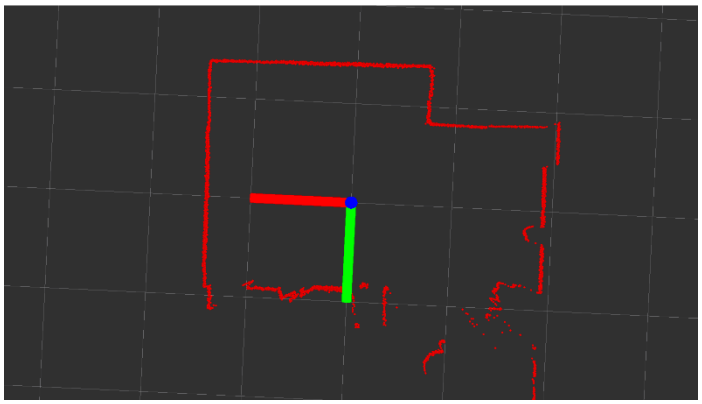
```
Header header               # timestamp in the header is the acquisition time
                            # of the first ray in the scan.
                            #
                            # in frame frame_id, angles are measured around
                            # the positive Z axis (counterclockwise, if Z is
                            # up)
                            # with zero angle being forward along the x axis

float32 angle_min           # start angle of the scan [rad]
float32 angle_max           # end angle of the scan [rad]
float32 angle_increment     # angular distance between measurements [rad]

float32 time_increment      # time between measurements [seconds] - if your
                            # scanner
                            # is moving, this will be used in interpolating posi
tion
                            # of 3d points
float32 scan_time           # time between scans [seconds]

float32 range_min           # minimum range value [m]
float32 range_max           # maximum range value [m]

float32[] ranges            # range data [m] (Note: values < range_min or >
                            # range_max should be discarded)
float32[] intensities       # intensity data [device-specific units].  If your
                            # device does not provide intensities, please
                            # leave the array empty.
```



Figuur 30: Puntenwolk opgemeten door R2000 laser range finder

3.7.2 RS232_tf_broadcaster

Er zijn verschillende RS-232 pakketten te vinden op de ROS wiki pagina maar geen ervan voldeed aan de eisen die nodig zijn om de odometrie gegevens van de AGV te ontvangen. Dit pakket is aan de hand van voorbeeldprogramma's geschreven om te voldoen aan de eisen. Het bericht dat ontvangen wordt bevat de snelheden in het assenstelsel van de AGV, namelijk: De waardes worden in mm/sec. mgraden/sec. doorgestuurd, dit omdat de SICK scanner (op basis van kunstmatige oriëntatiepunten) deze gegevens nodig heeft om te lokaliseren.

Een bericht ziet er als volgt uit:

$$\boldsymbol{X} < X - richting(mm/sec) > \boldsymbol{Y} < Y - richting(mm/sec) > \boldsymbol{W} < rotatie(mgraden/sec) > \boldsymbol{B}$$

X: Geeft het begin van het bericht aan, hierna volgt de snelheid in de X-richting.

Y: Geeft het einde van de snelheid in de X-richting aan, hierna volgt de snelheid in de Y-richting.

W: Geeft het einde van de snelheid in de Y-richting aan, hierna volgt de rotatiesnelheid.

B: Geeft zowel het einde van het bericht en het einde van de rotatiesnelheid aan.

3.7.2.1 Opbouw RS232_tf_broadcaster:

Verbinding maken
Het programma opent een ingestelde COM poort en leest deze aan een ingestelde baudrate uit. Wanneer er geen communicatie kan gelegd worden met deze COM poort, verschijnt een foutmelding.

Inlezen berichten
Berichten die via RS-232 binnen komen, worden initieel in een buffer geplaatst. Wanneer deze buffer een nieuw bericht (1 of meerdere karakters) binnen krijgt zal hij deze toevoegen aan eerder binnen gekregen karakters.

Controle
RS-232 is geen robuuste manier voor het verzenden van data, daarom is een controlemechanisme toegevoegd om ervoor te zorgen dat er steeds een correct bericht wordt ingelezen. Het programma controleert op volledigheid, volgorde en halve berichten.

volledigheid
Ten eerste wordt gecontroleerd of het bericht alle karakters bevat (X, Y, W en B), zo niet wachten we op nieuwe karakters.

volgorde
Als het bericht alle karakters bevat wordt gecontroleerd of deze in de juiste volgorde staan (X, Y, W en B).

Halve berichten
Soms wordt er maar een half bericht ingelezen, het volgende bericht wordt hieraan toegevoegd omdat het halve bericht dan bijvoorbeeld geen “B” bevat.

Voorbeeld foutief bericht:

$X10Y10X12Y12W0B$

Als hier de waardes van uitgelezen worden, krijgen we een foutieve Y waarde:

$X: 10$
$Y: 10X12Y12$
$W: 10$

Om deze fout te vermijden worden alle berichten gecontroleerd of ze dubbele X, Y of W waardes bevatten. De berichten met de dubbele waardes worden verwijderd.

Beperkingen
Er zijn nog altijd beperkingen aan het geschreven programma, waardoor er soms nog fouten worden ingelezen. De toevoeging van een pariteitsbit kan deze laatste fouten oplossen.

Een pariteitsbit geeft aan of een binaire code een even of oneven aantal logische enen heeft. Of de bit dan hoog is, of juist laag, hangt af van de afspraak die is gemaakt. Wanneer bij de verzending een fout optreedt waardoor een Oneven aantal bits van waarde verandert dan kan dit dankzij de pariteitsbit worden gedetecteerd. De ontvangen of teruggelezen rij bits heeft dan immers een oneven aantal ‘1’-bits. [24]

Extractie
Als het bericht door de controle geraakt worden de effectieve data hieruit afgeleid. Dit gebeurt door de posities van X, Y, W en B op te zoeken en de waardes hiertussen in elk een eigen integer te plaatsen.

Berekening odometrie
Uit de encoders van de AGV komt een snelheidsvector met volgende gegevens:

1) X-richting [millimeter/sec]
2) Y-richting [millimeter/sec]
3) Hoekverdraaiing [milligraden/sec]

Deze snelheden staan in het assenstelsel van de AGV (X-richting is vooruit, Y-richting naar links) en moeten dus omgevormd worden naar het assenstelsel van de map. De verplaatsing die volgens de odometrie gebeurt wordt elke keer een nieuw bericht ontvangen wordt upgedated. De afstanden worden berekend die de AGV heeft afgelegd sinds zijn vorige odometrie update. Deze afstanden worden daarna toegevoegd aan de vorige berekende afstanden.

```
    double dt = (current_time - last_time).toSec();
    double delta_x = (vx * cos(th) - vy * sin(th)) * dt;
    double delta_y = (vx * sin(th) + vy * cos(th)) * dt;
    double delta_th = vth * dt;
    x += delta_x;
    y += delta_y;
    th += delta_th;
```

3.7.3 Map_server

Het map_server pakket laat ons toe om opgebouwde mappen, zowel van HectorSLAM als GMapping, op te slaan als een bestand (.yaml) en deze door te sturen naar het AMCL algoritme. De map wordt opgeslagen als een raster. Elke cel kan een waarde hebben van 0 tot 255 met als betekenis: 0 geen bezetting tot 255 veel bezetting.

