

THE HAGUE

UNIVERSITY OF APPLIED SCIENCES

Magnetic mapping met de Turtlebot3

Afstudeerstage

2021-2022

NH4

Docent begeleider: J.A. Brons

Waarnemend docent begeleider: N. van der Kolk

Extra beoordelaar: S.A. van den Berg

Opdrachtgever: TNO

T.B. Putto — 18046061

31 mei 2022

Samenvatting

Voor het uitvoeren van navigatie zijn er diverse technieken. De voornaamste methode maakt gebruik van Inertial Measurement Units (IMU) en GPS. Echter is het GPS-signaal binnen gebouwen lastig te ontvangen. Ook is het GPS-signaal eenvoudig te spoofen waardoor deze navigatiemethode niet meer bruikbaar is.

Om toch binnen te kunnen navigeren kan er gebruik gemaakt worden van een IMU in combinatie met het aardmagnetisch veld. Dit veld ligt vast waardoor de positie bepaald kan worden. Ook is het mogelijk om met behulp van Magnetic SLAM dit veld in kaart te brengen terwijl simultaan de positie bepaald wordt.

Bij Magnetic SLAM wordt het magneetveld gereconstrueerd aan de hand van eigenwaardes en basisfuncties bepaald over een domein. Vervolgens is dit door middel van een Extended Kalman Filter te fuseren met de IMU om zo, aan de hand van een initiële positie inschatting, een zo goed mogelijke positie te bepalen.

Een andere methode maakt gebruik van een LIDAR om de ruimte in kaart te brengen en de positie te bepalen. Een LIDAR is een snel roterende afstandsmeter die gebaseerd is op de tijd die een lichtpuls er over doet om uitgezonden, gereflecteerd en weer ontvangen te worden. Zo kan in korte tijd bij elke graad rotatiehoek een afstand worden gemeten die leidt tot een puntenwolk die de ruimte in een horizontaal vlak beschrijft. Met deze puntenwolk kan het LIDAR SLAM-algoritme aan de hand van features de positie bepalen.

In dit verslag wordt toegelicht hoe er met behulp van de Turtlebot3 Waffle genavigeerd kan worden aan de hand van magneetvelden. Hiervoor wordt naast de Turtlebot3 gebruik gemaakt van de Twinleaf VMR om het magneetveld te meten. Uiteindelijk wordt er antwoord gegeven op de onderzoeksvraag: ***Biedt een magneetsensor meerwaarde bij het autonoom navigeren in een binnenomgeving, naast het gebruik van LIDAR en IMU?***

Om hier achter te komen is het Magnetic SLAM vergeleken met LIDAR SLAM. Hieruit is gebleken dat een magneetsensor zeker meerwaarde kan bieden naast het bereik van een IMU. Zo is de drift, die zich bevindt in de IMU positie, drastisch te verminderen. Echter dient er meer onderzoek gedaan te worden naar de te kiezen hyperparameters van het Magnetic SLAM algoritme om zo de nauwkeurigheid nog hoger te krijgen. De positie bepaald aan de hand van LIDAR SLAM blijkt namelijk nauwkeuriger dan bepaald aan de hand van Magnetic SLAM.

Wel is het een zeer interessante navigatiemethode die wellicht in de toekomst GPS kan vervangen. Zo maakt het navigeren aan de hand van magneetvelden mogelijk om een naadloze transitie te bieden in zowel indoor als outdoor navigatie.

Inhoudsopgave

1	Inleiding	1
1.1	Navigatie	1
1.2	Het Aardmagnetisch veld	2
1.3	Doel van de opdracht	2
1.4	Onderzoeksvragen	3
2	Magnetostatica	5
2.1	Magnetostatica	5
2.2	Magnetische dipool	8
2.3	In kaart brengen van het magnetisch veld	9
3	Turtlebot3 waffle	11
3.1	Wat is een Turtlebot?	11
3.2	Topics, Subscribers en Publishers	11
3.3	Communiceren met de Turtlebot	12
3.4	Toevoegen TIWNLEAF VMR	12
4	Turtlebot & Sensoren	15
4.1	Turtlebot	15
4.2	Karakterisering van de sensoren	15
4.3	Gyroscoop	16
4.3.1	Specificaties	16
4.3.2	Quaternionen	16
4.3.3	Analyse gedrag en karakterisering	18
4.4	Versnellingsmeter	20
4.4.1	Specificaties	21
4.4.2	Analyse gedrag en karakterisering	21
4.5	VMR	23
4.5.1	Magnetoresistiviteit	23
4.5.2	Specificaties	23
4.5.3	Analyse gedrag en karakterisering	24
4.6	LIDAR	26
4.6.1	Specificaties	26
5	Compensatie magneetsensor	28
5.1	CLAVIS	28
5.2	In kaart brengen magnetische verstoring	29
5.3	Compensatie model	31
5.4	Verzamelen van de benodigde data	32
5.5	Bepalen van de regressiecoëfficiënten	33
5.6	Resultaat na compensatie	34
6	LIDAR SLAM	36
6.1	Methode	36
6.2	Voorbeelden	36
6.3	Implementatie LIDAR SLAM	38
7	Magnetic SLAM	39
7.1	Magnetic mapping	39
7.2	Magnetic SLAM algoritme	41
7.3	Voorbeelden Magnetic SLAM	43

7.4	Implementatie Magnetic SLAM	44
8	Resultaten & Discussie	47
8.1	LIDAR SLAM	47
8.2	Magnetic SLAM	49
8.3	LIDAR SLAM vs Magnetic SLAM	52
9	Conclusie	54
9.1	Deelonderzoeksvragen	54
9.2	Hoofdonderzoeksvraag	55
	Referenties	57
A	Installatie van de Turtlebot	60
B	Specificaties IMU (MPU-9250)	63
B.1	Gyroscoop	63
B.2	Magnetometer	63
B.3	Versnellingsmeter	64
C	Specificaties LIDAR (LDS-01)	65
D	Kalman Filter en Extended Kalman Filter	66
D.1	Kalman Filter	66
D.2	Extended Kalman Filter	67
E	Extra	70
E.1	RRT*	70
E.1.1	Uitleg RRT*	70
E.1.2	Toepassen RRT*	71
E.2	Dynamisch model Turtlebot	72
F	Veel gebruikte commando's	74
F.1	ROS	74
F.2	Matlab	74
G	Datasheet Turtlebot	75
H	Achtergrond veld	76
I	50-cloud-init.yaml	78
J	Code; Uitlezen VMR	79

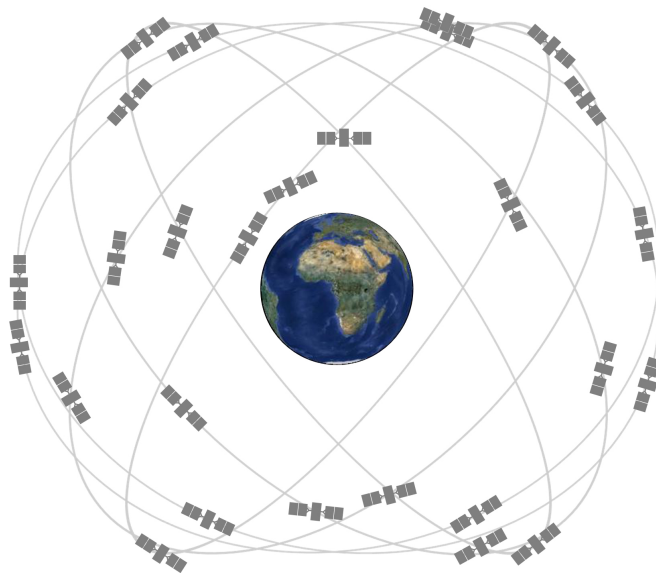
1 Inleiding

1.1 Navigatie

Stel je wilt afreizen naar de Haagse Hogeschool in Delft, maar je weet niet hoe je daar moet komen. Waarschijnlijk start je met het openen van “Google Maps”. Met deze applicatie op je telefoon is de snelste of kortste route naar een bestemming te bepalen, waarna het navigeren naar je bestemming kan gaan beginnen. Tot op heden is Google Maps de meest gebruikte applicatie voor navigatie. Andere applicaties zoals Waze, AppleMaps en MapQuest leveren vergelijkbare navigatie-mogelijkheden.

Google Maps maakt gebruik van het zogeheten Global Positioning System, zie Figuur 1. GPS is gebaseerd op de afstandsmeting tussen de ontvanger en een aantal satellieten. De afstand tussen de satelliet en de ontvanger wordt bepaald uit de looptijd van de radiogolven uitgezonden door de satelliet. Een faseverschil treedt tussen de ontvanger en de satelliet op dat omgerekend wordt tot de afstand. Aangezien de relatieve positie van de satellieten bekend is (ten opzicht van de Aarde), is het mogelijk om de positie van de ontvanger te bepalen [1].

De reden dat GPS het voornaamste hulpmiddel is voor navigatie komt door de vele voordelen. Zo is GPS gemakkelijk te gebruiken en is het niet alleen maar inzetbaar voor navigatie, maar ook voor bijvoorbeeld landmetingen, het volgen van objecten of personen, het vastleggen van verschillen kaarten en precieze tijdsynchronisatie. Een nadeel van GPS is dat het makkelijk verstoord kan worden door bijvoorbeeld bomen en hoge gebouwen. Daarnaast kan het GPS-sigitaal opzettelijk verstoord worden middels een jammer, waardoor het onbruikbaar wordt. Dit maakt het interessant om ook naar alternatieve technieken voor navigatie te kijken [2].



Figuur 1: Een illustratie van het GPS netwerk.

De positiebepaling aan de hand van GPS is niet perfect. Dit leidt ertoe dat de onzekerheid in de positiebepaling vaak in de orde van een aantal meter is, afhankelijk van het gebruikte systeem. Door het opnemen van bijvoorbeeld een Inertial Measurement Units (IMU), bestaande uit onder andere een gyroscoop en versnellingsmeter, kan de onzekerheid verkleind worden. Deze vorm van sensor fusie maakt het mogelijk om nauwkeurige plaatsbepalingen te krijgen. Sensorfusie is het proces waarbij data van sensoren gegevens afkomstig van verschillende bronnen zo worden gecombineerd dat de resulterende informatie minder onzeker is dan het geval zou zijn wanneer deze bronnen afzonderlijk zouden

worden gebruikt.

In sommige gevallen is het gebruik van GPS niet mogelijk. Denk aan navigatie binnenshuis of in tijden van oorlog waarbij de GPS-satellieten afgeschermd kunnen zijn. Van hieruit ontstaat de interesse om te kijken naar alternatieve navigatietechnieken. Voorbeelden hiervan zijn het gebruik van geografische informatie of radiosignalen die vanaf de aarde worden uitgezonden. Onlangs is ook de mogelijkheid onderzocht om het zwaartekrachtsveld of het magnetisch veld van de aarde voor navigatie te gebruiken. Hiervoor is het van belang dat er voldoende gevoelige sensoren zijn die de zeer kleine variaties in deze velden kunnen waarnemen.

In deze situatie is het ook mogelijk om aan de hand van een LIDAR te navigeren. Door gebruik te maken van een actieve lichtbron is de LIDAR instaat om hogere-nauwkeurigheid afstandmetingen uit te voeren. Hierdoor is LIDAR SLAM een robuust lokalisatie hulpmiddel voor het indoor navigeren. LIDAR SLAM systemen zijn gebaseerd op de scan-matching techniek om de verschillende gemeten afstanden samen te voegen en hieruit zo de ruimte in kaart te brengen, en de positie te bepalen. [3]

1.2 Het Aardmagnetisch veld

Overall is het Aardmagnetisch veld meetbaar, behalve als het opzettelijk is afgeschermd. Een illustratie van het Aardmagnetisch veld is te vinden in Figuur 2. Het Aardmagnetisch veld is het resultaat van een aantal componenten:

- Het magnetisch veld dat wordt opgewekt in de geleidende, vloeibare buitenkern;
- Het aardkorstveld dat in de aardkorst en de bovenmantel wordt opgewekt;
- Het gecombineerde stoorveld van elektrische stromen in de bovenste atmosfeer en de magnetosfeer, die elektrische stromen in de zee en de grond induceren.

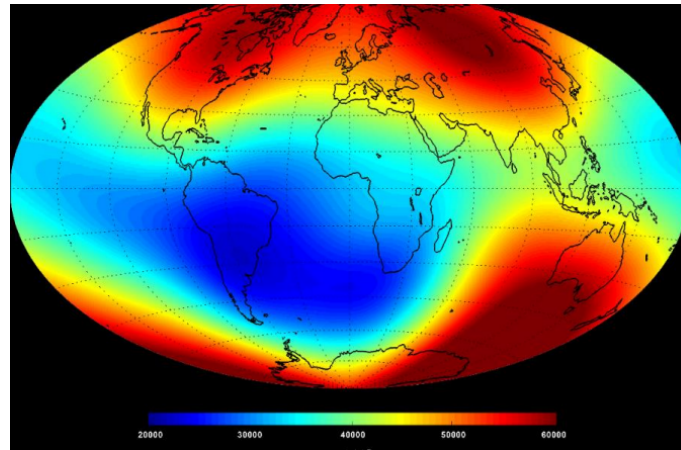
De kaart van het Aardmagneetveld wordt bepaald aan de hand van een grote collectie van metingen van het magnetisch veld over de hele Aarde. Deze metingen worden gebruikt om een multipool expansie te fitten, wat leidt tot het zogeheten World Magnetic Model (WMM). Dit model is te vinden op de website van National Center for Environmental Information (NOAA) [4].

Grote stalen voorwerpen zijn magnetisch van aard en beïnvloeden het lokale magnetisch veld. Wanneer deze voorwerpen niet van positie veranderen, ligt het statisch magnetisch veld vrijwel vast. Dit veld kan dan met een nauwkeurige magneetsensor worden waargenomen, waarna een magnetische kaart kan worden gemaakt. Navigatie is dan mogelijk door gebruik te maken van de magnetische kaart in combinatie met andere sensoren, zoals een IMU. De combinatie van een IMU en een magnetische kaart maakt het mogelijk om zonder GPS een nauwkeurige schatting van de positie te verkrijgen [5].

1.3 Doel van de opdracht

Het doel van de opdracht is het onderzoeken van magnetische navigatie. In het bijzonder kijken we of een magneet sensor meerwaarde kan bieden bij het autonoom navigeren in een gebouw. Waar navigeren aan de hand van GPS niet mogelijk is. Hiervoor zullen we naast een magneetsensor gebruik maken van Light Detection And Ranging of Laser Imaging Detection And Ranging (LIDAR) en een IMU. Er zal gebruik gemaakt worden van een Turtlebot die beschikt over een LIDAR en een IMU.

In een eerste stap zal aan de hand van de LIDAR een kaart gemaakt worden, waarmee de locatie van de Turtlebot3 waffle wordt bepaald. De positie inschatting door de LIDAR wordt al gebruikt binnen het werkveld en zal daarom als *ground truth* worden gebruikt.



Figuur 2: *Het Aardmagnetisch veld.*

Daarnaast gaan we kijken naar Simultaneous Localizing and Mapping (SLAM). SLAM is het proces voor het in kaart brengen van een ruimte terwijl tegelijk ook de positie van de Turtlebot wordt bepaald. Een gevoelige magnetische sensor is aan de Turtlebot toegevoegd voor het genereren van een magnetische kaart. Voor het bepalen van een magnetische map en het hierop navigeren wordt het zogeheten Magnetic SLAM-algoritme toegepast. Hiermee is het mogelijk om simultaan het lokale magneetveld in de ruimte in kaart brengen en hierbinnen de locatie van de Turtlebot te bepalen. Het algoritme dat we hiervoor gebruiken is een al bestaande methode ontwikkeld door onder andere Manon Kok en Arno Solin. De methode maakt gebruik van het gemeten veld, waarna door middel van een Gaussischregressie proces het magneetveld wordt geconstrueerd aan de hand van enkele basisfuncties om zo de complete ruimte in kaart te brengen. Vervolgens wordt het magneetveld gefuseerd met een initiële positie inschatting, bijvoorbeeld uit de IMU-data, door middel van een zogeheten Extended Kalman Filter.

Uiteindelijk zullen we beide kaarten gebruiken om te navigeren. De nauwkeurigheid van de positie bepaling aan de hand van beide kaarten zullen vergeleken worden. Op deze manier zal bepaald worden of een magneetsensor meerwaarde biedt bij het autonoom navigeren in een binnenomgeving.

1.4 Onderzoeksvragen

Bij het uitvoeren van het project wordt er onderzoek gedaan om antwoord te krijgen op onderstaande deelvragen. Deze deelvragen zullen samen met de hoofdonderzoeksvraag zorgen voor een rode draad door het verslag heen. De deelvragen die beantwoord gaan worden zijn hieronder opgenomen.

- Hoe Kan met behulp van de Turtlebot en een LIDAR een kaart worden gemaakt van de ruimte?
- Hoe kan de positie en oriëntatie van de magneetsensor in het globale coördinatensysteem worden bepaald om zo de magnetische metingen toe te voegen aan de kaart?
- Hoe kan het magnetisch veld worden bepaald op plekken waar niet gemeten wordt?
- Hoe kan er aan de hand van de gemaakte magnetische kaart worden genavigeerd?

Wanneer het geluk is om te navigeren aan de hand van de magnetische kaart zal de nauwkeurigheid van de navigatie met de magnetische kaart vergeleken worden met de situatie waarin de magnetische kaart niet gebruikt wordt. Op deze manier kan er antwoord gegeven worden op de hoofdonderzoeksvraag:

- **Biedt een magneetsensor meerwaarde bij het autonoom navigeren in een binnen omgeving, naast het gebruik van LIDAR en IMU?**

Als eerst zal de benodigde theorie in Hoofdstuk 2 met betrekking tot de veldvergelijkingen van de magnetostatica besproken worden. Tenslotte simuleren we een aantal voorbeelden van magnetische kaarten. Vervolgens wordt het gebruikte platform, de Turtlebot3 Waffle, toegelicht in Hoofdstuk 3. In dit hoofdstuk wordt de Turtlebot geïntroduceerd en werking hiervan toegelicht gevolgd door de communicatie methode en een toelichting over het opnemen van een extra magneetsensor.

Vervolgens zal de kwaliteit van de gebruikte sensoren in kaart worden gebracht om eventuele benodigde correcties in kaart te brengen in Hoofdstuk 4. Gevolgd door het onderzoeken van de invloed van de Turtlebot op het te meten magnetisch inductie-veld in Hoofdstuk 5, waarna de compensatie hiervoor ook toegepast zal worden.

Hierna zal LIDAR SLAM (LSLAM) worden toegelicht in Hoofdstuk 6. Als eerst wordt deze methode toegelicht waarna een voorbeeld gegeven wordt. Tot slot wordt dit hoofdstuk afgesloten met het bespreken van de implementatie van LSLAM op de Turtlebot.

Dit hoofdstuk zal gevolgd worden door een hoofdstuk met de werking van Magnetic SLAM (MSLAM). In dit hoofdstuk, Hoofdstuk 7 wordt toe gelicht hoe het magneetveld in kaart te brengen is, maar ook het Extended Kalman Filter van het MSLAM algoritme wordt toegelicht. Vervolgens word een voorbeeld van Magnetic SLAM gegeven samen met een toelichting over de implementatie hiervan.

Uiteindelijk worden de resultaten besproken in een apart hoofdstuk, Hoofdstuk 8. In dit hoofdstuk zal eerst de LSLAM behandeld worden gevolgd door de MSLAM resultaten. Tot slot wordt beide met elkaar vergeleken waarna er een discussie volgt over het verbeteren van MSLAM.

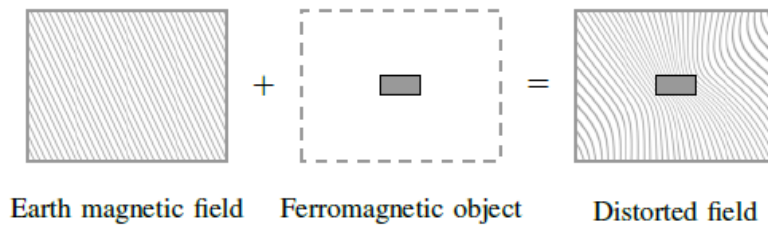
Tot slot zal in de conclusie besproken word wat de meerwaarde van magnetische navigatie is samen met de punten waar nog aan gewerkt moet/kan worden. In Bijlage E is overig werk met betrekking tot de Turtlebot opgenomen wat niet direct aansluit bij Magnetic Mapping met de Turtlebot3.

2 Magnetostatica

De theorie besproken in dit hoofdstuk ligt de basis waarop het magnetisch veld in kaart wordt gebracht. Aan de hand van de besproken relaties wordt uiteindelijk een dipoolveld gesimuleerd waarna de theorie gebruikt kan worden om een simulatie van een magnetische kaart te maken.

2.1 Magnetostatica

Overall, tenzij expres afgeschermd, is het Aardmagnetisch veld aanwezig. Dit veld wordt voor onze toepassing als statisch beschouwd, maar kan worden beïnvloed door ferromagnetische objecten. De invloed van een ferromagnetisch object is hieronder weergegeven in Figuur 3.

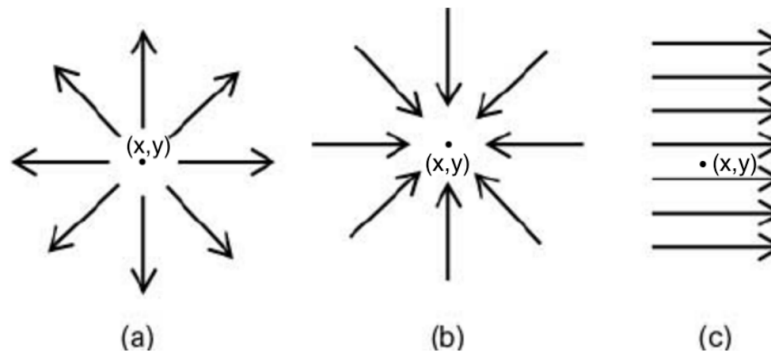


Figuur 3: Het effect van een ferromagnetisch object op het Aardmagnetisch veld. [6]

Met de hulp van de hieronder opgenomen veld-vergelijkingen voor magnetostatica is een statisch magneetveld te beschrijven. Deze vergelijkingen inclusief aannames vormen samen de basis voor het maken van een magnetische kaart. Formule 1 beschrijft de divergentie van het magnetisch inductie-veld:

$$\nabla \cdot \mathbf{B} = 0. \quad (1)$$

Hierin is \mathbf{B} het magnetisch inductieveld in Tesla en ∇ de gradiënt operator bestaande uit de partiële afgeleiden in de x, y en z -richting. Formule 1 laat zien dat de divergentie gelijk is aan nul wat betekent dat er geen punten zijn waaruit het magnetisch inductie-veld ontspringt. Figuur 4 laat drie mogelijke scenario's zien voor de divergentie van een vectorveld.



Figuur 4: Voorbeelden van de divergentie van een vectorveld waarbij de divergentie kleiner dan nul, groter dan nul en nul is.

Figuur 4a laat het effect van een divergentie groter dan nul zien. Hier wijst het veld vanuit punt (x, y) naar buiten. Dit wordt dan ook vaak een source node genoemd. Figuur 4b laat een divergentie kleiner dan nul zien. In dit geval wijst het veld punt (x, y) in. Tot slot laat Figuur 4c een voorbeeld zien waarin de divergentie van nul is. Dit is het geval zoals formule 1 beschrijft. De meeste divergentievrije vectorvelden zijn overigens niet constant. In deze

situatie ligt het magneetveld vast en is er geen punt waaruit het magnetisch inductie-veld ontspringt.

Het magneetveld is af te leiden aan de hand van Formule 2 hieronder:

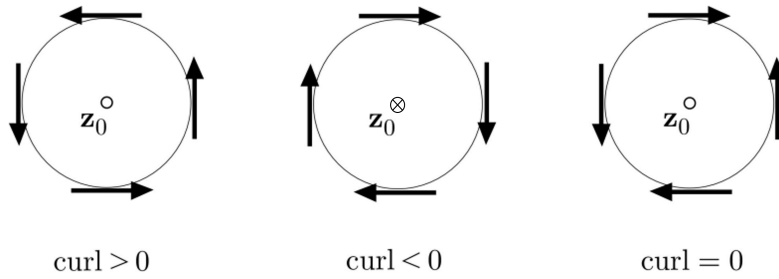
$$\nabla \times \mathbf{H} = \mu_0 \left(\mathbf{J} + \varepsilon_0 \frac{\partial \mathbf{E}}{\partial t} \right) \quad (2)$$

Hierin is \mathbf{H} het magneetveld in ampère per meter [A/m], μ_0 de magnetische permeabiliteit in vacuüm ($\mu_0 = 4\pi \cdot 10^{-7} \text{ N/A}^2$), \mathbf{J} de totale stroomdichtheid in A/m^2 , ε_0 de permittiviteit in vacuüm ($\varepsilon_0 = 8.85 \cdot 10^{-12} \text{ C}^2/\text{N} \cdot \text{m}^2$) en $\frac{\partial \mathbf{E}}{\partial t}$ de verandering van het elektrisch veld in de tijd [7].

In dit project zullen we alleen statische magneetvelden beschouwen. Dit heeft als gevolg dat de tijdsafhankelijke grootheden verdwijnen in de eerder genoemde formules, dat wil zeggen, $\frac{\partial \mathbf{E}}{\partial t} = \mathbf{0}$. Ook nemen we aan dat er geen vrije stromen in ons probleem bestaan, hetgeen betekent dat $\mathbf{J} = \mathbf{0}$. Substitutie hiervan in Formule 2 leidt tot:

$$\nabla \times \mathbf{H} = \mathbf{0}. \quad (3)$$

Formule 3 laat zien dat de rotatie van het magneetveld gelijk is aan nul. Dit houdt in dat de richting van het magneetveld niet verandert, maar constant blijft. Het effect van een rotatie kleiner en groter dan nul is samen met een rotatie gelijk aan nul schematisch weergegeven in Figuur 5.



Figuur 5: Effect van een negatieve en een positieve rotatie op vectoren. Samen met de weergave van een rotatie gelijk aan nul. Met curl de Engelse benaming van rotatie.

Figuur 5 laat zien dat wanneer de projectie van de rotatie op de z-as groter is dan nul, de vector rond punt z_0 tegen de klok in roteert. De richting van de rotatie is hierbij het blad uit. Is de rotatie kleiner dan nul dan staan de vectoren rond punt z_0 gericht met de klok mee en is gaat de rotatie het blad in. Bij een rotatie die gelijk is aan nul, laat figuur 5 zien dat de vectoren rond punt z_0 niet gericht zijn in een enkele klokwaartse oriëntatie. [8]

Bij het maken van de magnetische kaart wordt het magneetveld als functie van tijd als constant beschouwd. De relatie tussen het magnetische inductie-veld \mathbf{B} en het magneetveld \mathbf{H} is gegeven aan de hand van Formule 4, waarin \mathbf{M} de magnetisatie is, een vectorveld, in ampère meter kwadraat [A m^2]:

$$\mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M}). \quad (4)$$

Formule 4 laat zien dat het magnetisch inductie-veld \mathbf{B} bepaald kan worden als de som van het magneetveld \mathbf{H} en de magnetisatie \mathbf{M} maal een constante μ_0 . Hiermee is het magnetisch inductieveld te beschrijven aan de hand van het magneetveld en de magnetisatie.

De magnetisatie \mathbf{M} van een materiaal is hierbij gedefinieerd als het magnetische dipool moment per volume-eenheid. Magnetisatie is het gevolg van de uitlijning van dipool momenten \mathbf{m} in een materiaal.

Wanneer het magnetisch inductieveld buiten een magnetisch materiaal wordt beschreven geldt dat $\mathbf{B} = \mu_0 \mathbf{H}$. Buiten het magnetisch materiaal geldt namelijk dat de magnetisatie \mathbf{M} gelijk is aan nul. Deze vergelijking wordt ook wel de constitutieve relatie genoemd en geldt overal. Deze formule is af te leiden aan de hand van Formule 5:

$$\mathbf{B} = \mu_0(\mathbf{H} + \mathbf{M}) = \mu_0(\mathbf{H} + 0) = \mu_0 \mathbf{H}. \quad (5)$$

Aangenomen dat er geen vrije stromen aanwezig zijn, leidt Formule 3 tot een rotatie vrij \mathbf{H} -veld. Dit maakt het mogelijk om het magnetisch veld te beschrijven aan de hand van de scalaire magnetische potentiaal Φ . Hieruit volgt Formule 7:

$$\mathbf{H} = -\nabla\Phi. \quad (6)$$

Hierin is Φ de magnetische scalarpotentiaal in Ampere [A] en $\nabla := \left[\frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z} \right]^T$ de gradient operator uitgedrukt in de partiële afgeleiden naar x, y en z . Aan de hand van Formule 7 kan de magnetische kaart beschreven worden met alleen de scalaire potentiaal. Hiermee wordt het mogelijk om de drie vector componenten (H_x, H_y, H_z) van het magneetveld \mathbf{H} te bepalen wanneer de magnetische scalaire potentiaal bekend is.

Buiten het magnetisch materiaal is geen magnetisatie \mathbf{M} aanwezig. Aan de hand hiervan kan de laplaciaan worden opgesteld. Hiervoor dient eerst Formule 1 samengevoegd te worden met Formule 4 waaruit volgt dat:

$$\nabla \cdot \mathbf{B} = \mu_0 \nabla \cdot (\mathbf{H} + \mathbf{M}). \quad (7)$$

Merk op dat de divergentie van \mathbf{B} altijd nul is. Als we Formule 7 verder uitwerken vinden we:

$$0 = \mu_0 \nabla \cdot \mathbf{H} + \mu_0 \nabla \cdot \mathbf{M},$$

oftewel

$$-\nabla \cdot \mathbf{H} = \nabla \cdot \mathbf{M}.$$

Nu gebruiken we dat we het magnetisch veld kunnen uitdrukken in een scalaire potentiaal (Formule 6) en vinden we:

$$-\nabla \cdot (-\nabla\Phi) = \nabla \cdot \mathbf{M}.$$

Dit kunnen we herschrijven als

$$\nabla \cdot (\nabla\Phi) = \nabla \cdot \mathbf{M}.$$

Nu herkennen we de Laplaciaanoperator $\Delta \equiv \nabla \cdot \nabla$ in bovenstaande formule, waarbij $\nabla \cdot \mathbf{M}$ als bronterm voor de scalaire potentiaal gezien kan worden. Dit leidt tot Poisson's vergelijking, $\Delta\Phi = f$ met f een scalaire functie [9], voor de scalaire potentiaal:

$$\Delta\Phi = \nabla \cdot \mathbf{M}. \quad (8)$$

Als de bron in Formule 8 vooraf bekend is, dan kan de potentiaal voor het magnetisch veld berekend worden. Door nu gebruik te maken van $\mathbf{H} = -\nabla\Phi$, kan in een tweede stap het magneetveld bepaald worden. Merk op dat Poisson's vergelijking overal in de ruimte moet gelden. Buiten de magnetische bron zal dan voor de scalaire potentiaal gelden:

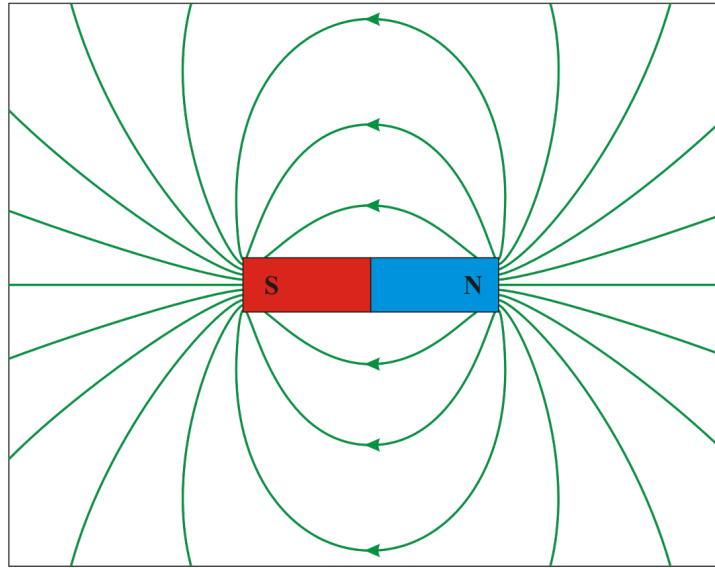
$$\Delta\Phi = 0 \quad (9)$$

Bovenstaande formules en aannames dienen als algemene achtergrond voor het maken van een magnetische kaart. De volgende paragraaf beschrijft de magnetische dipool die de basis zal vormen van het maken van een voorbeeld magnetische kaart.

2.2 Magnetische dipool

Deze paragraaf beschrijft de theorie behorend bij de magnetische dipool. Eerst wordt de magnetische dipool toegelicht. Vervolgens wordt er een verband toegelicht waarmee het mogelijk is om de sterkte van het magnetisch inductie-veld als functie van positie \mathbf{r} te bepalen.

In Figuur 6 is een magnetische dipool samen met de veldlijnen opgenomen. Een dipool bestaat een zuid- (rood) en een noord-pool (blauw). De veldlijnen lopen in een gesloten systeem van de noord-pool richting de zuid-pool. De uiteindelijk gemaakte magnetische kaart zal worden opgebouwd uit verschillende dipolen die samen het gemeten veld vormen.



Figuur 6: Schematische weergaven van een magnetische dipool. [10]

Het gemeten veld is een combinatie van het Aardmagnetisch veld en het lokale veld. Dit lokale veld bestaat uit bijvoorbeeld de invloed van elektrische stromen en/of ferromagnetische materialen in de ruimte. Denk hier bijvoorbeeld aan de metalen onderdelen van objecten, staal constructie in de fundering en stalen kasten.

Het magnetisch inductie-veld \mathbf{B} worden bepaald als functie van positie \mathbf{r} voor een magnetische puntdipool. Een puntdipool neemt geen volume in, in tegenstelling tot de dipool in Figuur 6. Echter heeft een puntdipool wel een dipoolmoment. Het verband is opgenomen in Formule 10 [11] hieronder. Bij Formule 10 wordt er vanuit gegaan dat de dipool zich bevindt in de oorsprong van het coördinatensysteem. Het magnetisch inductie-veld is te bepalen als functie van positie volgens:

$$\mathbf{B}(\mathbf{r}) = \frac{\mu_0}{4\pi} \left[\frac{3(\mathbf{m} \cdot \mathbf{r}) \cdot \mathbf{r}}{\|\mathbf{r}\|^5} - \frac{\mathbf{m}}{\|\mathbf{r}\|^3} \right]. \quad (10)$$

Hierin is \mathbf{r} de locatie t.o.v. de oorsprong in meter en \mathbf{m} is het magnetisch dipoolmoment in ampère meter kwadraat [Am^2].

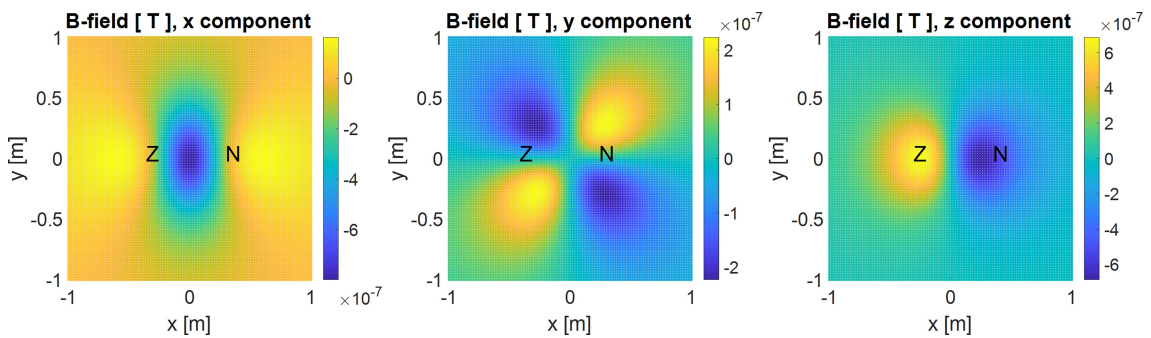
Een magnetische dipool bestaat uit zowel een noord als zuid pool met veld lijnen lopen van de noord- richting de zuid-pool. De sterkte van een magnetisch dipool als functie van positie \mathbf{r} kan bepaald worden aan de hand van Formule 10. Dit maakt het mogelijk om een magnetische kaart te maken door een superpositie van diverse magnetische dipolen. Dit is echter niet de basis van het Magnetic SLAM algoritme. Deze kennis wordt gebruikt

voor het simuleren van een magnetische kaart bestaande uit diverse punt-dipolen, om zo een beeld op te doen van een magnetische kaart.

