Het leren van een goal-based navigatietaak op een mobiele robot

Bachelorproject

Egbert van der Wal, 1398652
Begeleider: Dr. Marco Wiering

Samenvatting: Lerende mobiele robots stellen ons in staat taken te laten overnemen door deze robots. Het trainen van de robots kan op verschillende manieren gedaan worden. In dit onderzoek worden de prestaties van twee van deze leermethoden bij de toepassing op een goal-based navigatietaak onderzocht, namelijk direct leren en reinforcement learning met het CACLA algoritme. Ook worden de prestaties van verschillende configuraties van direct leren bekeken. Bruikbare resultaten voor het CACLA algoritme zijn niet gevonden, maar wel worden enkele vergelijkingen gemaakt binnen de geteste configuraties van direct leren waaruit blijkt dat demonstraties waarbij de robot in de juiste richting is gedraaid om het doel zo snel mogelijk te bereiken waarschijnlijk voldoende zijn om de robot in staat te stellen om vanuit een willekeurige positie en richting naar het doel te navigeren.

1 Inleiding

De toepassing van mobiele robots wordt pas interessant binnen de kunstmatige intelligentie als ze (semi-) intelligent gedrag vertonen. Hiervoor zijn verschillende mogelijkheden. Het sturen van de robot met behulp van *neurale netwerken* wordt bijvoorbeeld veel toegepast om de robots een bepaald gedrag te laten vertonen. Bij een andere, minder gebruikte methode wordt het gewenste gedrag bepaald, geanalyseerd en in regels omgezet. Deze regels worden dan vervolgens omgezet in programmacode waarmee de robot aangestuurd zal worden. Een groot nadeel van deze methode is dat de robot uiteindelijk alleen in de omgeving zal kunnen functioneren waarbinnen de regels zijn opgesteld. Bovendien is de methode tijdrovend en vereist veel precisie en door deze precisie zal de robot uiteindelijk slecht met ruis om kunnen gaan. Deze methode is dus alleen bruikbaar in zeer constante omgevingen en bij zeer geringe ruis van de sensoren.

Dit onderzoek richt zich op het toepassen van neurale netwerken. Aangezien het gebruik van neurale netwerken een algemene techniek is om een computer iets te leren, met vele toepassingen op het gebied van de kunstmatige intelligentie, is het niet verbazingwekkend dat er hiervoor meerdere methoden ontwikkeld zijn. De eenvoudigste is het direct leren op de sensorwaarden. Een wat geavanceerdere techniek is *reinforcement learning*. Beide technieken kunnen gebruik maken van *leren van demonstratie*.

In dit onderzoek wordt een vergelijking gemaakt tussen de prestaties van beide methoden bij het aanleren van een goal-based navigatietaak, een taak waarbij het de bedoeling is dat de robot vanuit een willekeurige plaats in een bepaalde omgeving zo snel mogelijk een van tevoren bepaalde doelpositie bereikt. Ook worden de prestaties vergeleken bij het varieren van enkele parameters bij de toepassing van direct leren.

1.1 Eerder onderzoek

Smart en Kaelbling (2002) hebben in hun onderzoek aangetoond dat met behulp van direct leren in combinatie met *reinforcement learning* een robot getraind kan worden om een bepaalde taak uit te voeren. Atkeson en Schaal (1997) tonen ook aan dat leren van demonstratie werkt, al hebben zij om tijd te besparen de robot voorbereid door er vooraf een model in te stoppen dat aangepast moest worden. Hado van Hasselt en Marco A. Wiering (2007) hebben een andere methode van reinforcement learning onderzocht, het *CACLA algoritme*. Ook is er onderzoek gedaan naar reinforcement learning waarbij een mens een meer directe controle heeft over de toegekende rewards (Thomaz & Breazeal, 2008). Een duidelijke vergelijking tussen de verschillende methodes is nog niet gemaakt.

1.2 Onderzoeksvraag

De vragen die onderzocht worden zijn de volgende: "Welke van de onderzochte leermethoden werkt beter bij het aanleren van een goal-based navigatietaak, direct leren of het CACLA algoritme?" en "Hoe presteert elk van de onderzochte leermethoden op een goal-based navigatietaak?"

2 Leeralgoritmen

2.1 Neurale netwerken

Een neuraal netwerk is een sterk versimpelde kopie van de structuur van de menselijke hersenen. Voor dit onderzoek wordt een specifiek type gebruikt, *feedforward neurale netwerken* (Gurney, 1997). Hierbij zijn er verschillende *nodes*, gerangschikt in een aantal lagen, die onderling met elkaar verbonden zijn. Deze verbindingen hebben elk een gewichtswaarde die de invloed van een node op een andere node bepaalt. De eerste laag van het netwerk is de invoerlaag. Hieraan kunnen sensoren en andere invoer gekoppeld worden. Als deze nu een invoersignaal leveren, dan produceert het netwerk via alle lagen een uitvoer in de laatste laag, de uitvoerlaag. Deze laag kan dan weer gekoppeld worden aan de motoren van de robot. Alle kennis in het netwerk ligt opgeslagen in de gewichtswaarden van de verschillende verbindingen. Het leren in het netwerk vindt dus plaats door de aanpassing van de gewichtswaarden. Dit gebeurt in het algemeen door een aantal trainingsronden waarin een invoer aangeboden wordt en de uitvoer vergeleken wordt met de gewenste uitvoer. Komen deze niet overeen dan worden de gewichtswaarden aangepast om meer te lijken op de gewenste uitvoer met behulp van het zogenaamde *backpropagation algoritme*. Hierbij wordt de fout van de uitvoernodes, die eenvoudig berekend kan worden door de gewenste uitvoerwaarde van de daadwerkelijke uitvoerwaarde af te trekken, gebruikt om de fout van de nodes in de *hidden layer* te bepalen waarbij wordt uitgegaan van het principe dat als een node in de hidden layer een grote invloed op een uitvoer-node heeft, hij dus ook wel een grote invloed op de fout van die node zal hebben. De berekende aanpassing wordt vervolgens vermenigdvuldigd met de leersnelheid α die bepaalt hoe groot de daadwerkelijke aanpassing is. Voor dit algoritme is het een vereiste om bij een aantal invoeren een gewenste uitvoer te hebben. Als maat van de fout van een neuraal netwerk op een bepaalde trainingsset wordt vaak de *Mean Squared Error* (MSE) gebruikt. De MSE wordt berekend door voor elk trainingspaar de som van de gekwadrateerde verschillen tussen de uitvoer van het netwerk en de gewenste uitvoer te nemen en op basis daarvan het gemiddelde over de gehele trainingsset te berekenen. Het backpropagation algoritme heeft tot doel de MSE te minimaliseren. Een mogelijk probleem dat daaruit voortkomt is *overfitting*, het fenomeen waarbij door het langdurig trainen op een trainingsset het neurale netwerk zich teveel aanpast aan deze trainingsset en het generaliseringsvermogen verliest, wat juist een van de sterke punten van neurale netwerken is.