3.7.3.1 Problemen map_server

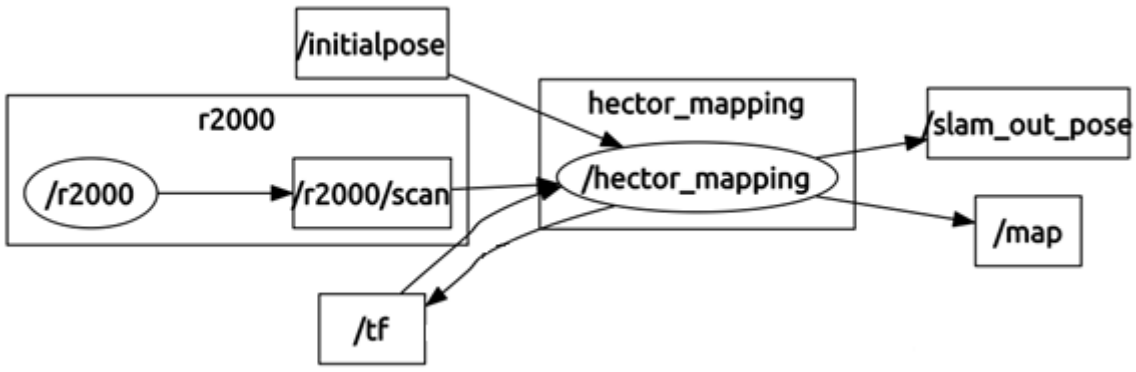
Het map_server pakket zit standaard in ROS. En kan dus zonder toevoegingen aan je catkin workspace gebruikt worden. Wanneer het AMCL pakket toegevoegd word aan je catkin workspace zal map_server niet meer werken. Dit komt omdat AMCL en mapserver samen met andere in de metapackeg "navigation-jade-devel" zitten. Als AMCL nu aan je catkin workspace wordt toegevoegd zullen alle verbindingen tijdens het compileren met de map_server verbroken worden. Dit kan simpel opgelost worden door map_server toe te voegen aan je catkin workspace.

3.7.4 HectorSLAM

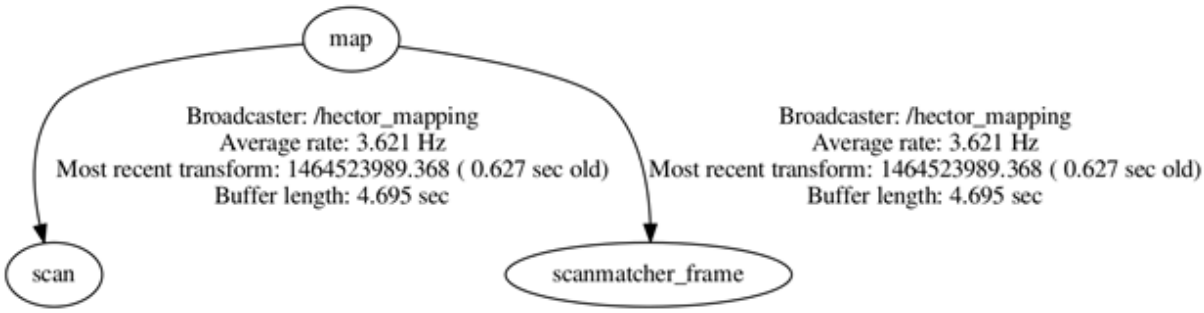
Het voordeel van HectorSLAM (Heterogeneous Cooperating Team Of Robots) is dat deze enkel data gebruikt van de laserscanner om een kaart op te bouwen. Er is dus nog geen AGV nodig om er testen mee uit te voeren. HectorSLAM is ontwikkeld in 2008 door Stefan Kohlbreche en Johannes Meyer aan de technische universiteit Darmstadt. In 2010 werd het algoritme aangepast zodat het binnen ROS kan werken [25].

3.7.4.1 Programma

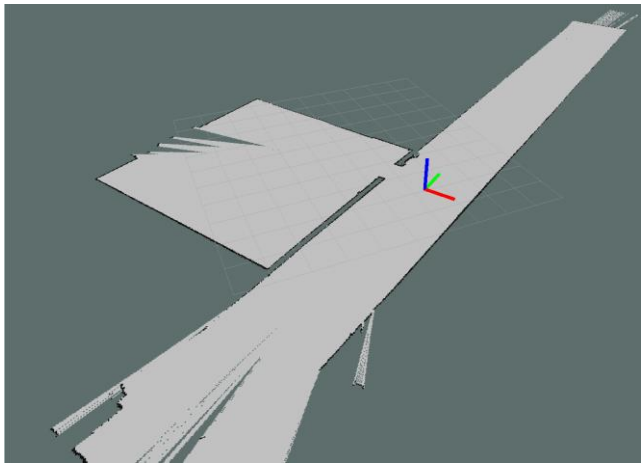
HectorSLAM heeft niet veel nodig om een kaart op te bouwen. Enkel de laserscanner-data, een initiële positie en een transformatiematrix. De transformatiematrix laat HectorSLAM weten op welke positie de scanner zich bevindt ten opzichte van het centrum van de AGV. Hier wordt de positie van de scanner beschouwd als het centrum van de AGV dus is er geen transformatie nodig. De assenstelsels kunnen gewoon doorgelinkt worden zonder deze aan te passen. De initiële positie kan naar wens ingesteld worden. De AGV zal in de meeste gevallen vanuit zijn laadpunt vertrekken om een kaart op te bouwen. Het laadpunt zal meestal als nulpunt beschouwd worden maar kan ingesteld worden naar de gewenste waarde. Eenmaal de mapping bezig is geeft HectorSLAM zijn positie en de map uit. De positie wordt in dit programma enkel gebruikt om de positie te laten zien in Rviz. De kaart wordt ook zichtbaar gemaakt in Rviz en kan wanneer de kaart compleet is opgeslagen worden. Deze kaart kan achteraf in het AMCL algoritme gestuurd worden.



Figuur 31: Rgt_graph HectorSLAM



Figuur 32: Transformaties HectorSLAM



Figuur 33: Map van klaslokaal en gang

3.7.4.2 Problemen HectorSLAM

HectorSLAM kan gebruikt worden wanneer er geen betrouwbare odometrie beschikbaar is. Toch moeten alle assenstelsels doorverbonden worden met elkaar. Als deze niet verbonden zijn zal HectorSLAM niet werken. Deze verbinding kunnen gemaakt worden in de launch file om HectorSLAM op te starten.

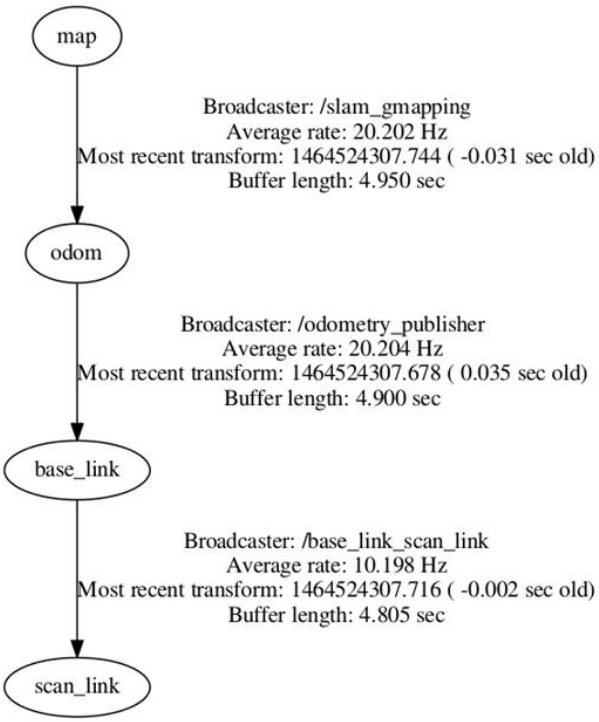
3.7.5 GMapping

Met het GMapping pakket kan een 2 dimensionale kaart van de omgeving opgebouwd worden. Het voordeel ten opzichte van HectorSLAM is dat dit algoritme gebruik maakt van odometrie. GMapping werd geschreven door Brian Gerkey. Buiten odometrie heeft GMapping ook nog nodig: scan-data, een initiële positie en transformaties tussen de verschillende assenstelsels (zie tf). De odometrie wordt uitgelezen door het RS232_tf_broadcaster pakket. De scan-data worden uitgelezen door het pepperl_fuchs_r2000 pakket. De opgebouwde kaart wordt uitgestuurd en kan gevisualiseerd worden. Eenmaal de kaart compleet is kan deze opgeslagen worden om later naar het AMCL algoritme te sturen [26] [27].