2.3 In kaart brengen van het magnetisch veld

In deze paragraaf zal er gekeken worden naar hoe het te meten magneetveld er uit kan zien wanneer er diverse objecten met een ferromagnetische eigenschap in de ruimte aanwezig zijn. Dit zal gedaan worden door middel van een simulatie waar diverse dipolen met elk een andere sterkte gesuperponeerd worden om zo de mogelijke variatie in het lokale veld in kaart te brengen. Uiteindelijk zal het magnetisch inductie-veld in kaart worden gebracht met algoritme beschreven in Hoofdstuk 7. Deze paragraaf geeft dus alleen een beeld bij het uiterlijk van een magnetische kaart. Alle waarden en variaties zijn dus niet representatief.

Het magnetische inductie-veld van een dipool kan beschreven worden door de al eerder genoemde Formule 10. Samen met de veldvergelijkingen 1 tot en met 9 is hiermee een dipool te simuleren, zie Figuur 7. Dit geeft inzicht in wat er uiteindelijk met de Turtlebot waargenomen kan worden.



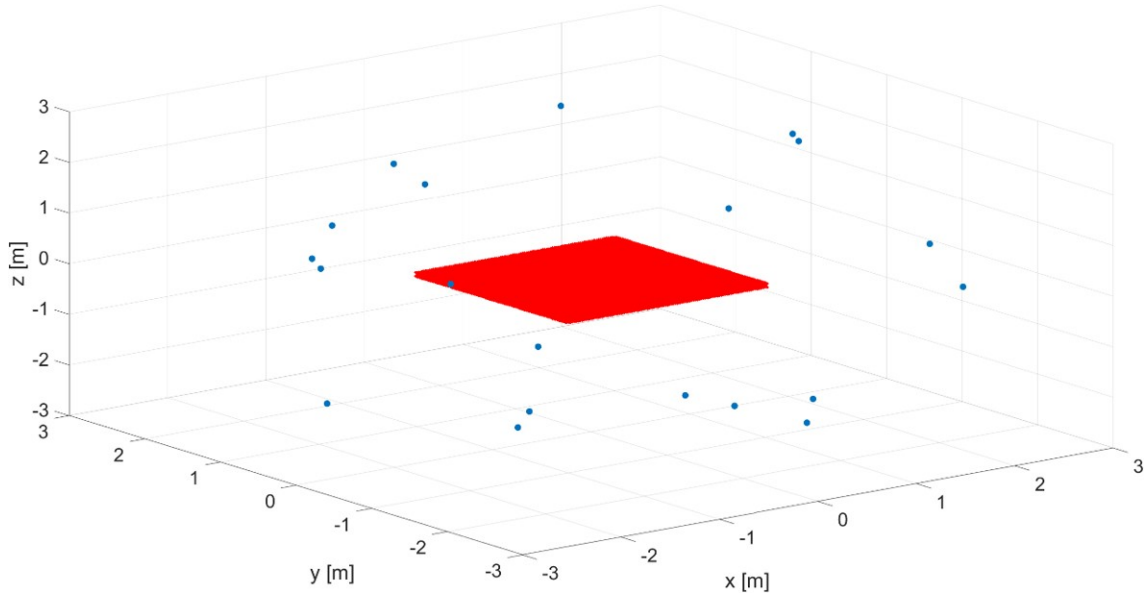
Figuur 7: De drie componenten van het magnetisch inductie-veld \mathbf{B} van een dipool uitgezet tegen de afstand in x en y richtingen in meter. Samen met de noord en zuidpool aangegeven.

Figuur 7 laat de grootte van het magnetisch inductie-veld \mathbf{B} zien voor alle drie de componenten uitgezet tegen de afstand in meters voor de x en y richting voor één dipool. Met links de x component B_x , in het midden de y component B_y en rechts de z component B_z .

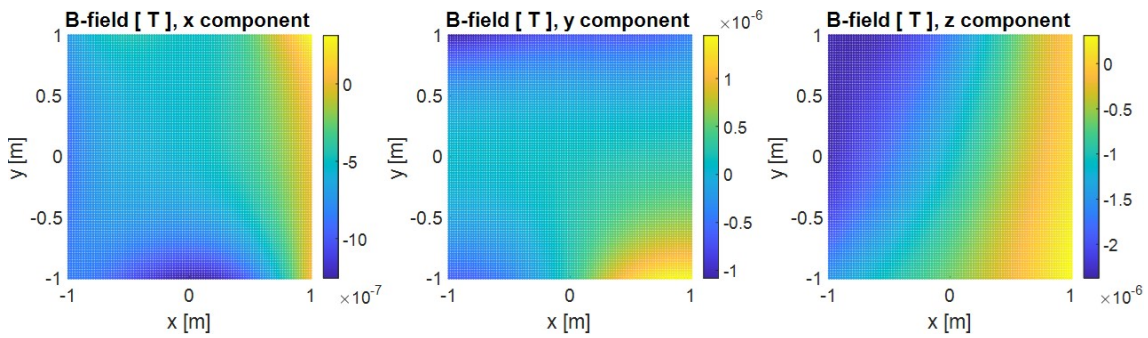
Nu bekend is hoe een gesimuleerde dipool er uit ziet, is de volgende stap om diverse dipolen in een ruimte te simuleren waarna de grootte van dit veld in kaart gebracht zal worden. Figuur 8 laat de gesimuleerde dipolen (blauwe stippen) die geplaatst zijn t.o.v. een grid (rood) zien.

Figuur 8 toont 20 dipolen met een magnetisch moment van 100 Am^2 geplaatst 3 meter van het rode vlak. Deze dipolen bevinden zich rond een opgesteld grid waarover het magnetisch inductie-veld berekend wordt. De grootte van het magnetisch inductie-veld is opgenomen in Figuur 9.

Figuur 9 geeft de grote van het magnetisch inductie-veld in tesla voor elk van de drie componenten x, y, z uitgezet tegen de afstand in x en y richting. De kleuren corresponderen met de grootte van het magnetisch inductie-veld. De kleurschaal is weergegeven aan de rechterkant van het figuur. De meeste variatie vindt plaats in de y -component. Hier varieert het veld van $-1 \cdot 10^{-6} \text{ T}$ tot $1,5 \cdot 10^{-6} \text{ T}$. Figuur 9 geeft een beeld hoe een mangetische kaart er uit zal kunnen zien. Zo wordt er verwacht dat diverse variaties in het te meten magnetisch inductie-veld zichtbaar zullen zijn afhankelijk van de omgeving. Deze variaties worden aan de hand van een kleurenschaal uitgezet tegen de afstand in x en y richting en zijn omgeving afhankelijk.



Figuur 8: Gesimuleerde dipolen (blauwe stippen) ten opzichte van een referentiekader (rode vlak).



Figuur 9: De drie componenten van het magnetisch inductie-veld B uitgezet tegen de afstand in x en y richtingen in meter.

De uiteindelijk te maken magnetische kaart zal wat weg hebben van Figuur 9. Hierin wordt de sterkte van het magnetische inductie-veld uitgezet tegen de positie in x en y richting. De sterkte van dit veld wordt door middel van een kleur verloop opgenomen. Hierdoor zijn de variaties in het magnetisch inductie-veld zichtbaar. Ook is de benodigde theorie voor het vastleggen hiervan in dit hoofdstuk besproken. Voordat het magnetisch inductie-veld in kaart kan worden gebracht dienen de benodigde sensoren onderzocht te worden, zie hiervoor volgend hoofdstuk.

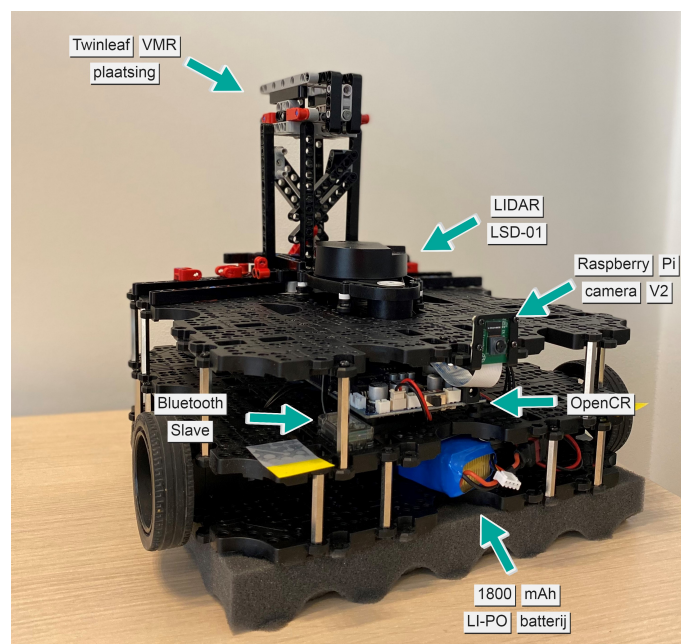
3 Turtlebot3 waffle

In dit hoofdstuk wordt de Turtlebot geïntroduceerd. De installatie van de Turtlebot, gevolgd door de gebruikte communicatie manier waarmee de Turtlebot bestuurd kan worden zijn opgenomen in Bijlage A. Tot slot wordt kort de werking van de Turtlebot binnen het zogehete 'ROS netwerk' toegelicht. Dit hoofdstuk wordt afgesloten met het opnemen van de gevoelige magneetsensor aan de Turtlebot en het Robot Operating System (ROS) netwerk.

3.1 Wat is een Turtlebot?

De Turtlebot3 waffle is het platform dat gebruikt zal worden voor het uitvoeren van SLAM. De gebruikte Turtlebot3 is weergegeven in figuur 10. Op de de Turtlebot zal ook een extra magnetische sensor geplaatst worden voor het waarnemen van het magneetveld. Naast de extra toegevoegde magnetische sensor is ook een versnellingsmeter en een gyroscoop aanwezig. De datasheet van de Turtlebot3 is opgenomen in bijlage G.

De Turtlebot draait op het Robot Operating System (ROS). ROS is een open source software development kit voor het aansturen van diverse robots. Het grote voordeel van ROS is dat het gebruikt wordt tijdens de ontwikkelfase van de robot tot aan de productie [12].



Figuur 10: De gebruikte Turtlebot 3 waffle met de zichtbare onderdelen benadrukt.

3.2 Topics, Subscribers en Publishers

Een topic bevat de data van een bepaalde sensor. Voor de LIDAR is bijvoorbeeld het topic /scan aanwezig. Elke sensor heeft zijn eigen topic met een unieke naam, en deze is uit te lezen met een 'Subscriber'.

Een subscriber is een ROS node, een python script, dat abonneert op de topic. Hierdoor kan een subscriber de message ontvangen wanneer er een message verzonden wordt. Oftewel een subscriber abonneert zich op een topic waarna de data van deze topic verder uit te pakken en verwerken is.

Ook is het mogelijk om data te sturen naar de Turtlebot. Hiermee is het bijvoorbeeld mogelijk om de robot vooraf bepaald pad te laten rijden. Het sturen van data naar de

Turtlebot gaat door middel van 'publishers'. In beide gevallen is het noodzakelijk dat het gewenste topic opgenomen wordt in de subscriber of publisher waarna de data uit te lezen of te verzenden is. Een overzicht van de gebruikte topics staat in Tabel 1.

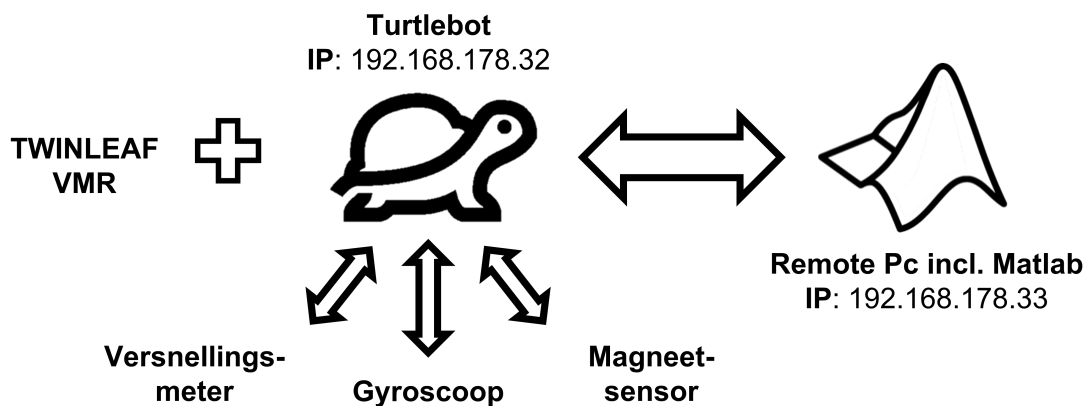
Tabel 1: *Gebruikte topics met korte toelichting*

Topic	Toelichting
/scan	LIDAR data
/odom	Positie en oriëntatie inschatting van de Robot
/IMU	Inertial Measurement Unit, diverse sensoren
/magneto	Data toegevoegde magneetsensor
/cmd_vel	Snelheid Turtlebot

3.3 Communiceren met de Turtlebot

Het aansturen van de Turtlebot is onder andere uit te voeren met de hulp van MATLAB. Hiervoor is een speciale toolbox beschikbaar genaamd: ROS Toolbox. Deze toolbox werkt met zowel ROS als ROS 2. Hierbij is het belangrijk dat de compatibiliteit tussen de ROS versie en de MATLAB versie gecontroleerd wordt. In dit geval is ROS Noetic gebruikt in combinatie met MATLAB R2021b. Een lijst met veel gebruikte commando's samen met een toelichting is opgenomen in Bijlage F

MATLAB kan met behulp van een master slave systeem verbinden met de Turtlebot waar ROS op draait. Op deze manier is de Turtlebot vanaf een remote PC te besturen en kan de data gelijk worden verwerkt. Voor de installatie van ROS is het stappenplan Robotis gevolgd [13].



Figuur 11: *Illustratie ter verduidelijking.*

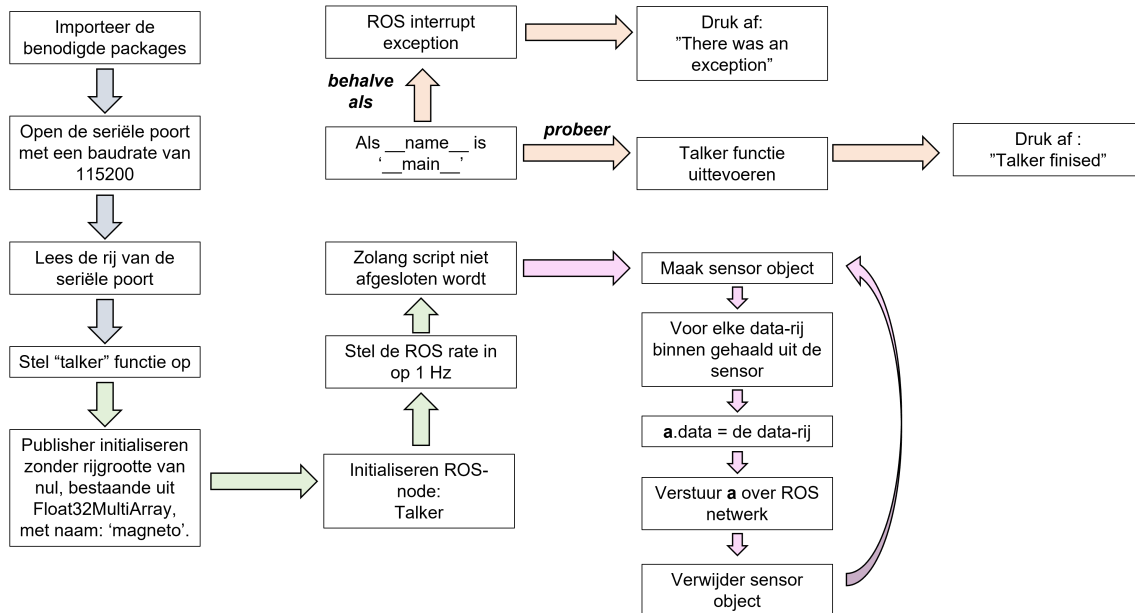
3.4 Toevoegen TIWNLEAF VMR

Deze paragraaf beschrijft de methode voor het toevoegen van de Twinleaf VMR. Door middel van een ROS node is het mogelijk om de seriële poort van de Raspberry pi, aanwezig op de Turtlebot, uit te lezen en deze data te verzenden over het ROS netwerk. Vervolgens zal worden toegelicht hoe deze ROS node te activeren is, waarna overige belangrijke informatie wordt toegelicht.

Om het magneetveld te meten wordt gebruik gemaakt van de Twinleaf VMR, zie Hoofdstuk 4.5. De sensor is geplaatst bovenop de Turtlebot en via USB aangesloten op de Raspberry Pi die aanwezig is op de Turtlebot. Door middel van een ROS Node, een python script, wordt de sensor uitgelezen en in een topic gestopt. Waarna door middel van een subscriber in MATLAB deze data wordt ingelezen. Het gebruikte python script is

een serial readout script in combinatie met een ROS publisher. De code is opgenomen in Bijlage J.

Het commando zorgt er voor dat de data van de magneetsensor 'gepublisht' wordt over de `magneto` topic. Door binnen MATLAB hierop te 'subscriben' kan de data worden ingeladen. Figuur 12 laat het blokschema van de code zien.



Figuur 12: *Blokschema van read_magneto.py.*

Als eerst worden de benodigde packages ingelezen zoals Figuur 12 laat zien. Naast een standaard package worden ook `tldevice` en `rospy` geïmporteerd. Deze packages zijn te downloaden via [14] en [15]. Vervolgens wordt de seriële poort geopend met een baudrate van 115200, waarna de eerste regel van de seriële poort wordt ingelezen.

Vervolgens wordt er een functie gemaakt die zorgt voor het verwerken en publiceren van de sensordata. Als eerst wordt er een ROS publisher geïntialiseerd onder de naam `magneto`, met een wachtrij grootte van 0 bestaande uit het data type `Float32 Multi Array`.

Vervolgens wordt de ROS standaard node "talker" wordt geïntialiseerd en de ROS rate wordt ingesteld op 1 Hz. Vervolgens wordt er een while-loop gestart die afgebroken wordt wanneer de het script wordt afgesloten. In de while-loop wordt een sensor-object aangeemaakt waarna voor elke data-rij binnen gehaald door het sensor-object, de data wordt gekoppeld aan een variabele die vervolgens wordt gepublicht over het ROS netwerk, gevolgd door het verwijderen van het sensor-object en het opnieuw beginnen van de while-loop.

Ook wordt er gecontroleerd of de `__name__` gelijk is aan `__main__`. Is dit het geval dan wordt er geprobeerd om de talker functie uit te voeren en wordt er naar afloop afgesloten met de tekst: "Talker finished" behalve als er een ROS interrupt exception voordoet waarna de tekst: "There was an exception" wordt afgebeeld in de terminal.

De magneetsensor is boven de Turtlebot geplaatst op 13 cm vanaf het bovenste platform. De VMR wordt door middel van Lego geplaatst boven de Turtlebot. Er is voor Lego gekozen omdat dit een non-ferromagnetisch materiaal is, waardoor er geen extra verstoringen in het te meten magneetveld worden geïntroduceerd. Ook biedt dit de mogelijkheid om snel en eenvoudig andere sensoren op te nemen in dien gewenst.

Met de nu opgenomen Twinleaf VMR-magneetveld sensor is het platform voor het uitvoeren van MSLAM compleet. Door gebruik te maken van de Twinleaf VMR kan het

magneetveld nu in kaart worden gebracht. Voordat het MSLAM algoritme uitgevoerd kan worden dient de theorie met betrekking tot magnetostatica nog behandeld te worden. Volgend hoofdstuk beschrijft deze theorie.

4 Turtlebot & Sensoren

In dit hoofdstuk worden de sensoren besproken en geanalyseerd. Als eerst wordt er een kort overzicht gegeven van de sensoren aanwezig op de Turtlebot. Vervolgens wordt de karakterisatiemethode toegelicht waarna elke sensor in een aparte paragraaf volledig besproken wordt. Denk hierbij aan de werking van de sensor, specificaties, validatie metingen, karakterisatie matrixen en waar nodig extra informatie over de sensor.

4.1 Turtlebot

De Turtlebot beschikt over een aantal sensoren. Deze bevinden zich op de OpenCR board van de Turtlebot. Door middel van de al eerder benoemde topics zijn deze sensoren uit te lezen uit het ROS netwerk, waarna hun data verwerkt kan worden.

Tabel 2: *Overzicht aanwezige sensoren op de Turtlebot3. Met hierbij hun grootheid, eenheid, of ze gebruikt worden voor MSLAM of LSLAM en de bijbehorende topic om ze uit te lezen.*

Sensor	Grootheid	Eenheid	LSLAM of MSLAM?	Topic
Gyroscoop (3 axis)	Oriëntatie	Quaternionen	MSLAM	\imu
Versnellingsmeter (3 axis)	Versnelling	[m/s ²]	-	\imu
Magneetveld sensor (3 axis)	Magnetisch inductieveld	[Tesla]	-	\magnetic_field
LIDAR	Afstand, Intensiteit	[meter], [W]	LSLAM	\scan

Een overzicht van de aanwezige sensoren op de OpenCR 1.0 is opgenomen in Tabel 2. De OpenCR 1.0 is onderdeel van de gebruikte Turtlebot3 waffle. In het overzicht is naast de sensor ook hun grootheid en eenheid opgenomen samen met een indicatie of de sensor gebruikt wordt voor LIDAR of Magnetic SLAM en de bijbehorende topic om deze uit te lezen.

De komende paragrafen analyseren de kwaliteit en het gedrag van de sensoren. Als eerst zal besproken worden hoe de sensoren gekarakteriseerd worden. Dit is van belang aangezien het Kalman filter, zie Bijlage D, uitgaat van Gaussische verdeelde ruis waarna voor elk van de sensoren, in een aparte paragraaf, de specificaties opgenomen, ruis in kaart wordt gebracht en geanalyseerd gevolgd door het opnemen van de karakterisatie matrixen.

4.2 Karakterisering van de sensoren

Voor elke sensor wordt in een statische omgeving de resultaten van een meting met een tijdsduur van 10 minuten in kaart gebracht. De histogrammen die hierbij opgenomen worden geven een beeld of de ruis redelijkerwijs normaal/Gaussisch verdeeld is.

Op basis van deze data is het mogelijk om een Gaussische verdeling te fitten op de data. Dit doen we door het empirische gemiddelde en de empirische variantie voor elke data reeks te bepalen [16]. Het empirisch gemiddeld $\hat{\mu}$ en variantie $\hat{\Sigma}$ van een data reeks

$\mathbf{X} = (\mathbf{x}_1, \mathbf{x}_2, \dots, \mathbf{x}_n)$ (met $\mathbf{x}_i \in \mathbb{R}^n$ de i -de sample) zijn gedefinieerd door [17]:

$$\hat{\mu} := \frac{1}{N} \sum_{i=1}^N \mathbf{x}_i, \quad (11)$$

$$\hat{\Sigma} := \frac{1}{N} \sum_{i=1}^N (\mathbf{x}_i - \hat{\mu})(\mathbf{x}_i - \hat{\mu})^T. \quad (12)$$

Op basis van het empirische gemiddelde en empirische variantie mogen we nu bij benadering schrijven

$$X \sim \mathcal{N}(\hat{\mu}, \hat{\Sigma}),$$

hetgeen betekent dat X normaal verdeeld is met gemiddelde $\hat{\mu}$ en variantie $\hat{\Sigma}$. In het bijzonder is $\hat{\mu}$ op te vatten als de bias in een sensor. We zullen zien dat het voor de implementatie van het Magnetic SLAM algoritme belangrijk is dat de onzekerheden in het systeem Gaussisch verdeeld zijn. De karakterisering van deze onzekerheden worden meegenomen in het Kalman filter om zo rekening te houden met de onnauwkeurigheden in een sensor.

4.3 Gyroscop

Deze paragraaf behandelt de specificaties van de gyroscop gevolgd door een toelichting over de eenheid Quaternionen. Tot slot zal het gedrag van de gyroscop geanalyseerd worden met afsluitend de karakteriseringsmatrix.

4.3.1 Specificaties

Op de OpenCR 1.0 is een gyroscop met 3 assen aanwezig. Dit houdt in dat er voor 3 componenten een hoekversnelling gemeten wordt. In dit geval gaat het hier om de x , y en z -component. De gyroscop is onderdeel van de MPU-9250 IMU. Deze gyroscop heeft een programmeerbaar bereik van ± 250 , ± 500 , ± 1000 , ± 2000 $^\circ/sec$ [18].

De Gyroscop heeft een temperatuur afhankelijke gevoeligheid. Bij een temperatuur van $25^\circ C$ is de gevoeligheid $\pm 3\%$. In een temperatuur bereik van $-40^\circ C$ tot en met $+84^\circ C$ neemt de gevoeligheid iets toe en is dit $\pm 4\%$. De volledige tabel met specificaties is opgenomen in Bijlage B.1. [18]

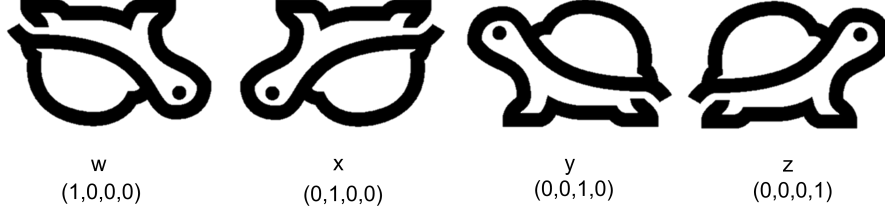
Hieruit is dus te concluderen dat de Turtlebot de hoekversnelling gemeten door de gyroscop omrekent naar quaternionen. Het omzetten van de gemeten hoekversnelling quaternionen gebeurt in een 'Black Box'. Er is niet bekend hoe deze vertaalslag wordt gemaakt. Vermoedelijk wordt dit gedaan door de gemeten versnelling twee maal te integreren om zo tot een hoek te komen, die vervolgens wordt omgezet naar quaternionen.

Quaternionen is een veel gebruikte beschrijving voor het weergegeven van de oriëntatie. Echter is deze eenheid niet heel bekend. Paragraaf 4.3.2 legt deze eenheid toe. Het nog te bespreken Magnetic SLAM (MSLAM) algoritme maakt hier nadrukkelijk gebruik van.

4.3.2 Quaternionen

Euler hoeken is een term om de driedimensionale rotatie met de die bijbehorende hoeken weer te geven om zo de oriëntatie van een object vast te leggen [19]. Net als Euler hoeken zijn quaternionen een manier om de oriëntatie van een object weer te geven. Bij Euler hoeken maakt de volgorde van rotatie uit. De uitkomst van de oriëntatie is afhankelijk van de volgorde van de as waar om geroteerd wordt. Roteer je een object eerst om de x -as en daarna om de y -as dan krijg je, wanneer er gebruik wordt gemaakt van Euler hoeken, een ander resultaat dan weer je eerst om de y -as roteert en dan om de x -as.

Bij quaternionen speelt dit geen rol. Quaternionen worden uitgedrukt in 4 dimensies gelabeld door x, y, z en w . Belangrijk hierbij is dat x, y, z niet representatief zijn voor één van de assen in het globale referentiekader. Elk van de 4 componenten zijn even belangrijk voor het weergegeven van de oriëntatie.



Figuur 13: *Oriëntatie van een schildpad aan de hand van quaternionen.*

Figuur 13 laat zien dat elke component op zichzelf een onafhankelijke oriëntatie representeert. De waarde 1 geeft 100 procent gewicht aan de oriëntatie waardoor, zoals te zien is in Figuur 13, de oriëntatie compleet in één van de vier richtingen verandert. Quaternionen kunnen dan ook enkel alleen waardes tussen -1 en 1 aannemen. Een min teken in de oriëntatie duidt op rotatie van het voorwerp in de tegengestelde richting dan waar de oriëntatie heen wees.

Een quaternion is een lineaire combinatie van een reëel deel en het “imaginaire” deel:

$$q = w + xi + yj + zk, \quad (13)$$

waarbij $\text{Re}(q) = w$ en $\text{Im}(q) = xi + yj + zk$. Hierin zijn i, j, k eenheden aan die voldoen aan de volgende relaties:

- $i^2 = j^2 = k^2 = -1$;
- $ij = -ji = k$
- $jk = -kj = i$
- $ki = -ik = j$

Quaternionen zijn op een unieke manier om te schrijven [20] naar een drie-dimensionale rotatie matrix die de oriëntatie van de Turtlebot beschrijft middels

$$R(q) = \begin{bmatrix} 2(w^2 + x^2) - 1 & 2(xy - wz) & 2(xz + wy) \\ 2(xy + wz) & 2(w^2 + y^2) - 1 & 2(yz - wx) \\ 2(xz + wy) & 2(yz + wx) & 2(w^2 + z^2) - 1 \end{bmatrix}. \quad (14)$$

Ook is het met quaternionen eenvoudig om de coördinaten van een punt te berekenen na een rotatie van θ . Neem punt $P = (m, n, o)$ dat gedraaid gaat worden over θ op het draaipunt $T = (a, b, c)$. Eerst dient hiervoor de quaternion te worden opgesteld waar over het punt P gedraaid gaat worden. Deze quaternion q is gedefinieerd als Formule 15

$$q = \cos(\theta/2) + \sin(\theta/2)(ti + uj + vk). \quad (15)$$

De quaternion q bestaat uit een eenheidsvector zodat $t^2 + u^2 + v^2 = 1$ en de hoek θ waarover gedraaid gaat worden. Vervolgens dient punt P als quaternion gedefinieerd te worden volgens $P = (m - a)i + (n - b)j + (o - c)k$. Waarna het geroteerde punt P' bepaald kan worden aan de hand van het product qpq^{-1} . Dit resulteert in: [21]

$$qpq^{-1} = w' + m'i + n'j + o'k. \quad (16)$$

Het vermenigvuldigen van quaternionen gaat volgens het zogeheten 'Hamilton product'. Het Hamilton product wordt bepaald aan de hand van het product van de basis elementen en hun distributieve eigenschap. De distributieve eigenschap houdt in dat het volgende verband geldt: $x \cdot (y + z) = x \cdot y + x \cdot z$ [22]. Het Hamilton product van twee quaternionen $q_1 = w_1 + x_1i + y_1j + z_1k$ en $q_2 = w_2 + x_2i + y_2j + z_2k$ is opgenomen in Formule 17 [23]

$$\begin{aligned} (q_1)(q_2) = & a_1a_2 + a_1b_2\mathbf{i} + a_1c_2\mathbf{j} + a_1d_2\mathbf{k} \\ & + b_1a_2\mathbf{i} + b_1b_2\mathbf{i}^2 + b_1c_2\mathbf{ij} + b_1d_2\mathbf{ik} \\ & + c_1a_2\mathbf{i} + c_1b_2\mathbf{ji} + c_1c_2\mathbf{j}^2 + c_1d_2\mathbf{jk} \\ & + d_1a_2\mathbf{k} + d_1b_2\mathbf{ki} + d_1c_2\mathbf{kj} + d_1d_2\mathbf{k}^2. \end{aligned} \quad (17)$$

Substitutie van de relaties tussen de eenheden i, j, k resulteert in Formule 18. Het product van de twee quaternionen kan gezien worden als een rotatie van q_2 gevolgd door een rotatie van q_1 .

$$\begin{aligned} (q_1)(q_2) = & a_1a_2 - b_1b_2 - c_1c_2 - d_1d_2 \\ & + (a_1b_2 + b_1a_2 + c_1d_2 + d_1c_2)\mathbf{i} \\ & + (a_1c_2 - b_1d_2 - c_1a_2 - d_1b_2)\mathbf{j} \\ & + (a_1d_2 + b_1c_2 + c_1b_2 + d_1a_2)\mathbf{k} \end{aligned} \quad (18)$$

Quaternionen is dus een manier om de oriëntatie van een voorwerp in drie dimensies weer te geven zonder de beperkingen van Euler hoeken. Nu bekend is hoe quaternionen werken is de volgende stap kijken naar de ruis in de gyroscoop. [24]

4.3.3 Analyse gedrag en karakterisering

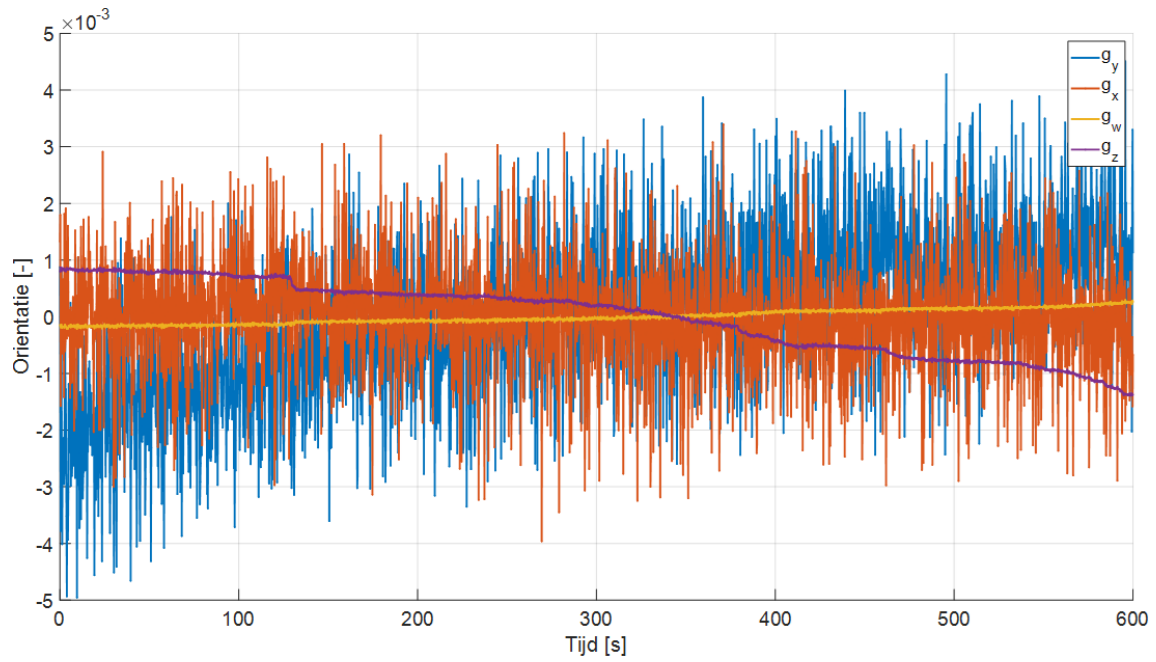
De ruis in wordt in kaart gebracht door gedurende 10 minuten de data vast te leggen waarbij de Turtlebot niet beweegt. De ruis in de gyroscoop data is bepaald door het gemiddelde van de oriëntatie te bepalen en deze van het gemeten signaal af te halen. Hiermee is de afwijking ten opzichte van het gemiddelde, de ruis, te bepalen. De ruis is weergegeven in Figuur 14 met de x y z en w -component in het geel, rood, blauw en paars uit gezet tegen de tijd in seconde. Hierbij corresponderen de x y z en w -component tot de quaternionen notatie.

Zoals te zien is in Figuur 14 dat er bij het stil staat voornamelijk ruis aanwezig is bij het bepalen van de oriëntatie van de x en y -component. Bij de x -component vindt een variatie van maximaal 0,0074 plaats voor de y -component is dit 0,0095. De oriëntatie van de z en w -component blijven minder constant over de tijd. Hoewel de x - en y - component het meeste ruis bevatten, blijven deze wel redelijk constant. De w -component van de oriëntatie weergegeven door quaternionen blijft vrij stabiel en bevat de minste ruis. De maximale variatie over de tijd heen van de w -component is slechts $5,1 \cdot 10^{-4}$.