Om de uitvoer van een node te bepalen wordt een functie toegepast op de activatie van die node. Dit kan simpelweg de totale activatie opleveren, een zogenaamde lineaire uitvoer. Ook kan een niet-lineaire functie gebruikt worden. Hiervoor wordt meestal een *sigmoidfunctie* gebruikt. De sigmoidfunctie die bij dit onderzoek gebruikt werd is gegeven in formule 2.1.

$$output = \frac{1}{1 + e^{-x}} \tag{2.1}$$

2.2 Leren van demonstratie

Bij deze methode krijgt de robot een compleet voorbeeld van het gedrag dat vertoond dient te worden. Deze voorbeelden zijn over het algemeen demonstraties waarbij de robot bijvoorbeeld met het toetsenbord gestuurd werd. Het systeem heeft tijdens de leerfase geen invloed op het gedrag maar kijkt slechts toe en probeert in elke situatie zelf te verzinnen wat het moet doen. Wijkt dat af van de demonstratie, dan worden de gewichtswaarden aangepast. Na een aantal complete demonstraties is het systeem in staat om de controle over te nemen en zelf het gedrag te vertonen. Een goed voorbeeld van deze methode in de vorm van reinforcement learning werd gepresenteerd door Smart en Kaelbling (2002)

2.3 Direct leren

Direct Leren is het rechtstreeks toepassen van leren van demonstratie op een neuraal netwerk. Er worden een aantal demonstraties gedaan en de sensorwaarden met de daaraan gekoppelde gewenste uitvoer worden gebruikt om het netwerk te trainen. Uiteindelijk kan het netwerk worden gebruikt om aan de hand van de sensorwaarden de robot te sturen.

Direct leren wordt ook gebruikt als voorbereiding op een vorm van reinforcement learning (Smart & Kaelbling, 2002), (Chohra, Benmehrez & Farah, 1998).

2.4 Reinforcement Learning

Reinforcement Learning is een methode waarbij een *policy* gevormd dient te worden die de robot zo snel mogelijk naar het doel brengt (Russell & Norvig, 2003). Hierbij zijn *rewards* gedefinieerd. Dat is doorgaans een grote positieve reward voor het bereiken van de doeltoestand en een kleine negatieve reward voor elke stap die gedaan wordt en, indien van toepassing, een grotere negatieve reward voor grotere fouten die gemaakt worden. Aan de hand van de toekomstige rewards wordt

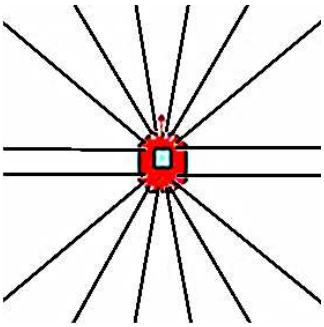
aan elke toestand een waarde gekoppeld die de rewards voorstelt die vanaf die toestand te verwachten zijn. Het leren gebeurt nu door de waarden van de toestanden te verhogen die tot een grotere positieve reward leiden en dus uiteindelijk tot het doel. Door vervolgens de acties te belonen die leiden tot een toestand met een hogere waarde, kan het systeem leren om in elke toestand de juiste actie te selecteren. Zoals de omschrijving doet vermoeden is dit systeem bij uitstek geschikt voor discrete toestanden en discrete acties, zodat voor zowel de acties als voor de toestanden een tabel met waarden en waarschijnlijkheden kan worden opgesteld.

Aangezien bij het aanvangen van het leerproces de tabel voor het toekennen van een waarde aan een toestand nog niet correct is, is het belangrijk dat niet altijd de voorgestelde actie wordt uitgevoerd, maar dat er ook genoeg *exploration* plaats vindt. Doorgaans wordt dit bepaald door een kans op een willekeurige actie, die bij aanvang hoog is en gedurende het leerproces steeds lager wordt.

Een combinatie van direct leren gevolgd door reinforcement learning is eerder onderzocht (Chohra et al., 1998). Ook is er onderzoek gedaan naar coöperatief leren bij een navigatietaak met meerdere robots (Melo & Ribeiro, 2008).

2.5 CACLA Algoritme

Het CACLA algoritme, het *Continuous Action Critic Learning Automaton* is een leeralgoritme voorgesteld door Hado van Hasselt en Marco A. Wiering(2007) en is een implementatie van reinforcement learning toegespitst op continue acties in continue omgevingen. Door het vervangen van de genoemde tabellen bij reinforcement learning voor discrete situaties door *Function Approximators* (FA) die aan de hand van de waarnemingen een voorspelling doen over de toestand, is het systeem geschikt te maken voor continue omgevingen. Deze FA's kunnen ingevuld worden door neurale netwerken. In dit geval worden er twee neurale netwerken gebruikt, waarvan er één aan de hand van de waarnemingen een waarde toekent aan de huidige situatie en de tweede aan de hand van de waarnemingen een actie voorstelt. Er wordt constant aangenomen dat de FA's optimaal zijn en dus zal de geselecteerde actie worden uitgevoerd. Blijkt dat de toestand die daardoor wordt bereikt een hogere waarde heeft dan de voorgaande waarde, dan wordt de actie die daartoe geleid heeft versterkt. In het geval van neurale netwerken gebeurt dat door backpropagation toe te passen met de invoer van de vorige toestand en de uitgevoerde actie als gewenste uitvoer. Om te zorgen dat altijd de



Figuur 3.1: De sonarlocaties op de robot

kortste weg naar het doel wordt gekozen, wordt echter niet de volledige waarde van de nieuwe toestand gebruikt maar wordt deze vermenigvuldigd met een *discount factor* γ, doorgaans iets minder dan 1.0. In elke toestand wordt altijd de waarde van de vorige toestand opnieuw berekend aan de hand van de huidige toestand, vermenigvuldigd met de discount factor en verhoogd met de reward verkregen in de huidige toestand, volgens formule 2.2.

$$V_t = R_{t+1} + \gamma \times V_{t+1} \tag{2.2}$$

Hierbij is V_t de waarde van de toestand op tijdstip t en R_{t+1} de reward verkregen door het bereiken van de toestand op tijdstip $t + 1$.