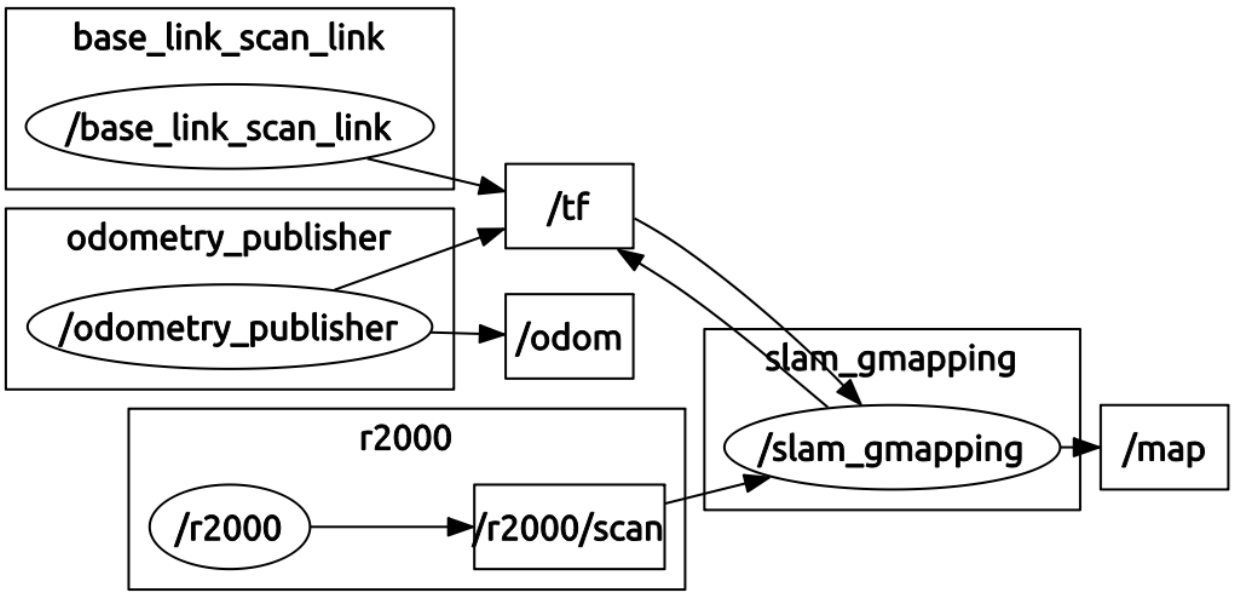
3.7.5.1 Coördinatentransformaties

Gmapping zorgt voor de transformatie van het kaartassenstelsel naar het odometrieassenstelsel. De transformatie van odometrie naar base_link (centrum van de AGV) wordt gedaan door de odometrie publischer die zich in de RS232_tf_broadcaster node bevindt. De scanner beweegt niet tegenover het centrum van de AGV dus daar wordt een statische transformatie aan toegewezen. Al deze tranformaties worden naar het topic tf gestuurd en worden hierin verwerkt. Tf is een standaard pakket dat in ROS

aanwezig is. Dit pakket stuurt transformaties tussen alles assen uit. Wanneer GMapping de tranformatie tussen de kaart en de scanner nodig heeft kan hij deze uit tf halen zonder deze zelf te moeten berekenen. Verschillende transformaties worden op verschillende tijden geüpdatet en transformaties tussen assenstelsel worden gevraagd op verschillende tijden. Om ervoor te zorgen dat de uitgestuurde transformatie zo goed mogelijk zijn, zal het tf pakket vaak moeten extrapoleren.



Figuur 34: GMapping transformaties



Figuur 35: Rqt_graph GMapping

3.7.5.2 Problemen GMapping

Gmapping heeft odometrie nodig hiervoor hebben we het pakket rs232_tf_broafcaster geschreven. In de eerste versie van het programma kwamen er veel foutieve odometrieberichten binnen waardoor GMapping zijn positie vaak verloor. We hebben de code aangepast zodat foute berichten niet doorgaan naar GMapping. GMapping heeft ook een limiet voor het aantal metingen per rotatie. Wij gebruiken standaard 3600 wat te hoog blijkt te zijn. Deze limiet stond nergens beschreven zowel op de infopagina als in de code. Het programma geeft een foutcode wanneer de limiet overschreven wordt, deze fout code geeft echter niet aan wat er juist fout is. Door trial and error op de parameters toe te passen zijn we erachter gekomen dat het groot aantal metingen de foutmelding veroorzaakte en dat het maximum op 2880 metingen per rotatie ligt. Dit kan met de gebruikte laptop te maken hebben, de specificaties van de gebruikte laptop zijn:

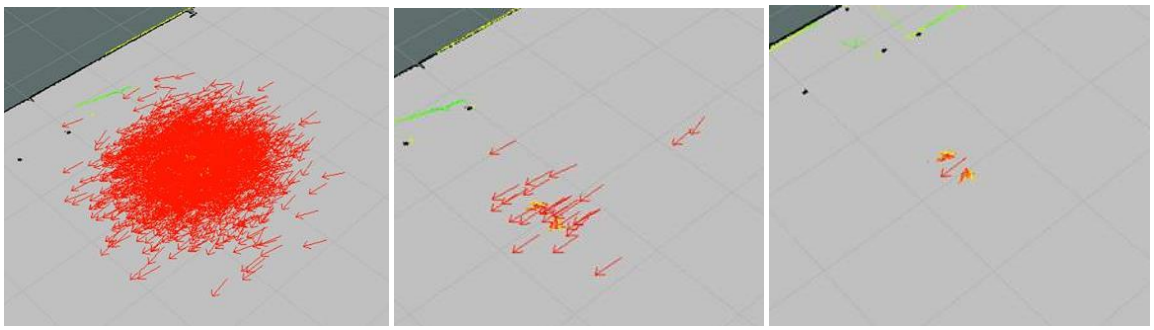
- Intel Core i3-5005U
- 8GB DDR3 RAM geheugen
- Intel HD graphics 5500

3.7.6 AMCL

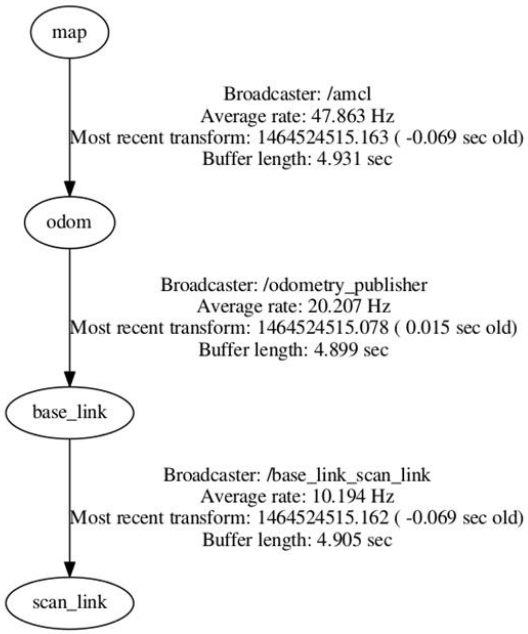
Het AMCL algoritme is gebruikt om de robotpose te bepalen in een eerder opgemeten kaart. De positie wordt bepaald aan de hand van de odometrie- en de scan-data. De odometrie wordt uitgelezen door het RS232_tf_broadcaster pakket. De scan-data worden uitgelezen door het pepperl_fuchs_r2000 pakket. Opnieuw word de tf functie gebruik om het algoritme van de nodige

transformaties te voorzien. De kaart wordt in het programma geladen via map server; dit kan eenmaal tijdens de start van het AMCL pakket of voortdurend. Wanneer er voortdurend een kaart in gestuurd wordt kan de map aangepast worden door bijvoorbeeld GMapping en AMCL simultaan te laten werken. Ook kunnen grote kaarten opgesplitst worden in kleinere delen. Dit vermindert de benodigde rekenkracht.