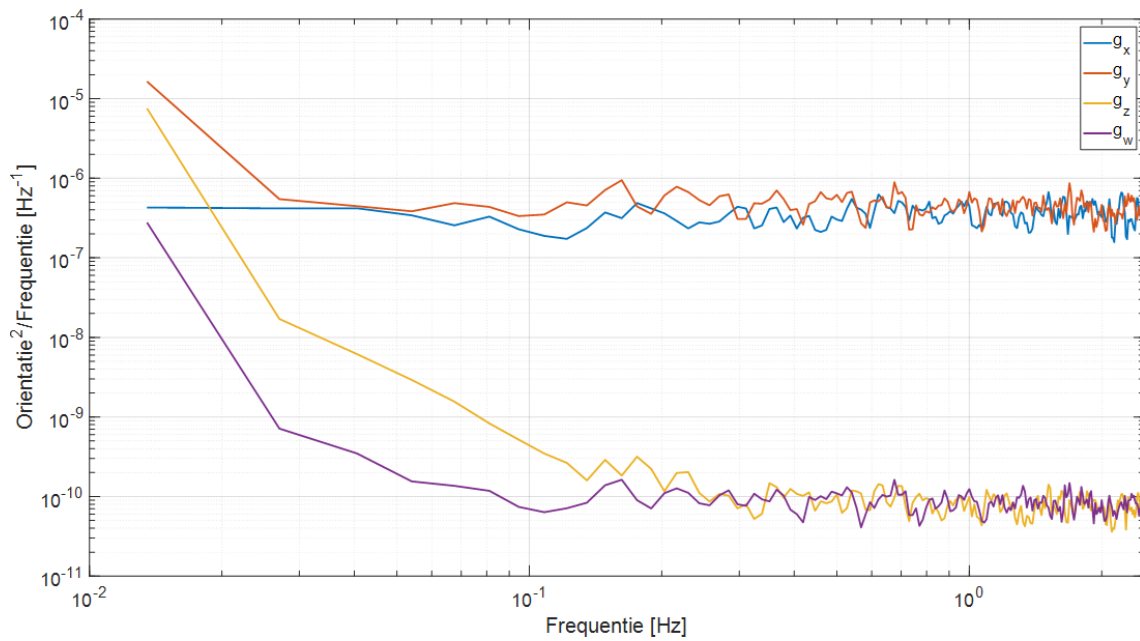
De z -component varieert over de tijd het meest. In het begin wordt er een waarde van ongeveer $1 \cdot 10^{-3}$ waargenomen waarna deze afneemt naar $-1,2 \cdot 10^{-3}$. Dit zou betekenen dat de oriëntatie van de Turtlebot veranderd is. Echter is dit niet geval en wordt deze afname veroorzaakt door de drift in de sensor. Er wordt verwacht dat wanneer de Turtlebot actief een pad rijdt deze drift minder aanwezig is.

Ook is voor de gyroscoop data de amplitude spectrum dichtheid bepaald. Deze is opgenomen in Figuur 15 waarbij Welch's methode is toegepast [25]. In Figuur 15 is het vermogen van het signaal uitgezet bij verschillende frequenties.

Figuur 15 laat zien dat voor de z en w -component de ruisvloer na 10^{-1} Hz bereikt wordt. De afname hiernaar toe is redelijk exponentiële zoals een ideale sensor hoort te beschrijven.



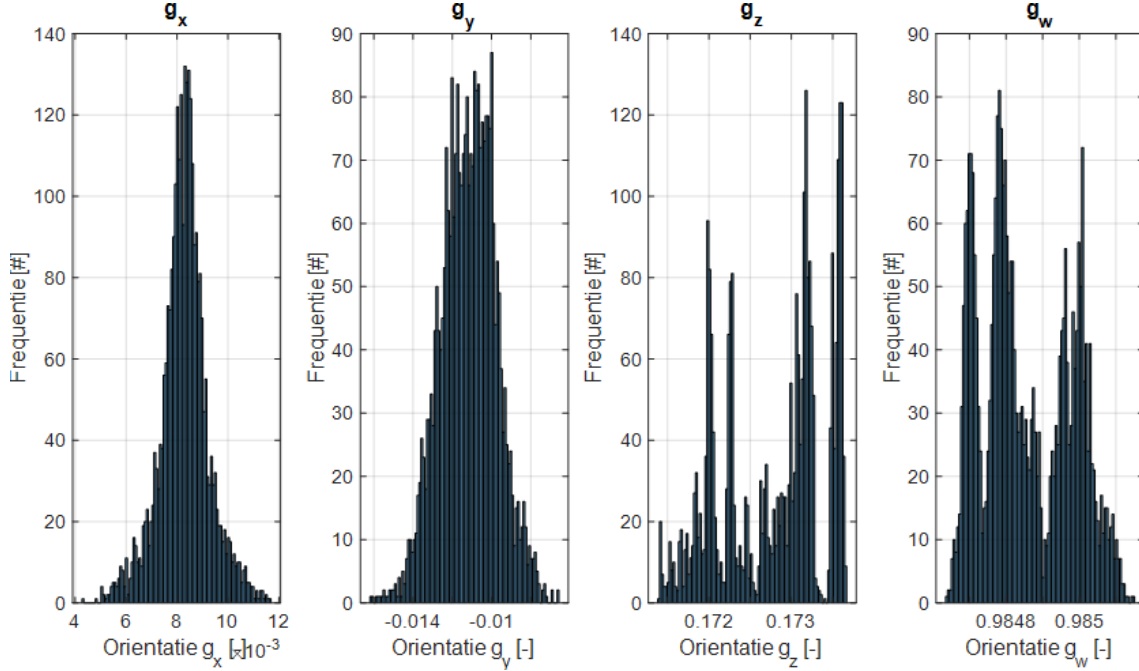
Figuur 14: Oriëntatie in quaternionen voor de x, y, z, w component min de gemiddelde waarde van de oriëntatie uitgezet tegen de tijd in secondes waarbij de Turtlebot 10 minuten stil staat.



Figuur 15: Spectrale dichtheid van de oriëntatie voor elk van de componenten (g_x, g_y, g_z, g_w)

De x -component is over het gehele frequentie bereik constant. Dit laat zien dat er alleen maar ruis gemeten wordt. Ook de y -component raakt de ruisvloer al snel. Echter is hier nog wel wat signaal aanwezig aangezien de ruisvloer niet direct geraakt wordt.

De vorm van de ruis wordt bekeken door het maken van een histogram. Figuur 16 laat de vorm van het signaal zien voor elk van de drie componenten. Hierbij is een bin grootte gebruikt van 100 om zo alle gemeten waardes die de oriëntatie aan heeft genomen te visualiseren. Waarna er iets over de vorm van de ruis gezegd kan worden.



Figuur 16: Histogram met de oriëntatie voor alle vier de componenten. Ter verduidelijking van de vorm van het signaal.

In Figuur 16 is te zien dat de x en y -component duidelijk een Gaussische vorm aannemen. de y -component bevat zichtbaar een minder sterke Gaussische vorm. De z en w -component nemen duidelijk geen Gaussische vorm aan. Door de drift aanwezig in de sensor loopt het signaal weg van de 'echte' oriëntatie waardoor er dus foutieve metingen worden verricht. Deze foutieve metingen zijn terug te zien in het histogram waardoor er deze componenten geen Gaussische vorm beschrijven.

Aan de hand van de meting is het gemiddelde van de variantie van de ruis te bepalen volgens Hoofdstuk 4.2. Het gemiddelde $\hat{\mu}$ en de variantie $\hat{\Sigma}$ zijn opgenomen als volgt:

$$\hat{\mu} = \begin{bmatrix} 0,0083 \\ -0,0112 \\ 0,1728 \\ 0,9849 \end{bmatrix} \quad \hat{\Sigma} = \begin{bmatrix} 9,1049 \cdot 10^{-7} & -7,0217 \cdot 10^{-8} & 4,6059 \cdot 10^{-8} & -1,6503 \cdot 10^{-8} \\ -7,0217 \cdot 10^{-8} & 1,9834 \cdot 10^{-6} & -5,5327 \cdot 10^{-7} & 1,2024 \cdot 10^{-7} \\ 4,6059 \cdot 10^{-8} & -5,5327 \cdot 10^{-7} & 3,9782 \cdot 10^{-7} & -7,6456 \cdot 10^{-8} \\ -1,6503 \cdot 10^{-8} & 1,2024 \cdot 10^{-7} & -7,6456 \cdot 10^{-8} & 1,4919 \cdot 10^{-8} \end{bmatrix}$$

Deze karakteriseringingen zijn belangrijk voor het Magnetic SLAM algoritme. Het Extended Kalman Filter in het Magnetic SLAM algoritme zal gebruik maken van deze onzekerheden om zo rekening te houden met de onnauwkeurigheid in de sensor.

4.4 Versnellingsmeter

De versnellingsmeter wordt niet direct gebruikt bij LSLAM of MSLAM. Wel kan voor het MSLAM algoritme gebruikt gemaakt worden van de odometry. Dit is een combinatie

van alle aanwezige bewegingssensoren op de Turtlebot [26]. Het karakteriseren van de versnellingsmeter is daarom alsnog interessant. Als eerst zullen de specificaties van de versnellingsmeter toegelicht worden gevolgd door de analyse van het gedrag van de sensor.

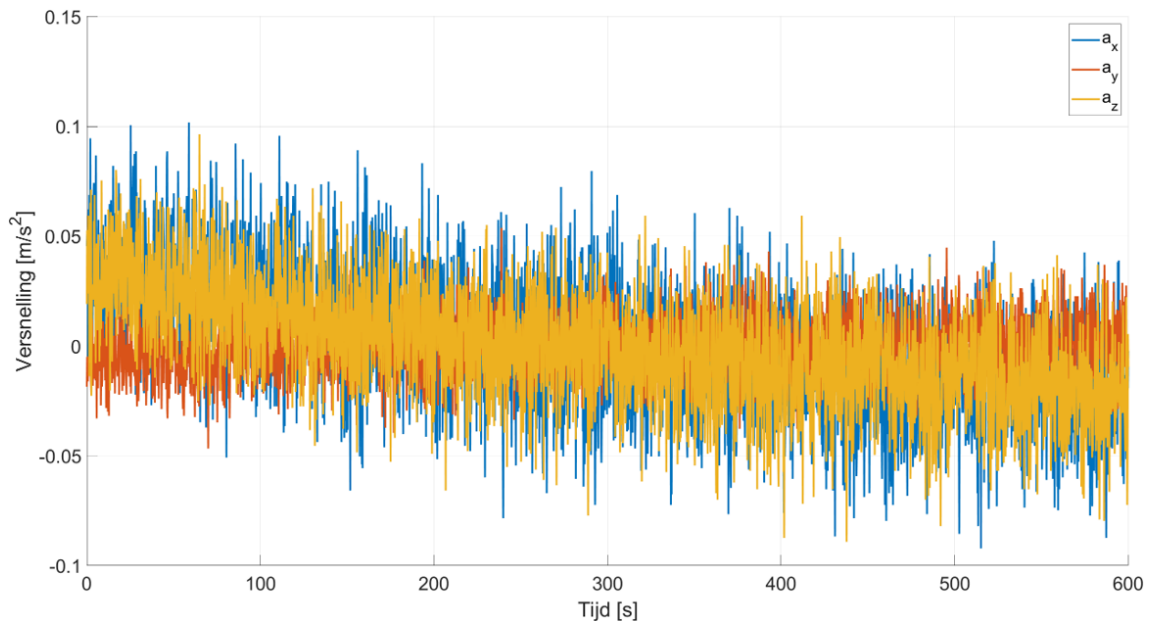
4.4.1 Specificaties

De versnellingsmeter heeft drie assen. Deze meet de versnelling voor de x , y en z -component. De componenten a_x , a_y en a_z worden weergegeven in de SI-eenheid m/s^2 vermoedelijk na een transformatie van een ROS node. De volledige specificaties tabel is opgenomen in Bijlage B.3.

De versnellingsmeter is ook onderdeel van de MPU-9250 IMU aanwezig op de OpenCR 1.0. Deze versnellingsmeter heeft een instelbaar bereik variërend van $\pm 2g$, $\pm 4g$, $\pm 8g$, $\pm 16g$. Ook de versnellingsmeter data zal dus waarschijnlijk binnen ROS om worden gezet naar de SI-eenheid m/s^2 . In het bereik van $\pm 2g$ heeft de versnellingsmeter een temperatuur afhankelijke gevoeligheid gedefinieerd als $\pm 0,026 \text{ } \%/^{\circ}C$ in een temperatuurbereik van $-40^{\circ}C$ tot $+85^{\circ}C$. [18]

4.4.2 Analyse gedrag en karakterisering

Om de ruis in de versnellingsmeter in kaart te brengen is gedurende 10 minuten de versnelling data vast gelegd. Figuur 17 laat de gemeten versnelling zien uitgezet tegen de tijd. Hierin is a_x opgenomen in het blauw, a_y in het oranje en a_z in het rood. Hierbij is het gemiddelde van elke dataset van de meting afgetrokken om zo de variatie in het te meten signaal beter in kaart te brengen.



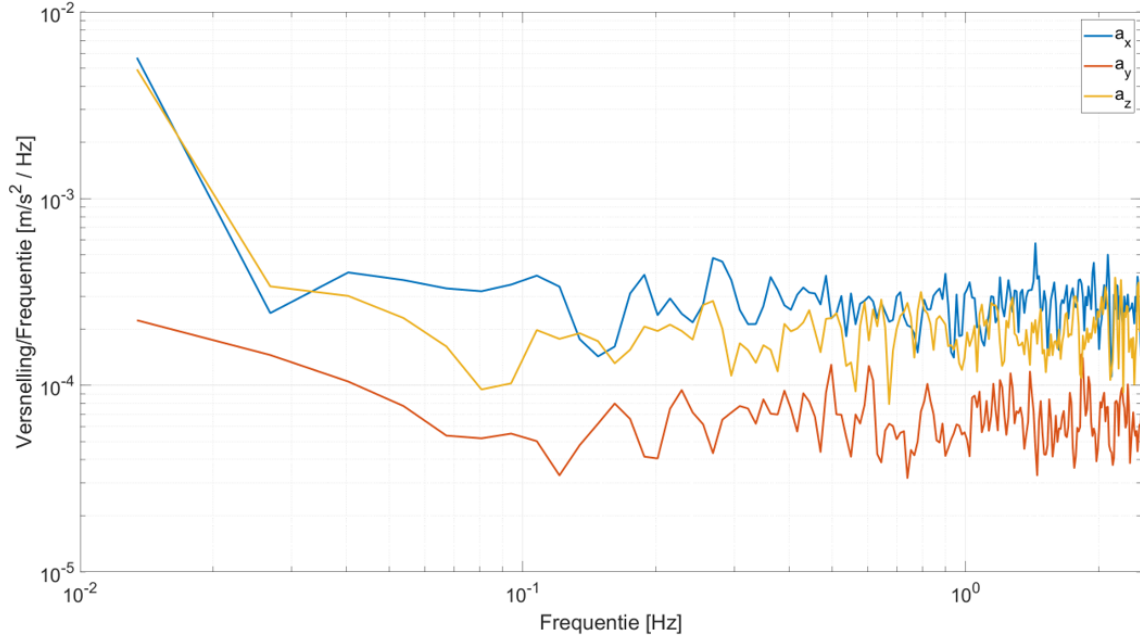
Figuur 17: Gemeten versnelling voor de x, y, z richting min de gemiddelde waarde van de versnelling uitgezet tegen de tijd in seconde waarbij de Turtlebot 10 minuten stil staat.

Figuur 17 laat zien dat er ondanks het corrigeren met het gemiddelde van de data er alsnog veel variatie aanwezig is in de gemeten versnelling. De grootste variatie vindt plaats in de x -component van de versnelling en is gelijk aan $0,1939 \text{ m/s}^2$. Voor de y - en z -component is dit respectievelijk $0,1006$ en $0,1856 \text{ m/s}^2$.

De bijbehorende amplitude spectrum dichtheid is opgenomen in Figuur 18. Hierbij is gebruik gemaakt van Welchs methode, waarbij ook hier de gemiddelde waarde van de metingen is afgehaald. Figuur 18 geeft het vermogen van het signaal, de gemeten versnelling,

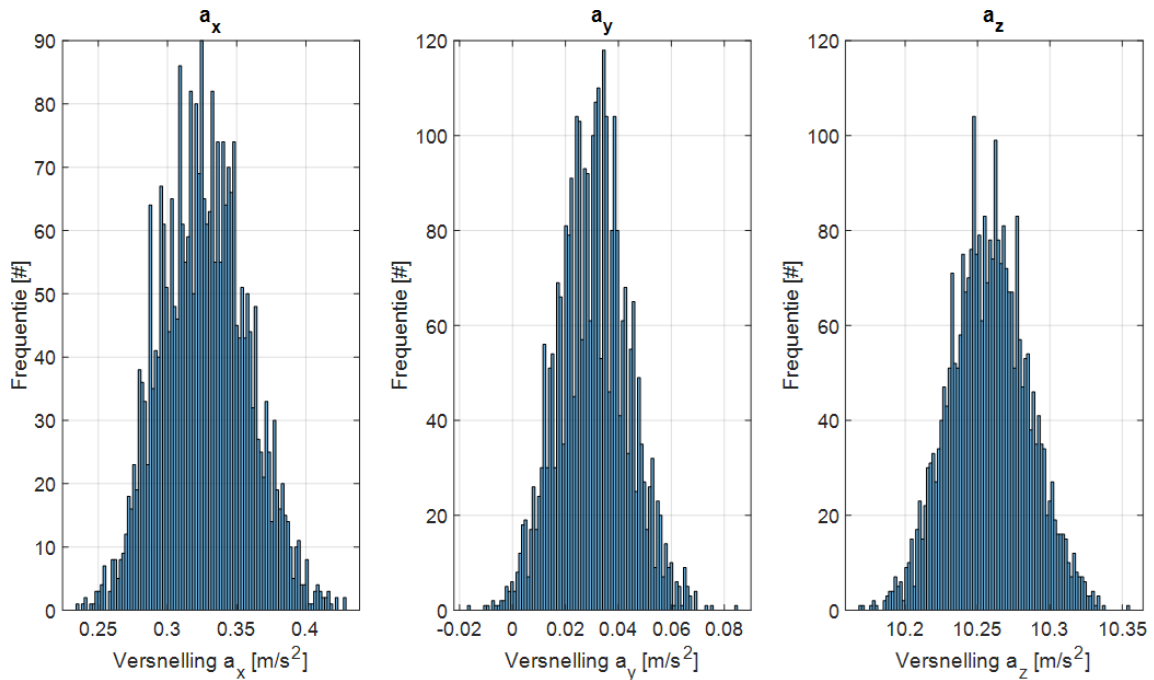
bij verschillende frequenties.

Figuur 18 laat zien bij een frequentie van 10^{-1} Hz de ruis vloer wordt bereikt. Hiervoor neemt de amplitude af. In dit gedeelte is het signaal aanwezig. Echter is ook duidelijk te zien dat het geen ideale sensor is. In het ideale geval zal het signaal exponentieel aflopen tot de ruisvloer bereikt wordt. Echter is dit duidelijk niet het geval.



Figuur 18: *Spectrale dichtheid van de versnelling voor elk van de componenten (a_x , a_y , a_z)*

Om een idee te krijgen van de vorm van het signaal is een histogram opgesteld. Hieruit zal te bepalen zijn of de ruis een Gaussische vorm aan neemt en wat de mogelijk bias in het signaal kan zijn. Figuur 19 laat de verdeling van de gemeten versnellingen zien voor elk van de drie componenten.



Figuur 19: *Histogram met versnelling in m/s^2 voor alle drie de componenten. Ter verduidelijking van de vorm van het signaal.*

Uit Figuur 19 is te halen dat de verdeling een Gaussische vorm aanneemt. Echter vindt er bij alle drie de componenten een bias plaats. De grootste bias is aanwezig bij de z-component. Dit is het gevolg van de aanwezige valversnelling op aarde. Echter wijkt de gemeten valversnelling wel af van de aanwezige valversnelling op aarde. Bij de x- en y-component is een bias aanwezig van 0,3237 en 0,034 m/s² samen met een standaard deviatie van $\pm 0,0255$ en $\pm 0,0130$ respectievelijk. Het gemiddelde $\hat{\mu}$ is hieronder opgenomen en is opgenomen voor elk van de drie componenten x, y, z . De bijbehorende variantie matrix $\hat{\Sigma}$ is hierbij ook zichtbaar.

$$\hat{\mu} = \begin{bmatrix} 0,2555 \\ 0,1284 \\ 10,2704 \end{bmatrix} \quad \hat{\Sigma} = \begin{bmatrix} 6,4969 \cdot 10^{-4} & -4,4467 \cdot 10^{-5} & 3,5115 \cdot 10^{-4} \\ -4,4467 \cdot 10^{-4} & 1,6794 \cdot 10^{-4} & -3,0176 \cdot 10^{-5} \\ 3,5115 \cdot 10^{-4} & -3,0176 \cdot 10^{-5} & 0,0010 \end{bmatrix}$$

4.5 VMR

Zoals al eerder benoemd beschikt de Turtlebot van zichzelf als over een magneetveld sensor. Echter zal deze niet gebruikt worden tijdens het uitvoeren van het Magnetic SLAM-algoritme. Hiervoor zal een externe sensor worden toegevoegd aan de Turtlebot. In dit geval is dat de Twinleaf VMR. Het toevoegen van de VMR is toegelicht in Hoofdstuk 3.4. Deze paragraaf geeft een korte toelichting van deze sensor samen met de technische specificaties van de VMR. Hierna zal de kwaliteit van de VMR en de overige gebruikte sensoren onderzocht worden.

4.5.1 Magnetoresistiviteit

De Twinleaf VMR is een magnetoresistieve vector magnetometer. Dit betekent dat de werking van de VMR gebaseerd is op de magnetoresistieve eigenschappen van een materiaal waarmee het magnetisch inductieveld voor de drie componenten $\mathbf{B} = (B_x, B_y, B_z)$ gemeten wordt.

Magnetoresistance is de neiging van een materiaal, vaak ferromagnetisch, om een verandering in elektrische weerstand te ondergaan wanneer er een extern magnetisch inductie veld wordt aangelegd. Deze verandering in elektrische weerstand is te meten waarna hieruit een magnetisch inductie veld afgeleid kan worden.

4.5.2 Specificaties

De VMR is een kost effectieve sensor voor het meten van het vector magnetisch inductieveld. De VMR is bruikbaar voor metingen, waarbij precisie in nanotesla gewenst is. In het geval van lokale magnetische velden in een gebouw geldt als vuistregel dat er variaties optreden in de orde grootte van $30\mu T$ tot $60\mu T$. Dit betekent dat een precisie van enkele nanotesla voldoende is voor onze toepassing. De specificaties van de TWINLEAF VMR zijn opgenomen in Tabel 3. [27]

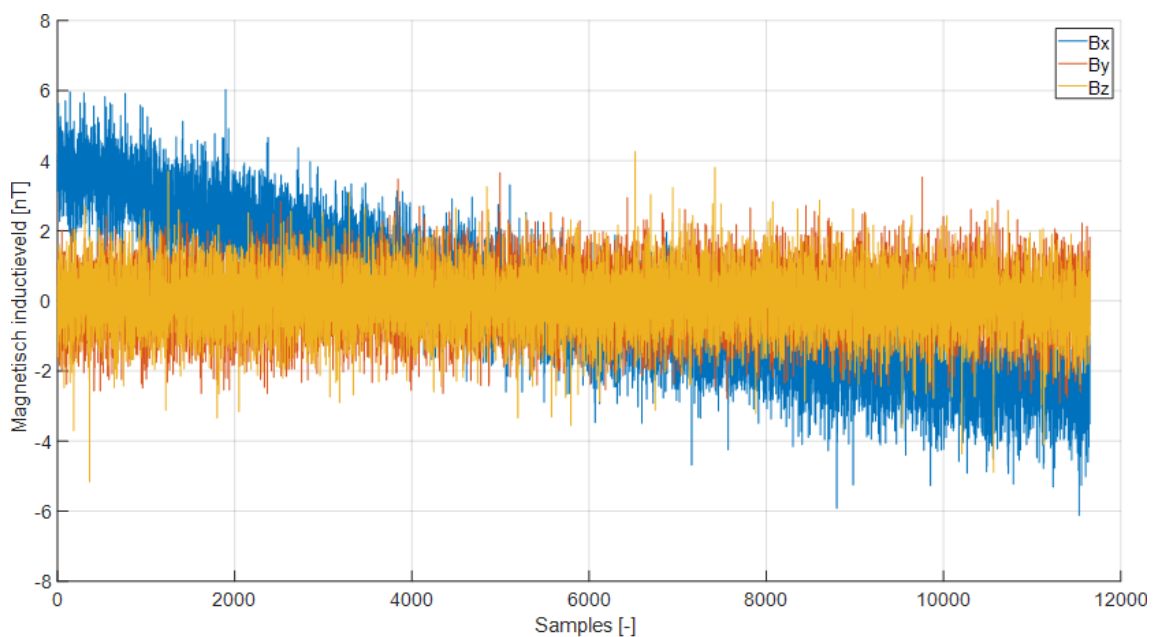
Tabel 3: *Specificaties van de Twinleaf VMR. [27]*

Specifications VMR	
Number of axis	3
Sensitivity	300 pT/Hz
Field range	$\pm 1 \cdot 10^{-4}$ T (nonlinear to $\pm 2 \cdot 10^{-4}$ T)
Simultaneous sampling rate	200 Hz
Power	450 mW
Volume	16x16x87 mm ³
Mass	40 g

Naast dat de VMR in staat is het magnetisch inductieveld te meten bevat deze sensor ook een gyroscoop, versnellingsmeter en een barometrische druk sensor. Met de barometrische druk sensor is het mogelijk om de hoogte te bepalen met een resolutie van 0,2 m. Deze extra sensoren zullen bij het MSLAM algoritme niet worden gebruikt. De VMR wordt enkel gebruikt voor het waarnemen van het magnetisch inductieveld.

4.5.3 Analyse gedrag en karakterisering

Ook bij de VMR is gedurende 10 minuten het magnetisch inductie-veld gemeten. Hierbij is de VMR in een magnetische shielding chamber geplaatst. Met deze opstelling is het mogelijk om het achtergrondveld vrijwel geheel af te sluiten. Hierdoor kan goed de ruis van de VMR in kaart worden gebracht. Het verloop van deze meting is opgenomen in Figuur 20. Hierbij is de gemiddelde waarde van elk van de componenten afgehaald om zo de variatie nog beter in kaart te brengen.



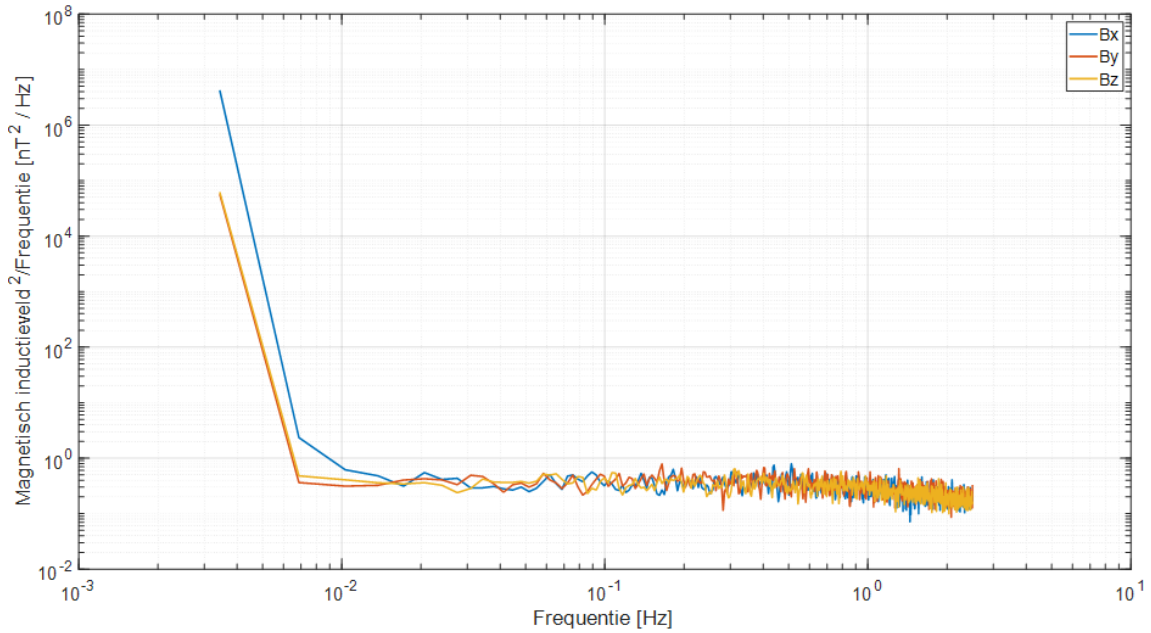
Figuur 20: *Gemeten magnetisch inductieveld voor de x, y, z component min de gemiddelde waarde van het gemeten magnetisch inductieveld uitgezet tegen de samples waarbij de VMR gedurende 10 minuten geplaatst is in een magnetic shielding chamber.*

Figuur 20 laat zien dat voor de y en z -component de ruis een variatie heeft van ± 2 nT. Voor de x -component is dit ook rond de ± 2 nT. B_x laat een aflopend verband zit. Dit wordt mogelijk veroorzaakt doordat de sensor niet precies in het midden van de magnetic shielding camber geplaatst was. Alleen in het midden hiervan is het veld namelijk voor elk van de componenten echt 0. Iets buiten het midden zal dus een klein magneetveld gemeten worden. Wanneer er alleen gekeken wordt naar het verloop van het signaal is ook een offset zichtbaar in de gemeten componenten. Hoewel dit niet zichtbaar is in Figuur 20

De x -component ondervindt een offset van 209 nT. De y en z -componenten ondervinden respectievelijk een offset van 24 en 25 nT. Ook is er een afnemend verband zichtbaar in de x -component. Dit is mogelijk het gevolg van een on bereikt evenwichtssituatie. De VMR dient namelijk 10 minuten geacclimatiseerd te zijn voordat er een meting uitgevoerd wordt. In dit geval is dat niet van toepassing geweest. Waardoor er een afnemend verband in de x -component te zien is.

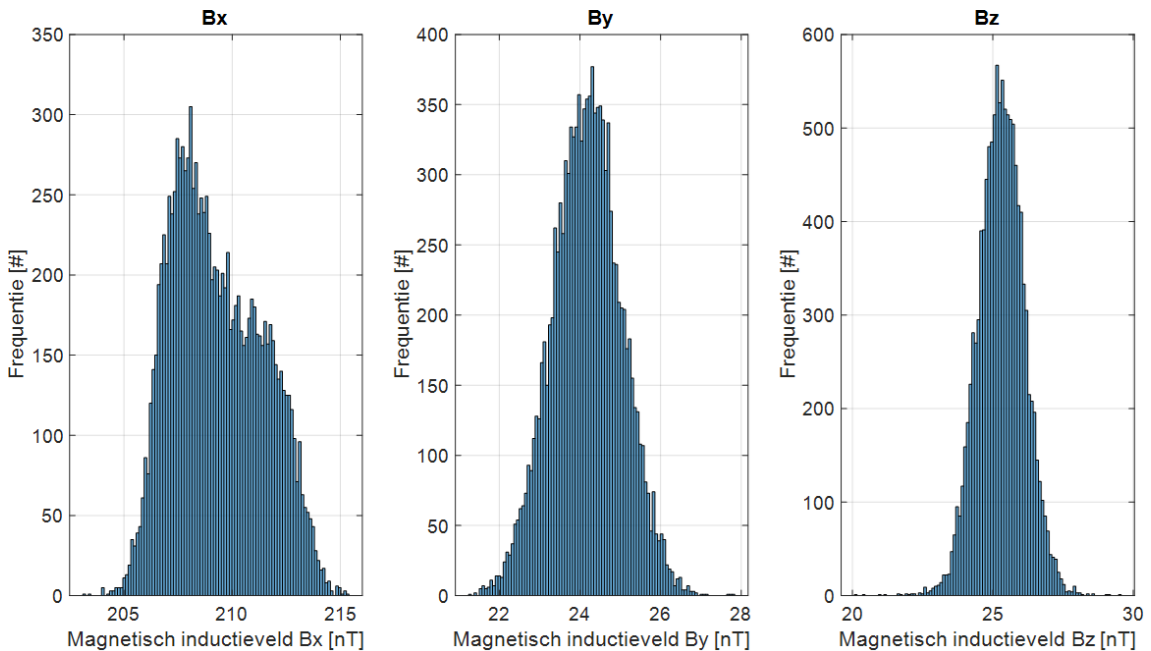
Met de nu in kaart gebracht offset en het verloop wordt vervolgens gekeken naar de spectrale dichtheid van het gemeten signaal. Hiervoor wordt gebruik gemaakt van Welchs methode. Het resultaat hiervan is te zien in Figuur 21 hieronder.

Figuur 21 laat zien dat de ruisvloer van de VMR vrijwel direct voor elk van de componenten geraakt wordt. Bij deze ruisvloer wordt er witte ruis gemeten door de sensor. Het verloop hier naar toe is voor elk van de componenten ver van ideaal.



Figuur 21: *Spectrale dichtheid van magnetisch inductieveld voor elk van de componenten (B_x , B_y , B_z)*

De vorm van de ruis wordt in kaart gebracht door het maken van een histogram van het gemeten signaal. Hiervoor zijn 100 bins opgenomen samen met het gemeten magnetisch inductieveld van elk van de componenten. Het resultaat hiervan is hieronder in Figuur 22 opgenomen.



Figuur 22: *Histogram met het magnetisch inductieveld voor alle drie de componenten. Ter verduidelijking van de vorm van het signaal.*

Figuur 22 laat zien dat de ruis een de y en z -component een gaussische vorm bevat. De x -component laat dit minder duidelijk zien. Ook laat Figuur 22 de offset in het

gemeten signaal zien voor elk van de componenten zien. De piek van de verdeling bevindt zich namelijk niet in de oorsprong. Het gemiddelde $\hat{\mu}$ en de variatie $\hat{\Sigma}$ bepaald volgens Hoofdstuk 4.2 is volgt:

$$\hat{\mu} = \begin{bmatrix} 2,0930 \cdot 10^{-7} \\ 2,4173 \cdot 10^{-8} \\ 2,5302 \cdot 10^{-8} \end{bmatrix} \quad \hat{\Sigma} = \begin{bmatrix} 4,3853 \cdot 10^{-18} & -1,4737 \cdot 10^{-19} & 4,2266 \cdot 10^{-20} \\ -1,4736 \cdot 10^{-19} & 7,3665 \cdot 10^{-19} & 7,4589 \cdot 10^{-22} \\ 4,2266 \cdot 10^{-20} & 7,4589 \cdot 10^{-22} & 6,7487 \cdot 10^{-19} \end{bmatrix}.$$

4.6 LIDAR

De LIDAR bevindt zich als enige sensor niet op de OpenCR 1.0, maar is uit te lezen via een seriële poort op de raspberry PI. De data van de LIDAR is te verkrijgen door de `/scan` topic. De LIDAR meet een afstand en de intensiteit van het gereflecteerde licht per graad van rotatie. De werking van de LIDAR wordt toegelicht in Hoofdstuk 6. Deze paragraaf beschrijft de specificaties van de LIDAR en wat dit betekend voor het maken van de kaarten.

4.6.1 Specificaties

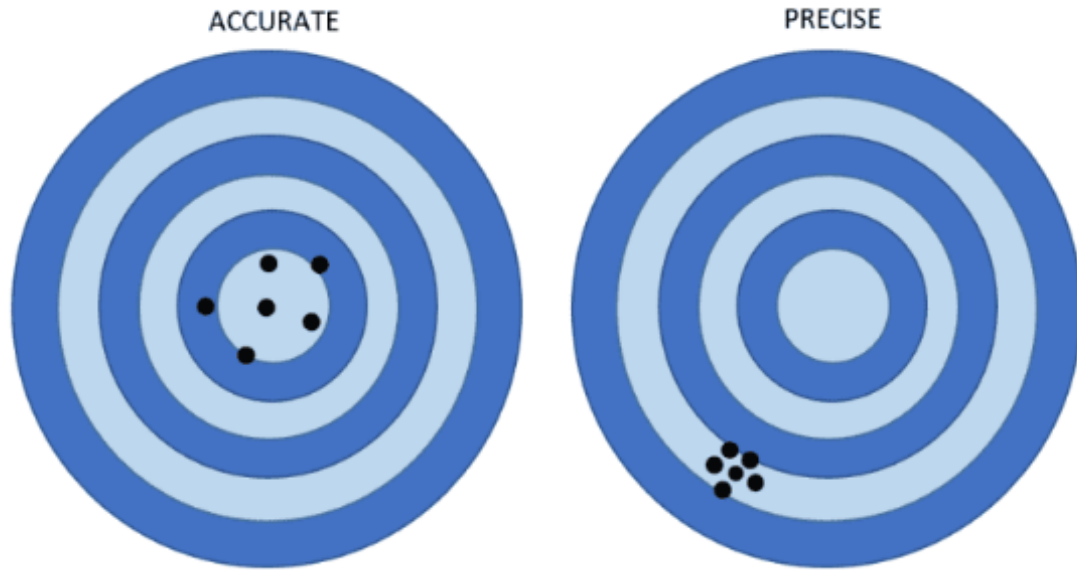
In Bijlage C zijn de volledige specificaties van de gebruikte LIDAR LDS-01 opgenomen. In deze paragraaf worden enkele specificaties toegelicht samen met de nauwkeurigheid en het gevolg hiervan op de nauwkeurigheid van de LIDAR SLAM. De lichtbron van de LDS-01 LIDAR bestaat uit een laserdiode met een golflengte van $\lambda = 785$ nm. Deze golflengte bevindt zich in het IR-gebied [28].