De FA voor het toekennen van waarden aan toestanden wordt getraind om aan de hand van de vorige waarnemingen de nieuw berekende waarde als uitvoer op te leveren, in het geval van neurale netwerken wederom met behulp van backpropagation. Het belangrijkste verschil met andere aanpakken voor reinforcement learning in continue omgevingen is dat de omvang van de verandering in waarde van de toestand niet bepalend is voor de omvang van de reinforcement van de actie in de voorgaande toestand, maar slechts het teken van die verandering.

In dit onderzoek werd een discount factor 0.99 gebruikt, stapkosten van 0.1, een reward van 100 voor het bereiken van het doel en een negatieve reward van 10 voor het naderen van een muur dichter dan 5 cm, wat gezien werd als een aanrijding.

3 Methode

3.1 Apparatuur

Alle tests zijn uitgevoerd in een simulatie met behulp van MobileSim versie 0.4.0 (`http://robots.mobilerobots.com/MobileSim/`), die een robot in een omgeving kan simuleren. De gesimuleerde robot is een model van de Pioneer

P2DX robot, zoals die ook bij de Rijksuniversiteit Groningen aanwezig zijn. De gesimuleerde robot heeft zestien sonars rondom gepositioneerd zoals in figuur 3.1. Elk van deze sonars geeft de afstand tot het dichtstbijzijnde obstakel door, tot een maximum van 1072.0 mm. Verder heeft de robot twee wielen die elk afzonderlijk kunnen worden ingesteld op een snelheid tussen de -500 en de 500 mm/s. Het geheel werd aangestuurd door de ARIA library versie 2.5.1 (`http://www.activrobots.com/SOFTWARE/aria.html`), waarbij de aansturingsprocedure *fire* tien keer per seconden wordt aangeroepen om de volgende actie van de robot te bepalen. Bovendien werd de ArActionStallRecover module van de ARIA library gebruikt die op het moment dat beide motoren ongewenst stilstaan ten gevolge van bijvoorbeeld een botsing een uitwijkmanoeuvre uitvoert om de stilstand op te heffen.

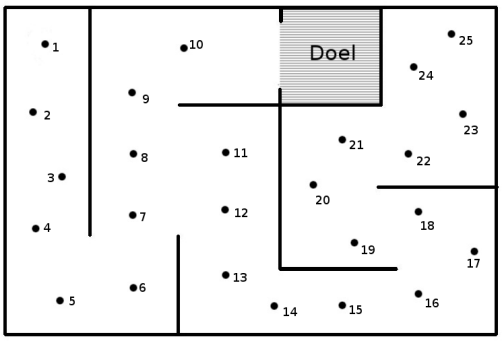
De gesimuleerde omgeving is een omgeving met enkele muren en gangen en een doelgebied dat door de robot bereikt dient te worden. De omgeving is te zien in figuur 3.2.

3.2 Stimuli / invoer

De stimuli, ofwel de waarden die als invoer aan het systeem werden gegeven, zijn de waarden van de zestien sonars rondom. Deze waarden werden door de software van tevoren aangepast om te bewerkstelligen dat naar mate een obstakel dichterbij de robot is het netwerk een grotere invoerwaarde krijgt, in plaats van een kleinere. De uiteindelijke waarde die als invoer aan het netwerk gegeven wordt, wordt berekend met behulp van formule 3.1.

$$invoerWaarde = \frac{100}{sonarWaarde + 0.1} \quad (3.1)$$

Bovendien worden door het programma de vorige waarden van de sonars bewaard net als de vorige uitvoer van het netwerk. Als parameter kan het aantal stappen geschiedenis van zowel de sensors als de vorige acties afzonderlijk opgegeven worden.



Figuur 3.2: De omgeving van de robot

Een stap geschiedenis betekent hier de sensorwaarden en de uitgevoerde actie van de vorige aanroep van de beslissingsfunctie door de ARIA library. Het vergroten van de geschiedenis heeft tot gevolg dat de robot meer een idee heeft in welke toestand hij zich bevindt, maar heeft ook een groter aantal invoerwaarden tot gevolg wat het weer moeilijker maakt te generaliseren.

3.3 Programmatuur

Het gebruikte programma is zo modulair mogelijk geschreven om het eenvoudig te maken om snel van beslissingsmechanisme te wisselen en om de omgeving van de verschillende leermethoden zo veel mogelijk gelijk te houden. Bij starten van het programma wordt een leeralgoritme gekozen en deze wordt vervolgens aan de module voor de aansturing aangeboden, die deze vervolgens vraagt welke actie uit te voeren. De module voor de aansturing verzamelt de invoer van de sonars en is bovendien uitgerust om informatie van een camera te accepteren via een beeldverwerkingsmodule, om zo de uitbreiding van het onderzoek toe te laten voor complexere omgevingen of taken.

Elke leermodule heeft de beschikking over een invoermodule waarmee bijvoorbeeld invoer van het toetsenbord of een joystick gebruikt kan worden om de robot te sturen. Dit kan gebruikt worden voor het doen van demonstraties. De leermodules zijn bovendien uitgerust om deze demonstraties op te slaan zodat er later op getraind kan worden. Hierbij wordt de complete invoer van het systeem opgeslagen, inclusief alle geschiedenis. Ook wordt de gewenste actie, afkomstig uit de besturingsmodule, opgeslagen.