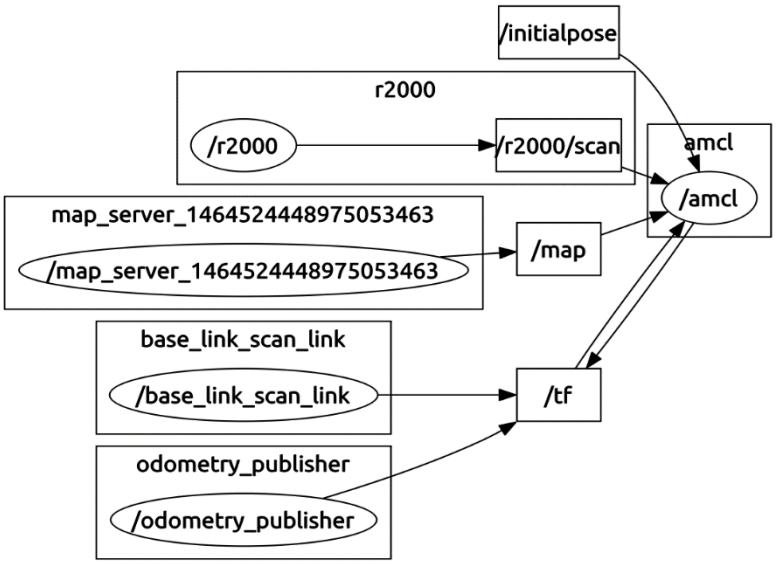
AMCL zal een start-pose vereisen, deze positie moet in de buurt liggen van zijn start positie. AMCL zal nu een set mogelijk positie particles rondom deze start positie plaatsen. Als de AGV nu beweegt zal het algoritme aan de hand van de odometrie en scan data nieuwe particles genereren en plaatsen op de map. De particles met de beste overeenkomst met de laser scandata worden behouden. De hoeveelheid particles zal dalen naargelang de zekerheid van de positie stijgt.



Figuur 36: Aantal particles



Figuur 37: Transformatie AMCL



Figuur 38: Rqt_graph AMCL

3.7.6.1 Problemen AMCL

Het AMCL was een pakket dat direct werkend was nadat het geïnstalleerd was. Echter positioneert AMCL zichzelf niet zonder een initiële positie te ontvangen of als de robot stilstaat. Dat een initiële positie nodig is hadden we vrij snel op de forums gevonden. Dat AMCL zichzelf niet positioneerde wanneer deze stilstond was een punt waar we lang achter gezocht hebben. Dit kwam voornamelijk omdat de programmatie niet bij Mabo gebeurde. Om het programma te testen simuleerde we een stilstaande robot. Eenmaal bekend was dat het programma beweging nodig had konden we verder. Ook zou AMCL om moeten kunnen gaan met “robot kidnapping” dit blijkt niet zo te zijn, wanneer de positie verloren raakt is er vrij weinig kans dat deze teruggevonden wordt. De map moet in vergelijking met GMapping en HectorSLAM zeer compleet zijn om een betrouwbare positionering te doen. Dit kan voor problemen zorgen wanneer er veranderingen in de omgeving gebeuren maar niet aangepast worden in de map. Dit kan opgelost worden door regelmatig, al dan niet automatisch, een nieuwe kaart op te bouwen.

4 Testen

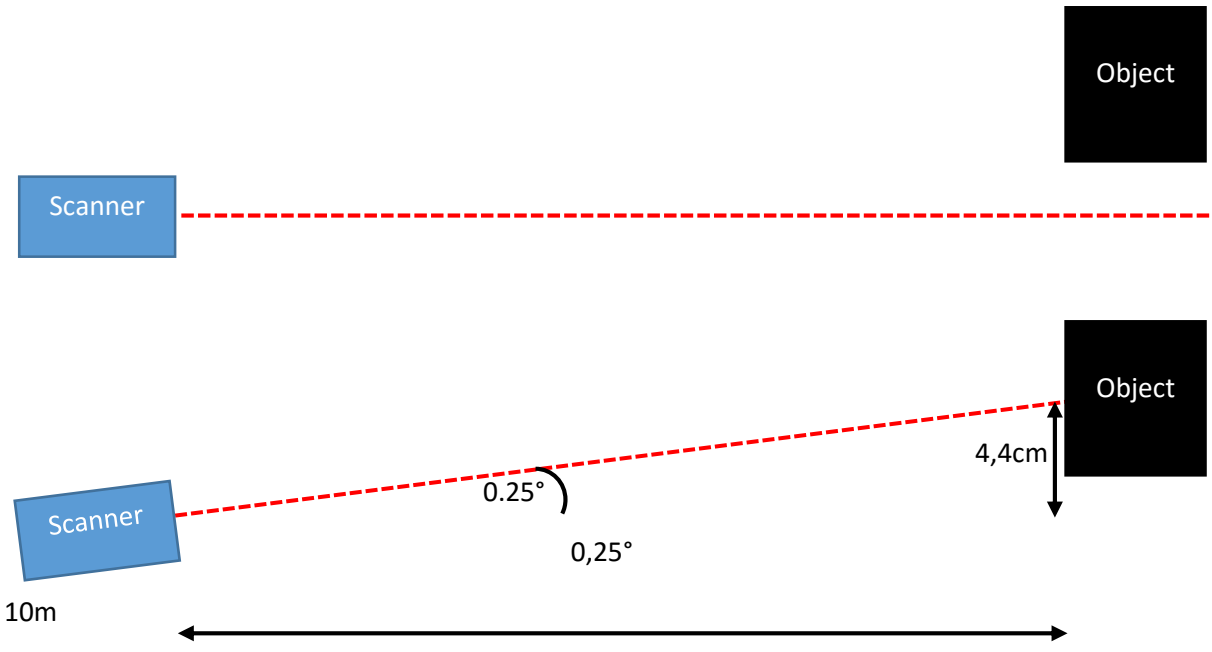
Het testen van het systeem ging initieel bij Sirris in Diepenbeek gebeuren, door tijdsgebrek is dit niet kunnen doorgaan. In plaats daarvan zijn de testen bij Mabo nv in lier zelf gedaan. De uitkomst van deze testen geven aan in welke maten het systeem toepasbaar is en of het een mogelijke vervanger kan zijn voor het systeem dat werkt op kunstmatige oriëntatiepunten.