De LDS-01 is een 2D laser scanner met een bereik van 360° graden en is in staat om in stappen van 1° graad te meten. Het bereik van de LDS-01 is $120 \sim 3500$ mm. De nauwkeurigheid en de precisie van de LDS-01 afhankelijk van het bereik is opgenomen in Tabel 4.

Tabel 4: *Nauwkeurigheid en precisie van de gemeten afstand door de LDS-01 LIDAR. [29]*

Nauwkeurigheid (120 mm ~ 499 mm)	± 15 mm
Nauwkeurigheid (500 ~ 3500 mm)	$\pm 5,0$ %
Precisie (120 mm ~ 499 mm)	± 10 mm
Precisie (500 ~ 3500 mm)	$\pm 3,5$ %

We gaan er vanuit dat de LIDAR tijdens het uitvoeren van LIDAR SLAM afstanden meet in het bereik van $500 \sim 3500$ mm. Dit betekend dat we een meetnauwkeurigheid van $\pm 5\%$ hebben. Binnen dit bereikt, $500 \sim 3500$ mm, heeft de LIDAR een precisie van $\pm 3.5\%$. De precisie van de LIDAR beschrijft hoe veel verschillende meetpunten van elkaar afwijken terwijl de nauwkeurigheid beschrijft hoeveel de LIDAR meting afwijkt van de 'waarheid'. Figuur 23 geeft het verschil tussen de nauwkeurigheid en de precisie van een meting schematisch weer. [29] [30]



Figuur 23: Het verschil tussen de nauwkeurigheid en de precisie van een meting schematisch weergegeven. Met links de betekenis van de nauwkeurigheid en rechts de betekenis van precisie. [30]

De LIDAR is dus in staat om binnen een bereik van 500 ~ 3500 mm de afstand $\pm 5\%$ nauwkeurig te meten. Dit zorgt voor een betrouwbare *ground truth* die gebruikt zal worden als vergelijking van Magnetic SLAM met het lidarSLAM algoritme van MATLAB 100% nauwkeurig is.

5 Compensatie magneetsensor

In Hoofdstuk 2 is besproken dat ferromagnetisch materiaal voor een verstoring in het magneetveld zorgt. Ferromagnetisch materialen zoals een schroef of de magneten in de motoren van de Turtlebot, zullen dus het te meten veld beïnvloeden. Daarnaast is ook te verwachten dat

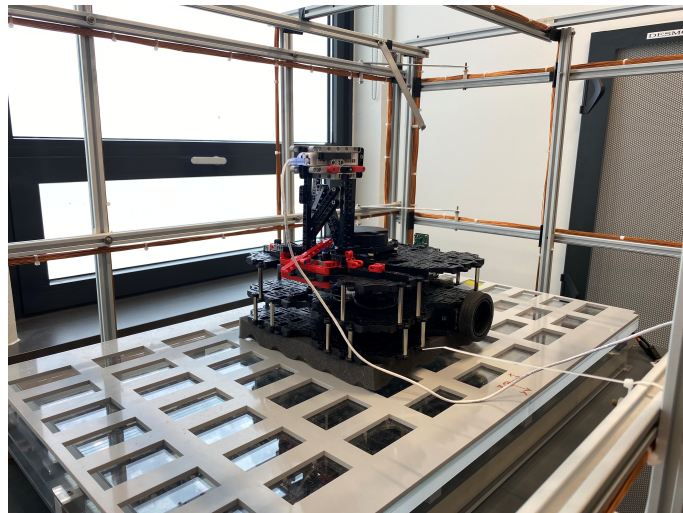
- de spoelen in de motoren,
- de elektrische componenten in de accu,
- de plate supports, en
- de overige elektrische componenten op de Turtlebot

voor een grote invloed op het te meten veld zullen zorgen. De verstoring van het te meten magneetveld is niet gewenst aangezien dit de kwaliteit van de magnetische map en het Magnetic SLAM-algoritme zal beïnvloeden. Door hiervoor te compenseren is te verwachten dat de kwaliteit van een magnetische map toe neemt.

5.1 CLAVIS

Het stoorveld van de Turtlebot zal onderzocht worden in CLAVIS. CLAVIS is een Helmholtz spoelen opstelling, waarin het mogelijk is om een bekend magneetveld aan te leggen tot een veldsterkte van $400 \mu\text{T}$. Verder bestaat CLAVIS uit een sensor array met tweehonderd 3-assige RM3100 magneetsensoren. Deze sensoren kunnen real-time uitgelezen worden, waardoor het magnetisch inductieveld gevisualiseerd en bemeaten kan worden.

Ook biedt CLAVIS de mogelijkheid voor het aanleggen van een nulveld. Bij een aangelegd nulveld is er geen magnetisch veld binnen CLAVIS aanwezig. Dit maakt het mogelijk om de permanente verstoring in kaart te brengen, waarna er voor deze verstoringen gecompenseerd kan worden. Figuur 24 laat de Turtlebot in CLAVIS zien.



Figuur 24: *De Turtlebot in CLAVIS.*

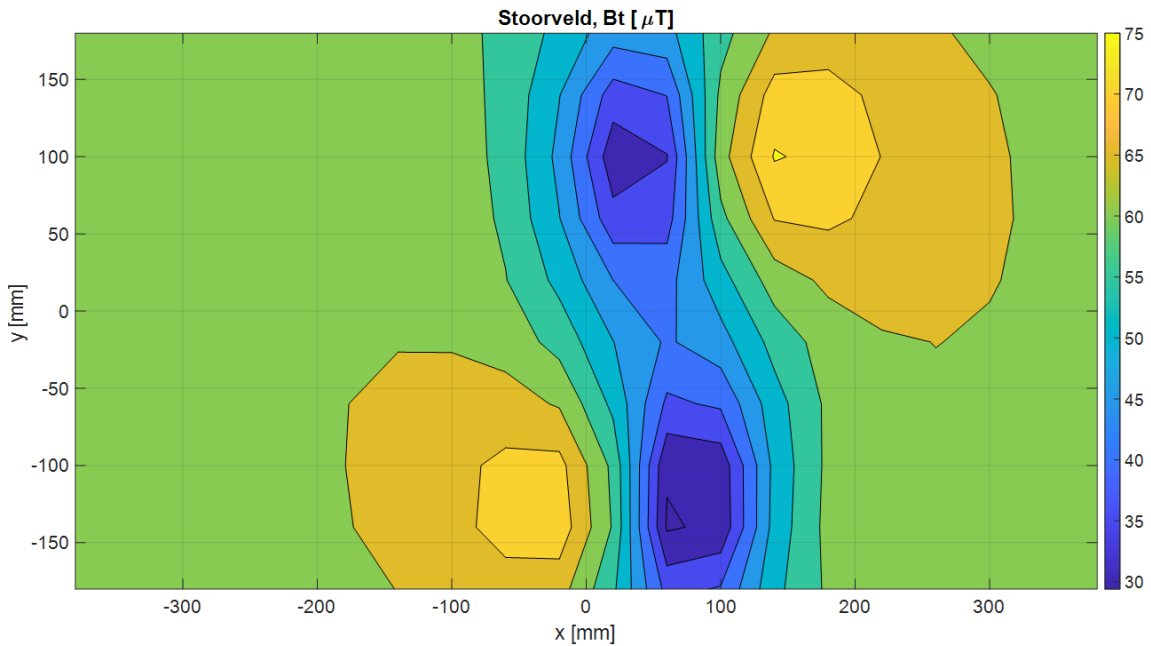
De invloed van de Turtlebot op het magneetveld zal voor twee situaties onderzocht worden. Als eerst wordt de situatie waar de Turtlebot in Idle staat onderzocht. Idle houdt in dat de Turtlebot aan staat maar niet actief bezig is met het uitvoeren van opdrachten. In dit geval gaat het dan ook om de rust toestand van de Turtlebot [31]. Hiermee kan de invloed van de aanwezige ferromagnetische materialen en de magneten in kaart worden gebracht.

Vervolgens wordt de invloed van de bewegende motoren onderzocht. Hiervoor wordt de Turtlebot in CLAVIS geplaatst, waarna er het effect van de bewegende motoren op het te magnetisch inductie-veld in kaart wordt gebracht.

Doordat het aangelegde veld in CLAVIS bekend is, kunnen we het stoorveld van de Turtlebot op de VMR bepalen, door het verschil van de meting en de bekende waarden van elkaar af te trekken.

5.2 In kaart brengen magnetische verstoring

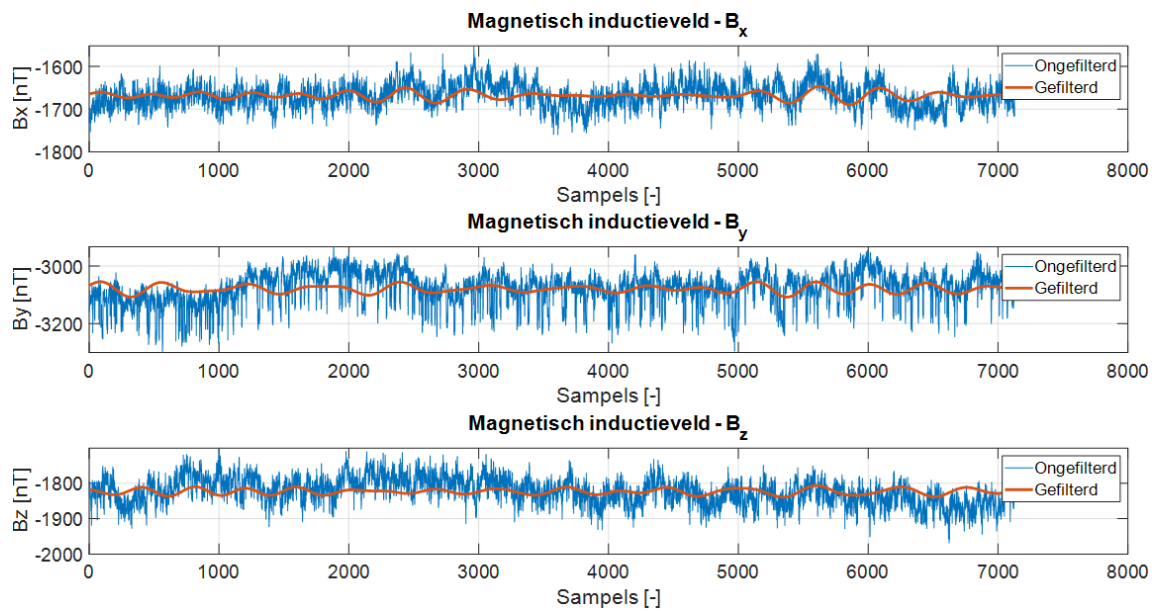
Als eerst is er met CLAVIS het achtergrondveld gemeten. De sterkte van dit veld varieert tussen $42,8 \mu\text{T}$ en $44,2 \mu\text{T}$. De verdeling van dit veld is opgenomen in bijlage H. Het stoorveld is in kaart gebracht aan de hand van: $\mathbf{B}_{\text{stoorveld}} = \mathbf{B}_{\text{Achtergrond}} - \mathbf{B}_{\text{meting}}$. Dit leidt tot Figuur 25.



Figuur 25: Het totale stoorveld \mathbf{B}_t weergegeven in μT uitgezet tegen de afstand in horizon en verticale richting in mm.

Het totale stoorveld weergegeven in Figuur 25 uitgezet tegen de afstand in millimeter. Dit veld is gemeten op de sensor array dat zich 5 cm onder de Turtlebot bevindt. Het daadwerkelijke stoorveld zal dus op locatie van de Turtlebot groter zijn. De kleuren corresponderen met de kleur schaal aan de rechter kant van het figuur en is representatief voor de grootte van het magneetveld in μT . In Figuur 25 is duidelijk te zien dat er magneten aanwezig zijn in de motoren. Deze bevinden zich op (100,100) mm en (20,-120) mm.

Aangezien bij het uitvoeren van Magnetic SLAM gebruik gemaakt zal worden van de VMR zal ook de data hiervan moeten worden onderzocht. De compensatie zal dan ook aan de hand van de VMR uitgevoerd moeten worden. Figuur 26 laat het verloop van het gemeten magneetveld met de VMR zien voor elk van de drie componenten waarbij er gemeten is in een nulveld.



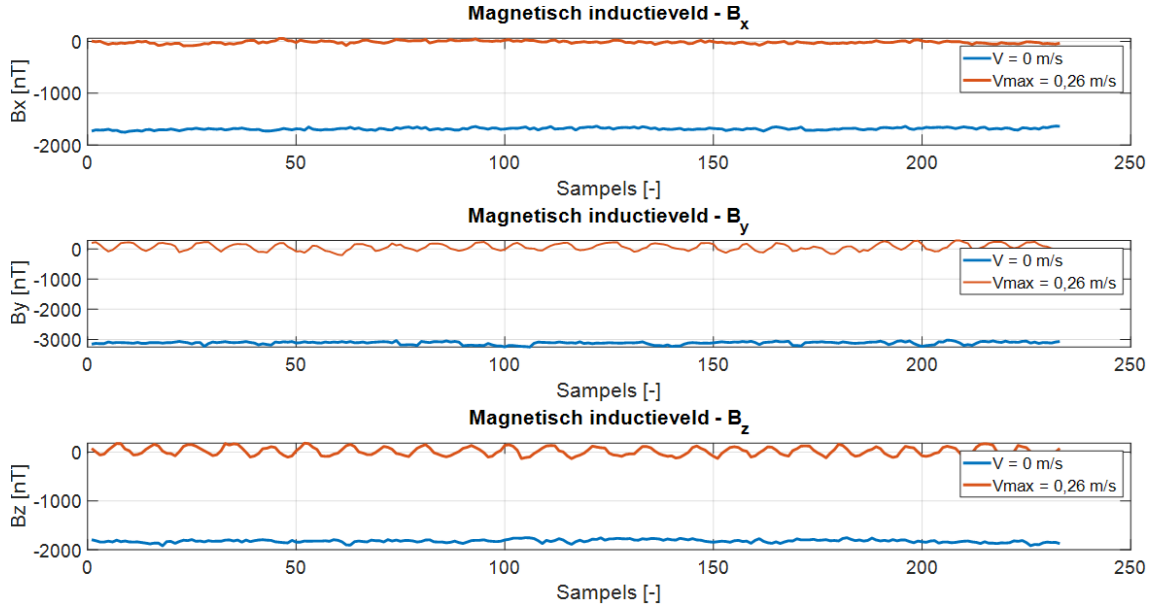
Figuur 26: *Stoorveld van elk van de componenten in nT uitgezet tegen de samples gemeten door de VMR waarbij de Turtlebot gedurende 10 minuten idle staat in een nulveld.*

Figuur 26 laat het gemeten signaal zien in het blauw samen met het frequentie gefilterde signaal in het oranje. Bij het gefilterde signaal wordt gebruikt gemaakt van een bandpass filter die de frequenties in het bereik van 0,5 tot 1 Hz doorlaat. Deze frequentie band is gekozen om zo de laag frequente verstoringen in kaart te brengen.

Nu bekend is wat de VMR waarneemt in wanneer de Turtlebot Idle is, is de volgende stap het bekijken van de VMR-data wanneer de Turtlebot met de maximale snelheid van 0,26 m/s voorwaarts beweegt. Figuur 27 laat het totale stoorveld zien wanneer de Turtlebot met maximale snelheid voorwaarts beweegt samen met het stoorveld wanneer de Turtlebot Idle is. Om het stoorveld als gevolg van voorwaartse beweging in kaart te brengen is er gecompenseerd voor de permanente invloed van de Turtlebot op het magnetisch inductieveld. Dit leidt tot de oranje lijn uit Figuur 27.

Figuur 27 laat zien dat de invloed van het bewegen met maximale snelheid ten opzicht van de Idle stand verwaarloosbaar klein is. Door het bewegen van de motoren wordt het stoorveld zeker aangepast, maar op een veel kleinere schaal. Aangezien de verstoring door beweging zo klein is, zal hier niet nog extra gecompenseerd voor worden.

Nu de verstoringen van de Turtlebot op het te meten veld in kaart gebracht zijn, is de volgende stap compenseren hiervoor. Er zal alleen gecompenseerd worden voor de statische verstoringen door de Turtlebot. Denk hierbij aan verstoring door de magneten, ferromagnetische materialen in de Turtlebot en overige statische elektrische componenten.



Figuur 27: Stoorveld van elk van de componenten in nT uitgezet tegen de samples gemeten door de VMR, waarbij de Turtlebot gedurende 20 seconde Idle staat. Samen met de geïsoleerde invloed van het bewegen van de motoren bij een maximale snelheid gedurende 20 seconde.

5.3 Compensatie model

Nu bekend is dat de magneten in de motoren voor een aanzienlijke verstoring zorgen en de invloed van de roterende motoren te verwaarlozen is, is de volgende stap het wegregelen hiervan. Het wegregelen van de invloed van de Turtlebot op het te meten magnetisch inductie-veld gebeurt aan de hand van een lineair model. Dit model gaat ervan uit dat de meting verricht met de VMR bestaat uit:

$$\mathbf{B}_{VMR} = \mathbf{B}_{local} + \mathbf{B}_p^{TB} + \mathbf{B}_{ind}^{TB} \quad (19)$$

hierin is \mathbf{B}_{VMR} het gemeten magneetveld met de VMR, \mathbf{B}_p^{TB} de permanente verstoring van de Turtlebot, \mathbf{B}_{ind}^{TB} de verstoring geïnduceerd door het daadwerkelijke lokale veld en \mathbf{B}_{local} het lokale magneetveld.

We veronderstellen dat de geïnduceerde verstoring lineair is in \mathbf{B}_{local} . Hierdoor is het mogelijk om \mathbf{B}_{ind}^{TB} als functie van \mathbf{B}_{local} te schrijven. Dit resulteert in:

$$\mathbf{B}_{ind}^{TB} = \mathbf{B}_{ind}^{TB}(\mathbf{B}_{local}) = A\mathbf{B}_{local}. \quad (20)$$

Door middel van lineaire regressie is een matrix A op te stellen waarmee de \mathbf{B}_{ind}^{TB} bepaald kan worden als functie van het lokale magneetveld \mathbf{B}_{local} . Dit resulteert in:

$$\mathbf{B}_{VMR} = \mathbf{B}_p^{TB} + A\mathbf{B}_{local} + \mathbf{B}_{local}. \quad (21)$$

Met het nu bekende model is de volgende stap het verzamelen van de data, om zo matrix A te kunnen bepalen. Hiermee wordt het mogelijk om te compenseren voor de invloed van de Turtlebot op het te meten magnetisch inductie-veld. Na het toelichten welke data en hoe deze verzameld is, is de volgende stap het omzetten van deze data naar matrix A .

5.4 Verzamelen van de benodigde data

Voordat matrix A bepaald kan worden moet eerst de benodigde data worden verkregen. Als eerst is de permanente invloed van de Turtlebot op het te meten magneetveld \mathbf{B}_p^{TB} bepaald. \mathbf{B}_p^{TB} is opgenomen in Tabel 5.

Tabel 5: \mathbf{B}_p^{TB} ; permanente invloed van de Turtlebot op het te meten magneetveld uitgedrukt in tesla's.

	\mathbf{B}_p^{TB} [T]
x	$-2,2221 \cdot 10^{-5}$
y	$-3,8130 \cdot 10^{-6}$
z	$-1,7147 \cdot 10^{-6}$

Vervolgens is bij verschillende aangelegde velden het magneetveld gemeten. Voor elk van de drie componenten (B_x, B_y, B_z) is individueel een veld aangelegd van zowel $-50 \mu\text{T}$ als $50 \mu\text{T}$. Verdeeld in twee situaties waarbij de Turtlebot aanwezig is, en een situatie waar dit niet zo is. Hiervoor is het belangrijk dat de VMR constant op dezelfde plek blijft ook wanneer er niet Turtlebot niet aanwezig is. Dit komt doordat het veld in CLAVIS net niet helemaal homogeen is. Hiervan is vervolgens de permanente invloed van afgehaald.

Voor de situatie waarbij de Turtlebot aanwezig is leidt dit tot Tabel 6. Hierin is de geïnduceerde verstoring opgenomen volgens: $\mathbf{B}_{ind}^{TB} = \mathbf{B}_{VMR} - \mathbf{B}_p^{TB}$.

Tabel 6: Gemeten magnetisch inductieveld door de VMR met de aanwezigheid van de Turtlebot. Hierbij is al gecorrigeerd voor de permanente invloed van de Turtlebot.

	nBx [T]	pBx [T]	nBy [T]	pBy [T]	nBz [T]	pBz [T]
x	$3,0132 \cdot 10^{-7}$	$-3,1906 \cdot 10^{-7}$	$2,3935 \cdot 10^{-6}$	$-2,4577 \cdot 10^{-6}$	$-7,6458 \cdot 10^{-7}$	$7,7158 \cdot 10^{-7}$
y	$-2,4305 \cdot 10^{-6}$	$2,3156 \cdot 10^{-6}$	$4,9359 \cdot 10^{-8}$	$-7,9281 \cdot 10^{-8}$	$3,0764 \cdot 10^{-6}$	$-4,2875 \cdot 10^{-7}$
z	$6,1972 \cdot 10^{-6}$	$-7,3634 \cdot 10^{-7}$	$-5,4499 \cdot 10^{-7}$	$3,3225 \cdot 10^{-7}$	$-4,3256 \cdot 10^{-7}$	$2,1441 \cdot 10^{-7}$

Tabel 6 laat het gemiddelde zien van het magnetisch inductieveld zien gemeten door de VMR met de Turtlebot aanwezig waarbij er voor elke component een veld van $-50 \mu\text{T}$ en $50 \mu\text{T}$ is aangelegd. **nBx** geeft dus het gemiddelde van het gemeten magnetisch inductieveld van elke component bij een aangelegd veld van $-50 \mu\text{T}$ in de x -richting en **pBx** bij $+50 \mu\text{T}$.

Vervolgens is dezelfde meting uitgevoerd, maar dan zonder de aanwezigheid van de Turtlebot. Hiervoor is, zoals al eerder benoemd, de locatie van de VMR onveranderd gebleven. Dit resulteert in onderstaande tabel. Hierbij is niet gecorrigeerd voor de permanente invloed van de Turtlebot op het te meten veld \mathbf{B}_p^{TB} aangezien de Turtlebot hier niet aanwezig is. Deze meting wordt gebruikt om zo het verschil in het waargenomen magnetisch inductieveld in kaart te brengen en is opgenomen in Tabel 7.

Tabel 7: Gemeten magnetisch inductieveld door de VMR zonder de aanwezigheid van de Turtlebot.

	nBx [T]	pBx [T]	nBy [T]	pBy [T]	nBz [T]	pBz [T]
x	$-4,5146 \cdot 10^{-5}$	$4,5165 \cdot 10^{-5}$	$-4,2828 \cdot 10^{-7}$	$6,7288 \cdot 10^{-7}$	$-1,4758 \cdot 10^{-6}$	$1,7038 \cdot 10^{-6}$
y	$7,6300 \cdot 10^{-7}$	$3,9934 \cdot 10^{-7}$	$-4,4357 \cdot 10^{-5}$	$4,5700 \cdot 10^{-5}$	$1,4387 \cdot 10^{-7}$	$1,0235 \cdot 10^{-6}$
z	$2,2623 \cdot 10^{-6}$	$-8,0996 \cdot 10^{-7}$	$2,2744 \cdot 10^{-6}$	$-8,5763 \cdot 10^{-7}$	$-4,3298 \cdot 10^{-5}$	$4,5803 \cdot 10^{-5}$

5.5 Bepalen van de regressiecoëfficiënten

Met de benodigde data nu verzameld is de volgende stap het bepalen van de regressiecoëfficiënten. Na het bepalen van deze coëfficiënten kan er gecompenseert worden voor de invloed van de Turtlebot op het te meten magnetisch inductieveld.

Aangezien er een permanente verstoring is kan hiervoor gecompenseerd worden. Dit gebeurt volgens $\mathbf{B}_{VMR} - \mathbf{B}_p^{TB} = \hat{\mathbf{B}}$. Waarna substitutie hiervan in Formule 21 resulteert in:

$$\hat{\mathbf{B}} = A\mathbf{B}_{local} + I\mathbf{B}_{local}. \quad (22)$$

Aangezien A een matrix is wordt er nog een identiteitsmatrix aan \mathbf{B}_{local} toegevoegd om zo uiteindelijk het lokale magneetveld \mathbf{B}_{local} te kunnen bepalen. Aan de hand van Formule 22 kan het lokale magneetveld worden bepaald:

$$\mathbf{B}_{local} = (A + I) \setminus \hat{\mathbf{B}}. \quad (23)$$

Hierin wordt \mathbf{B}_{local} geschreven als functie van $\hat{\mathbf{B}}$ met hierbij matrix A en identiteitsmatrix I . In Formule 23 correspondeert \setminus tot een matrix linkse deling [32]. Voordat het lokale veld gemeten kan worden dient matrix A te worden opgesteld. Deze matrix op te stellen aan de hand van lineaire regressie. Het gemeten veld b_n kan geschreven worden als een product van een *3times3* matrix A en het aangelegde veld x_n :

$$\begin{aligned} Ax_1 &= b_1 \\ Ax_2 &= b_2 \\ &\vdots \\ Ax_6 &= b_6 \end{aligned} \quad (24)$$

hierin is x_n het aangelegde veld, A de constante en b_n met $n = 1, 2, \dots, 6$ het gemeten veld met de VMR. Vervolgens kan b_n opgenomen worden in matrix B . Dit leidt tot $B = [b_1 \dots b_6]$. Vervolgens kan B geschreven worden als:

$$\begin{aligned} B &= [b_1 \dots b_6] \\ &= [Ax_1 \dots Ax_6] \\ &= A[x_1 \dots x_6] \\ &= AX. \end{aligned} \quad (25)$$

Matrix B en X bestaan uit Tabel 6 en Tabel 7 respectievelijk. Vervolgens kan de matrix A bepaald worden. Dit is te doen aan de hand van Formule 26 hieronder [33]:

$$A = BX^+ \quad (26)$$

hierin wordt de pseudoinverse van X genomen vermenigvuldigd met de matrix met de aangelegde magneetvelden X . Dit leidt tot de beste afbeelding van de aangelegde velden naar de gemeten geïnduceerde velden door de Turtlebot.

Met de nu bekende matrix A is het mogelijk om te compenseren voor de invloed van de Turtlebot op het te meten magneetveld met de VMR voor stationaire metingen. Matrix A is opgenomen in Tabel 8.

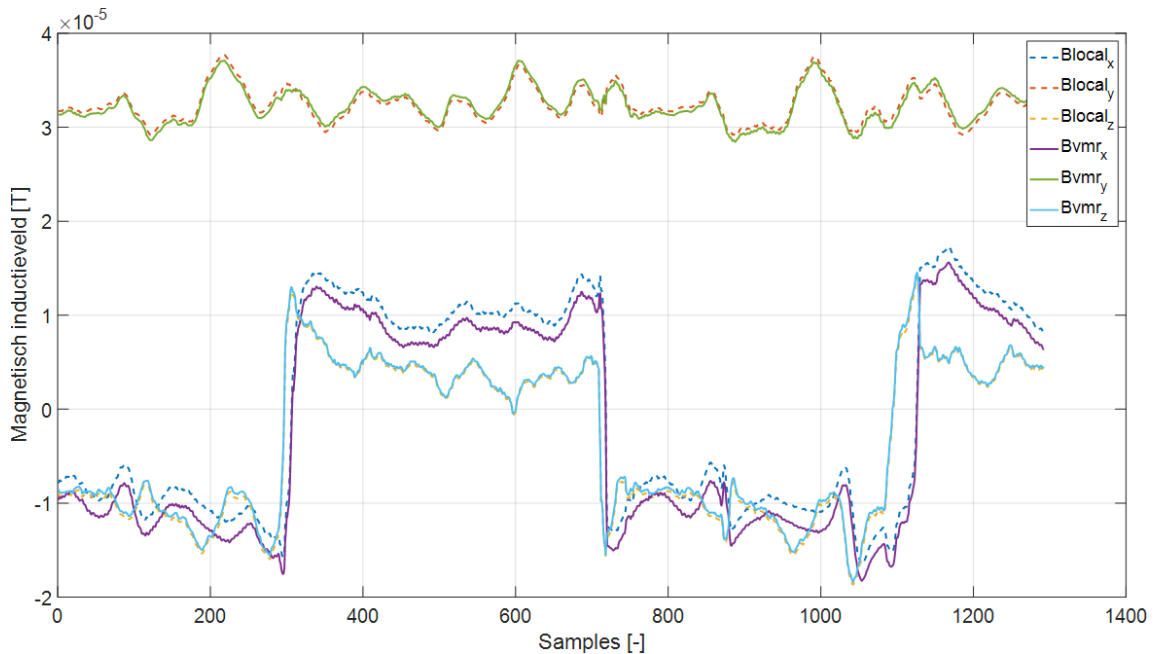
Tabel 8: *Filter matrix A bepaald aan de hand van de data uit Tabel 5, Tabel 6 en Tabel 7 voor het compenseren van de invloed van de Turtlebot op het te meten magneetveld.*

	A [T]		
x	-0,0065	-0,0532	0,0180
y	0,0533	-0,0025	-0,0102
z	-0,0147	0,0101	0,0075

5.6 Resultaat na compensatie

Vorige paragraaf beschreef het bepalen van de regressiecoëfficiënt. Met deze nu bepaald is de volgende stap het toepassen van de compensatie. Deze paragraaf laat het gevolg van de regressie zien aan de hand van een Figuur 28. Hierin is het effect van de compensatie te zien.

Figuur 28 laat het lokale werkelijke magneetveld \mathbf{B}_{local} zien zoals hierboven beschreven wordt. Hiervoor is een dataset gebruikt waarbij de Turtlebot op verschillende plekken in de ruimte het magneetveld meet.



Figuur 28: *Gemeten magneetveld, B_{vmr} , weergegeven over de tijd heen uitgezet tegen de samples waarbij de Turtlebot rond rijdt en op diverse plekken het magneetveld gemeten heeft. Samen met B_{local} , het daadwerkelijke magneetveld, waarbij voor de invloed op het magneetveld van de Turtlebot gecompenseerd wordt.*

Naast het \mathbf{B}_{local} laat Figuur 28 ook het \mathbf{B}_{vmr} zien. \mathbf{B}_{local} is het daadwerkelijke magneetveld en \mathbf{B}_{vmr} het on gecompenseerde magneetveld, de data direct uit de VMR tijdens het rondrijden. In \mathbf{B}_{vmr} is de invloed van de Turtlebot dan ook nog aanwezig.

Wanneer \mathbf{B}_{local} met \mathbf{B}_{vmr} vergeleken wordt dan is te zien dat \mathbf{B}_{local} over het algemeen net wat zwakker is voor de y en z -component. De x -component van \mathbf{B}_{local} ligt wat hoger dan het gemeten veld. Door de verstoring van het Turtlebot is het mogelijk dat er door de VMR te lage waarden of te hoge waarden gemeten worden. Hiervoor is nu echter te corrigeren.

Figuur 28 laat zien dat het uitvoeren van de compensatie het lokale magneetveld \mathbf{B}_{local} beter beschrijft dan \mathbf{B}_{vmr} . Door het uitvoeren van de compensatie zou de magnetische map nauwkeuriger moeten worden aangezien hierin nu niet meer de Turtlebot opgenomen

wordt. Het verwerken van \mathbf{B}_{local} in het Magnetic SLAM algoritme leidt dus tot een magnetische kaart representatief voor het magneetveld. Wanneer \mathbf{B}_{vmr} wordt gebruikt is dit niet het geval aangezien deze data ook de permanente magneten en andere verstoringen van de Turtlebot mee meet. De nauwkeurigheid van \mathbf{B}_{local} is niet onderzocht.

Voor het uitvoeren van deze compensatie is gebruik gemaakt van de aanname dat de geïnduceerde verstoring lineair is in \mathbf{B}_{local} . Het is nu dus mogelijk om te corrigeren voor de geïnduceerde verstoringen door de Turtlebot op het te meten magnetisch inductieveld. Dit heeft als gevolg dat de te maken magnetische kaart meer overeenkomt met het achtergrondveld. Wat leidt tot een hogere kwaliteit en nauwkeurigere magnetische kaart.

6 LIDAR SLAM

Het maken van een kaart aan de hand van LIDAR SLAM wordt in dit hoofdstuk besproken. SLAM staat voor Simultaneous Localization And Mapping. Bij LIDAR SLAM wordt een LIDAR gebruikt voor het uitvoeren van SLAM. SLAM is de methode die gebruikt wordt om met de Turtlebot om een kaart van de omgeving te maken terwijl de locatie van bot tegelijkertijd wordt bepaald [34].

6.1 Methode

Bij LIDAR SLAM wordt er aan de hand van de LIDAR sensor de gelijktijd een kaart gemaakt en de positie bepaling uitgevoerd. LIDAR (Light Detection And Ranging) is een methode om afstand te meten. Een LIDAR bestaat voornamelijk uit een laser en een detector om de afstand te bepalen [35].

De laser zend licht uit waarna het licht vervolgens door een object terug gereflecteerd wordt op de detector. De tijd die het kost om weer terug te komen op de detector wordt gemeten waarna de afstand bepaald wordt. Het bepalen van de afstand gebeurt aan de hand van de formule 27 [36]. Echter geeft de sensor gelijk een afstand terug, het uitvoeren van deze berekening is in dit geval dus niet nodig.

$$s = \frac{1}{2}(c \cdot t) \quad (27)$$

Hierin is s de afstand tot het object in meter, c de lichtsnelheid in m/s ($c = 3.0 \cdot 10^8$ m/s [7]) en t de benodigde tijd in seconde. Deze afstand wordt samen met de bijbehorende hoek opgenomen in de SLAM functie binnen MATLAB. Door het continue toevoegen van de LIDAR data wordt hiermee uiteindelijk een kaart gevormd.

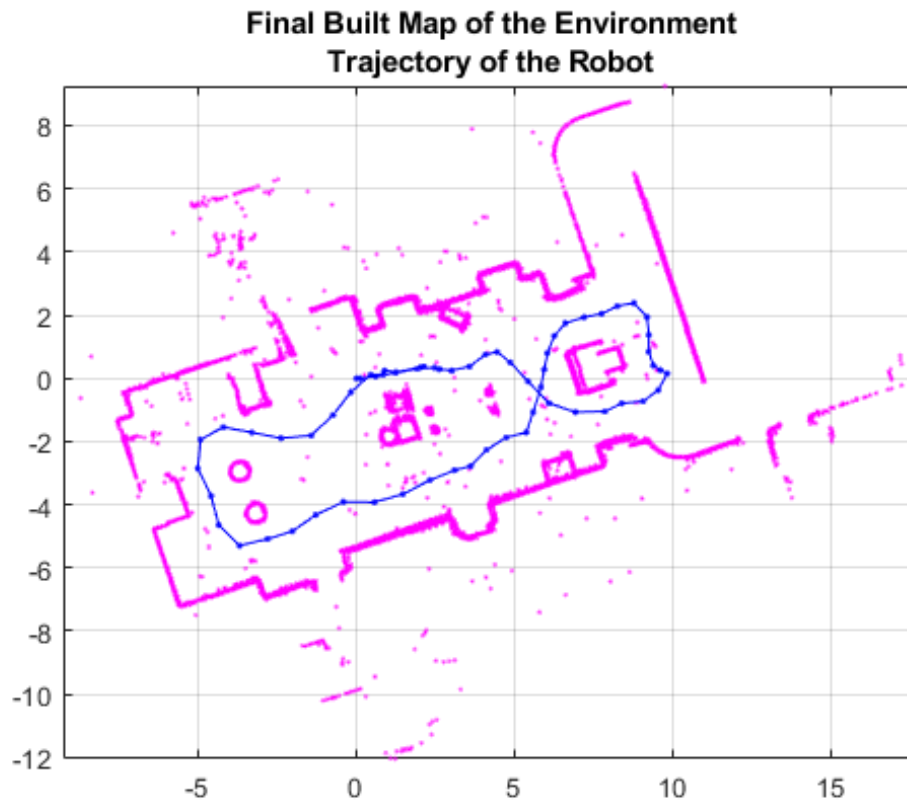
De LIDAR data bestaat uit de zogeheten range, een afstand, en de hoek. De LIDAR meet voor elke graad een afstand en draait rond van 0 tot 360°. Doordat de hoek en de afstand bekend zijn is het mogelijk om een puntenwolk te generen en de afstand te bepalen.