3.4 Trainingsset

Voor het trainen van de beide methoden zijn 50 demonstraties opgenomen, waarvan 25 vanuit verschillende posities in de omgeving waarbij de robot begint in de goede richting en 25 waarbij de robot begon in de verkeerde richting waardoor de robot eerst gekeerd moest worden. In figuur 3.2 zijn de 25 startposities te zien als cirkels en vanaf elke positie werd de robot zowel in de goede als in de verkeerde richting gestart. Van elke positie werd de robot zo snel mogelijk naar het doel gestuurd, waarbij constant een redelijke afstand tot de muren werd behouden. Bij de startposities waarbij de robot in de verkeerde richting gedraaid was, werd er rekening mee gehouden dat er een aantal stappen nodig is voordat redelijkerwijs verwacht mag worden dat aan de hand van de sonarwaarden gezien kan worden dat de robot de verkeerde kant oprijdt. Bij de draaimanoeuvre werd de snelheid verlaagd en een redelijke afstand tot de muren bewaard.

3.5 Training

De training bestaat uit het trainen op de demonstraties gedurende een aantal epochs. Voor het direct leren werd batch learning gebruikt, wat betekent dat eerst alle gewichtsaanpassingen voor de gehele trainingsset opgeteld werden voordat ze werden toegepast. Dit maakt het mogelijk om een *momentum term* λ te gebruiken (Gurney, 1997): de toevoeging van een bepaalde factor maal de vorige gewichtsaanpassing. Dit laat het toe om in het geval van veel gewichtsaanpassingen in dezelfde richting, de leersnelheid tijdelijk te verhogen terwijl de momentum term aanpassingen in de tegengestelde richting juist verkleint. De formule voor de uiteindelijke aanpassing is gegeven in formule 3.2. Hierbij is ΔW_{ij}^t de aanpassing van het gewicht tussen node i en node j op tijdstip t.

$$\Delta W_{ij}^t = \Delta W_{ij}^t + \lambda \times \Delta W_{ij}^{t-1} \tag{3.2}$$

De momentum term wordt bij het CACLA algoritme niet gebruikt omdat de waarden van de toestanden bepalen of er getraind wordt of niet en daardoor batch learning niet mogelijk is.

3.6 Test

De robot werd in de gegeven omgeving op de 25 trainingsposities geplaatst en vervolgens werd gemeten of de robot het doel bereikte en hoe lang hij daar over deed. Vanuit elke positie werd de robot zowel in de juiste richting als in de verkeerde richting gestart. Ter referentie werd de robot vanuit elke positie ook handmatig zo snel mogelijk naar het doel gestuurd. Bovendien werd bij elke test het aantal aanrijdingen geteld om een andere vorm van prestatie bij te houden.

3.7 Metingen

Met behulp van het UNIX time-commando werd de tijd bijgehouden tussen het starten van het programma en het beëindingen van het programma na het bereiken van het doel. Van de uitvoer werd de *real time* gebruikt, welke overeenkomt met de daadwerkelijk verstreken tijd. Het bereiken van het doel is hierbij gedefinieerd als het moment dat de gehele robot binnen het doelgebied is. Om eventuele cache-effecten uit te schakelen werd van tevoren het programma eerst vijf keer gestart en beëindigd.

3.8 Dataverwerking

Voor elke geteste configuratie werd het tijdsverschil met de bijbehorende demonstratie berekend. Hierdoor is het duidelijk te zien wat het verschil is tussen de configuraties. Bovendien is er een vergelijking gemaakt tussen het aantal aanrijdingen wat bij elke configuratie plaatsvond. Voor alle tijdgegevens is een foutmarge van vijf procent gebruikt, terwijl voor het aantal aanrijdingen een foutmarge van één aanrijding werd gebruikt.

4 Resultaten

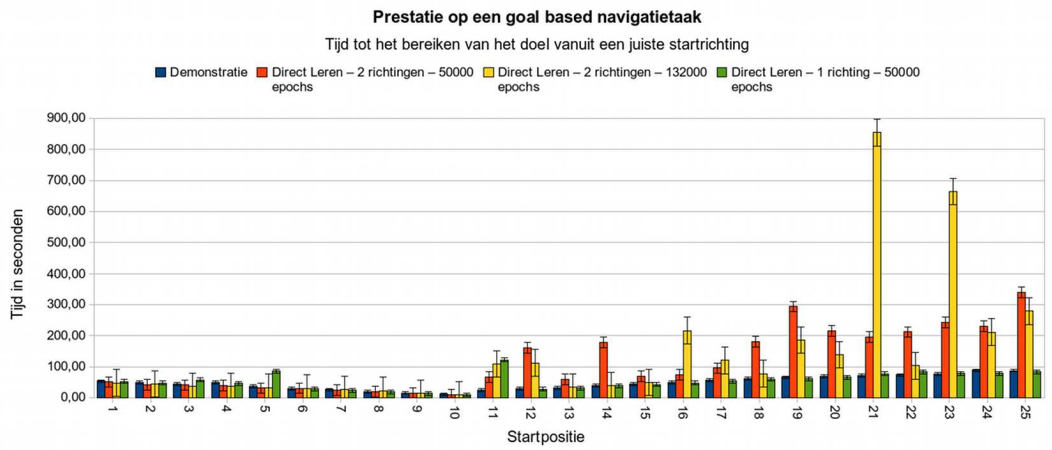
Het is niet gelukt om een goede vergelijking te maken tussen het leren van de taak met behulp van direct leren en het leren van de taak met behulp van het CACLA algoritme, omdat het bij het CACLA algoritme niet gelukt is om enige vorm van prestatie te bereiken. Bij het direct leren is dit wel gelukt.

Bij het direct leren zijn drie configuraties getest. De eerste is een training op alle 50 demonstraties vanaf de 25 startposities in beide richtingen, gedurende 50000 epochs. De tweede is een training op alleen de 25 demonstraties in de juiste richting gedurende 50000 epochs. Deze beide configuraties vertoonden grote prestatieverschillen, maar verschilden beide ook in de resulterende MSE. Daarom werd het netwerk van de eerste configuratie doorgetraind om de MSE naar een gelijk niveau te brengen als de tweede configuratie. Dit niveau werd bereikt na in totaal 132000 epochs, wat de derde geteste configuratie is. Alle configuraties werden getest op alle 25 startposities in zowel de juiste als de verkeerde richting.