4.1 Omgeving

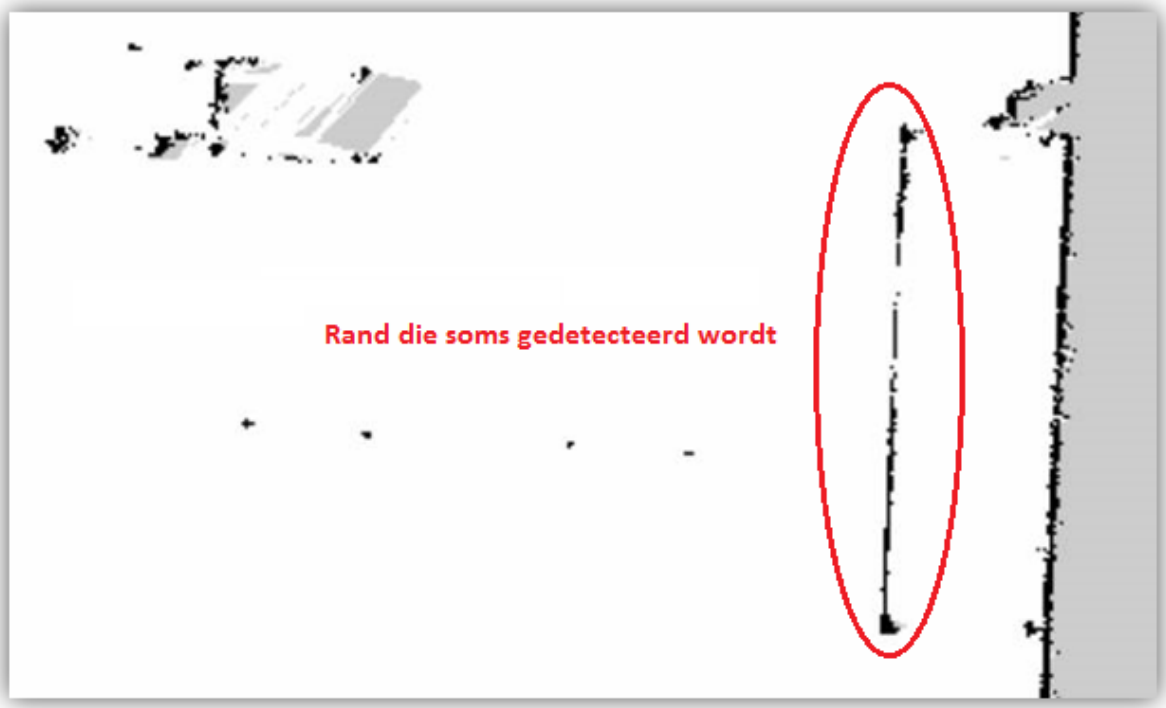
De ruimte die ter beschikking gesteld is bij Mabo nv is de hal waar heftrucks geplaatst worden voor reparatie of nieuwe/gerepareerde heftrucks die naar de klant moeten. De ruimte heeft een grootte van 100m bij 50m, hiervan is een vrije oppervlakte van 20m bij 10m vrijgemaakt voor het testen. Tijdens het testen is het niet mogelijk alle bezigheden in de hal stil te leggen hierdoor zaten er verschillen in de opgemeten map en de werkelijke map. Enkel voorbeelden van de grote verschillen zijn:

- Andere heftrucks die rondrijden in de omgeving
- Heftrucks die verplaats zijn
- Camions die heftrucks komen laden en lossen
- Materialen die toegevoegd/weggehaald worden
- Een door zichtbaar plastiek zeil dat voortduren geopend en gesloten word
- Veel randen op hoogte lasserscanner

Het is dus niet een ideale omgeving om de nauwkeurigheid te controleren. Wel stelt het een werkelijke situatie in een industriële omgeving voor. Het doorzichtbare plastiek zeil zorgt voor twee problemen. Het eerste probleem is dat de scanner het zeil soms detecteren soms kijkt hij er dwars doorheen. Het tweede probleem ontstaat wanneer iemand het zeil opent en sluit, de contour van het zeil zal nu niet meer overeenstemmen met de contour dat op de ingemeten kaart staat. Tegen de muren staan rekken deze rekken hebben een rand die toevallig op de een gelijke hoogte staat met de laserscanner. Door zeer kleine oneffenheden in het oppervlakte van de vloer kan het punt waarop de afstand gemeten word enkele centimeters verschuiven, dit resulteert in metingen waar het rek soms waargenomen word en soms niet.



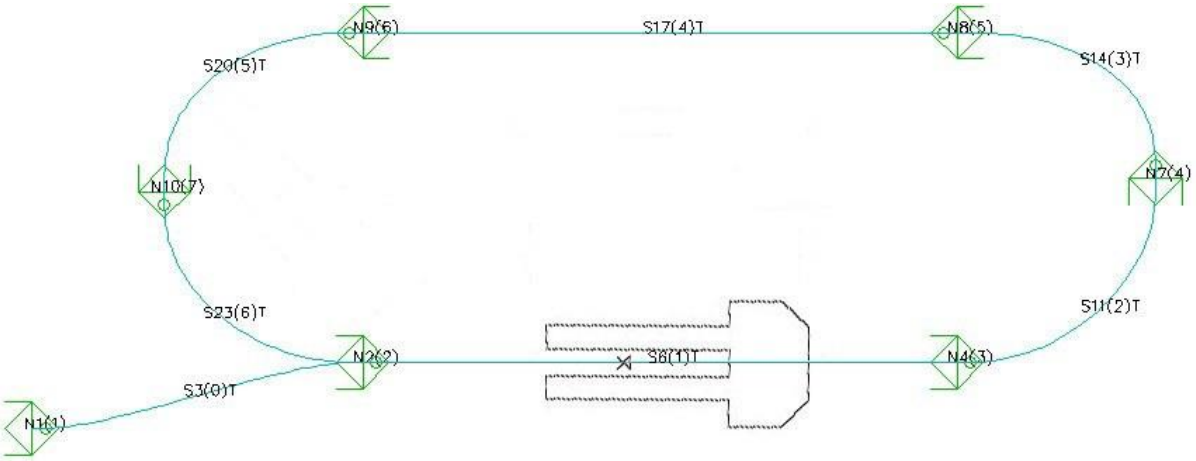
Figuur 39: Invloed oneffenheden vloer



Figuur 40: Gemeten rand

4.2 Gebruikte test

De AGV zal een rechte lijn volgen van 10m lang hierna draait de AGV linksom met een radius van 2m en rijd terug om weer een linkse draai te nemen met een radius van 2m. De AGV zal dit pad meerdere malen volgen. De snelheid van de AGV is 150mm/seconden. Een pen is bevestigd op de AGV en trekt een lijn over een papier elke keer als hij voorbij punt 3 en 6 Rijd, in deze test meten we de dynamische nauwkeurigheid. Op punt 6 Stopt de AGV en duiden we deze positie ook aan, dit zal de statische nauwkeurigheid aanduiden. De twee punten zijn gekozen omdat het ene punt zich bevindt bij het plastiek zeil waar vaak camions staan die niet in de map zitten en veel verkeer is, dit punt wordt vanaf hier gerefereerd als “het onstabiele punt”.. Het andere punt staat in de een stabielere omgeving waar niet veel veranderingen gebeuren, dit punt wordt vanaf hier gerefereerd als “het stabiele punt”. De pathplanning van de AGV zal voor 70% rekening houden met de positionering vanuit het ROS systeem dit is gekozen omdat de posities worden verstuurd met een snelheid van 5hz. De pathplanning heeft een hogere snelheid nodig en gebruikt zijn odometrie om de benodigde tussentijdse posities op te vullen.



Figuur 41: Pathplanning

4.2.1 Statisch

Statische nauwkeurigheid is de meest belangrijke, anders zal de heftruck falen om correct in de openingen van een pallet te rijden. De nauwkeurigheid is getest door bij stilstand zijn werkelijke positie te vergelijkingen met de positie in de software.

4.2.2 Dynamisch

Dynamische nauwkeurigheid is minder belangrijk, zolang de AGV maar op een correcte manier op zijn volgende positie geraakt. Om hiervan de nauwkeurigheid te testen, laten we de AGV constant in een vierkant rijden. Hierop is een stift geplaatst die steeds op het zelfde punt tijdens het rijden zijn positie aanduidt. Nadien kan dan gekeken worden of er veel verschil zit tussen deze lijnen. Het rijgedrag van de AGV geeft ook de nauwkeurigheid van de lokalisatie aan. Wanneer de AGV zijn pad al slalommend volgt kan men hieruit afleiden dat de lokalisatie niet goed is.