Het afgelegde pad wordt bepaald aan de hand van de punten wolk gemaakt door de SLAM functie. Door het matchen van zogenoemde 'features' in de punten wolk is samen met de afstand tussen deze punten en de Turtlebot en de hoek de positie te bepalen. Deze wordt vervolgens weergegeven [37].

6.2 Voorbeelden

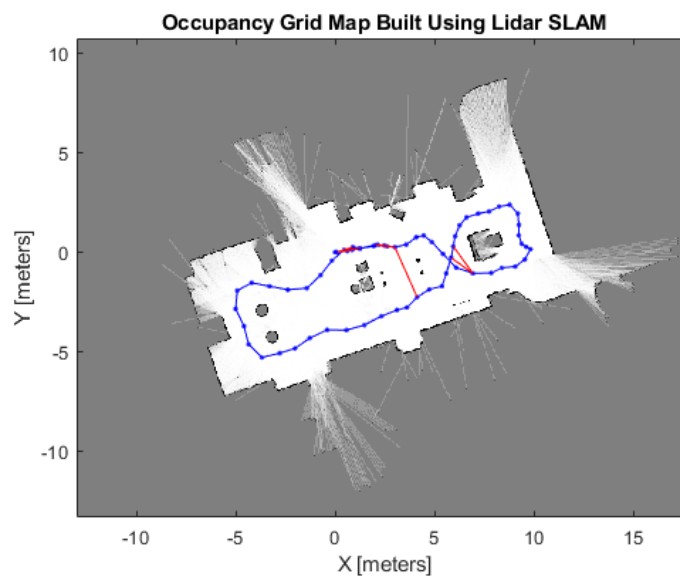
In Figuur 29 is een voorbeeld van een punten wolk opgenomen. Deze punten wolk wordt gemaakt aan de hand van de afstand gemeten bij de bijbehorende hoek. Doordat de LIDAR rond draait, wordt het mogelijk om continu deze punten wolk met de nieuwe informatie te updaten. Waardoor er steeds dikke roze lijnen zichtbaar worden zoals Figuur 29 laat zien.

De positie van de LIDAR wordt in Figuur 29 weergegeven door de blauwe lijn. Door middel van scan matching is het algoritme achter het maken van de map uit 29 instaat om de positie te bepalen. Bij scan matching zoek het algoritme naar loop closures en naar waar scans overlappen met al eerder in kaart gebrachten gebieden om zo de positie te bepalen. [38]



Figuur 29: Voorbeeld LIDAR SLAM puntenwolk en hierin het afgelegde pad van de LIDAR. [39]

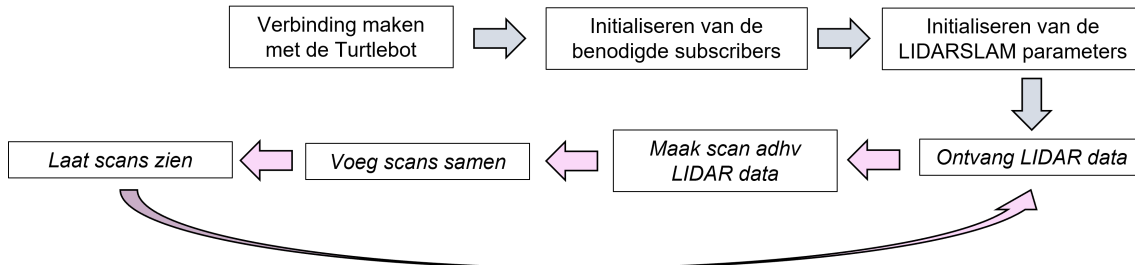
Vervolgens is de punten wolk om te zetten naar een occupancy map. Een occupancy map geeft de bezetting van de ruimte aan waarbij een bepaalde grijswaarde gekoppeld is aan de zekerheid van bezetting. Dit leidt tot een wit, grijs / zwarte kaart van de ruimte. Waarbij de witte gedeelten representatief zijn voor geen bezetting. De zwart delen geven de randen van de ruimte waar volledige bezetting plaatsvind. In de gedeeltes in het grijs is de bezetting niet bekend. Figuur 57 geeft de bijbehorende occupancy kaart weer behorend tot figuur 29.



Figuur 30: Voorbeeld occupancy kaart behorend bij Figuur 29 met hierin het afgelegde pad. [39]

6.3 Implementatie LIDAR SLAM

Deze paragraaf beschrijft de implementatie van LIDAR SLAM. Aan de hand van een blokschema zal de MATLAB code uitgelegd worden. Belangrijk hiervoor is dat de ROS Toolbox [40] en de Navigation Toolbox [41] gedownload worden. Enkele functies uit de code komen voort uit deze Toolboxen. Figuur 31 laat het blokschema voor het uitvoeren van LIDAR SLAM zien.



Figuur 31: *Blokschema implementatie LIDAR SLAM op de Turtlebot3.*

Figuur 31 laat zien er als eerst verbinding met de Turtlebot wordt gemaakt. Vervolgens worden de benodigde subscribers geïnitieerd. De subscribers zijn verantwoordelijk voor het aanroepen van de LIDAR data. Vervolgens worden de benodigde parameters voor de lidarSLAM functie [38] geïnitieerd.

Vervolgens wordt er een loop gestart. In deze loop wordt constant de LIDAR data ontvangen en omgezet naar een scan. Deze scan wordt vervolgens samengevoegd waarna de scan afgebeeld wordt. Door constant scans toe te voegen wordt uiteindelijk een figuur zichtbaar dat lijkt op Figuur 29.

Vervolgens kan dit zogenoemde 'slamobject' omgezet worden naar de verschillende scans en een positie matrix. Beide dienen samen met de al eerder geïnitieerd parameters als input voor de functie waarmee de occupancy map, Figuur 57, te maken is.

Met de werking van de LIDAR nu bekend samen met een verwachting hoe de kaarten er uit komen te zien, en bekend is hoe het toegepast wordt op de Turtlebot, is de volgende stap het uitvoeren van LIDAR SLAM. De resultaten worden volgend hoofdstuk besproken. De LIDAR SLAM resultaten specifiek zijn opgenomen in Hoofdstuk 8.1.

7 Magnetic SLAM

Dit hoofdstuk beschrijft de theorie achter Magnetic SLAM. Als eerst wordt de theorie van het algoritme toegelicht. Dit is gebaseerd op [6, 42, 43]. Het is belangrijk om op te merken dat de precieze afleiding van het Magnetic SLAM algoritme niet opgenomen is in dit verslag. De belangrijkste aspecten van het Magnetic SLAM algoritme zullen behandeld worden. Denk hierbij aan de fysische onderbouwing van het magnetic mapping gedeelte en het algoritme dat ten grondslag ligt aan Magnetic SLAM. Hierna worden een paar voorbeelden van Magnetic SLAM gegeven. Tot slot wordt er een toelichting gegeven van de implementatie van het algoritme op de Turtlebot.

7.1 Magnetic mapping

Deze paragraaf beschrijft de theorie achter het in kaart brengen van het magneetveld. Hiervoor nemen we aan dat er geen vrije stromen aanwezig zijn en dat we ons buiten de magnetisatie bron bevinden. Formule 9 in Hoofdstuk 2 stelt dat de magnetische scalar potentiaal Φ buiten de magnetische bronnen dan voldoet aan:

$$\Delta\Phi = 0. \quad (28)$$

Het idee achter magnetic mapping is het inschatten van de scalaire potentiaal die het lokale magneetveld beschrijft; we vatten de scalaire potentiaal op als bron voor het magneetveld. Zonder verdere kennis over deze potentiaal is het vrijwel onmogelijk om tot zo'n inschatting te komen.

Merk op dat de laplaciaan Δ een lineaire operator is. Dit wilt zeggen dat voor alle scalaire potentialen Φ_1, Φ_2 en alle reële scalaren getallen $c_1, c_2 \in \mathbb{R}$ geldt:

$$\Delta(c_1\Phi_1 + c_2\Phi_2) = c_1\Delta\Phi_1 + c_2\Delta\Phi_2. \quad (29)$$

Uit de wiskunde weten we dat de Laplaciaan operator zich netjes gedraagt. Dit betekent dat de Laplaciaan zich gedraagt als een *matrix*. Voor symmetrische matrices A , $A^T = A$ [44], weten we dat deze altijd een orthogonale diagonalisering toelaten [45]. Een orthogonale diagonaliseerbare matrix houdt in dat de matrix A geschreven kan worden als een product van matrices. Het bewijs dat de Laplaciaan symmetrisch is, is te vinden in [46]. Dit impliceert dat er een verzameling van *eigenfuncties* $\{\Phi_j\}_{j=1}^{\infty}$ en *eigenwaarden* $\lambda_j \in \mathbb{R}$ bestaan, zodat

$$\Delta\Phi_j = \lambda_j^2\Phi_j. \quad (30)$$

De verzameling $\{\Phi_j\}_{j=1}^{\infty}$ vormt een oneindige basis voor alle scalaire potentialen waarvoor geldt dat $\lim_{\|\mathbf{r}\| \rightarrow \infty} \Phi(\mathbf{r}) = 0$, naarmate de absolute positie $\|\mathbf{r}\|$ richting oneindig gaat $\Phi(\mathbf{r})$ naar nul. In het bijzonder geldt nu dat elke scalaire potentiaal Φ , die voldoet aan $\Delta\Phi = 0$, te schrijven is als eindige lineaire combinatie van deze basisfuncties, dus

$$\Phi := \sum_{j=1}^N c_j \Phi_j. \quad (31)$$

Merk op dat N uit Formule 31 zeer groot kan zijn. Voor het magneetveld buiten de magnetische bron, dat de scalaire potentiaal Φ beschrijft, geldt vervolgens (vanwege lineariteit, Formule 5 en Formule 6 uit Hoofdstuk 2)

$$\mathbf{B} = -\mu_0 \sum_{j=1}^N c_j \nabla \Phi_j. \quad (32)$$

We zien dus dat het magneetveld te schrijven is als een oneindige lineaire combinatie in de basisfuncties $\nabla \Phi_j$. In het vervolg zullen we $\nabla \Phi_j$ de j -de mode noemen voor het magneetveld.

Voor het Magnetic SLAM algoritme moeten we de basisfuncties $\nabla\Phi_j$ vooraf bepalen. Hiertoe beschouwen we een rechthoekig volume $\Omega = [-L_1, L_1] \times [-L_2, L_2] \times [-L_3, L_3] \subseteq \mathbb{R}^3$ (met L_1, L_2, L_3 reële getallen) dat zó gekozen is, dat het gebied waarin het algoritme wordt toegepast ruim erbinnen valt. We lossen nu het volgende probleem op:

$$\begin{cases} -\Delta\Phi_j = \lambda_j^2\Phi_j & \text{in } \Omega \\ \Phi_j = 0 & \text{op de rand van } \Omega \end{cases}, \quad (33)$$

waarbij we eisen dat de functies $\Phi_j = 0$ op de rand van Ω zijn. Dit laat zien dat het rechthoekig volume voldoende groot moet zijn, anders valt de potentiaal te snel af naar 0. Dit zou dan leiden tot niet-fysische basisfuncties, want voor een scalaire potentiaal die een magneetveld beschrijft moet gelden dat $\lim_{\|\mathbf{r}\| \rightarrow \infty} \Phi(\mathbf{r}) = 0$.

Het probleem beschreven in Formule 33 kan analytisch opgelost worden [6] voor een rechthoekig volume Ω . Vooraf kiezen we een natuurlijk getal groter dan 0 voor $m \in \mathbb{N}_{>0}$. Dit leidt tot de volgende beschrijving van de eigenfuncties en eigenwaarden voor $j = 1, \dots, m$:

$$\Phi_j(\mathbf{x}) = \prod_{d=1}^3 \frac{1}{\sqrt{L_d}} \sin\left(\frac{\pi n_{j,d}(x_d + L_d)}{2L_d}\right) \quad \text{en} \quad \lambda_j^2 = \sum_{d=1}^3 \left(\frac{\pi n_{j,d}}{2L_d}\right)^2. \quad (34)$$

Hierin is L_d de lengte van zijde d met $\mathbf{n} \in \mathbb{R}^{m \times 3}$ bestaat uit een index set van permutaties van gehele getallen. Dit houdt in dat \mathbf{n} een matrix is van alle mogelijk combinaties van indices, en ziet er als volgt uit:

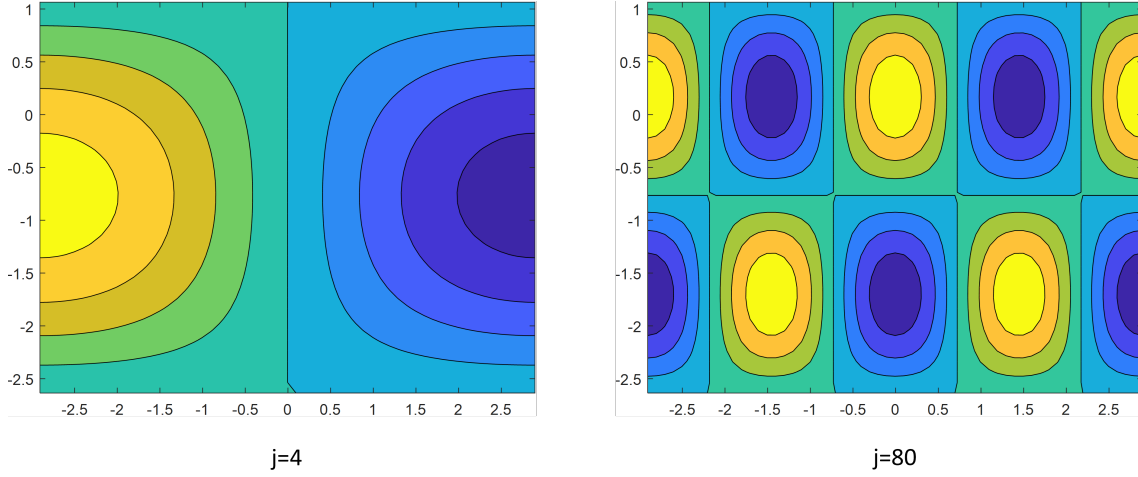
$$\mathbf{n} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 2 \\ \vdots & \vdots & \vdots \\ 1 & 1 & m \\ 1 & 2 & 1 \\ 1 & 2 & 2 \\ \vdots & \vdots & \vdots \\ m & m & m \end{bmatrix}, \quad (35)$$

waarvoor geldt dat $n_{j,d}$ het getal is op de i -de rij en d -de kolom van \mathbf{n} . Met deze matrix \mathbf{n} kunnen alle mogelijke combinaties van eigenwaarden en eigenfuncties bepaald worden. Hiermee bepalen we de belangrijkste $N_m < m$ eigenfuncties, door de N_m **kleinste eigenwaarden** te kiezen. Met N_m kunnen we vooraf instellen hoeveel (lokale) variaties we toelaten in de scalaire potentiaal. Dit leidt dan tot een eindige verzameling van eigenfuncties en bijbehorende eigenwaarden, waarmee een magnetische kaart te maken is. Met deze eindige verzameling is het mogelijk om een magnetische kaart numeriek in eindige tijd te kunnen inschatten.

De verschillende modi voor het beschrijven van het magneetveld volgen nu door de gradiënt van elke functie Φ_j te nemen. Figuur 32 laat x -component van de vierde en 80^e orde mode voor een rechthoekig blok zien.

Figuur 32 laat zien dat een hogere mode resulteert in meer variaties in het magnetisch inductieveld. Dit maakt het mogelijk om lokale variaties in het veld te kunnen vangen in een magnetische kaart. Door een lineaire combinatie van deze eigenfuncties wordt het mogelijk om het gemeten veld in kaart te brengen. Ook kunnen we, gegeven de inschatting van een magnetische kaart, ook het magneetveld bepalen op plekken waar niet direct gemeten is.

Nu bekend is hoe we een magnetische kaart kunnen maken in een ruimte, is de volgende stap het begrijpen hoe metingen met de VMR magneetsensor gebruikt kan worden om een inschatting te kunnen doen van een magnetische kaart, en hoe met deze kaart de positie inschatting van de Turtlebot geüpdatet kan worden.



Figuur 32: *Eigenfuncties opgelost voor een rechthoekig blok domein; links de vierde mode ($j = 4$) en rechts de 80^e mode ($j = 80$).*

7.2 Magnetic SLAM algoritme

Nu we weten hoe het magneetveld in kaart gebracht kan worden, kunnen we positie in deze magneetkaart proberen te bepalen. Deze paragraaf beschrijft de werking van het Magnetic SLAM algoritme. Dit algoritme is gebaseerd op een Extended Kalman Filter (EKF). Aan de hand van Figuur 33 wordt de werking van het Magnetic SLAM algoritme beschreven. Figuur 33 laat de pseudocode zien van het Extended Kalman Filter voor het Magnetic SLAM algoritme.

Zoals beschreven is in Bijlage D en Figuur 33, bestaat een Extended Kalman Filter uit drie stappen: de dynamic update, de measurement update en de relinearization. Aan de hand van Figuur 33 wordt elk van deze stappen toegelicht.

In Figuur 33 is te zien dat de input van het algoritme bestaat uit een drietal datasets: de veranderingen in de positie $\Delta p_{t|t}^w$ en de veranderingen in de oriëntatie $\Delta q_{t|t}^b$; deze worden beide gemeten door de IMU. De input $y_{t|t}^b$ correspondeert met het gemeten magneetveld. Voor alle drie de datasets zijn N samples nodig. Het subscript geeft het tijdstip t aan, waarop de metingen zijn gedaan.

Vervolgens worden de drie stappen (dynamic update, de measurement update en de relinearization) voor tijdstappen $t = 1$ tot en met tijdstap N , doorlopen. Dit resulteert in de output van het EKF, dat bestaat uit de positie $\hat{p}_{t|t}^w$, de oriëntatie $\hat{q}_{t|t}^{wb}$ uitgedrukt in quaternionen en de magneetveldmatrix $\hat{m}_{t|t}^w$ van de magneetkaart. De magneetveldmatrix $\hat{m}_{t|t}^w$ is een vector dat N_m lang is, en kan gebruikt worden om een inschatting van het lokale magnetisch veld te kunnen doen middels Formule 32.

Als eerst wordt er een voorspelling gedaan van de volgende positie $\hat{p}_{t|t-1}^w$, de oriëntatie $\hat{q}_{t|t-1}^{wb}$, de magneetveldmatrix $\hat{m}_{t|t-1}$ en de covariantiematrix $\hat{P}_{t|t-1}$. Zowel de voorspelde positie als oriëntatie worden bepaald aan de hand van het verschil in positie en oriëntatie, zoals opgenomen in 26a en 26b in Figuur 33. De positieinschatting is een lineaire update aan de positie, terwijl de verandering in de oriëntatie middels het quaternionenproduct \odot wordt gedaan. Dit is een efficiënte implementatie voor rekenen met quaternionen. Formule 26c laat zien blijft de magneetveldmatrix onveranderd. Dit houdt in dat we aannemen dat het lokale magneetveld niet in de tijd veranderd.

Na de dynamic update stap volgt het fuseren van gemeten magneetveld. Dit is onderdeel van de measurement update. De magneetkaart wordt middels een Kalman filter in de tijd bepaald. In 27a uit Figuur 33 wordt de magneetveld data gefuseerd door het

Algorithm 1: EKF for magnetic field SLAM

Input: $\{\Delta p_t^w, \Delta q_t^b, y_t^b\}_{t=1}^N$
Output: $\{\hat{p}_{t|t}^w\}_{t=1}^N, \{\hat{q}_{t|t}^{wb}\}_{t=1}^N, \{\hat{m}_{t|t}\}_{t=1}^N$
Initialisation: $\hat{p}_{0|0}^w = 0_{3 \times 1}, \hat{q}_{0|0}^{wb} = q_0^{wb}, \hat{m}_{0|0} = 0_{(N_m+3) \times 0}, (19)$

- 1: for $t = 1$ to N do
- 2: Dynamic update

$$\hat{p}_{t|t-1}^w = \hat{p}_{t-1|t-1}^w + \Delta p_t^w \quad (26a)$$

$$\hat{q}_{t|t-1}^{wb} = \hat{q}_{t-1|t-1}^{wb} \odot \Delta q_t^b \quad (26b)$$

$$\hat{m}_{t|t-1} = \hat{m}_{t-1|t-1} \quad (26c)$$

$$P_{t|t-1} = P_{t-1|t-1} + Q \quad (26d)$$

- 3: Measurement update

$$z_t = \hat{R}_{t|t-1}^{wb} y_t^b - \nabla \Phi_p(\hat{p}_{t|t-1}^w) \hat{m}_{t|t-1} \quad (27a)$$

$$S_t = H_t P_{t|t-1} H_t^\top + \sigma_m^2 \mathcal{I}_3 \quad (27b)$$

$$K_t = P_{t|t-1} H_t^\top S_t^{-1} \quad (27c)$$

$$\hat{\zeta}_t = K_t z_t \quad (27d)$$

$$P_{t|t} = P_{t|t-1} - K_t S_t K_t^\top \quad (27e)$$

- 4: Relinearization

$$\hat{p}_{t|t}^w = \hat{p}_{t|t-1}^w + \hat{\delta}_t^w \quad (28a)$$

$$\hat{q}_{t|t}^{wb} = \exp_q(\hat{\eta}_t^w) \odot \hat{q}_{t|t-1}^{wb} \quad (28b)$$

$$\hat{m}_{t|t} = \hat{m}_{t|t-1} + \hat{v}_t \quad (28c)$$

- 5: end for

Figuur 33: *Pseudocode Extended Kalman Filter ter illustratie van de werking van het Magnetic SLAM algoritme. [47]*

opnemen van rotatie matrix $\hat{R}_{t|t-1}^{wb}$ corresponderend tot de oriëntatie $\hat{q}_{t|t-1}^{wb}$, het gemeten magneetveld Δy_t^b , de modi voor het magneetveld als functie van positie $\nabla \Phi_p(\hat{p}_{t|t-1}^w)$ en de magneetveldmatrix $\hat{m}_{t|t-1}$.

Vervolgens wordt het gelineairiseerd aan de hand van 28a tot en met 28c uit Figuur 33 uitgevoerd. Hier geldt dat

$$\hat{\delta}_t^w = p_t^w - \hat{p}_{t|t}^w. \quad (36)$$

Formule 36 laat zien dat de gefilterde positie schatting fout $\hat{\delta}_t^w$ bestaat uit de positie, p_t^w , op tijdstip t en de positie, $\hat{p}_{t|t}^w$, op tijdstip t gegeven tijdstip t . In 28b Figuur 33 is de gefilterde oriëntatie schatting fout opgenomen als $\hat{\eta}_t^w$.

Tot slot is de gefilterde magneetveld fout opgenomen als \hat{v}_t^w zoals Formule 37 laat zien. Formule 37 stelt dat

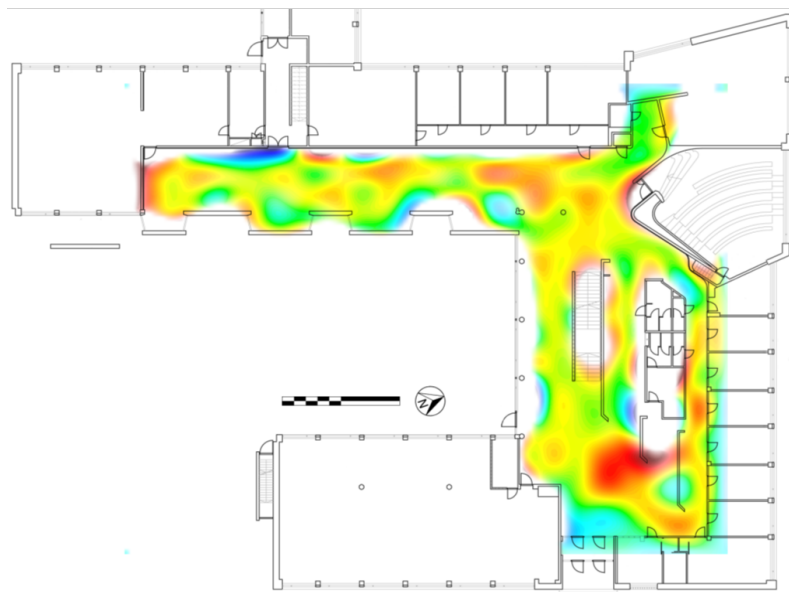
$$\hat{v}_t^w = m - \hat{m}_{t|t}^w. \quad (37)$$

De gefilterde magneetveld fout is afhankelijk van magneetveld matrix m en de magneetveld matrix op tijdstip t gegeven tijdstip t , $\hat{m}_{t|t}^w$.

Het Magnetic SLAM algoritme is gebaseerd op het vastleggen van het gemeten veld zoals besproken in Hoofdstuk 7.1 en het hierboven besproken Extended Kalman Filter [47]. Met de theorie van Magnetic SLAM nu behandeld is de volgende stap het bespreken van enkele voorbeelden waarna het algoritme implementeert kan worden.

7.3 Voorbeelden Magnetic SLAM

Bij Magnetic SLAM wordt gebruik gemaakt van een magneetsensor voor het waarnemen van het magnetisch inductieveld. Met behulp van deze data en het hierboven besproken Magnetic SLAM-algoritme wordt het mogelijk om het magneetveld te inter- en extra- te poleren waarna een kaart gemaakt kan worden. Een voorbeeld van een magnetische kaart is opgenomen in Figuur 34.



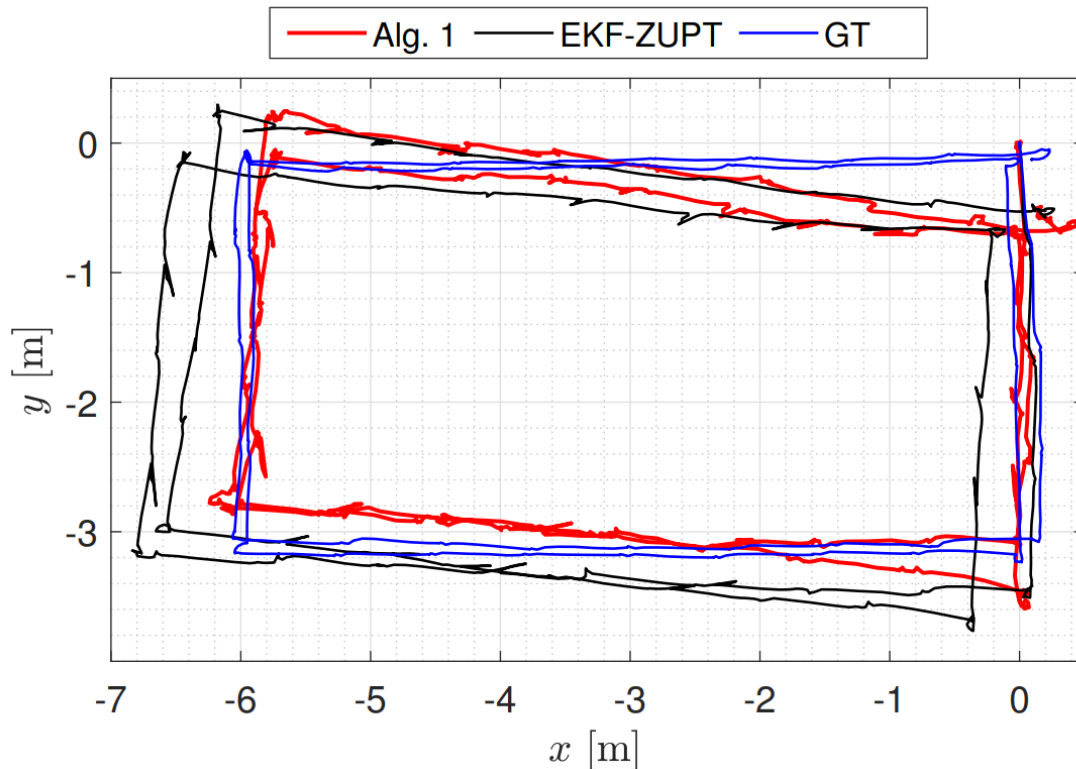
Figuur 34: Voorbeeld magnetische kaart waarbij de variaties in het magnetisch inductie-veld zijn opgenomen aan de hand van een kleurschaal. Met in donkerblauw 60 tot 30 tot donkerrood 60. [6]

Figuur 34 laat het geïnterpoleerde magneetveld sterkte zien variërend van 30, in het donkerblauw, tot 60, donker rood, μT . Zoals te zien is vindt er veel variatie plaats in de sterkte van het magneetveld. Het doel is om een soort gelijk figuur te kunnen maken met behulp van de Turtlebot en de benodigde sensoren.

Het gemeten magnetisch inductieveld zal er voorzorgen dat aan de hand van het algoritme een positie te bepalen is waar geen of minder drift plaatst vind, dan wanneer de positie alleen bepaald wordt aan de hand van de IMU. Figuur 35 laat een voorbeeld zien waarin drie verschillende afgelegde paden zijn opgenomen. In Figuur 35 is de afgelegde pad weergegeven voor drie verschillende methodes.

De rode lijn uit Figuur 35 beschrijft het afgelegde pad bepaald met het algoritme. De zwarte lijn representeert de positie bepaald aan de hand van het EKF-ZUPT algoritme. Dit algoritme simuleert de odometry data van deze toepassing. De ground truth, het werkelijke afgelegde pad, is in de voorbeeldkaart in Figuur 35 opgenomen in het blauw.

Figuur 35 laat zien dat alleen de odometry positie, EKF-ZUPT, onvoldoende is om de positie te bepalen. Er vindt een duidelijk verschil plaats tussen de zwarte en de blauwe lijn. De oorzaak hiervan is de aanwezige drift. Deze drift is er uit te halen door middel



Figuur 35: Voorbeeld afgelegd pad kaart dat aan de hand van het algoritme tot stand zal komen. Met hierin het afgelegde pad bepaald aan de hand van het algoritme in het rood, EKF-ZUPT algoritme voor de odometry in het zwart en de ground thruth in het blauw [48]

van het opnemen van het magnetisch inductieveld. Deze positie opgenomen in het rood, komt al beter overeen met de ground truth.

Wanneer het algoritme toegepast zal worden zal de odometry direct uit de Turtlebot gebruikt worden om deze zwarte lijn te representeren. De ground truth zal gerepresenteerd worden door de positie bepaald aan de hand van de LIDAR om zo beide posities met elkaar te vergelijken.

7.4 Implementatie Magnetic SLAM

Met de voorbeelden nu besproken is de volgende stap het implementeren van het algoritme. Als basis is het algoritme, EKFMagSLAM van Frida Viset gebruikt. [42] Door middel van enkele aanpassingen is het mogelijk om met de Turtlebot magnetic SLAM uit te voeren.

Als eerst is gekeken naar de benodigde input parameters. Om zo een read-out voor de Turtlebot te schrijven die de benodigde data in de juiste vorm opslaat voor het EKFMagSLAM algoritme. De benodigde input argumenten met hun vorm en een korte toelichting zijn opgenomen in Tabel 9.

Tabel 9 laat zien dat er al een positie bekend moet zijn. Hiervoor kan, bij gebruik van de Turtlebot, gebruik worden gemaakt van de positie en oriëntatie schatting aan de hand van de odometry topic of bepaald worden aan de hand van de IMU-data. Beide bevatten drift die door middel van het toevoegen van de magneetvelddata verminderd/verholpen zal gaan worden.

Met de nu ingelezen data is de volgende stap het bepalen van de hyperparameters. Deze parameters zijn voor elke magnetische map identiek heb hebben grote invloed op het de kwaliteit van het Magnetic SLAM algoritme. Aangezien het ene moment bijvoorbeeld

Tabel 9: Tabel met de benodigde input variabelen voor het uitvoeren van het EKFMagSLAM algoritme van Frida Viset [42] toegepast voor MSLAM met de Turtlebot3.

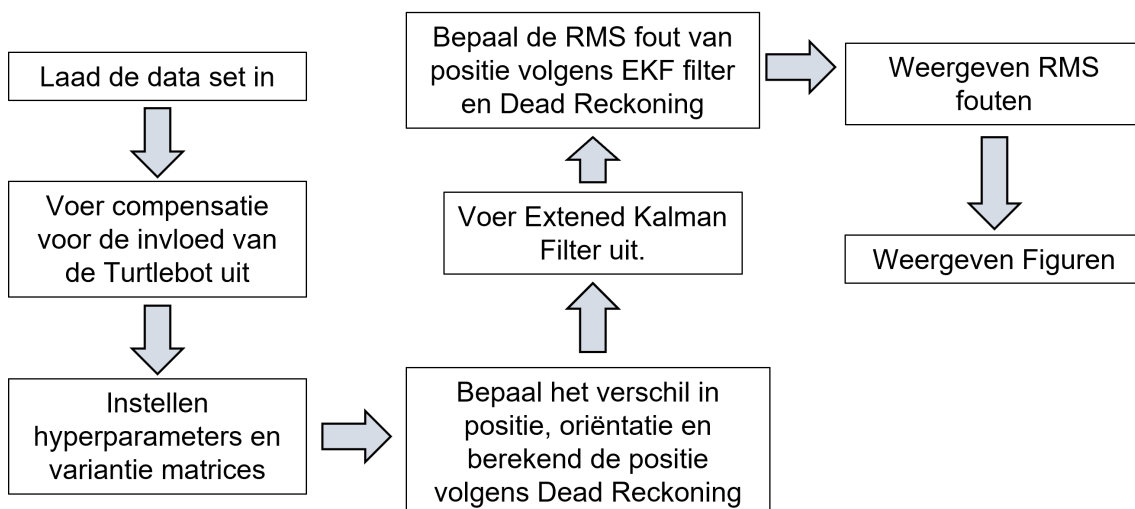
Variabele	Toelichting	Vorm
N	Aantal samples	1x1
p_odom	Initiele positie schatting	3xN
p_0	Begin positie schatting	3x1
q	Initiele orientatie schatting	4xN
q_0	Begin orientatie	4x1
y_mag	Magneetsensor data	3xN

de het afgelegde pad groter of kleiner is waardoor de grootte van het domein dus ook veranderd. Tabel 10 laat de aan te passen hyperparameters zien met een toelichting en hun standaard waarde.

Tabel 10: Tabel met de aan te passen hyperparameters voor het uitvoeren van het EKFMagSLAM algoritme van Frida Viset [42] toegepast voor MSLAM met de Turtlebot3 samen met hun standaard waarde.

Hyperparameter	Toelichting	Standaard waarde
margin	Marge om pad voor het definiëren van het domein	1
N_m	Aantal basis functies dat gebruikt wordt	100
sigma_SE	Ruis behorend tot κ_{SE}	$1 \cdot 10^{-6}$
l_SE	Lengte schaal voor het bepalen van κ_{SE}	1
sigma_lin	Ruis behorend tot κ_{lin}	$1 \cdot 10^{-6}$
sigma_y	Ruis in het gemeten magneetveld in tesla	$1 \cdot 10^{-5}$

Nu bekend is welke data er nodig is, en de hyperparameters zijn uitgelegd is de volgende stap het uitvoeren van het algoritme. Figuur 36 laat het blokschema van de implementatie van het EKFMagSLAM algoritme van Frida Viset zien. Vervolgens zal aan de hand van dit blokschema de werking van het EKFMagSLAM algoritme worden uitgelegd. Hiervoor zijn geen speciaale Toolboxes nodig. Wel is het belangrijk dat de gebruikte dataset bestaat uit de variabele van Tabel 9 en dat de vorm hiervan overeenkomt zoals beschreven.



Figuur 36: Blokschema implementatie EKFMagSLAM algoritme van Frida Viset.

Als eerst wordt de dataset ingeladen. Het script van het EKFMagSLAM algoritme is zo gemaakt dat elke variabele uit de dataset gelijk in de goede functies wordt opgeno-

men. Vervolgens wordt de compensatie van de invloed van de Turtlebot op het te meten magnetisch inductieveld uitgevoerd.