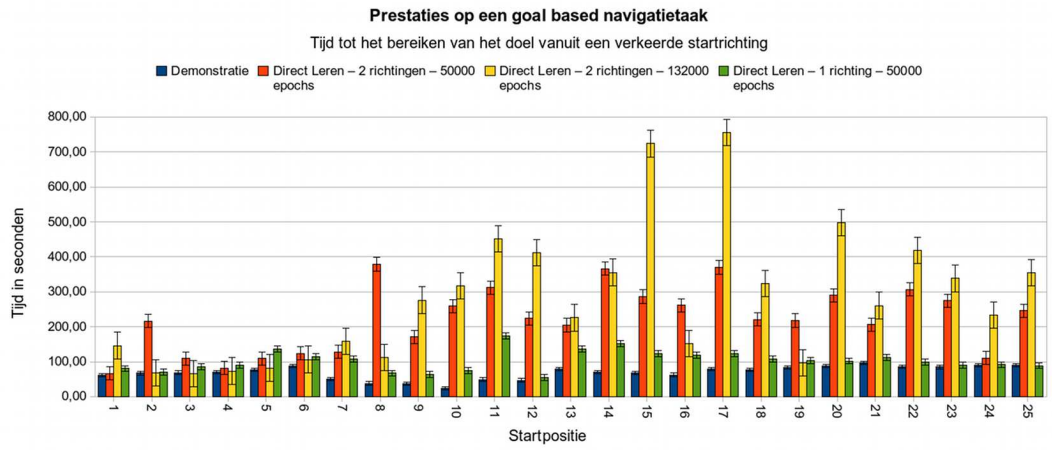
Alle geteste configuraties netwerken hebben twee uitvoerwaarden. De eerste is de bewegingssnelheid en de tweede is de draaisnelheid. Ook maken alle netwerken gebruik van tien stappen geschiedenis voor zowel de sensorwaarden als voor de uitgevoerde acties. De netwerken hebben dus in totaal 196 invoerwaarden.

Als overige parameters werden voor alle configuraties dezelfde waarden gebruikt. Voor de leersnelheid α was dat 6×10^{-5}, de momentum term λ was 0.8. Deze parameters bleken tot een redelijk snelle convergentie te leiden. Het vergroten van de leersnelheid had tot gevolg dat het netwerk niet meer in staat was te convergeren. De invloed van de momentum term op de tijd benodigd om te leren was klein. Er werden steeds twaalf hidden units gebruikt. Dit aantal is proefondervindelijk vastgesteld. Minder hidden units bleken niet in staat om voldoende detail te herkennen en meer hidden units vertraagden het leerproces zonder betere resultaten op te leveren.

In figuren 4.1 en 4.2 zijn de tijden benodigd voor het bereiken van het doel te zien. In figuren 4.3 en 4.4 zijn de verschillen van de drie configuraties met de demonstratie te zien en de gemiddelden.



Figuur 4.1: Tijd tot bereiken doel vanuit juiste startrichting



Figuur 4.2: Tijd tot bereiken doel vanuit verkeerde startrichting

In figuren 4.5 en 4.6 zijn het aantal aanrijdingen van de drie configuraties te zien. In tabel 4.1 zijn de gemiddelden en standaarddeviates van de drie configuraties te zien.

Bij de twee configuraties waarbij zowel de demonstraties in de juiste richting als de demonstraties in de verkeerde richting werden gebruikt, werden steeds dezelfde fouten gemaakt met aanrijdingen tot gevolg. Dit was niet het geval bij de configuratie waarbij alleen de demonstraties in de juiste richting gebruikt werden. De robot bereikte uiteindelijk wel vanuit elke positie het doel.

In appendix A zijn alle resultaten van de drie configuraties te zien.

5 Discussie

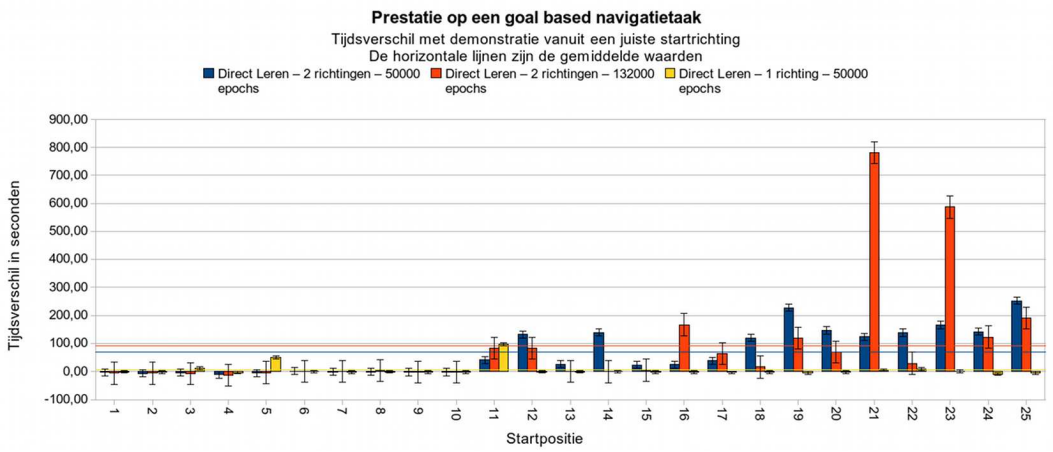
Eerst worden de resultaten van de drie configuraties afzonderlijk besproken. Daarna wordt besproken wat uit deze resultaten kan worden afgeleid en wordt getracht de onderzoeksvraag te beantwoorden.