4.3 Test resultaten

De lokalisatie is getest met verschillende variabelen dit vooral om de invloed hiervan te bepalen. De volgende variabelen zijn getest:

- Resolutie opgebouwde kaart
- Rotatiesnelheid
- Aantal metingen per rotatie

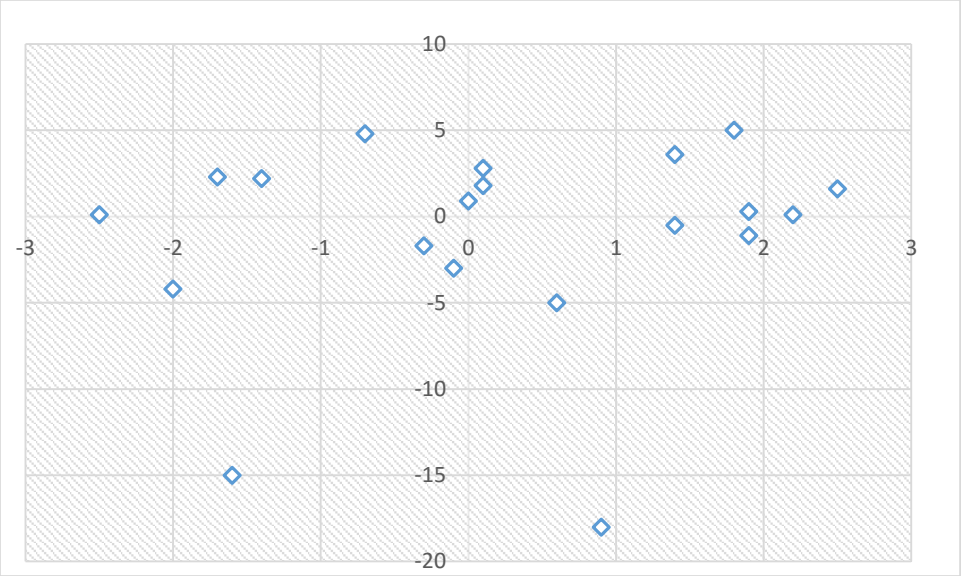
4.3.1 Rastergroote 200mm, maximum snelheid

Bij deze test is de rastergrootte(grootte van de cellen op de kaart) groot ingesteld en de scanner ingesteld op zijn maximum rotatiesnelheid en zijn bijhorende maximum aantal metingen per rotatie. De instellingen bedragen:

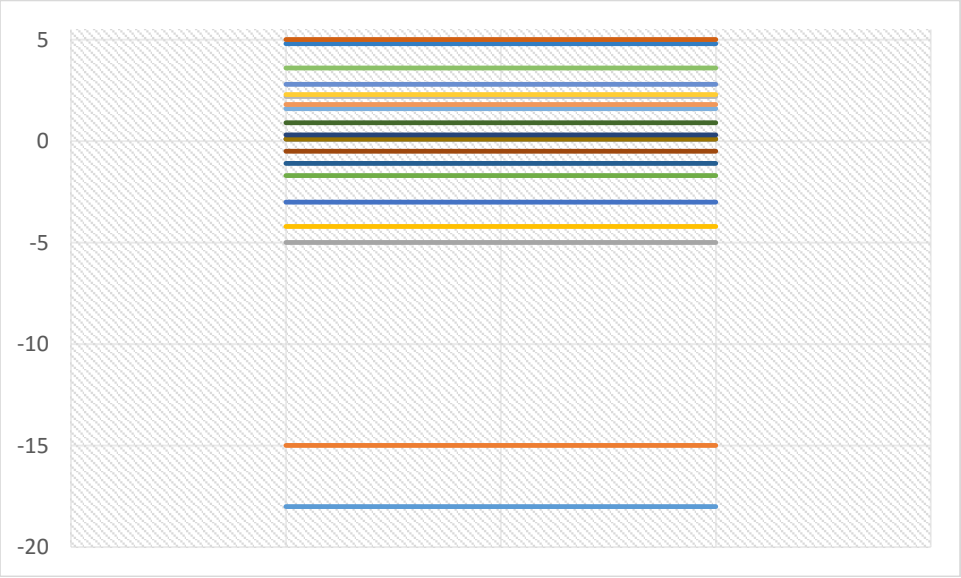
- Rastergroote: 200 mm
- Rotatiesnelheid: 50hz
- Aantal metingen per rotatie: 3600

4.3.1.1 Resultaten

De AGV volgde de lijn met een variatie van ± 5mm in het stabiele punt en stopte met een nauwkeurigheid van ±2.5mm. Bij het onstabiele punt was de nauwkeurigheid ±15mm met 2 uitschieters.



Figuur 42: Statische nauwkeurigheid



Figuur 43: Dynamische nauwkeurigheid

4.3.1.2 Besluit

Deze test geeft een goede basis om de komende testen met te vergelijken.

4.3.2 Rastergrootte 5mm maximum snelheid

Bij deze test is de map resolutie verhoogt en de scanner ingesteld op zijn maximum rotatiesnelheid en zijn bijhorende maximum aantal metingen per rotatie. De instellingen bedragen:

- Rastergrootte: 5 mm
- Rotatiesnelheid: 50hz
- Aantal metingen per rotatie: 3600

4.3.2.1 Resultaten

De AGV volgde de lijn met een variatie van ± 8mm in het stabiele punt en stopte met een nauwkeurigheid van ±3mm. Bij het onstabiele punt was de nauwkeurigheid ±20mm, er waren geen uitschieters

4.3.2.2 Besluit

Zoals verwacht resulteert een grotere rastergrootte in een minder nauwkeurige lokalisatie met minder uitschieters. Er zijn verschillende redenen waarom geopteerd kan worden voor een grotere rastergroote. De resolutie van de laserscanner kan niet voldoende zijn om hoger in resolutie te gaan. De gebruikte computer kan niet overweg met te grootte kaarten, de lokalisatie algoritmes zijn geoptimaliseerd om op meerdere CPU cores te werken en belasten 1 core heel intensief. Er is een robuust systeem vereisten in een dynamische omgeving, lokalisatie in mappen met een lagere resolute blijkt een grotere robuustheid te hebben.

4.3.3 Rastergrootte 5mm lagere snelheid

Bij deze test is de rastergrootte niet veranderd tegenover de vorige test en de scanner is ingesteld op een lagere rotatiesnelheid en een lager aantal metingen per rotatie. Dit om een goedkopere scanner te simuleren. De instellingen bedragen:

- Rastergroote: 200 mm
- Rotatiesnelheid: 10hz
- Aantal metingen per rotatie: 360

4.3.3.1 Resultaten

De AGV volgde de lijn met een variatie van ±6mm in het stabiele punt en stopte met een nauwkeurigheid van ±2.5mm. Bij het onstabiele punt was de nauwkeurigheid ±30mm met 5 uitschieters.

4.3.3.2 Besluit

De statische nauwkeurigheid is identiek gebleven, 360 meetpunten per rotatie lijkt genoeg te zijn om een nauwkeurige positionering te doen. Dynamisch is de nauwkeurigheid achteruit gegaan, het algoritme heeft minder punten om te vergelijken op een bewegend voertuig. Hierdoor zal het systeem ook minder robuust zijn zo als men kan zien aan de uitschieters. Toch is er een bruikbare lokalisatie mogelijk gemaakt wat aanduid dat goedkopere laserscanners zeker een plaats hebben op de markt.

4.4 Algemeen besluit testen

Uit deze minimale testen kunnen we afleiden dat wanneer de kaart vrij goed overeenstemt met het werkelijke dat lokalisatie met natuurlijke oriëntatiepunten mogelijk is. Variaties in de omgeving tegenover de kaart hebben een invloed op de lokalisatie en moeten minimaal gehouden worden. De snelheid in deze test was laag vergeleken met andere systemen, om te kijken hoe de lokalisatie daarop reageert moet nog getest worden.