Met de compensatie uitgevoerd dienen de hyperparameters ingesteld te worden en worden de variantie matrixen opgenomen voor het Extended Kalman Filter. Vervolgens wordt de positie en de oriëntatie in een functie geladen die hieruit het verschil tussen twee punten bepaald en de positie volgens Dead Reckoning bepaald.

Vervolgens wordt het Extended Kalman filter uitgevoerd waarna de Root Mean Square (RMS) fout wordt bepaald voor de positie volgens het Extended Kalman Filter (EKF), de positie uit de odometry topic en Dead Reckoning ten opzichte van de LIDAR positie. Dit het mogelijk wat over de kwaliteit van de positie bepaald volgens Magnetic SLAM te zeggen. Wanneer de positie volgens Magnetic SLAM geheel overeenkomt met de positie volgens LIDAR SLAM dan is de RMS fout nul. Door de RMS fouten van Magnetic SLAM en odometry met elkaar te vergelijken wordt duidelijk of het opnemen van een magneetsensor meer waarde biedt.

Tot slot wordt worden de RMS fouten weergegeven waarna de figuren worden geplot. Er wordt een figuur weergegeven wat iets wegheeft van Figuur 34 met hierin het pad volgens LIDAR en Magnetic SLAM, een figuur zoals opgenomen is in Figuur 35 en een figuur waarbij de positie in x en y richting voor LIDAR SLAM, Magnetic SLAM en odometry topic tegen elkaar wordt uitgezet als functie van het aantal samples. Het laatste figuur zal samen met de RMS fouten, in Hoofdstuk 8.3, gebruikt worden om de meerwaarde van de magneetsensor te beschrijven.

Aan de hand van de input argumenten uit Tabel 9 en het blokschema, Figuur 36, is het mogelijk om het EKFMagSLAM algoritme van Firda Viset uit te voeren. Voordat hier een correcte magneetveld kaart uitkomt moeten eerst de hyperparameters goed gekozen worden. De in te stellen parameters zijn samen met hun standaard waarde opgenomen in Tabel 10. Vervolgens is het dan mogelijk om na het verzamelen van de data Magnetic SLAM uit te voeren. De resultaten hiervan worden besproken in volgend hoofdstuk.

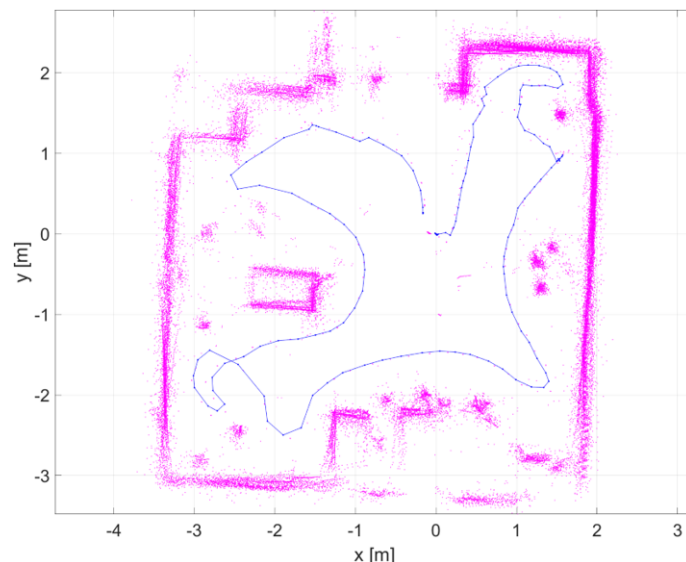
8 Resultaten & Discussie

De resultaten zijn de gemaakte kaarten zijn gemaakt door de methodes LIDAR SLAM en Magnetic SLAM. Als eerst worden de resultaten met betrekking tot LIDAR SLAM besproken gevolgd door de resultaten van Magnetic SLAM. Tot slot zullen beide methodes met elkaar vergeleken worden door het uitzetten van beide posities over tijd.

8.1 LIDAR SLAM

In deze paragraaf zijn kaarten gemaakt aan de hand van LIDAR SLAM opgenomen. De punten wolk die hier te zien zijn, zijn gemaakt aan de hand van de functie lidarSLAM in MATLAB [38]. Zowel de puntenwolken met hierin het afgelegde pad weergegeven als de gemaakte occupancy kaarten worden toegelicht.

De ruimte is door middel van LIDAR SLAM in kaart gebracht. Het resultaat hiervan is terug te zien in Figuur 37. Hierin is in het roze een punten wolk opgenomen die bestaat uit de data van de LIDAR. Het afgelegde pad van de robot is opgenomen in het blauw. Deze positie wordt ook aan de hand van de LIDAR data bepaald.



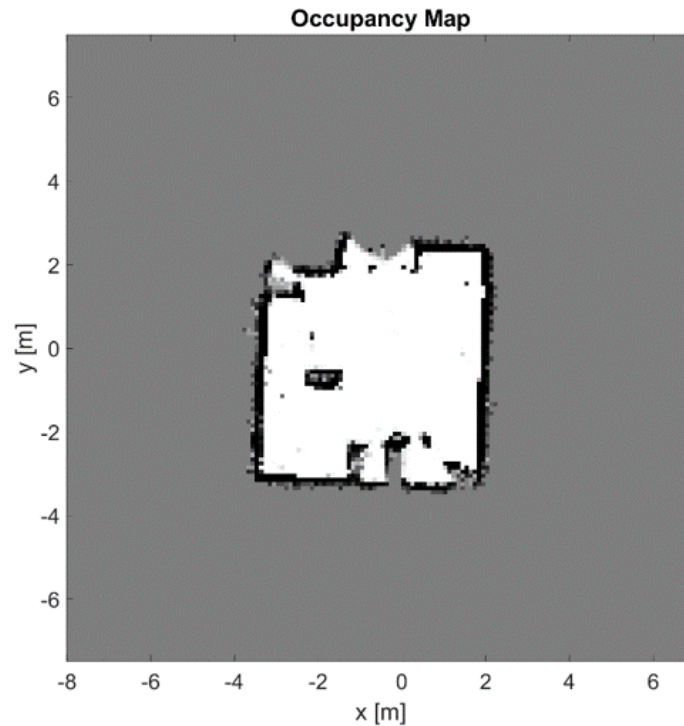
Figuur 37: *Punten wolk samen met het gereden pad geplot tegen een afstand in x en y richting in meter.*

Figuur 37 laat de ruimte zien met de punten wolk in het roze en het afgelegde pad in het blauw uit gezet tegen de afstand in x en y richting in meter. Zoals Figuur 37 laat zien zorgt het gebruik maken van LIDAR SLAM voor een nauwkeurige kaart van de ruimte. Ook laat Figuur 37 clusters van punten in de ruimte zien. Deze worden veroorzaakt door kleine object in de ruimte. Denk hierbij aan bureau en stoel-poten, kleine kasten of dozen.

De gemaakte kaart laat ook zien dat het mogelijk is om de muren achter objecten in kaart te brengen. Zolang er maar een lichtstraal kan komen. Dit is te zien rond punt (1,5; -2), waar het pad van de Turtlebot langs objecten komt, maar er toch nog voldoende licht wordt weerkaatst om achter het object in punt (0,75; -2) de muur in kaart te brengen.

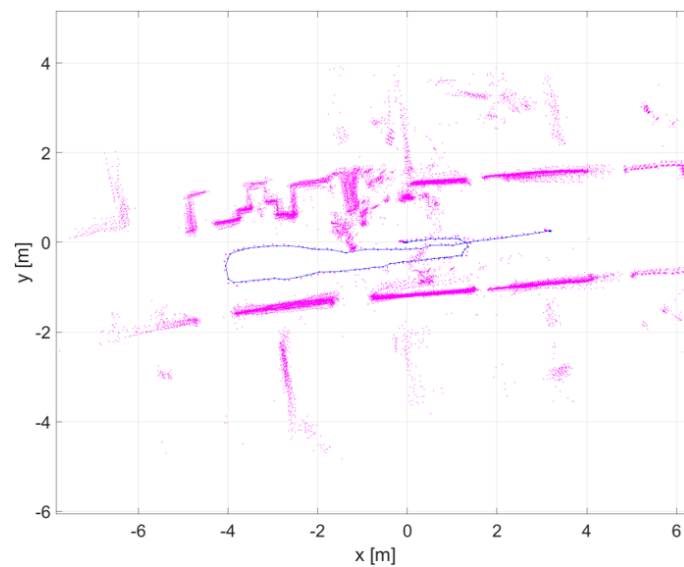
Vervolgens is de occupancy map van de punten wolk gegenereerd. Figuur 38 laat deze occupancy kaart zien. De occupancy map laat zien dat de ruimte vrijwel niet bezet is. De enkele zwarte stippen in de occupancy map duiden op de aanwezigheid van kleine objecten zoals bureau poten.

Figuur 39 laat een andere LIDAR kaart zien. Hierbij is een deel van de gang in kaart gebracht. De puntenwolk met het afgelegde pad is in Figuur 39 opgenomen. Figuur 39



Figuur 38: *Occupancy kaart behorend tot de data uit Figuur 37*

laat zien dat wanneer er genoeg punten in de wolk aanwezig zijn er een goede map en locatie bepaling uitgevoerd kan worden. Het is dus mogelijk om het de LIDAR de locatie van de Turtlebot te bepalen.



Figuur 39: *Puntenwolk met in het roze met het afgelegde pad in het blauw*

De positiebepaling aan de hand van de LIDAR is een goede manier om de locatie te bepalen. De positie aan bepaald met de LIDAR vormt hierdoor een goede standaard om het positie bepaald aan de hand van MSLAM mee te vergelijken. Wanneer er vanuit wordt gegaan dat de MATLAB functie 100% nauwkeurig is, dan hangt de nauwkeurigheid de kaart af van de nauwkeurigheid van de LIDAR. In dit geval zal de *ground truth* dan met een nauwkeurigheid van $\pm 5\%$ bepaald zijn.

8.2 Magnetic SLAM

Deze paragraaf laat de resultaten behaald met betrekking tot Magnetic SLAM zien. Denk hierbij aan de gemaakte kaarten maar ook een analyse over de nauwkeurigheid hiervan. De nauwkeurigheid van de positie bepaling aan de hand van het Magnetic SLAM-algoritme wordt uitgedrukt in de RMS waarde.

De RMS waarde vergelijkt de ground truth, LIDAR SLAM positie, met de positie volgens Magnetic SLAM. Ook wordt de RMS waarde voor de odometry opgenomen om zo het verschil in nauwkeurigheid duidelijk aan te geven. De RMS waarde van de positie volgens Magnetic SLAM zal dus vergeleken worden met de RMS waarde van de positie volgens odometry.

Voor het maken van de resultaten is het Magnetic SLAM-algoritme toegepast. De benodigde input variabelen worden door middel van read-out algoritme als eerst verkregen. Dit read-out algoritme slaat alle input variabelen samen op in een dataset. Deze code is opgeslagen onder de naam 'MSLAM_readout.mat'

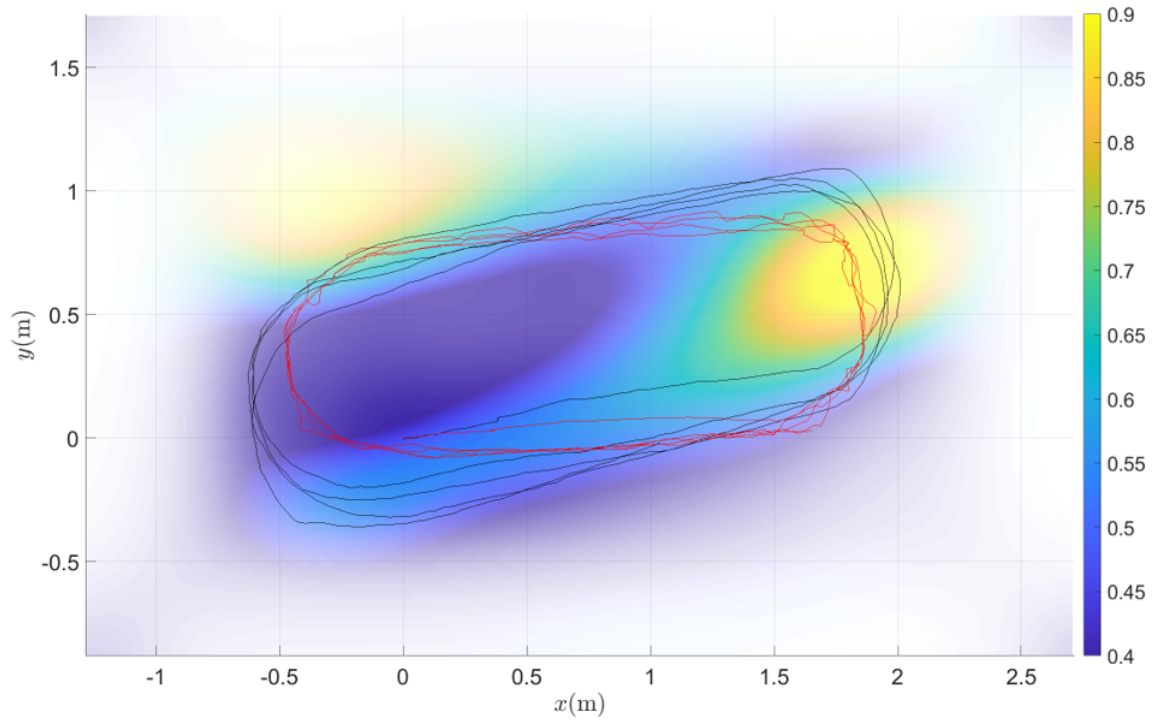
Het Magnetic SLAM-algoritme laad als eerst deze dataset in. Vervolgens is het mogelijk om de keuze te maken welke magneetsensor gebruikt wordt. De VMR zal voornamelijk gebruikt worden als magneetsensor. Ook is het mogelijk om de data verkregen door de magneetsensor van de Turtlebot zelf te gebruiken. De oriëntatie data kan verkregen worden aan de hand van de gyroscoop of uit de odometry topic. Aangezien de oriëntatie data uit de odometry topic binnen in het ROS netwerk bewerkt wordt zal de oriëntatie data uit de gyroscoop gebruikt worden.

Figuur 40 laat de magnetische kaart zien samen met het afgelegde pad volgens het algoritme in het zwart en het pad volgens LIDAR SLAM in het rood. Bij Figuur 40 is er gebruik gemaakt van de magneetsensor van de Turtlebot zelf. Hierbij zijn de hyperparameters als volgt gekozen: $\sigma_{SE} = 1$, $l_{SE} = 0,8$, $\sigma_{lin} = 1$ en $\sigma_y = 0,5$ G. Let op dat hier de hyperparameters nog in Gauss geven zijn aangezien het hier om een oud resultaat gaat. In Figuur 40 is nog niet gecorrigeerd voor de invloed van de Turtlebot op het te meten magneetveld.

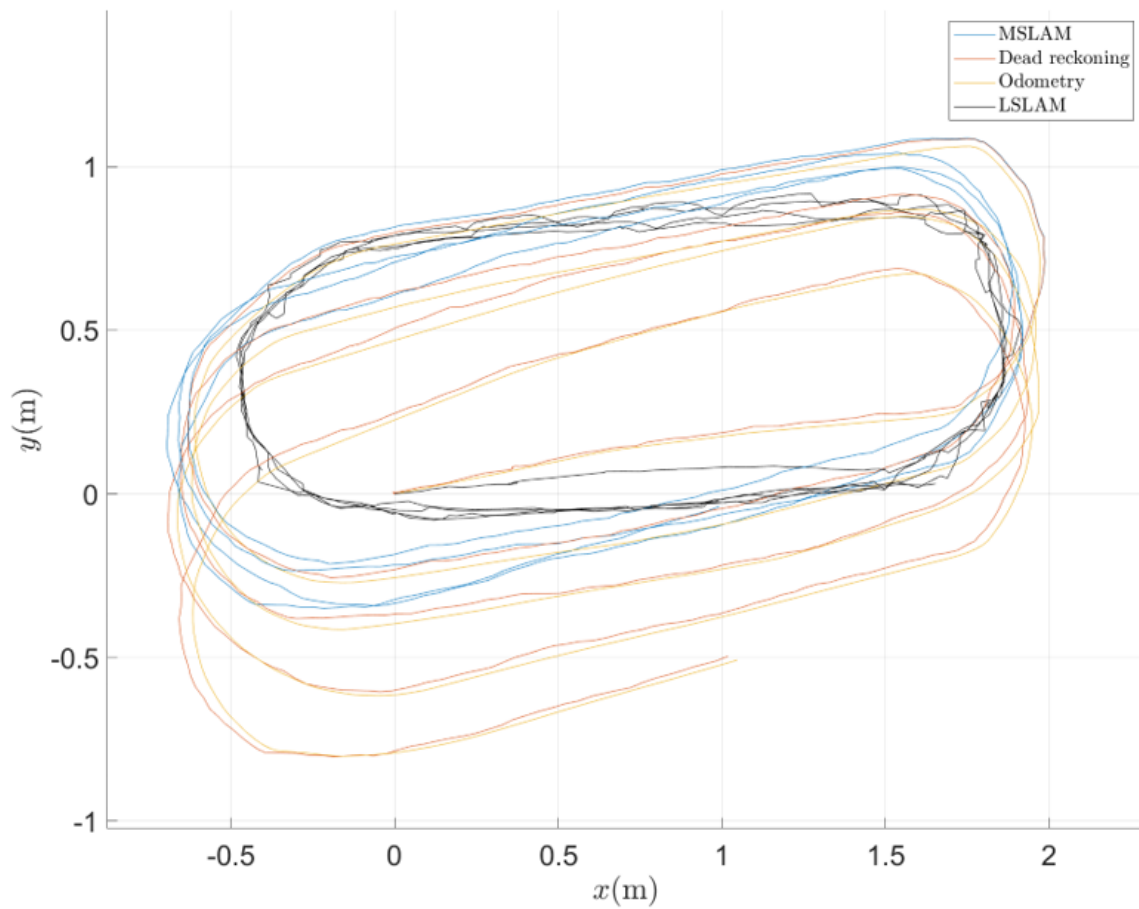
Naast de afgelegde paden is ook de sterkte van het magneetveld weergegeven in Figuur 40. De sterkte van het magneetveld is weergegeven in diverse kleuren waarbij elke kleur correspondeert tot een magneetveld met een sterkte gemeten in Gauss, ($1 \text{ G} = 1,0 \cdot 10^{-4} \text{ T}$). De kleurenschaal is aan de rechter zijkant van Figuur 40 opgenomen.

Naast het vergelijken van het gereden pad bepaald van de hand van MSLAM en LSLAM kunnen dit ook vergeleken worden met de positie bepaald a.d.h.v. dead reckoning [49] en de positie uit de odometry topic. Figuur 41 laat het afgelegde pad zien bepaald aan de hand van deze diverse methodes.

Figuur 41 laat zien dat de beginpositie voor elke methode gelijk is. Echter begint het afgelegde pad volgens de odometry positie en dead reckoning weg te lopen als gevolg van drift. Wanneer het magneetsensor wordt samengevoegd met de odometry positie, het MSLAM algoritme, is de drift minder aanwezig. Echter komt het afgelegde pad volgens magnetic SLAM nog niet overeen met het LIDAR SLAM-pad. Er wordt bij het optimaliseren van het MSLAM algoritme aangenomen dat de LSLAM posities de ground truth zijn.



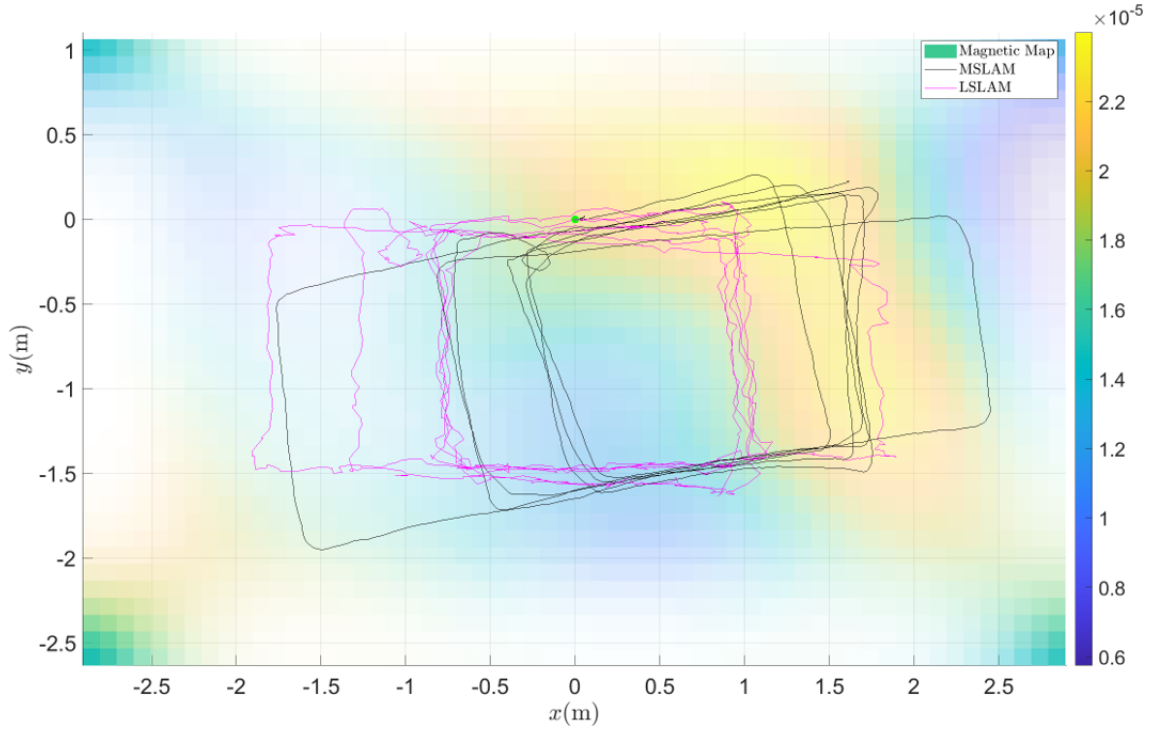
Figuur 40: *Magnetische kaart met het gereden pad volgens het Magnetic SLAM algoritme in het zwart en het pad volgens LIDAR SLAM in het rood samen met de sterkte van het gemeten magneteveld in een kleurenschaal in Gauss uitgezet tegen de afstand in x en y richting in meter.*



Figuur 41: *Het gereden pad volgens het Magnetic SLAM algoritme in het blauw, het pad volgens LIDAR SLAM in het zwart samen, dead reckoning in het oranje en de positie uit de odometry topic in het geel uitgezet tegen de afstand in x en y richting in meter.*

Vervolgens is er een magnetische kaart gemaakt waarbij de magneet-data verzameld is met de VMR, direct aan gesloten op de Turtlebot. Hierbij zijn alle hyperparameters omgezet naar tesla's en zijn als volgt: $\sigma_{SE} = 1 \cdot 10^{-6}$, $l_{SE} = 1$, $\sigma_{lin} = 1 \cdot 10^{-6}$ en $\sigma_y = 1 \cdot 10^{-5}$ T.

In tegenstelling tot Figuur 40 is hierbij wel gecorrigeerd voor de invloed van de Turtlebot op het te meten veld. Aangezien hierbij ook gemeten is met de Twinleaf VMR. De magnetische kaart samen met de positie volgens MSLAM en LSLAM is opgenomen in Figuur 42

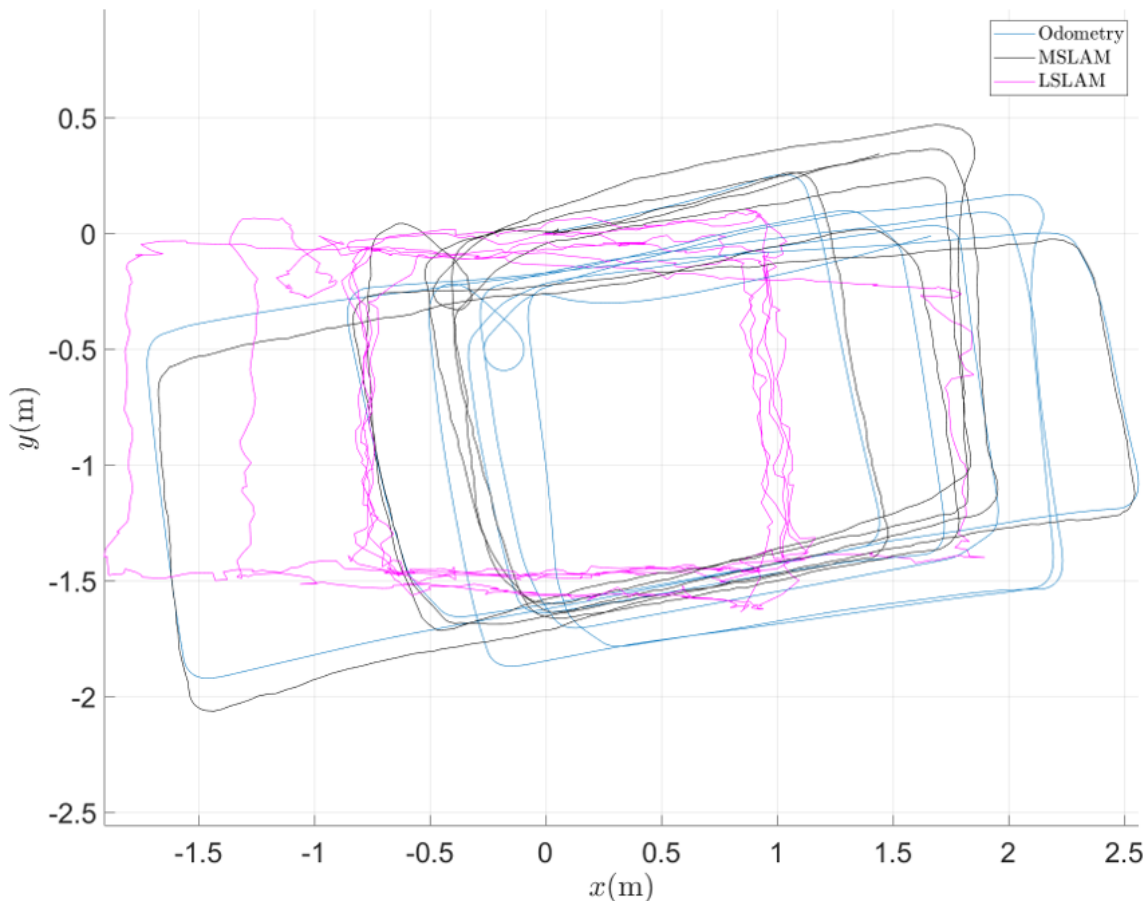


Figuur 42: Magnetische kaart met het gereden pad volgens het Magnetic SLAM algoritme in het zwart en het pad volgens LIDAR SLAM in het paars samen met de sterkte van het gemeten magneetveld in een kleurschaal in Tesla uitgezet tegen de afstand in x en y richting in meter.

Figuur 42 laat de magnetische kaart zien met de positie volgens MSLAM in het zwart, en de positie volgens LSLAM in het paars. Wanneer deze met elkaar vergeleken worden is ook hier weer een duidelijke draaiing zichtbaar. Door het beter te kunnen instellen van de hyperparameters wordt het vermoedelijk mogelijk om deze draaiing eruit te halen. Echter dienen deze erg zorgvuldig gekozen te worden aangezien deze ook verantwoordelijk zijn voor de variatie in de te maken magnetische kaart.

Aangezien er bij deze meting direct langs een permanente magneet is gereden, varieert het magnetisch inductieveld van $0,8 \cdot 10^{-5}$ T tot $2,4 \cdot 10^{-5}$ T zoals Figuur 42 laat zien. Echter kan een onjuiste keuze van hyperparameters ervoor zorgen dat deze magneet niet zichtbaar is. Zelfs met de volgende hyperparameters: $\sigma_{SE} = 1 \cdot 10^{-6}$, $l_{SE} = 1$, $\sigma_{lin} = 1 \cdot 10^{-6}$ en $\sigma_y = 1 \cdot 10^{-5}$ T blijft de magnetische kaart nog niet helemaal te kloppen. Zo bevond de permanente magneet zich rond $(0, -1)$ terwijl deze volgens Figuur 42 zich bevindt rond $(1, 0)$. Een goede hyperparameter keuze is dus van uiterst belang.

Wanneer de verschillende posities met elkaar vergeleken wordt, blijkt ook hier weer de drift in de positie bepaald aan de hand van MSLAM aanzienlijk minder maar alsnog aanwezig is. Figuur 43 laat zien dat de positie volgens MSLAM, in het zwart, in punt $(1; -1,5)$ bijna 0,15 m minder ver weg drift ten opzichte van de positie bepaald aan de hand van de odometry, in het blauw. De positie volgens de odometry heeft namelijk een y waarde van $-1,65$ bij de dezelfde x -coördinaat als het punt $(1; -1,5)$.



Figuur 43: Het gereden pad volgens het Magnetic SLAM algoritme in het zwart, het pad volgens LIDAR SLAM in het paars samen en de positie uit de odometry topic, verkregen uit o.a. de IMU, in het blauw uitgezet tegen de afstand in x en y richting in meter.

Met resultaten voor zowel LIDAR SLAM als Magnetic SLAM nu behandeld is de volgende stap het vergelijken van LIDAR SLAM en Magnetic SLAM om zo antwoord te krijgen op de hoofdonderzoeksvraag: **”Biedt een magneetsensor meerwaarde bij het autonoom navigeren in een binnen omgeving, naast het gebruik van LIDAR en IMU?”**.

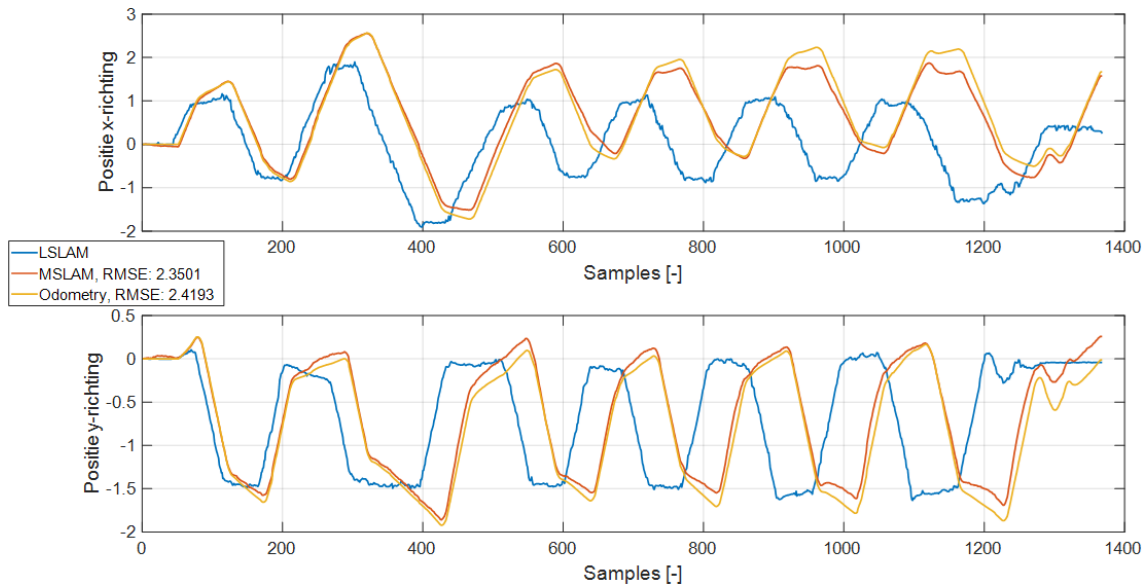
8.3 LIDAR SLAM vs Magnetic SLAM

In deze paragraaf wordt aan de hand van het resultaat opgenomen in Figuur 43 antwoord gegeven op de hoofdonderzoeksvraag. Dit zal gedaan worden door de twee verschillende posities bepaald door Magnetic SLAM en de positie uit de odometry topic te vergelijken met de *ground truth* verkregen aan de hand van LIDAR SLAM.

Magnetic SLAM bepaald de positie zoals beschreven in Hoofdstuk 7. De positie uit de odometry maakt gebruik van de IMU waardoor er in de loop van de tijd drift ontstaat, als gevolg van het twee maal integreren van de versnelling naar positie. Figuur 44 laat de x en y posities uitgezet tegen de samples zien.

In Figuur 44 is duidelijk zichtbaar dat de positie volgens Magnetic SLAM en de odometry voor zowel in de x en y richting alsnog drift bevatten. De eind posities komen namelijk niet overeen met de eind positie volgens LIDAR SLAM. Ook vind er een verschuiving plaats bij de positie bepaald aan de hand van Magnetic SLAM en de odometry.

Figuur 44 laat ook de RMS fout voor Magnetic SLAM en de odometry zien. De RMS fout is een waarde voor de afwijking van de positie ten opzicht van de *ground truth*. Voor Magnetic SLAM is de RMS fout 2,3501 en voor de odometry komt dit neer op 2,4193.



Figuur 44: De x en y positie volgens het Magnetic SLAM algoritme in het rood, het pad volgens LIDAR SLAM in het blauw samen, odometry positie in het geel uitgezet tegen de samples. Met hierbij de Root Means Square fout (RMSE) opgenomen voor Magnetic SLAM en de odometry positie voor het totale pad.

De RMS fout van de Magnetic SLAM is kleiner dan de RMS fout van de odometry. De positie volgens Magnetic SLAM komt dus meer overeen met de positie volgens LIDAR SLAM dan de positie volgens de odometry. Dit is ook terug te zien in Figuur 43.

Het verschil in de RMS fouten is aanwezig maar nog klein. Ook is een RMS fout voor Magnetic SLAM van 0 gewenst aangezien in dit geval de positie volgens Magnetic SLAM gelijk is aan de *ground truth*. De positie volgens Magnetic SLAM is nauwkeuriger te krijgen, dus met een lagere RMS fout, door de hyperparameters te kiezen. Echter doordat er vijf hyperparameters zijn, is het volledig optimaliseren hiervan een lang proces. Dit is dan ook niet gelukt. Er wordt dan ook aangeraden om hier meer onderzoek naar te doen, om zo de RMS fout te verkleinen en de nauwkeurigheid van het Magnetic SLAM algoritme te vergroten.

Het opnemen van een magneetsensor, Magnetic SLAM positie, naast een IMU, odometry positie, biedt dus zeker meerwaarde zoals Figuur 44 laat zien. Door het opnemen van de magneetsensor is de drift, die in de positie volgens de odometry topic aanwezig is, verminderd. Wel dient voor een nauwkeurigere positie bepaling nog stappen gemaakt te worden in het bepalen van de hyperparameters. De hyperparameters veel invloed hebben de positie bepaling en het in kaart gebrachte magneetveld en de positie hierin.

9 Conclusie

Twee verschillende manieren van het in kaart brengen van een ruimte terwijl tegelijkertijd de positie wordt bepaald zijn besproken. Als eerst is LIDAR SLAM (LSLAM) behandeld. Deze methode maakt enkel gebruik van een LIDAR die aan de hand van een gereflecteerde lichtstraal, een afstand bepaalt waarna door middel van een algoritme de diverse scans worden samengevoegd om zo een kaart te maken.

Magnetic SLAM (MSLAM) maakt gebruik van een de Inertial Measurement Unit (IMU) en een magneetveld sensor. In dit geval is hiervoor een Twinleaf VMR gebruikt. Aan de hand van de metingen wordt door het MSLAM algoritme een magnetische kaart gemaakt waarin het afgelegde pad, de positie, uit bepaald wordt.

Ondanks dat de term SLAM, Simultaneous Localizing And Mapping, constant gebruikt wordt is het belangrijk om aan te geven dat, bij Magnetic SLAM geen echte SLAM is toegepast. Het uitvoeren van het Magnetic SLAM zoals in dit verslag is geschreven is gebaseerd op een voorafgaande meeting waarna de positie en het magneetveld in kaart wordt gebracht. Dit is dus niet simultaan, Simultaneous, uitgevoerd waardoor de Magnetic SLAM niet helemaal correct is.

9.1 Deelonderzoeksvragen

Met beide methodes behandeld en geoptimaliseerd, denk hierbij aan het compenseren voor de invloed van het platform, de Turtlebot3, op het te meten veld. Maar ook het onderzoek doen naar de kwaliteit van de gebruikte sensoren en het vergelijken van de positie bepaald aan de hand van MSLAM en LSLAM zijn behandeld. Hiermee is het mogelijk om antwoord te geven op de onderzoeksvragen. Als eerst wordt er antwoord gegeven op de deelvragen om zo op te bouwen naar het antwoord op de Hoofdonderzoeksvraag.