5.1 Configuratie 1: 2 richtingen, 50000 epochs

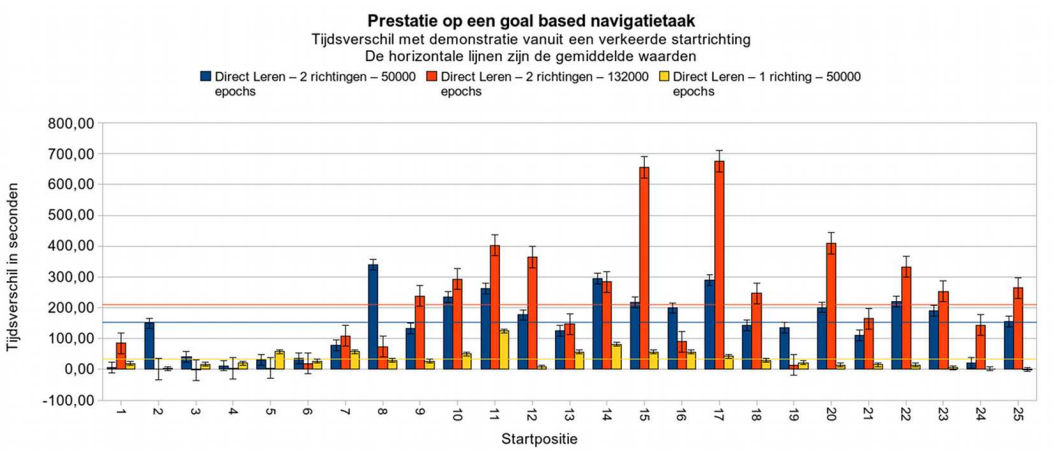
Het resultaat van deze configuratie is vrij matig. Er worden, vooral in de rechterhelft van de gebruikte omgeving, veel aanrijdingen gedaan. Bovendien bleek het daar moeilijk om de juiste actie te ondernemen, de robot kwam regelmatig in hoeken vast te zitten wat al snel twee of meer aanrijdingen tot gevolg had. Bovendien was de robot niet in staat om de bocht bij de centrale kruising naar rechts te nemen maar nam deze altijd naar links, waardoor een extra draai gemaakt moest worden, wat duidelijk extra tijd heeft gekost.

5.2 Configuratie 2: 2 richtingen, 132000 epochs

Deze configuratie is de doorgetrainde versie van configuratie 1. Bij deze configuratie is het resultaat slechter dan bij configuratie 1: de robot bleef lan-



Figuur 4.3: Tijdsverschil met demonstratie vanuit juiste startrichting



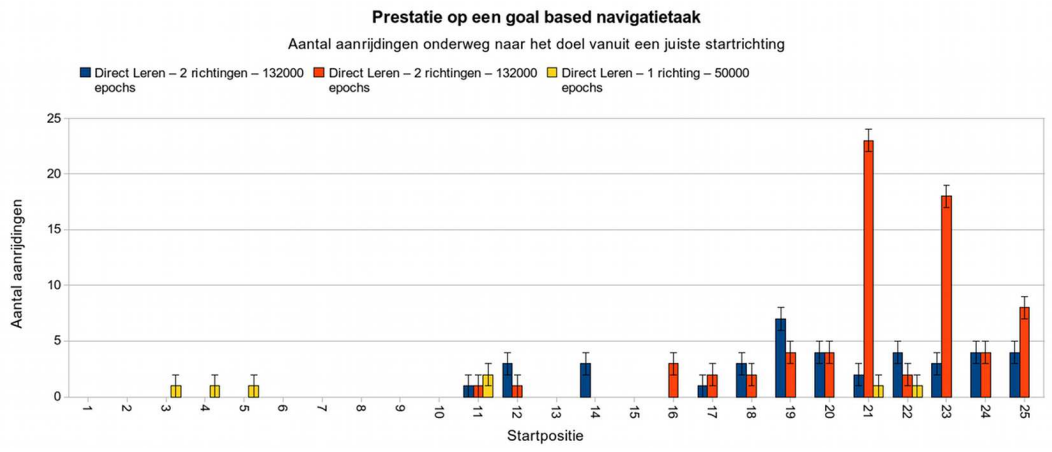
Figuur 4.4: Tijdsverschil met demonstratie vanuit verkeerde startrichting

ger in hoeken vastzitten, maakte meer aanrijdingen en had meer tijd nodig om het doel te bereiken, zoals uit tabel 4.1 af te lezen valt. Er is duidelijk sprake van overfitting: ondanks dat de MSE van deze configuratie aanmerkelijk lager is dan de configuratie met een training van 50000 epochs presteert deze configuratie significant slechter met een tijd die gemiddeld 37 procent langer was dan bij de vorige configuratie en met 71 procent meer aanrijdingen.

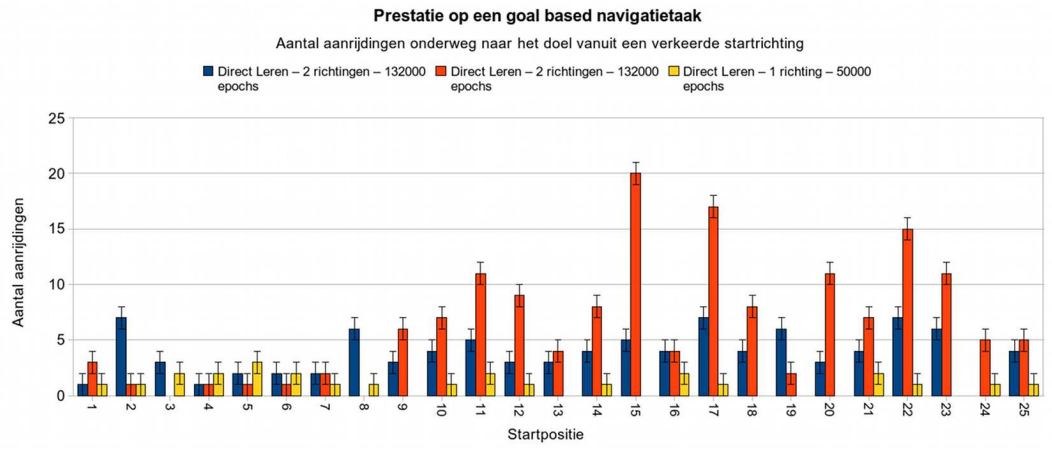
5.3 Configuratie 3: 1 richting, 50000 epochs

Bij deze configuratie zijn alleen de demonstraties waarbij in de juiste richting gestart werd gebruikt. Hierdoor werd het aantal voorbeelden waarop getraind kon worden meer dan gehalveerd. Dit had ook een daling van de MSE tot gevolg. Uit de tests blijkt dat het netwerk bij deze configuratie veel beter in staat is om te generaliseren. De robot was vanuit negentien startposities waarbij de robot in de juiste richting begon in staat om zonder aanrijdingen het doel te bereiken, zoals te zien in figuur 4.5, en vanuit zeven startposities in de verkeerde richting zonder aanrijdingen, zoals te zien in figuur 4.6. Dit zijn in de juiste richting 46 procent meer aanrijdingsvrije ritten dan de andere configuraties en in de verkeerde richting 250 procent meer aanrijdingsvrije ritten dan configuratie 1 en 700 procent meer aanrijdingsvrije ritten dan configuratie 2. Ook de tijd benodigd om het doel te bereiken is significant lager, namelijk 87.3 procent lager dan configuratie 1 en 82.6 procent lager dan configuratie 2. Deze verschillen komen vooral voort uit de prestaties bij het vinden van het doel vanuit de verkeerde startpositie en uit de prestaties bij het vinden van het doel vanuit de juiste startrichting op een positie aan de rechterkant van de omgeving.