5 Vergelijking natuurlijke- met kunstmatige oriëntatiepunten

Een onderdeel van deze masterproef is onderzoeken of lokalisatie met natuurlijke oriëntatiepunten een mogelijke vervanger kan zijn voor lokalisatie met kunstmatige oriëntatiepunten. Zoals al eerder gebleken is gaat dit afhankelijk zijn van verschillende factoren.

5.1 Kostprijs

De kostprijs van een lokalisatiesysteem gaat afhankelijk zijn van verschillende factoren. Volgende factoren hebben het meeste invloed op de prijs.

- Groote van het terrein
- Gewenste nauwkeurigheid (statisch)
- Gewenste snelheid (dynamische nauwkeurigheid)

5.1.1 Laserscanner

De eigenschappen van de lasserscanner gaan bepalen of deze geschikt is om aan deze parameters te voldoen. Laserscanners voor kunstmatige oriëntatiepunten moeten minder meetpunten hebben dan laserscanner voor natuurlijke oriëntatiepunten. Er moet immers enkel de afstand en hoek tot enkele reflectoren gemeten worden om een lokalisatie mogelijke te maken. De laserscanner voor natuurlijk oriëntatiepunten moet veel meer meetpunten hebben om een bruikbare lokalisatie te verwezenlijken. Ook de maximale te meten afstand ligt over het algemeen hoger bij kunstmatige oriëntatiepunten, dit voornamelijk omdat reflectoren gemeten worden en de scanner deze door hun grote reflectie factor op grotere afstanden kan detecteren. Deze eigenschappen bepalen voornamelijk prijs van de laserscanner. De laserscanner voor kunstmatige urgentiepunten zal het goedkoopste zijn.

Belangrijke eigenschappen lasserscanner kunstmatige oriëntatiepunten

- Maximale afstand
- Nauwkeurigheid afstandsmeting
- Nauwkeurigheid hoekresolutie

Belangrijke eigenschappen lasserscanner natuurlijke oriëntatiepunten

- Maximale afstand
- Nauwkeurigheid afstandsmeting
- Aantal metingen per rotatie
- Nauwkeurigheid hoekresolutie
- Rotatiesnelheid

5.1.2 Computer

De Computer zal zowel bij Natuurlijke als kunstmatige oriëntatiepunten de data uit de scanners en odometrie omzetten naar een locatie. Bij systemen met kunstmatige oriëntatiepunten zitten de scanner en computer meestal in 1 pakket. Dit pakket zal daarom duurder zijn dan de laserscanner voor natuurlijke oriëntatiepunten. Voor lokalisatie met natuurlijke oriëntatiepunten moet nog een aparte computer voorzien worden. In deze masterproef is gebruikt gemaakt van een laptop, in de praktijk kan hiervoor een computer zonder randapparatuur voor gekozen worden wat de prijs zal doen zakken. Ook een mogelijk is om de berekeningen door de computer te laten gebeuren die al aanwezig is op de AGV voor pathplanning etc. Natuurlijk moet deze, meestal embedded computer, voldoende capaciteit hebben om de berekeningen uit te voeren.

5.1.3 Inmeten reflectoren

Bij het systeem met kunstmatige oriëntatiepunten moeten de reflectoren ingemeten worden met een theodoliet. De kostprijs om iemand hiervoor in te huren kost 125€/uur. Te tijd nodig om alle reflectoren in te meten is afhankelijk van het aantal reflectoren en grootte van het terrein. Bij het systeem met natuurlijke oriëntatie punten moet de AGV de map opbouwen. De map opbouw wordt gedaan door de AGV handmatig het terrein rond te rijden, dit kan zelf gedaan worden en neemt aanzienlijk minder tijd in beslag.

5.1.4 Vergelijking

Een definitieve vergelijking is niet mogelijk dit omdat elk systeem zijn eigen voor en nadelen heeft. Toch gaan we ons systeem (natuurlijke oriëntatiepunten) vergelijken met het systeem (kunstmatige oriëntatiepunten) dat initieel op de AGV stond.

Tabel 3: Vergelijking verschillende systemen

	Natuurlijke oriëntatiepunten	Kunstmatige oriëntatiepunten
laserscanner:	Pepperl-Fuchs OMD30M-R2000-B23-V1V1D-T-1L	SICK NAV350
prijs laserscanner:	€ 5 880	€ 6 790
prijs computer:	€ 1 000	in scanner
prijs inmeten reflectoren:	niet nodig	125€/uur x 3uur*
	€ 6 880	7165

*gemiddeld 3 uur voor 40 reflectoren

Bij deze vergelijking moet wel rekening gehouden worden dat de OMD30M-R2000 scanner van Pepperl-Fuchs een van de betere scanners is, er zou een goedkopere scanner gekozen kunnen worden. Ook de computer is hier een laptop en is zeker niet de meest kost efficiënte oplossing.

5.2 Nauwkeurigheid

Vanuit Mabo nv weten we dat de SICK NAV350 een nauwkeurigheid haalt van ±15mm, deze nauwkeurigheid haalt hij in vrijwel alle omgevingen. Uit de testen blijkt dat met natuurlijke oriëntatiepunten deze nauwkeurigheid zeker kan geëvenaard worden. Maar de nauwkeurigheid is zeer afhankelijk van de omgeving en de gekozen scanner.

5.3 Betrouwbaarheid

5.3.1 Kunstmatige

De betrouwbaarheid van een systeem met kunstmatige oriëntatiepunten is afhankelijk van het aantal reflectoren die de scanner kan waarnemen. wanneer reflectoren geblokkeerd worden kan het systeem zijn positie verliezen. Ook reflecterende objecten als roesvrijstalen buizen of ramen kunnen aanzien worden als reflectoren waardoor het systeem zijn positie verliest.

5.3.2 Natuurlijke

Bij een systeem met kunstmatige oriëntatiepunten is de betrouwbaarheid afhankelijk van verschillende factoren.

- Correctheid opgebouwde map
- Map resolutie
- Slecht opmeetbare objecten zoals: zeilen, ramen,…
- Grote van de omgeving

Met al deze factoren moet rekening gehouden worden en eventueel geëlimineerd worden. Zeilen kunnen vervangen worden door schuifdeuren en op ramen kan een witte band geplaatst worden ter hoogte van de scanner om maar enkele voorbeelden te geven. Het kaartopbouw algoritme kan om simultaan werken met het lokalisatiealgoritme om zo regelmatig een kaart update te voorzien.

5.3.3 besluit betrouwbaarheid

Wanneer een keuze gemaakt wordt tussen de twee systemen als meest betrouwbare moet gekeken worden naar de omgeving. Wanneer de reflectoren geblokkeerd worden bij een systeem op kunstmatige oriëntatiepunten zal de betrouwbaarheid dalen. Het systeem op natuurlijke oriëntatiepunten kan om met veranderingen zolang deze maar niet te groot zijn. Meer testen in een industrieel representatieve omgeving zijn nodig om hier een verder besluit uit te trekken.

5.4 Besluit

Uit deze vergelijking zien we dat een systeem op natuurlijke oriëntatiepunten zeker een mogelijke vervanger is voor de systemen op kunstmatige oriëntatiepunten. Zeker bij toepassingen waar de eisen voor nauwkeurigheid en snelheid niet zo hoog zijn. Hier kunnen goedkope scanners gebruikt worden. Wanneer er in toekomst betere lokalisatie algoritmes gecreëerd worden en de prijs voor de benodigde rekenkracht daalt zal het moeilijk zijn deze techniek te negeren.

6 Besluit

Het doel van deze masterproef was een lokalisatie creëren met natuurlijke oriëntatiepunten om deze te vergelijken met de andere navigatiemethodes met in het specifieke de kunstmatige oriëntatiepunten. In het eerste semester werd eerst een uitgebreide literatuurstudie gedaan naar alle gebruikte systemen. Toen we meer bekend waren in de AGV wereld zijn we dieper ingegaan op de systemen met natuurlijke oriëntatiepunten. We hebben de verschillende algoritmes bestudeert en mogelijke problemen en aandacht punten bekennen. Verschillende leveranciers zijn ook gecontacteerd om een prijsvergelijking tussen de beschikbare sensoren op te stellen.