Als eerst is er onderzocht '*Hoe kan met behulp van de Turtlebot en een LIDAR een kaart worden gemaakt van de ruimte?*'. MATLAB biedt de mogelijkheid om te verbinden met het ROS netwerk waarop de Turtlebot is aangesloten. Hiermee wordt de benodigde data van de scan topic binnen gehaald. Deze data bestaat uit een intensiteit en een afstand, ook wel de range genoemd. Aan de hand van de range en de bijbehorende hoek, de lijst met mogelijke hoeken dient zelf geïnitieerd te worden, wordt het met de lidarSLAM functie van MATLAB [38] mogelijk om LIDAR SLAM toe te passen.

Deze functie geeft een roze puntenwolk terug samen met het afgelegde pad in het blauw. De puntenwolk is vervolgens om te zetten naar een zogeheten 'Occupancy map'. Dit is een kaart van de ruimte, gemaakt aan de hand van de afgelegde positie en de puntenwolk, waarin de bezetting van de ruimte opgenomen is aan de hand van een grijswaardeschaal. Zowel de Occupancy map als de puntenwolk representeren een kaart van de ruimte.

Vervolgens is het MSLAM algoritme onderzocht. Dit algoritme is dan ook gelijk het antwoord op de volgende twee deelvragen: '*Hoe kan de positie en oriëntatie van de magneet sensor in het globale coördinatensysteem worden bepaald om zo de magnetische metingen toe te voegen aan de kaart?*' en '*Hoe kan het magnetisch veld worden bepaald op plekken waar niet gemeten wordt?*'.

Het MSLAM algoritme construeert het gemeten magneetveld aan de hand van een aantal basisfuncties. Een lineaire combinatie van deze basisfuncties wordt samengesteld waardoor het geconstrueerde veld zoveel mogelijk overeenkomt met het gemeten veld. Door gebruik te maken van de oriëntatie, verkregen uit de IMU, weet het MSLAM algoritme in welke oriëntatie het magneetveld gemeten wordt.

De positie wordt vervolgens door middel van een Extended Kalman Filter, het EKFmag-

SLAM algortime van [50], gefuseerd met de positie uit de odometry topic. De odometry maakt gebruik van alle bewegingssensoren op de Turtlebot en ook dus de IMU. Deze positie is te onnauwkeurig om te gebruiken voor navigatie. Door het Extended Kalman Filter word er een zo goed mogelijke schatting van de positie gemaakt aan de hand van de magneetsensor en de IMU.

Ook geeft het Extended Kalman Filter de matrix n terug waarmee het gemeten magneetveld zo goed mogelijk beschreven kan worden als een lineaire combinatie van basisfuncties. Deze basis functies worden vooral over een bepaald domein bepaald. Dit domein wordt zo gekozen dat het gereden pad zich in het domein bevind met een ingestelde marge hier omheen. Zo bevind het gereden pad zich dus altijd in het zelfde domein waarover ook de basisfuncties uitgerekend worden. Dit maakt het mogelijk om op plekken waar de Turtlebot niet direct gereden heeft toch het magneetveld in kaart te brengen.

De laatste deelvraag '*Hoe kan er aan de hand van de gemaakte magnetische kaart worden genavigeerd?*' blijft onbeantwoord. Tijdens de afstudeeropdracht was er onvoldoende tijd beschikbaar om aan de hand van een magnetische kaart te navigeren. Het implenmenteren van Magnetic SLAM op de Turltebot3 en hierover kennis op nemen nam meer tijd in beslag waardoor het uitvoeren van navigatie aan de hand van een magnetische kaart niet meer mogelijk was.

Wel is het idee dat het mogelijk is om met de Turltebot3 MSLAM toe te passen en te navigeren aan de hand van een al eerder gemaakte magnetische kaart. Echter blijkt het aansturen van de Turtlebot vanuit MATLAB wat problemen met zich mee te brengen. Er wordt dan ook aan geraden om voor het navigeren aan de hand van een magnetische kaart een eigen ROS script te schrijven aangezien op deze manier veel meer mogelijk is. MATLAB biedt veel mogelijkheden met betrekking tot het aansturen van de Turtlebot, maar om echt goede navigatie toe te passen wordt er aan geraden om je eigen ROS node te schrijven in python, deze vervolgens uit te voeren om zo te navigeren aan de hand van een magnetische kaart. Dit maakt het vervolgens ook mogelijk om zelf de verwerking van de sensordata uit te voeren waardoor alles net even wat duidelijker en wellicht eenvoudiger wordt.

9.2 Hoofdonderzoeksvraag

Met de deelonderzoeksvragen beantwoord kan er antwoord gegeven worden op de hoofdonderzoeksvraag. Om hier antwoord op te kunnen geven zijn diverse kaarten gemaakt. Figuur 43 liet bijvoorbeeld het afgelegde pad bepaald aan de hand van verschillende manieren zien. Aan de hand van de resultaten kan er antwoord gegeven worden op de hoofdonderzoeksvraag: '*Biedt een magneetsensor meerwaarde bij het autonoom navigeren in een binnenomgeving, naast het gebruik van LIDAR en IMU?*'

Zoals te zien was zorgt het opnemen van een magneetsensor naast het gebruik maken van een IMU voor een positiebepaling waarbij de drift verminderd wordt. Zo loopt de positie in het punt $(1; -1, 5)$ waarbij de MSLAM positie gelijk is aan de LSLAM positie volgens de odometry 0,15 m weg.

Dat de positie bepaald aan de hand van MSLAM meet overeenkomt dan de positie volgens de odometry is ook terug te zien in de Root Means Square (RMS) fout. Voor de positie volgens het Magnetic SLAM algortime is de RMS fout 2,3501 en volgens de odometry 2,4193. Dit is een aanzienlijke verbetering. Echter is deze positie bepaling minder accuraat dan de positie bepaald aan de hand van LSLAM. Er bevind zich namelijk toch nog wat drift in de positie bepaald aan de hand van MSLAM.

Het is voor een goede positie aan de hand van MSLAM erg belangrijk dat de hyperparameters gekozen worden. Met een project specifiek gericht op het goed kiezen van

de hyperparameters is het dan ook zeker mogelijk om een hogere of wellicht de zelfde nauwkeurigheid te behalen als LSLAM.

Er moet hierbij wel vermeld worden dat de positie van MSLAM vergeleken wordt met LSLAM. De positie bepaald aan de hand van LSLAM wordt hierbij als 100% nauwkeurig beschouwd terwijl dit wellicht niet het geval is. Er wordt daarom ook aangeraden om beide methodes te vergelijken met een ground truth die op een andere manier bepaald wordt. Deze nieuwe ground truth dient met een zeer hoge nauwkeurigheid de positie te bepalen. Dit zo mogelijk moeten zijn met bijvoorbeeld een OptiTrack [51].

Kortom, het opnemen van een magneetsensor, in een geval waarbij geen LIDAR aanwezig is en alleen een IMU, biedt zeker meer waarde. De drift ontstaan door het integreren van de IMU is drastisch verminderd. Echter in het geval waarbij wel een LIDAR aanwezig is, biedt de magneetsensor met hyperparameter keuze geen meerwaarde in het verkrijgen van een nauwkeurigere positie. Wel wordt er verwacht dat na een hyperparameter onderzoek de magneetsensor zeker meerwaarde zal bieden naast een IMU en LIDAR.

In de toekomst, wanneer de hyperparameters beter gekozen worden, zal de magneetsensor zeker meerwaarde bieden onafhankelijk van de aanwezigheid van een LIDAR. Denk hierbij aan een situatie waarbij het mogelijk wordt om met de magneetsensor en de IMU in je smartphone te navigeren. De eerste keer dat iemand met zijn smartphone komt, wordt de magnetische kaart gemaakt en opgeslagen. Wanneer er vervolgens een ander iemand komt kan hij of zij navigeren aan de hand van de al bestaande kaart. Aangezien bijna iedereen tegenwoordig een smartphone gebruikt wordt er verwacht dat het in kaart brengen van het magnetieveld, bijvoorbeeld in heel Nederland, snel gebeurd zal zijn, waarna het vervolgens mogelijk is om met je smartphone zonder GPS alsnog te kunnen navigeren. Hierdoor is er geen verschil in indoor en outdoor-navigatie en loopt alles vlekkeloos in elkaar over.

Referenties

- [1] Wikipedia. Global positioning system. [Online]. Available: https://en.wikipedia.org/wiki/Global_Positioning_System
- [2] GPS-SYSTEEM.NL. Gps systeem, hoe het werkt, toepassingen en voordelen. [Online]. Available: <https://www.gps-systeem.nl/gps-systeem/>
- [3] Q. Zou, Q. Sun, L. Chen, B. Nie, and Q. Li, “A comparative analysis of lidar slam-based indoor navigation for autonomous vehicles,” 2020.
- [4] N. C. for Environmental Information. World magnetic model. [Online]. Available: <https://www.ngdc.noaa.gov/geomag/WMM/>
- [5] A. Canciani. Magnetic navigation. [Online]. Available: <https://www.gps.gov/governance/advisory/meetings/2018-12/canciani.pdf>
- [6] A. Solin, M. Kok, N. Wahlstrom, T. B Schon, and S. Sark, “Modeling and interpolation of the ambient magnetic field by gaussian processes.”
- [7] D. C. Giancoli, *Natuurkunde deel 2; Elektriciteit, magnetisme, optica en moderne fysica*. Pearson Benelux, 2016.
- [8] T. Vertregt, “Designing a search-algorithm that uses an autonomously moving swarm of drones to efficiently detect drifting steel shipping containers, using magnetic anomaly detection and simulated annealing.” Ph.D. dissertation, 2020.
- [9] Wikipedia. Poissonvergelijking. [Online]. Available: <https://nl.wikipedia.org/wiki/Poissonvergelijking>
- [10] “Magnetic dipole.” [Online]. Available: https://www.wikilectures.eu/w/Magnetic_dipole
- [11] W. Nederpel, “Magnetic interpolation using gaussian processes,” Ph.D. dissertation, 2020.
- [12] R. wiki. Why ros? [Online]. Available: <https://www.ros.org/blog/why-ros/>
- [13] “Handleiding turttebot.” [Online]. Available: https://emmanual.robotis.com/docs/en/platform/turtlebot3/sbc_setup/#sbc-setup
- [14] “Python twinleaf i/o.” [Online]. Available: <https://github.com/twinleaf/tio-python>
- [15] “rospy.” [Online]. Available: <http://wiki.ros.org/rospy>
- [16] Wikipedia. Centrale limietstelling. [Online]. Available: https://nl.wikipedia.org/wiki/Centrale_limietstelling
- [17] E. S. D. Calvetti, *Introduction to Bayesian Scientific Computing*. Springer Text, 2007.
- [18] “Mpu-9250 product specification revision 1.1.” [Online]. Available: <https://invensense.tdk.com/wp-content/uploads/2015/02/PS-MPU-9250A-01-v1.1.pdf>
- [19] NetinBag. Wat is een euler-hoek? [Online]. Available: <https://www.netinbag.com/nl/science/what-is-a-euler-angle.html>
- [20] automaticaddison. How to convert a quaternion to a rotation matrix. [Online]. Available: <https://automaticaddison.com/how-to-convert-a-quaternion-to-a-rotation-matrix/>

- [21] deplorablemountaineer. Introduction to quaternions for 3d rotation computations. [Online]. Available: <https://www.deplorablemountaineer.com/tutorial/mathematics/introduction-to-quaternions-for-3d-rotation-computations/?msclkid=ffa318b4cf8511ecb8f5da58b48ec75d>
- [22] Wikipedia. Distributive property. [Online]. Available: https://en.wikipedia.org/wiki/Distributive_property
- [23] —. Quaternion. [Online]. Available: <https://en.wikipedia.org/wiki/Quaternion>
- [24] sociamix. 10 mins gamedev tips - quaternions. [Online]. Available: <https://www.youtube.com/watch?v=1yoFjjJRnLY>
- [25] MathWorks. pwelch; welch’s power spectral density estimate. [Online]. Available: <https://nl.mathworks.com/help/signal/ref/pwelch.html>
- [26] “What is the /odom topic?” [Online]. Available: <https://answers.ros.org/question/342361/what-is-the-odom-topic/>
- [27] Twinleaf. Vmr magnetoresistive vector magnetometer. [Online]. Available: <https://twinleaf.com/vector/VMR/>
- [28] ICNIRP. Infrared radiation 780 nm - 1000 μm . [Online]. Available: <https://www.icnirp.org/en/frequencies/infrared/index.html>
- [29] Robotis. 23. appendix lds-01. [Online]. Available: <https://www.robotshop.com/media/files/pdf/robotis-360-laser-distance-sensor-lds-01-datasheet.pdf>
- [30] Precisa. What is the difference between accuracy and precision measurements? [Online]. Available: <https://www.precisa.co.uk/difference-between-accuracy-and-precision-measurements/>
- [31] “Betekenis idle.” [Online]. Available: <https://www.betekenis-definitie.nl/idle>
- [32] MathWorks. mldivide, \. [Online]. Available: <https://nl.mathworks.com/help/symbolic/mldivide.html>
- [33] J. N. Steven L. Brunton, *KutzData-Driven Science and Engineering*. Cambridge University Press, 2020.
- [34] T. B. Hugh Durrant-Whyte, “Simultaneous localisation and mapping (slam): Part i the essential algorithms,” pp. 1–9.
- [35] N. O. Service and N. O. A. A. U. D. of Commerce”. What is lidar? [Online]. Available: <https://oceanservice.noaa.gov/facts/lidar.html>
- [36] B. Sharma. What is lidar technology and how does it work? [Online]. Available: <https://www.ros.org/blog/why-ros/>
- [37] “Understanding slam using pose graph optimization — — autonomous navigation, part 3.” [Online]. Available: <https://www.youtube.com/watch?v=saVZtgPyyJQ>
- [38] MathWorks. lidarslam. [Online]. Available: <https://nl.mathworks.com/help/nav/ref/lidarslam.html>
- [39] —. Implement simultaneous localization and mapping (slam) with lidar scans. [Online]. Available: <https://nl.mathworks.com/help/nav/ug/implement-simultaneous-localization-and-mapping-with-lidar-scans.html>
- [40] —. Ros toolbox. [Online]. Available: https://nl.mathworks.com/help/ros/index.html?s_tid=CRUX_lftnav

- [41] ——. Navigation toolbox. [Online]. Available: <https://nl.mathworks.com/products/navigation.html>
- [42] F. Viset, R. Helmons, and M. Kok, “An extended kalman filter for magnetic field slam using gaussian process regression,” 2022.
- [43] A. Solin and S. Särkkä, “Hilbert space methods for reduced-rank gaussian process regression.”
- [44] T. Delft. Symmetrische matrices. [Online]. Available: <http://homepage.tudelft.nl/11r49/onderw0001/linalgwb/college/week26.pdf>
- [45] D. Lay and S. Lay, *Linear Algebra and Its Applications*, 3rd ed. Addison Wesley Publishing Company, 2005.
- [46] R. Haberman, *Applied Partial Differential Equations with Fourier Series and Boundary Value Problems*, 5th ed. Addison Wesley Publishing Company, 2013.
- [47] Wikipedia. Extended kalman filter. [Online]. Available: https://en.wikipedia.org/wiki/Extended_Kalman_filter
- [48] M. Osman, F. Viset, and M. kok, “Indoor slam using a foot-mounted imu and the local magnetic field,” 2022.
- [49] S. Edelkamp and S. Schrödl, *Heuristic Search: Theory and Applications*, 1st ed. Elsevier, 2013, ch. 17.
- [50] F. Viset. Ekfmagslam. [Online]. Available: <https://github.com/fridaviset/EKFMagSLAM>
- [51] OptiTrack. Optitrack for movement sciences. [Online]. Available: <https://optitrack.com/applications/movement-sciences/>
- [52] Robotis. Sbc setup. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/sbc_setup/#sbc-setup
- [53] ——. Opencr setup. [Online]. Available: https://emanual.robotis.com/docs/en/platform/turtlebot3/opencr_setup/#opencr-setup
- [54] “Kalman filter.” [Online]. Available: https://en.wikipedia.org/wiki/Kalman_filter
- [55] K. Law, A. Stuart, and K. Zygalakis, *Data Assimilation*, 1st ed. Dordrecht ZH, Nederland: Springer, 2015, ch. 4.1.
- [56] —, *Data Assimilation*, 1st ed. Dordrecht ZH, Nederland: Springer, 2015, ch. 4.2.2.
- [57] Z. H. Iram Noreen, Amna Khan, “A comparison of rrt, rrt* and rrt*-smart path planning algorithms,” 2016.
- [58] “Tree construction process of the rrt algorithm.” [Online]. Available: https://www.researchgate.net/figure/Tree-construction-process-of-the-RRT-algorithm-3_fig2_276089220
- [59] S. Mehdi. Mapping path-following for a two-wheeled robot. [Online]. Available: <https://medium.com/@sarim.mehdi.550/mapping-path-following-for-a-two-wheeled-robot-b8bd55214405>

Appendices

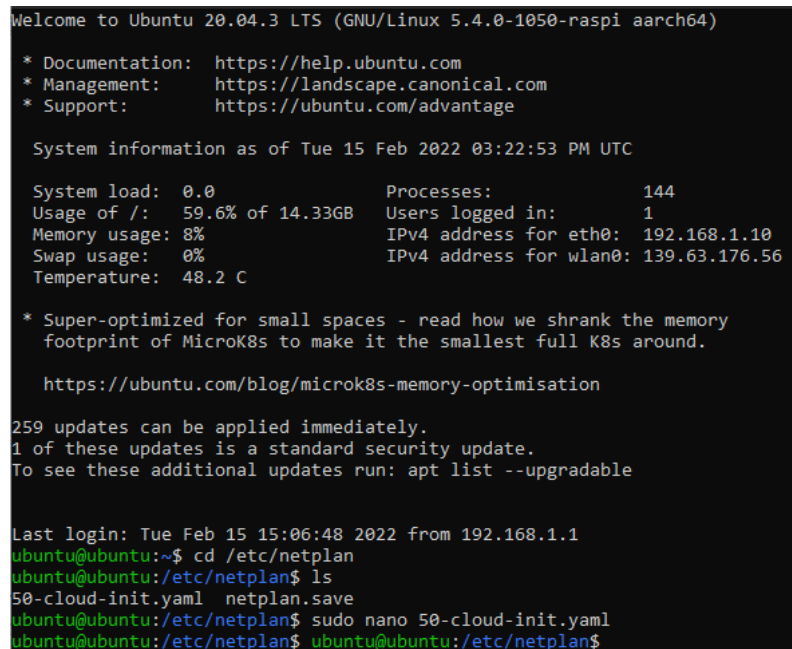
A Installatie van de Turtlebot

De Turtlebot wordt als bouw pakket geleverd. Als eerst dient deze in elkaar gezet te worden. Volg hiervoor het bijgeleverde handboek. Wanneer de Turtlebot in elkaar is gezet is de volgende stap het uploaden van de gewenste ROS-versie. Deze paragraaf beschrijft de installatie van ROS Noetic.

Als eerst dient de gewenste ROS-versie gedownload te worden. Let hierbij goed op of dat de versie die gekozen wordt compatibel is met de gebruikte Raspberry Pi. Wanneer het downloaden voltooid is, dient de map uitgepakt te worden waarna het .img bestand opgeslagen moet worden op de lokale opslag. Vervolgens dient de image op een SD-kaart gebrand te worden. Dit is onder andere te doen met: Raspberry Pi Imager. Dit programma is te downloaden via [52]. Ook is hier het gevolgde stappenplan te vinden.

Vervolgens kan de SD-kaart in de Raspberry Pi geplaatst worden. Sluit een toetsenbord en scherm aan op de Pi en zet de Turtlebot aan. Laat het systeem opstarten. Wanneer het systeem opgestart is kan er door middel van de gebruikersnaam: **'ubuntu'** en het wachtwoord: **'turtlebot'** worden ingelogd.

Om de Turtlebot te verbinden met het gewenste wifi netwerk dient het 50-cloud-init.yaml bestand te worden aangepast. Door het verbinden van de Turtlebot met wifi wordt het later mogelijk draadloos met de Turtlebot te communiceren. Ook wordt het hierdoor mogelijk om eenvoudig de sensordata over te dragen. Figuur 45 hieronder laat zien hoe dit bestand te bereiken is:



```
Welcome to Ubuntu 20.04.3 LTS (GNU/Linux 5.4.0-1050-raspi aarch64)

* Documentation:  https://help.ubuntu.com
* Management:    https://landscape.canonical.com
* Support:       https://ubuntu.com/advantage

System information as of Tue 15 Feb 2022 03:22:53 PM UTC

System load:  0.0               Processes:            144
Usage of /:   59.6% of 14.33GB   Users logged in:     1
Memory usage: 8%               IPv4 address for eth0: 192.168.1.10
Swap usage:   0%               IPv4 address for wlan0: 139.63.176.56
Temperature: 48.2 C

* Super-optimized for small spaces - read how we shrank the memory
  footprint of MicroK8s to make it the smallest full K8s around.

https://ubuntu.com/blog/microk8s-memory-optimisation

259 updates can be applied immediately.
1 of these updates is a standard security update.
To see these additional updates run: apt list --upgradable

Last login: Tue Feb 15 15:06:48 2022 from 192.168.1.1
ubuntu@ubuntu:~$ cd /etc/netplan
ubuntu@ubuntu:/etc/netplan$ ls
50-cloud-init.yaml  netplan.save
ubuntu@ubuntu:/etc/netplan$ sudo nano 50-cloud-init.yaml
ubuntu@ubuntu:/etc/netplan$ ubuntu@ubuntu:/etc/netplan$
```

Figuur 45: Bereiken van 50-cloud-init.yaml voor het instellen van het gewenste netwerk na het opstarten van de Turtlebot.

Met het geopende bestand is de volgende stap het toevoegen van de wifi SSID en het wachtwoord. Figuur 46 laat zien waar de SSID en het wachtwoord ingevuld dienen te worden. Na het aanpassen van het 50-cloud-init.yaml bestand kan het opgeslagen worden door de toets Ctrl+s en is het af te sluiten met Ctrl+z.

Vervolgens dient het commando `sudo netplan apply` uitgevoerd te worden om het netplan

```

network:
  version: 2
  renderer: networkd
  ethernet:
    eth0:
      dhcp4: yes
      dhcp6: yes
      optional: true
  wifi:
    wlan0:
      dhcp4: yes
      dhcp6: yes
      access-points:
        WIFI_SSID:
          password: WIFI_PASSWORD

```

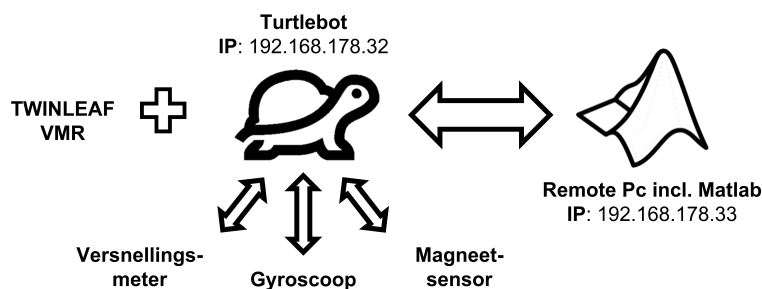
Figuur 46: Instellen van het netwerk SSID samen met het wachtwoord. [52]

te updaten waarna een de Turtlebot verbinding maakt met het ingestelde netwerk. Het netplan zoals ingesteld voor de gebruikte accespoint is opgenomen in bijlage I. Hierin is ook te zien hoe de Turtlebot met het eduroam netwerk verbonden kan worden.

Het is nu mogelijk om met een remote Pc, die verbonden is met hetzelfde netwerk als de Turtlebot, in te logen op te Turtlebot. Dit is te doen door gebruikt te maken van het commando: `ssh ubuntu@<IP_VAN_TURTLEBOT>`. Het bijbehorende wachtwoord is: `'turtlebot'`.

Tot slot moet de software op de OpenCR geïnstalleerd worden. Hiervoor kan de e-manual van Robotis gevolgd worden [53]. Vervolgens kan er gewerkt worden aan de communicatie tussen de Turtlebot en MATLAB om zo, de Turtlebot aan te sturen.

Afwijkend van het stappenplan is de "ROS Network Configuration". In tegenstelling tot het stappenplan dient een ROS_MASTER_URI ingesteld die gelijk is aan `http://<IP_RASPBERRY_PI>:1131`. Voor de situatie in Figuur 47 betekent dit dus: `export ROS_MASTER_URI=http://192.168.178.32:11311`. Dit zorgt er voor dat de `~/.bashrc` file van de Turtlebot correct wordt ingesteld om zo een actieve verbinding te verkrijgen.



Figuur 47: Illustratie ter verduidelijking voorbeeld.

Figuur 48 hieronder laat de aangepaste `~/.bashrc` file zien. Zoals te zien is, zijn beide variabele nu gekoppeld aan het correcte IP adres. Ook dient hier het gebruikte LIDAR model opgenomen te worden. Afhankelijk van de LIDAR is dit LDS-01 of LDS-02. Tot slot dient het commando `source ~/.bashrc` uitgevoerd te worden om de aanpassingen te verwerken.

De laatste stap voor het verkrijgen van een actieve verbinding tussen MATLAB en de

```
# Replace {IP_ADDRESS_OF_REMOTE_PC} with the IP address of remote pc
# Both Remote PC and Raspberry Pi should be connected in the same local network
export ROS_MASTER_URI=http://192.168.178.32:11311
# Replace {IP_ADDRESS_OF_RASPBERRY_PI} with the IP address of Raspberry Pi
export ROS_HOSTNAME=192.168.178.32
export LDS_MODEL=LDS-01
export LDS_MODEL=LDS-01
```

Figuur 48: Voorbeeld van `~/bashrc` file met de ingestelde variabelen.

Turtlebot is het aanpassen van de firewall. MATLAB dient toegang te krijgen om door de firewall heen te gaan om zo data te ontvangen en te versturen. Wanneer Windows Defender als firewall gebruikt wordt gaat dit als volgt. Open Windows Defender Firewall with Advanced Security navigeer naar 'Inbound Rules'. Hieronder zijn de instellingen van deze nieuwe 'Rule' opgenomen:

- **Rule Type:** Custom
- **Program:** All programs
- **Protocol en Ports:** Laat ongewijzigd
- **Scope:** Local IP is het IP adres van de computer met Matlab, Remote IP is het IP van de Turtlebot
- **Action:** Allow the connection
- **Profile:** Laat ongewijzigd
- **Name:** Matlab - Turltebot (Andere naam ook mogelijk, dit maakt het terugvinden eenvoudig)

Vervolgens dient er ook een 'Outbound Rule' gemaakt te worden met dezelfde instellingen als hierboven beschreven. Met een nu actieve verbinding is het mogelijk om de zogeheten 'Topics' uit te lezen.

B Specificaties IMU (MPU-9250)

Deze bijlage bevat alle volledige specificaties van de sensoren MPU-9250 IMU op de Turtlebot3 Waffle.

B.1 Gyroscop

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Full-Scale Range	FS_SEL=0		±250		°/s
	FS_SEL=1		±500		°/s
	FS_SEL=2		±1000		°/s
	FS_SEL=3		±2000		°/s
Gyroscope ADC Word Length			16		bits
Sensitivity Scale Factor	FS_SEL=0		131		LSB/(°/s)
	FS_SEL=1		65.5		LSB/(°/s)
	FS_SEL=2		32.8		LSB/(°/s)
	FS_SEL=3		16.4		LSB/(°/s)
Sensitivity Scale Factor Tolerance	25°C		±3		%
Sensitivity Scale Factor Variation Over Temperature	-40°C to +85°C		±4		%
Nonlinearity	Best fit straight line; 25°C		±0.1		%
Cross-Axis Sensitivity			±2		%
Initial ZRO Tolerance	25°C		±5		°/s
ZRO Variation Over Temperature	-40°C to +85°C		±30		°/s
Total RMS Noise	DLPFCFG=2 (92 Hz)		0.1		°/s-rms
Rate Noise Spectral Density			0.01		°/s/√Hz
Gyroscope Mechanical Frequencies		25	27	29	KHz
Low Pass Filter Response	Programmable Range	5		250	Hz
Gyroscope Startup Time	From Sleep mode		35		ms
Output Data Rate	Programmable, Normal mode	4		8000	Hz

Figuur 49: *Specificaties gyroscop.* [18]

B.2 Magnetometer

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
MAGNETOMETER SENSITIVITY					
Full-Scale Range			±4800		μT
ADC Word Length			14		bits
Sensitivity Scale Factor			0.6		μT / LSB
ZERO-FIELD OUTPUT					
Initial Calibration Tolerance			±500		LSB

Figuur 50: *Specificaties magnetometer.* [18]

B.3 Versnellingsmeter

PARAMETER	CONDITIONS	MIN	TYP	MAX	UNITS
Full-Scale Range	AFS_SEL=0		±2		g
	AFS_SEL=1		±4		g
	AFS_SEL=2		±8		g
	AFS_SEL=3		±16		g
ADC Word Length	Output in two's complement format		16		bits
Sensitivity Scale Factor	AFS_SEL=0		16,384		LSB/g
	AFS_SEL=1		8,192		LSB/g
	AFS_SEL=2		4,096		LSB/g
	AFS_SEL=3		2,048		LSB/g
Initial Tolerance	Component-Level		±3		%
Sensitivity Change vs. Temperature	-40°C to +85°C AFS_SEL=0 Component-level		±0.026		%/°C
Nonlinearity	Best Fit Straight Line		±0.5		%
Cross-Axis Sensitivity			±2		%
Zero-G Initial Calibration Tolerance	Component-level, X,Y		±60		mg
	Component-level, Z		±80		mg
Zero-G Level Change vs. Temperature	-40°C to +85°C		±1.5		mg/°C
Noise Power Spectral Density	Low noise mode		300		µg/√Hz
Total RMS Noise	DLPFCFG=2 (94Hz)			8	mg-rms
Low Pass Filter Response	Programmable Range	5		260	Hz
Intelligence Function Increment			4		mg/LSB
Accelerometer Startup Time	From Sleep mode		20		ms
	From Cold Start, 1ms V _{DD} ramp		30		ms
Output Data Rate	Low power (duty-cycled)	0.24		500	Hz
	Duty-cycled, over temp		±15		%
	Low noise (active)	4		4000	Hz

Figuur 51: *Specificaties versnellingsmeter.* [18]

C Specificaties LIDAR (LDS-01)

Items	Specifications
Operating supply voltage	5V DC $\pm 5\%$
Light source	Semiconductor Laser Diode($\lambda=785\text{nm}$)
LASER safety	IEC60825-1 Class 1
Current consumption	400mA or less (Rush current 1A)
Detection distance	120mm ~ 3,500mm
Interface	3.3V USART (230,400 bps) 42bytes per 6 degrees, Full Duplex option
Ambient Light Resistance	10,000 lux or less
Sampling Rate	1.8kHz
Dimensions	69.5(W) X 95.5(D) X 39.5(H)mm
Mass	Under 125g

Figuur 52: *Algemene specificaties LIDAR (LDS-01).* [29]

Items	Specifications
Distance Range	120 ~ 3,500mm
Distance Accuracy (120mm ~ 499mm)	$\pm 15\text{mm}$
Distance Accuracy(500mm ~ 3,500mm)	$\pm 5.0\%$
Distance Precision(120mm ~ 499mm)	$\pm 10\text{mm}$
Distance Precision(500mm ~ 3,500mm)	$\pm 3.5\%$
Scan Rate	$300 \pm 10 \text{ rpm}$
Angular Range	360°
Angular Resolution	1°

Figuur 53: *Meetbereik LIDAR (LDS-01).* [29]

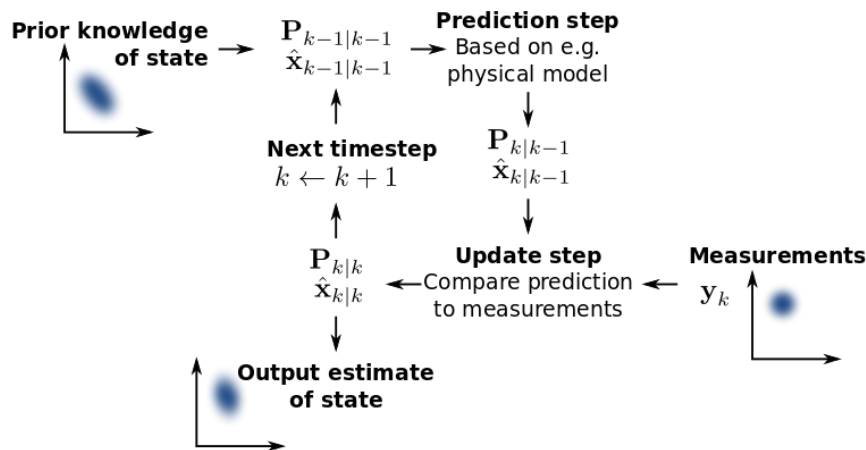
D Kalman Filter en Extended Kalman Filter

Kalman filter is een methode waarmee de ruis in een meeting wordt onderdrukt om zo tot een best passende uitkomst te komen. Een Kalman filter wordt toegepast op een proces dat gemodelleerd is. Het model voorspelt de waarde op elk moment. De meeting wordt verrekend met de voorspelde waarde tot een gewogen gemiddelde. De uitkomst hiervan wordt gelijk gebruikt voor het update van het model.

Een Kalman filter wordt specifiek gebruikt bij lineaire systemen. Bij niet-lineaire systemen dient een Extended Kalman Filter (EKF) te worden toegepast. Een EKF benaderd de non-lineariteit door middel van het nemen van de eerste en tweede afgeleiden. Kalman filters zijn breed toepasbaar aangezien ze in real-time voorspellingen kunnen doen. Zo worden ze onder andere gebruikt bij navigatie systemen om zo de positie van een voertuig te bepalen.

D.1 Kalman Filter

Het algoritme van een Kalman Filter bestaat uit een tweedelig proces. De voorspellingsfase schat het Kalman filter de huidige state variabelen samen met de bijbehorende onzekerheid. Wanneer het resultaat van de volgende meeting is waargenomen, worden deze geschatte variabelen geüpdatet met het gewogen gemiddelde. Hierbij wordt de waarde van geschatte punten met een kleine onzekerheid verhoogd.



Figuur 54: Schematische weergave van de werking van het Kalman filter. [54]

Een Kalman filter is te schrijven in de vorm van formule 38 hieronder.

$$\hat{x}_k = A\hat{x}_{k-1} + B\mu_k + K_k(y_k - C(A\hat{x}_{k-1} + B\mu_k)) \quad (38)$$

Het eerste deel, $A\hat{x}_{k-1} + B\mu_k$, voorspelt de huidige staat door het gebruikt te maken van de staat voorspelling van voorafgaand tijdstap en de huidige input. Dit deel is dan ook te schrijven als \hat{x}_k^- , ook wel de priori estimate genoemd. Substitutie hiervan geeft:

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-) \quad (39)$$

Het tweede deel gebruikt de meeting y_k en verwerkt dit in de voorspelling \hat{x}_k^- door een update. Het systeem wordt geüpdatet door de term $K_k(y_k - C\hat{x}_k^-)$. Hierin is K_k de Kalman Gain op tijdstip k . Het resultaat hiervan \hat{x}_k wordt de posteriori estimate genoemd.