Een opvallend resultaat is dat juist deze configuratie, die niet getraind is op de demonstraties in de verkeerde richting, beter presteert op de startposities in de verkeerde richting.



Figuur 4.5: Aantal aanrijdingen onderweg naar doel vanuit de juiste startrichting



Figuur 4.6: Aantal aanrijdingen onderweg naar doel vanuit de verkeerde startrichting

5.4 Wel of geen voorbeelden in verkeerde richting

Uit de tests blijkt dat het netwerk dat alleen getraind is op de demonstraties waarbij de robot in de juiste richting startte, aanmerkelijk beter presteert op zowel de tests in de juiste richting (figuur 4.3) als in de verkeerde richting(figuur 4.4). Bovendien is het aantal aanrijdingen in beide richtingen (figuren 4.5 en 4.6) lager, vooral voor de tests in de rechterhelft van de testomgeving. Deze resultaten suggereren dat het toevoegen van de demonstraties in de verkeerde richting eerder een negatieve dan een positieve bijdrage levert. Dit is waarschijnlijk te wijten aan het moeten aanpassen aan beide soorten demonstraties van het neurale netwerk, wat een compromis in het midden tot gevolg heeft. Bij het afwezig zijn van deze demonstraties zullen de toestanden waarbij de robot in de verkeerde richting rijdt weliswaar niet herkend worden, maar daaruit kan geconcludeerd worden dat de richting verkeerd is waardoor de robot een keerpoging zal ondernemen.

Het vermijden van obstakels gaat ook aanmerkelijk beter bij slechts één soort demonstraties. Dit is waarschijnlijk het gevolg van het gebrek aan expliciete draaimanoeuvres in de demonstraties waarbij de robot onnodig dicht bij de muren komt. Bovendien zullen de draaimanoeuvres in de demonstraties met een verkeerde startrichting tot gevolg hebben dat de robot vaker een draaimanoeuvre inzet.

Het feit dat de laatste configuratie het best presteert op de startposities in de verkeerde richting suggereert sterk dat de demonstraties in de verkeerde richting overbodig zijn en zelfs een negatieve invloed op de prestaties hebben.

5.5 Het CACLA algoritme

Het is niet gelukt om enige vorm van prestaties te verkrijgen bij het toepassen van het CACLA algoritme. Hiervoor zijn een aantal mogelijke verklaringen. De focus van dit onderzoek heeft gedurende het grootste deel van de looptijd

Tabel 4.1: Overzicht prestaties direct leren

Configuratie	**Mean Squared Error**	**Gemiddeld tijdsverschil**	**Standaarddeviatie tijdsverschil**	**Totaal aantal aanrijdingen**
2 richtingen, 50000 epochs	0.010155	110.33	97.3	135
2 richtingen, 132000 epochs	0.009249	151.18	197.54	231
1 richting, 50000 epochs	0.009040	19.24	29.28	33

gelegen op de implementatie van de direct leren methode, aangezien ook deze methode eerst geen resultaat opleverde. Hierdoor bleef er onvoldoende tijd over om eventuele fouten in de software voor het CACLA algoritme op te sporen en om met de parameters te experimenteren.

Eerder onderzoek naar het CACLA algoritme (Wiering & Hasselt, 2007) was wel succesvol, maar in dat onderzoek ging het om andere taken: het volgen van een andere agent en het balanceren van een paal op een wagon. Bij deze taken werden rewards vaker uitgedeeld, omdat bij deze taken vaker duidelijk was of de agent een goede policy had. In dit onderzoek krijgt de robot slechts een reward aan het einde van een sessie. Dit verschil zou ook een reden kunnen zijn waarom het niet gelukt is om het CACLA algoritme toe te passen.

Een andere mogelijke bijdrage is de trainmethode. Het CACLA algoritme heeft, net als andere implementaties van reinforcement learning, vooral de mogelijkheid om zonder demonstraties te leren. Door *exploration* is het systeem in staat om de huidige policy te verbeteren en zo naar de optimale policy te convergeren. Bij de in dit onderzoek gebruikte methode wordt het systeem echter getraind op de opgenomen demonstraties, waarbij het systeem dus per definitie de actie behorende bij de optimale policy aangeleverd krijgt. Hierdoor wordt het grootste voordeel van deze methode overbodig gemaakt en krijgt het mogelijk een nadeel ten opzichte van de direct leren methode. Hierdoor is het niet gerechtvaardigd om het CACLA algoritme negatief te beoordelen.

6 Conclusie

De resultaten suggereren dat voor deze taak het direct leren algoritme beter geschikt is, waarbij alleen getraind wordt op de demonstraties in de juiste richting. Hierbij dient wel opgemerkt te worden dat de gebruikte trainmethode waarschijnlijk niet geschikt is voor het CACLA algoritme, waardoor dit algoritme benadeeld kan zijn. Er is dus geen eenduidig antwoord te geven op de onderzoeksvraag aan de hand van de verkregen resultaten. Hiervoor is verder onderzoek benodigd.

6.1 Suggesties voor verder onderzoek

Het gebruiken van de ingebouwde odometrie van de ARIA library als invoer voor de leeralgoritmes zou mogelijk kunnen leiden tot betere prestaties. Hierbij zou bijvoorbeeld de afgelegde afstand en hoek ten opzichte van een aantal stappen geleden als invoer gegeven kunnen worden. Door het toepassen van deze informatie zal de robot beter in staat zijn een schatting te maken van waar hij zich bevindt. Deze informatie maakt de robot misschien wel meer afhankelijk van de situatie waarin de robot getraind is: variaties in de onderlinge afstanden zullen dan minder makkelijk opgevangen kunnen worden. Het effect van deze toevoeging is een interessant onderwerp voor verder onderzoek.