In het tweede semester is de programmatie en implementatie van start gegaan. Het programma werd gemaakt in ROS, hiervoor was eerst de nodige kennis van het ROS systeem nodig. Door de ROS wiki pagina samen met zijn tutorials volledig te doorlopen zijn we stillaan bekend geraakt in de ROS wereld. Door verschillende ROS pakketten te combineren met zelf geschreven code is een werkende mapping en lokalisatie programma gecreëerd.

De testen gingen in eerste instantie bij het onderzoekcentrum Sirris in Diepenbeek gebeuren, dit is niet meer gelukt vanwege een tekort aan tijd. We hebben de AGV getest bij Mabo nv, uit deze testen blijkt dat lokalisatie met natuurlijke oriëntatiepunten een mogelijke vervanger is voor het systeem met kunstmatige oriëntatiepunten.

De verschillende scanners zijn ook niet getest kunnen worden vanwege een tekort aan tijd. In de plaats hiervan zijn parameters van de gebruikte scanner aangepast om een slechtere scanner voor te stellen. Hieruit hebben we een idee welke invloed een goedkopere scanner heeft op de uiteindelijke prestaties.

AGV’s komen steeds vaker voor in de industrie, door deze masterproef hebben we ons inzicht hierin vergroot. Het onderzoeksproces heeft ons vele nieuwe aspecten geleerd die ook buiten de AGV wereld bruikbaar zijn. De programmatie in ROS zal ons zeker bij blijven en we zullen de ontwikkeling hiervan zeker verder volgen.

Verwijzingen

[1] Sirris, „Innoveren is ons vak,” Sirris, 5 Januari 2016. [Online]. Available: http://www.sirris.be/nl.

[2] SICK, „Operating instructions NAV350,” 27 Januari 2014. [Online]. Available: https://www.mysick.com/saqqara/im0040143.pdf.

[3] S. Thrun, W. Burgard en D. Fox, Probabilistic Robotics, 1999.

[4] „Automated guided vehicle,” [Online]. Available: https://en.wikipedia.org/wiki/Automated_guided_vehicle.

[5] Götting, „AGV Guidance Technologies,” Götting, [Online]. Available: http://www.goetting-agv.com/dateien/downloads/2012_02_16_automatic-guidance-of-vehicles-presentation_E_R07_TD.pdf.

[6] R. Demuth, „Solutions,” Goetting, 6 Juni 2013. [Online]. Available: http://www.goetting-agv.com/solutions.

[7] AGVS, „How do the vehicles work in an AGV system?,” MHI, 2012. [Online]. Available: http://www.mhi.org/downloads/industrygroups/agvs/elessons/vehicles-work-agv.pdf.

[8] R. Demuth, „Hannover Messe International 2013,” Goetting, 18 Juni 2013. [Online]. Available: http://www.goetting-agv.com/news/2013/hmi.

[9] S. Report, „Custom AGV eliminates need for employee to shuttle carts between departments,” Sustainable, 24 Oktober 2014. [Online]. Available: http://sustainablemfr.com/energy-efficiency/custom-agv-eliminates-need-employee-shuttle-carts-departments.

[10] B. V. A. M. Evens, „Haalbaarheidsstudie naar het inzetten van AGV's in een flexibele werkvloeromgeving,” 2015.

[11] Transbotics, „Guidance / Navigation,” Transbotics, [Online]. Available: https://www.transbotics.com/learning-center/guidance-navigation.

[12] T. M. Anandan, „Intelligent Robots: A Feast for the Senses,” Robotic Industries Association, 25 Juni 2015. [Online]. Available: http://www.robotics.org/content-detail.cfm/Industrial-Robotics-Industry-Insights/Intelligent-Robots-A-Feast-for-the-Senses/content_id/5530.

[13] A. Censi, „A comparison of algorithms for likelihood”.

[14] F. Dellaert, D. Fox, W. Burgard en S. Thrun, „Monte Carlo Localization for Mobile Robots”.

[15] „Monte Carlo Localization,” Wikipedia, 19 April 2015. [Online]. Available: https://en.wikipedia.org/wiki/Monte_Carlo_localization.

[16] D. Fox, „Adapting the Sample Size in Particle Filters Through KLD-Sampling,” p. 11.

[17] H. Durrant-Whyte, „Simultaneous Localisation and Mapping (SLAM):,” pp. 1-2.

[18] S. Thrun, „Learning metric-topological maps for indoor,” *Elsevier,* p. 245, 1996.

[19] S. Riisgaard en M. Rufus Blas, „SLAM for Dummies," pp. 10-15.

[20] G. Grisetti, C. Stachniss en W. Burgard, „GMapping," OpenSLAM, [Online]. Available: https://www.openslam.org/gmapping.html.

[21] G. Grisetti, G. D. Tipaldi, C. Stachniss, W. Burgard en D. Nardi, „Fast and Accurate SLAM with Rao-Blackwellized Particle Filters," p. 1, 2006.

[22] M. Quigley, B. Gerkey, K. Conley, J. Faust, T. Foote, J. Leibs, E. Berger, R. Wheeler en A. Ng, „ROS: an open-source Robot Operating System".

[23] „Wiki ROS," ROS, [Online]. Available: http://wiki.ros.org/.

[24] „Pariteitsbit," Wikipedia, 17 Juni 2015. [Online]. Available: https://nl.wikipedia.org/wiki/Pariteitsbit.

[25] S. Kohlbrecher, „hector_slam," ROS, [Online]. Available: http://wiki.ros.org/hector_slam.

[26] G. Grisetti, C. Stachniss en W. Burgard, „Improved Techniques for Grid Mapping With Rao-Blackwellized Particle Filters," *IEEE TRANSACTIONS ON ROBOTICS,* nr. 23, 2007.

[27] B. Gerkey, „GMapping," ROS, [Online]. Available: http://wiki.ros.org/gmapping.

Auteursrechtelijke overeenkomst

Ik/wij verlenen het wereldwijde auteursrecht voor de ingediende eindverhandeling:
Kaartopbouw en lokalisatie voor automatisch geleide voertuigen met natuurlijke oriëntatiepunten

Richting: **master in de industriële wetenschappen: energie-automatisering**
Jaar: **2016**

in alle mogelijke mediaformaten, - bestaande en in de toekomst te ontwikkelen - , aan de Universiteit Hasselt.

Niet tegenstaand deze toekenning van het auteursrecht aan de Universiteit Hasselt behoud ik als auteur het recht om de eindverhandeling, - in zijn geheel of gedeeltelijk -, vrij te reproduceren, (her)publiceren of distribueren zonder de toelating te moeten verkrijgen van de Universiteit Hasselt.

Ik bevestig dat de eindverhandeling mijn origineel werk is, en dat ik het recht heb om de rechten te verlenen die in deze overeenkomst worden beschreven. Ik verklaar tevens dat de eindverhandeling, naar mijn weten, het auteursrecht van anderen niet overtreedt.

Ik verklaar tevens dat ik voor het materiaal in de eindverhandeling dat beschermd wordt door het auteursrecht, de nodige toelatingen heb verkregen zodat ik deze ook aan de Universiteit Hasselt kan overdragen en dat dit duidelijk in de tekst en inhoud van de eindverhandeling werd genotificeerd.

Universiteit Hasselt zal mij als auteur(s) van de eindverhandeling identificeren en zal geen wijzigingen aanbrengen aan de eindverhandeling, uitgezonderd deze toegelaten door deze overeenkomst.

Voor akkoord,

Vandenboer, Stijn **Wellens, Miggel**

Datum: **6/06/2016**