De meting y_k is te schrijven als formule 40. Hierin wordt de meting beschouwd aan de hand van de measurement functional C , \hat{x}_k de posteriori estimate en de sensorruis v_k . Hierbij geldt dat de expectation gelijk is aan $E[v_k] = 0$ en $E[vv^T] = R$.

$$y_k = Cx_k + v_n \quad (40)$$

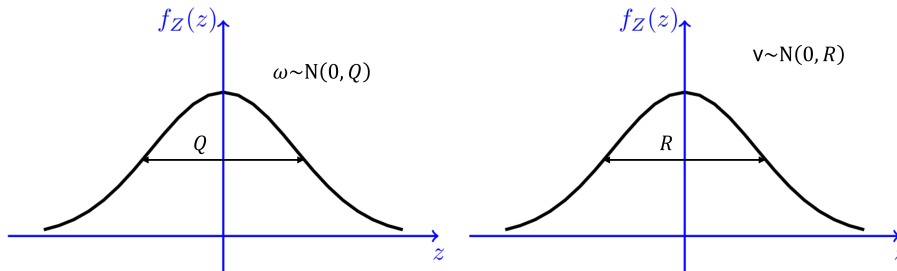
Tijdens voorspelling wordt het systeem model gebruikt om het een inschatting te maken van de staat in voorafgaand punt \hat{x}_k^- samen met de bijbehorende fout covariantie P_k^- . P_k^- kan gezien worden als een onzekerheid behorende tot de voorspelde staat. Deze onzekerheid ontstaat uit proces ruis en de propagatie van \hat{x}_k^- . Formules behorend tot de voorspelling zijn opgenomen in formule 41 en 42

$$\hat{x}_k^- = A\hat{x}_{k-1} + B\mu_k + G\omega_k \quad (41)$$

Hierin is A de systeemmatrix, \hat{x}_{k-1} de a priori inschatting, B de control matrix, μ_k de control input en $G\omega_k$ de systeem ruis. Belangrijk is dat aangenomen mag worden dat elke ruis Gaussisch verdeeld is.

$$P_k^- = AP_{k-1}A^T + Q \quad (42)$$

Belangrijk is dat de ruis als gaussisch verdeeld, wordt beschouwd. Dit houdt in dat voor de sensor en model ruis ω en v geldt dat $\omega \sim N(0, Q)$ en $v \sim N(0, R)$ met Q en R als covariantie als weergegeven hieronder.



Figuur 55: Schematische weergave verdeling ruis.

De prior estimate's worden tijdens de update fase gebruikt om uiteindelijk de posteriori estimate te bepalen. De Kalman Gain K_k wordt zo bepaald dat de posteriori error covariantie P_k^- geminimaliseerd wordt. De formules behorend tot de update fase zijn opgenomen in formule 43 tot 45. [55]

$$K_k = P_k^- C^T (C P_k^- C^T + R)^{-1} \quad (43)$$

$$\hat{x}_k = \hat{x}_k^- + K_k(y_k - C\hat{x}_k^-) \quad (44)$$

$$P_k = (I - K_k C) P_k^- \quad (45)$$

D.2 Extended Kalman Filter

Extended Kalman Filters (EKF) wijken iets af van de hierboven besproken theorie over de Kalman filters. Extended kalman filters werken in tegenstelling tot het hierboven besproken Kalman filter ook voor niet lineaire systemen. Hierdoor kan de staat en observatie geschreven worden als een differentieerbare functie zoals hieronder te zien is:

$$x = f(x_{k-1}, u_k) + \omega_k \quad (46)$$

$$z_k = h(x_k) + v_k \quad (47)$$

hierin zijn ω_k en v_k het proces en observatie ruis. Er wordt aangenomen dat zowel ω_k en v_k gaussisch verdeeld zijn rond om 0 met covariantie Q_k en R_k . De controle vector wordt hierin opgenomen als u_k . De voorspelvergelijkingen van een EKF wijken ook iets af. Voor een EKF is de voorspelde staat te benaderen als volgt:

$$\hat{x}_{k|k-1} = f(\hat{x}_{k-1|k-1}, u_k) \quad (48)$$

hierin is $\hat{x}_{k|k-1}$ \hat{x} op tijdstip k gegeven tijdstip k-1, $f(\hat{x}_{k-1|k-1}, u_k)$ de functie waarmee de staat voorspeld kan worden door middel een niet lineair systeem afhankelijk van $\hat{x}_{k-1|k-1}$ en de controle vector u_k . Hierbij behoort ook de voorspelde covariantie. Het voorspelde covariantie gemiddelde is met onderstaand verband te benaderen:

$$P_{k|k-1} = F_k P_{k-1|k-1} F_k^T + Q_k \quad (49)$$

hierin wordt de covariantie bepaald aan de hand van de afgeleide van functie f , de covariantie op vorige tijdstap k-1 gegeven k-1 en de procesruis Q_k . De volgende stap is het updaten van het systeem. De afgeleiden van de transitie en observatie functies f en h zijn gedefinieerd als volgt:

$$F_k = \frac{\delta f}{\delta x} \big|_{\hat{x}_{k-1|k-1}, u_k} \quad (50)$$

$$H_k = \frac{\delta h}{\delta x} \big|_{\hat{x}_{k|k-1}} \quad (51)$$

De Innovation of de measurement residual is te bepalen aan de hand de observatie op tijdstip k, z_k , en de voorspelde staat $\hat{x}_{k|k-1}$:

$$\tilde{y}_k = z_k - h(\hat{x}_{k|k-1}) \quad (52)$$

Vervolgens kan de covariantie op tijdstip k S_k , geüpdatet worden waarna het ook mogelijk wordt om de Kalman gain op tijdstip k K_k te bepalen. Hiervoor zijn onderstaande verbanden te gebruiken:

$$S_k = H_k + P_{k|k-1} H_k^T + R_k \quad (53)$$

$$K_k = P_{k|k-1} H_k^T S_k^{-1} \quad (54)$$

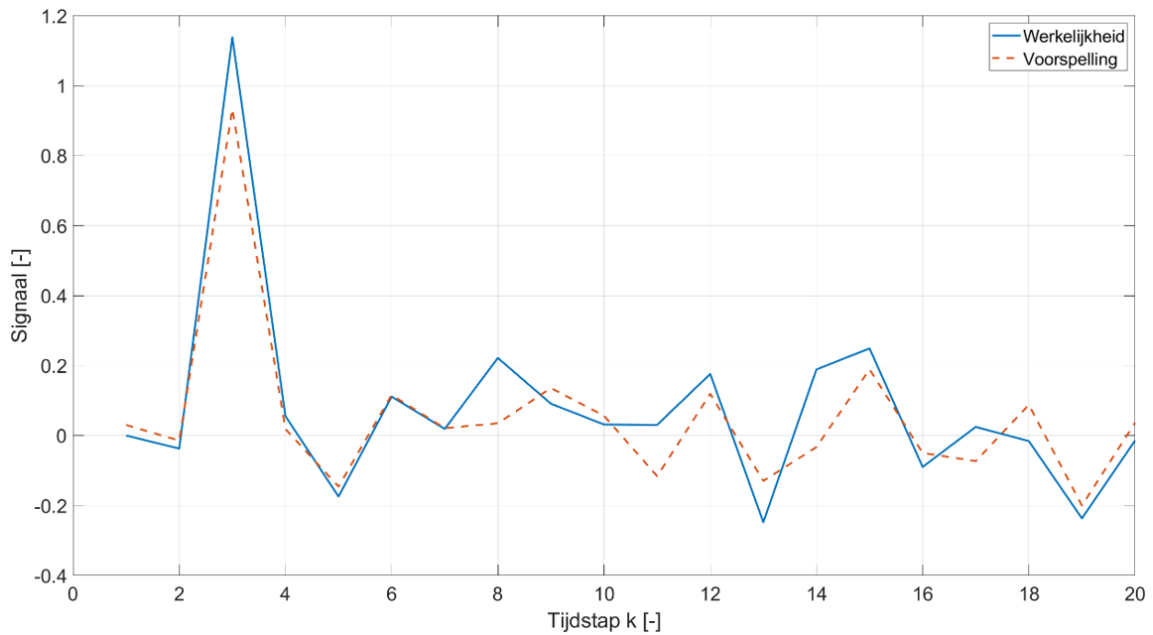
Uiteindelijk is het dan mogelijk om de staat en de covariantie te updaten waardoor $\hat{x}_{k|k}$ en $P_{k|k}$ bekend worden. De staat $\hat{x}_{k|k}$ wordt bepaald aan de hand van de voorspelde staat $\hat{x}_{k|k-1}$, de Kalman gain K_k en de measurement residual.

$$\hat{x}_{k|k} = \hat{x}_{k|k-1} + K_k \tilde{y}_k \quad (55)$$

De bijbehorende covariantie $P_{k|k}$ wordt bepaald aan de hand van I de ..., de Kalman gain, de observatie afgeleide en de voorspelde covariantie $P_{k|k-1}$. [47]

$$P_{k|k} = (I - K_k H_k) P_{k|k-1} \quad (56)$$

De beschreven formules 46 tot en met 56 maken het mogelijk om aan de hand van een signaal met ruis of drift een zo goed mogelijk inschatting te maken van de werkelijk waarde [56]. Figuur 56 laat een Extended kalman filter zien.



Figuur 56: Waarde van de staat uitgezet tijden de bijbehorende tijdstap voor de werkelijkheid en de geschatte waarde met behulp van een EKF.

Hierin is de waarde van de staat uitgezet tegen de tijdstap k voor zowel de werkelijkheid als de geschatte waarde met behulp van het Kalman filter. Zoals te zien is is de waarde van de staat redelijk te bepalen met behulp van het Kalman filter.

Zoals te zien was zijn Kalman filters te gebruiken voor lineaire systemen en werken Extended Kalman filters voor niet lineaire systemen. Een EKF lineariseert het niet lineaire systeem door het nemen van de afgeleiden. Hiervoor is het belangrijk dat de afgeleiden van dit systeem bestaan. Nu bekend is hoe een Kalman filter en een Extended Kalman filter beide werken is de volgende stap het toepassen hiervan in het algoritme wat uiteindelijk Magnetic SLAM mogelijk maakt.

E Extra

In deze bijlage is al het werk met betrekking tot de Turtlebot opgenomen wat waarde heeft voor de opdrachtgever maar wat minder dicht bij de rode draad van het onderzoek staat. Inzichten die tijdens het onderzoeken van onderstaande onderwerpen zijn verkregen hebben wel geleid tot de progressie die gemaakt is voor het uitvoeren van magnetic mapping met de Turtlebot3

E.1 RRT*

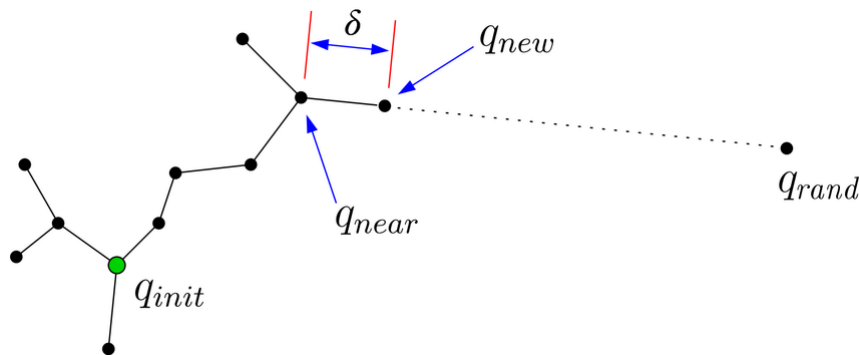
Aan de hand van een gemaakte occupancy kaart is het mogelijk om een pad tussen twee punten te genereren. Hiervoor wordt gebruik gemaakt van een RRT* algoritme. Het algoritme zoekt het pad zonder botsingen met de minste kosten van punt q_{init} naar punt q_{goal} .

E.1.1 Uitleg RRT*

RRT* staat voor Rapid exploring Random Tree star. Het algoritme maakt een random gegenereerde boom en begint in punt q_{init} , waarna de boom begint uit te breiden tot punt q_{goal} . Bij elke iteratie breidt de boom uit en wordt een nieuw punt geselecteerd q_{rand} . Als q_{rand} in een de vrije ruimte ligt wordt de dichtsbijzijnde node gezocht aan de hand van een ingestelde metric ρ . Deze node wordt punt q_{near} genoemd.

Wanneer q_{rand} binnen de ingestelde afstand, δ , vanaf q_{near} ligt, breidt de boom uit door deze punten met elkaar te verbinden. Is dit niet het geval wordt een nieuw punt q_{new} bepaald aan de hand van een stuur functie waarna de boom uitbreidt door de punten q_{near} en q_{new} met elkaar te verbinden.

Aan de hand van een 'Boolean collision checking process' wordt bepaald of het bepaalde pad mogelijk is of voor botsingen zorgt. Dit proces controleert aan de hand van booleans of het pad tussen q_{new} en q_{near} botsingvrij is. Zelfs wanneer een pad tussen q_{init} en q_{goal} gevonden is blijft het Boolean collision checking process zich herhalen tot vooraf bepaalde tijd verstrijkt of een aantal iteraties uitgevoerd zijn. [57]

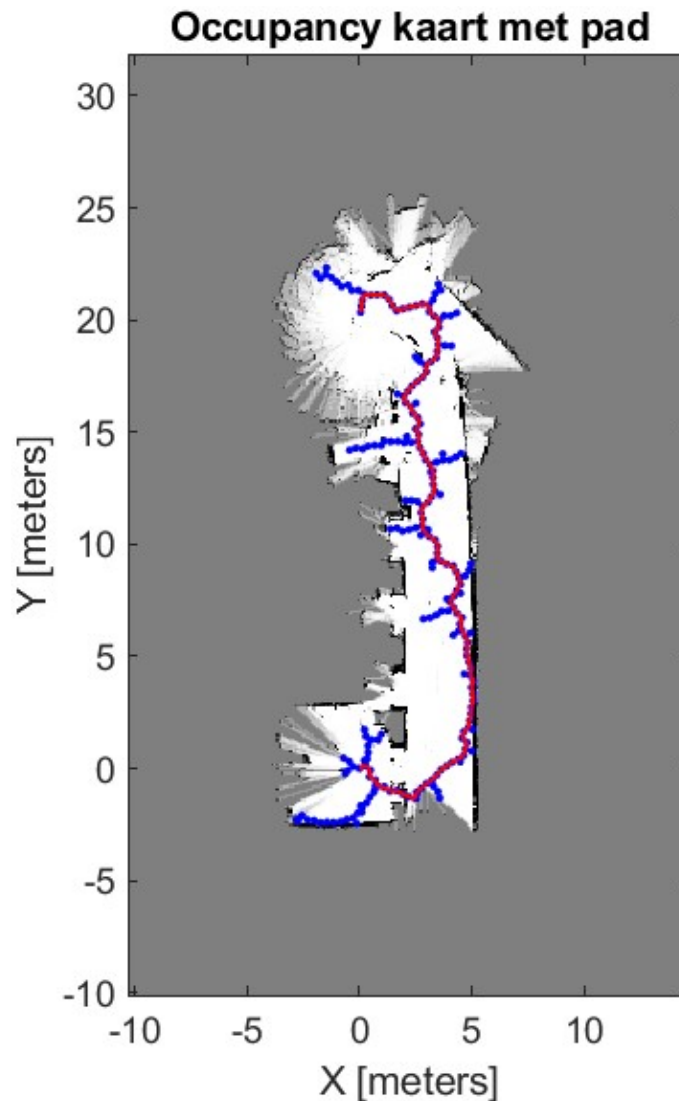


Figuur 57: Schematische weergaven uitbreiden boom RRT* padzoek algoritme. [58]

Uiteindelijk is het gemaakte pad te gebruiken om de bijbehorende snelheid v en hoeksnelheid ω te bepalen aan de hand van het dynamisch model. Hiermee wordt het mogelijk om de Turtlebot autonoom een route te laten afleggen aan de hand van een al gemaakte kaart.

E.1.2 Toepassen RRT*

Het RRT* algoritme is toegepast om een route tussen twee punten te bepalen aan de hand van onderstaande occupancy kaart. Hiervoor wordt de occupancy map geopend in Matlab waarna het algoritme geïnitialiseerd en uitgevoerd wordt. Het algoritme is toegepast om een pad te bepalen tussen punt (0,0) en punt (0,20). Het resultaat hiervan is opgenomen in figuur 58.



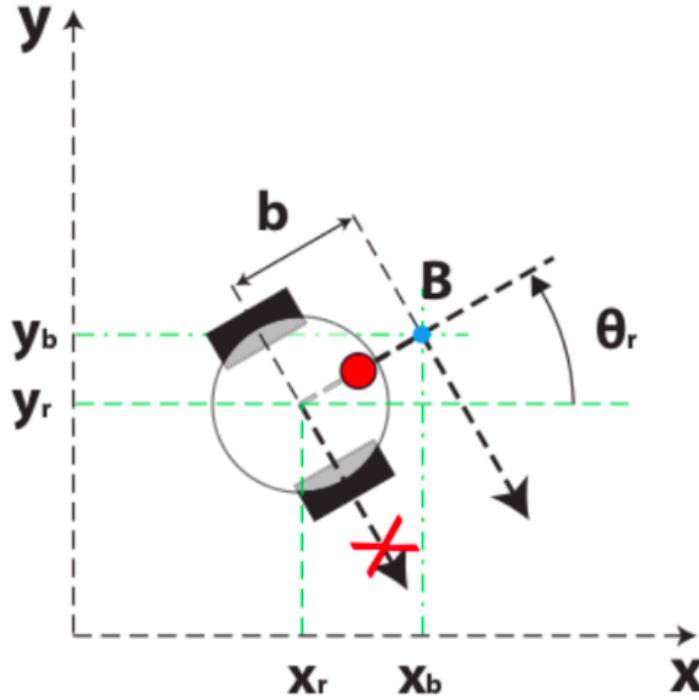
Figuur 58: Pad tussen (0,0) en (0,20) opgenomen in het rood bepaald door het RRT* algoritme.

Figuur 58 laat het uiteindelijke bepaalde pad zien in het rood. In het blauw is de boom opgenomen waaruit het meest kost effectieve pad bepaald is. Met het RRT* algoritme wordt het mogelijk om een pad tussen twee punten uit te rekenen waarna deze gebruikt kan worden om de Turtlebot te laten rijden. Hierdoor verkleint de toevallige fout aangezien de Turtlebot dan alleen nog maar rijdt aan de hand van een pad.

Ook maakt dit het mogelijk om de locatiebepaling aan de hand van de LIDAR te vergelijken. Door de Turtlebot een vooraf bepaald pad te laten rijden is de locatie hiervan vrijwel zeker. Door deze te vergelijken met de het pad bepaald wordt aan de hand van de LIDAR data valt wat over deze positie bepaling te zeggen.

E.2 Dynamisch model Turtlebot

De Turtlebot heeft slechts 2 wielen. Dit maakt het dynamisch model eenvoudig zoals te zien is in figuur 59. Het centrum van Turtlebot bevindt zich in (x_r, y_r) . De hoek θ_r geeft de rotatie van de Turtlebot ten opzichte de positieve x-as. Een punt B in (x_b, y_b) is te definiëren voor de robot. De afstand tot punt B is b lang, bevindt zich voor het middelpunt van de robot en ligt op de vector van de voorwaarts bewegende Turtlebot



Figuur 59: *Dynamisch model tweewielige robot.* [59]

Hiermee is punt B te bepalen aan de hand van het middelpunt van de Turtlebot. Punt B (x_b, y_b) is te beschrijven aan de hand van formule 57 en 58. Hierbij geldt: $b \neq 0$.

$$x_b = x_r + b \cdot \cos \theta_r \quad (57)$$

$$y_b = y_r + b \cdot \sin \theta_r \quad (58)$$

Aan de hand van de afgeleiden in positie (x_b, y_b) is de snelheid $(v_{x,b}, v_{y,b})$ in dit punt te bepalen. Dit leidt tot het verband in formule 59 en 60.

$$\dot{x}_b = v_{x,b} \quad (59)$$

$$\dot{y}_b = v_{y,b} \quad (60)$$

De afgeleiden van formule 57 en 58 resulteert in onderstaande formules:

$$\dot{x}_b = \dot{x}_r - b \cdot \omega \sin \theta_r \quad (61)$$

$$\dot{y}_b = \dot{y}_r + b \cdot \omega \cos \theta_r \quad (62)$$

Hierin is de afgeleiden van θ_r naar de tijd gelijk aan hoeksnelheid ω . Dit is de snelheid waarmee de Turtlebot draait. De afgeleiden van het middelpunt is gegeven door \dot{x}_r en \dot{y}_r . Vervolgens zijn formule 61 en 62 te herschrijven naar 63 en 64.

$$\dot{x}_b = v \cos \theta_r - b \cdot \omega \sin \theta_r \quad (63)$$

$$\dot{y}_b = v \sin \theta_r - b \cdot \omega \cos \theta_r \quad (64)$$

De lineaire snelheid van de Turtlebot is te definiëren als een vector die vanuit het midden van de Turtlebot gericht is in de richting van punt B. De horizontale en verticale componenten van deze vector geeft ons de afgeleiden van \dot{x}_r en \dot{y}_r . Hiermee zijn formule 63 en 64 om te zetten naar onderstaande matrix

$$\begin{bmatrix} \dot{x}_b \\ \dot{y}_b \end{bmatrix} = \begin{bmatrix} \cos \theta_r & -b \sin \theta_r \\ \sin \theta_r & b \cos \theta_r \end{bmatrix} \begin{bmatrix} v \\ \omega \end{bmatrix} \quad (65)$$

Aangezien de Turtlebot bestuurd kan worden met een input van de lineaire snelheid v en de hoeksnelheid ω wordt de inverse van formule 65 genomen om deze te bepalen. Dit resulteert in formule 66, het dynamisch model waarmee de Turtlebot te beschrijven is. Hierbij geldt nog steeds dat $b \neq 0$. Met het idee dat \dot{x}_b en \dot{y}_b bepaald worden voor een lijst coördinaten die het te rijden pad beschrijven [59].

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} \cos \theta_r & \sin \theta_r \\ -\frac{1}{b} \sin \theta_r & \frac{1}{b} \cos \theta_r \end{bmatrix} \begin{bmatrix} \dot{x}_b \\ \dot{y}_b \end{bmatrix} \quad (66)$$

F Veel gebruikte commando's

F.1 ROS

<code>ifconfig</code>	Verkrijgen van netwerk status
<code>sudo nano /etc/netplan/50.cloud-init.yaml</code>	Bewerken netwerkconfiguratie
<code>sudo nano ~/.bashrc</code>	Bewerken .bashrc file
<code>source ~/.bashrc</code>	Sourcen van de bashrc file
<code>sudo reboot now</code>	Reboot van de Raspberry pi
<code>roslaunch turtlebot3_bringup turtlebot3_robot.launch</code>	Opstarten van de Turtlebot
<code>rostopic list</code>	Verkrijgen lijst actieve topics
<code>rostopic list echo <topic></code>	Printen van topic output

*Locaties van bestanden kunnen afwijken afhankelijk van de gebruikte image op de Pi.
Bovenstaande locaties gelden voor image van Handleiding Turtlebot ROS versie Noetic
[13]

F.2 Matlab

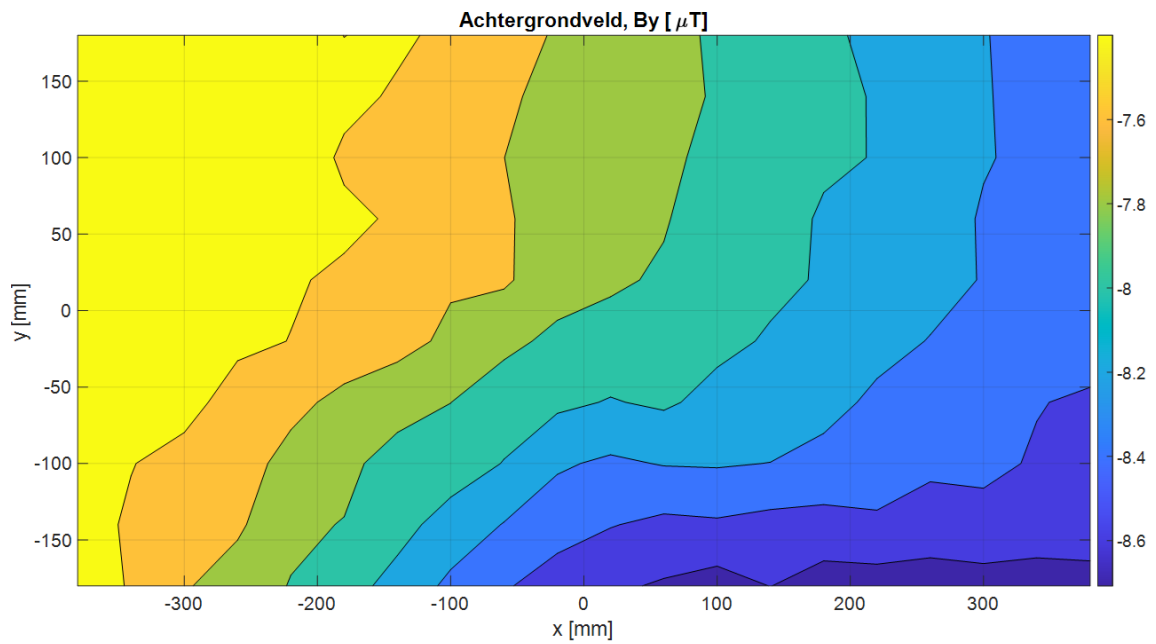
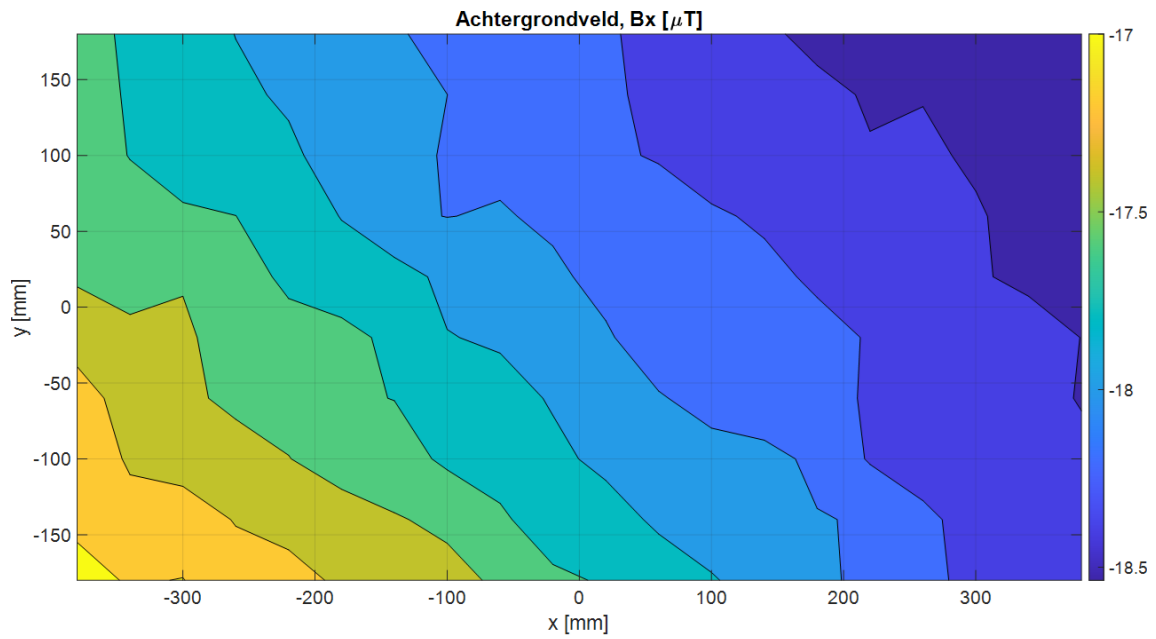
<code>rosinit('<IP_TURTLEBOT>')</code>	Verbinden turtlebot
<code>roscrate(<GEWENSTE_FREQUENTIE>)</code>	Instellen loop frequentie
<code>rossubscriber("<topic>", "DataFormat", "struct")</code>	Ontvangen van topic data
<code>rospublisher("<topic>", "DataFormat", "struct")</code>	Versturen van topic data
<code>rosmesssage("<message>", "DataFormat", "struct")</code>	Maken van een ROS message
<code>receive(<Subscriber>)</code>	Uitpakken topic data van subscriber
<code>lidarSLAM(<resolution>, <maxRange>)</code>	Initialiseren SLAM object
<code>lidarScan(<ranges>, <angles>)</code>	Maken plot
<code>addScan(<slamObj>, <scan>)</code>	Toevoegen plot aan SLAM object
<code>show(<slamObj>)</code>	Weergeven plot
<code>scansAndPoses(<slamObj>)</code>	Teruggeven positie LIDAR data
<code>buildMap(<scansSLAM>, <poses>, <resolution>, <maxRange>)</code>	Maken Occupancy kaart
<code>show(<occMap>)</code>	Weergeven Occupancy kaart

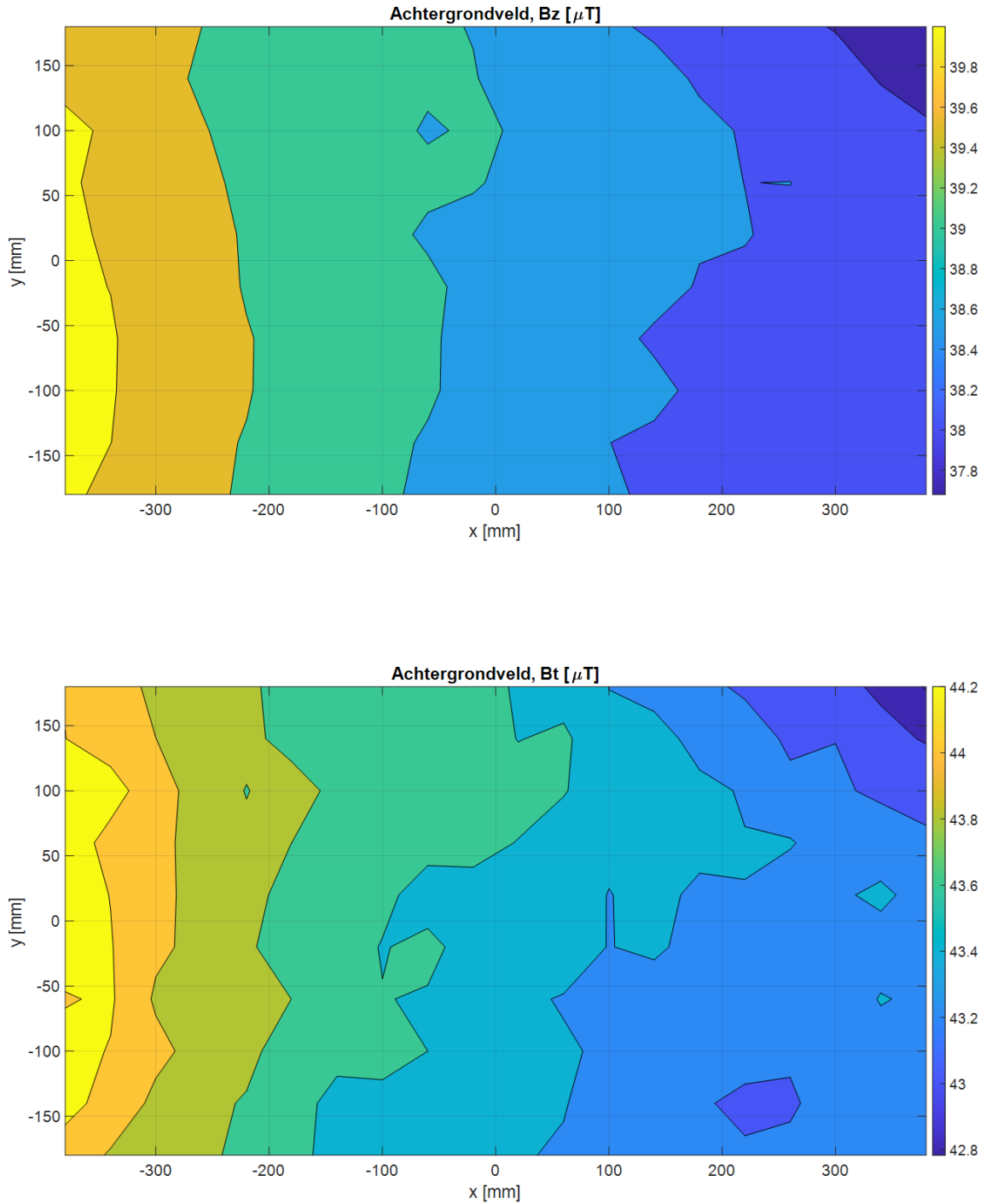
*Alles binnen <...> zijn variabelen.

G Datasheet Turtlebot

Items	Burger	Waffle Pi
Maximum translational velocity	0.22 m/s	0.26 m/s
Maximum rotational velocity	2.84 rad/s (162.72 deg/s)	1.82 rad/s (104.27 deg/s)
Maximum payload	15kg	30kg
Size (L x W x H)	138mm x 178mm x 192mm	281mm x 306mm x 141mm
Threshold of climbing	10 mm or lower	10 mm or lower
Expected operating time	2h 30m	2h
Expected charging time	2h 30m	2h 30m
SBC	Raspberry Pi 32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)	Raspberry Pi 32-bit ARM Cortex®-M7 with FPU (216 MHz, 462 DMIPS)
MCU		RC-100B + BT-410 Set (Bluetooth 4, BLE)
Remote Controller	-	
Actuator	XL430-W250	XM430-W210
LDS (Laser Distance Sensor)	360 Laser Distance Sensor LDS-01 or LDS-02	360 Laser Distance Sensor LDS-01 or LDS-02
IMU	Gyroscope 3 Axis Accelerometer 3 Axis	Gyroscope 3 Axis Accelerometer 3 Axis
Power connectors	3.3V / 800mA 5V / 4A 12V / 1A	3.3V / 800mA 5V / 4A 12V / 1A
Expansion pins	GPIO 18 pins Arduino 32 pin	GPIO 18 pins Arduino 32 pin
Peripheral	UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4	UART x3, CAN x1, SPI x1, I2C x1, ADC x5, 5pin OLLO x4
DYNAMIXEL ports	RS485 x 3, TTL x 3	RS485 x 3, TTL x 3
Audio	Several programmable beep sequences	Several programmable beep sequences
Programmable LEDs	User LED x 4	User LED x 4
Status LEDs	Board status LED x 1 Arduino LED x 1 Power LED x 1	Board status LED x 1 Arduino LED x 1 Power LED x 1
Buttons and Switches	Push buttons x 2, Reset button x 1, Dip switch x 2	Push buttons x 2, Reset button x 1, Dip switch x 2
Battery	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C	Lithium polymer 11.1V 1800mAh / 19.98Wh 5C
Firmware upgrade	via USB / via JTAG	via USB / via JTAG
Power adapter (SMPS)	Input : 100-240V, AC 50/60Hz, 1.5A @max Output : 12V DC, 5A	Input : 100-240V, AC 50/60Hz, 1.5A @max Output : 12V DC, 5A

H Achtergrond veld





Figuur 60: Gemeten achtergrond veld voor het plaatsen van de Turtlebot in CLAVIS. Waar het gemeten veld uit voor elk van de drie componenten x, y, z uitgezet is tegen de positie samen met het de totale sterkte B_t

I 50-cloud-init.yaml

```
GNU nano 4.8
# /etc/cloud/cloud.cfg.d/99-disable-network-config.cfg with the following:
# network: {config: disabled}
network:
  version: 2
  renderer: networkd
  ethernets:
    eth0:
      dhcp4: no
      addresses: [192.168.1.10/24]
      gateway4: 192.168.1.1
#   dhcp6: yes
#   optional: true
  wifis:
    wlan0:
      dhcp4: yes
      dhcp6: yes
      access-points:
#         "eduroam":
#             auth:
#                 key-management: eap
#                 method: peap
#                 identity: "tim.putto@tno.nl"
#                 password: "TNOstage1!"
      TurtleBot:
        password: TN02022!
```

Figuur 61: Netplan ingesteld om verbinding te maken met de accespoint, met de mogelijkheid tot het verbinden met het eduroam netwerk in het blauw en de mogelijkheid om te verbinden met een netwerkkabel waarbij de PC een IP adres heeft van 192.168.1.1

J Code; Uitlezen VMR

```
1 #!/usr/bin/env python3
2 import serial
3 import tldevice
4 import rospy
5 from std_msgs.msg import Float32MultiArray
6
7 ser = serial.Serial('/dev/ttyUSB1', 115200)
8 print(f'Port name = {ser.name}')
9 line = ser.readline()
10
11 def talker():
12     pub = rospy.Publisher('magneto', Float32MultiArray, queue_size=0)
13     rospy.init_node('talker')
14     rate = rospy.Rate(1)
15     a = Float32MultiArray()
16     count = 0
17
18     while not rospy.is_shutdown():
19         sensor = tldevice.Device('/dev/ttyUSB1')
20         print(f'Count={count}')
21         for row in sensor.data.iter():
22             #row = sensor.data.iter()
23             a.data = row
24             #print(f"Row = {row}")
25             pub.publish(a)
26             #rate.sleep()
27         del sensor
28         print('Deleted sensor object')
29
30     print('Rospy.is_shutdown became true')
31
32 if __name__ == '__main__':
33     try:
34         talker()
35         print('Talker finished')
36     except rospy.ROSInterruptException:
37         print('There was an exception')
```