Een ander interessant onderzoek zou het gebruiken van de camera op de Pioneer P2DX robots zijn. Dit vereist dat de tests uitgevoerd worden op een echte robot in plaats van in simulatie, aangezien MobileSim geen camerasimulatie ondersteund en dat is ook niet eenvoudig te realiseren. Door het gebruik van de camera zal de robot beter in staat zijn om de omgeving te herkennen en om het doel te herkennen. Dit zal waarschijnlijk vooral voordelen hebben voor het CACLA algoritme, die het doel aan bepaalde visuele stimuli kan herkennen en zo ook in onbekende omgevingen het doel sneller zal kunnen vinden.

Om een betere vergelijking met het CACLA algoritme te maken kan het CACLA algoritme getraind worden zonder demonstraties, maar alleen door de reward toe te kennen zodra de robot het doel bereikt heeft. Na training kunnen dan de prestaties vergeleken worden met het direct leren algoritme. De robot is dan zelf in de gelegenheid gesteld om de optimale policy te bepalen en zou daardoor beter kunnen presteren. Om het leerproces te versnellen zou als initieel actienetwerk het netwerk geproduceerd door het

direct leren algoritme gebruikt kunnen worden. Aangezien dit netwerk het doel al weet te bereiken kan aan de hand hiervan het waarde netwerk getraind worden om de doeltoestand te herkennen. Als dat gebeurd is, kan vervolgens het CACLA algoritme zijn werk doen. Wellicht dat door de combinatie van de twee netwerken het netwerk verder verbeterd kan worden, verder dan met alleen direct leren mogelijk zou zijn. Dit is meer in overeenstemming met een eerder onderzochte aanpak van reinforcement learning (Chohra et al., 1998).

Ook zou gekeken kunnen worden hoe de configuratie waarbij alleen getraind werd op de voorwaartse demonstraties van dit onderzoek verder verbeterd kan worden. Hierbij moet vooral gedacht worden aan het minimaliseren van het aantal aanrijdingen. Dit zou gerealiseerd kunnen worden door aan de trainingsset een aantal demonstraties toe te voegen zonder geschiedenis waarbij alleen de actuele sensorwaarden gebruikt worden, waarbij wordt weggedraaid van obstakels. Door de geschiedenis weg te laten bij deze demonstraties kan de training zich focussen op de actuele sensorwaarden die bij een naderende aanrijding het belangrijkst zijn.

Er zijn een aantal parameters die weinig of helemaal niet gevarieerd zijn. De voornaamste hiervan is de geschiedenis die aan het netwerk wordt meegegeven. In alle configuraties was deze tien stappen lang. Een langere geschiedenis meegeven vereist waarschijnlijk meer demonstraties om het netwerk in staat te stellen te generaliseren maar zou ook de mogelijkheid kunnen hebben om complexere omgevingen aan te leren. Ook is er mogelijk een verschil tussen de bijdrage van de sensorgeschiedenis en de bijdrage van de actiegeschiedenis die verder onderzocht zou kunnen worden. Een groter aantal stappen sensorgeschiedenis heeft een groter effect op de leersnelheid omdat dit het aantal invoerwaarden veel sterker vergroot dan het vergroten van het aantal stappen actiegeschiedenis. Wat wel gebleken is uit de eerste experimenten is dat tien stappen geschiedenis betere prestaties toelaat dan slechts één stap geschiedenis wat bij de eerste experimenten werd gebruikt.

Verder zou het interessant zijn om de prestaties met andere leermethoden te vergelijken. Er is bijvoorbeeld al onderzoek gedaan naar *robotic clicker training* (Kaplan, Oudeyer, Kubinyi & Miklósi, 2002) waarbij een techniek toegepast wordt die veel gebruikt wordt bij het trainen van dieren. Dit is een vorm van reinforcement learning maar bij deze methode worden er veel vaker rewards uitgedeeld, namelijk bij elke actie in de juiste richting. Hiervoor hoeft de doeltoestand dus niet eerst bereikt te worden. Deze methode vereist tijdens het trainen meer menselijke input maar zou mogelijk sneller in staat zijn om het gewenste gedrag aan te leren dan andere vormen van reinforcement learning.

Referenties

Atkeson, C. & Schaal, S. (1997). Robot learning from demonstration. *Machine learning: Proceedings of the fourteenth Internation Conference (CML '97)*, 12 – 20.

Chohra, A., Benmehrez, C. & Farah, A. (1998). Neural navigation approach for intelligent autonomous vehicles (iav) in partially structured environments. *Applied Intelligence: The International Journal of Artificial Intelligence, Neural Networks, and Complex Problem-Solving Technologies*, *8*(3), 219 – 233.

Gurney, K. (1997). *An introduction to neural networks.* CRC.

Kaplan, F., Oudeyer, P. Y., Kubinyi, E. & Miklósi, A. (2002). Robotic clicker training. *Robotics and Autonomous Systems*, *38*(3-4), 197 – 206.

Melo, F. S. & Ribeiro, M. I. (2008). Reinforcement learning with function approximation for cooperative navigation tasks. *2008 Ieee International Conference On Robotics And Automation. The Half-day Workshop On: Towards Autonomous Agriculture Of Tomorrow*, 3321 – 3327.

Russell, S. & Norvig, P. (2003). *Artificial intelligence, a modern approach, second edition.* Prentice Hall.

Smart, W. & Kaelbling, L. P. (2002). Effective reinforcement learning for mobile robots. *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No.02CH37292)*, 3404 - 10 vol.4.

Thomaz, A. & Breazeal, C. (2008). Teachable robots: understanding human teaching behavior to build more effective robot learners. *Artificial Intelligence*, *172*(6-7), 716 – 737.

Wiering, M. A. & Hasselt, H. v. (2007). Reinforcement learning in continuous action spaces. *Proceedings of the 2007 IEEE Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 272 – 279.

Appendices

A Ruwe data

In tabel A.1, vervolgd in tabel A.2 zijn alle metingen uit dit onderzoek te vinden.

Tabel A.1: De uitgevoerde metingen op de drie configuraties

Start		**Demo**	**Direct Leren - 50000 epochs**			**Direct Leren - 132000 epochs**			**Direct Leren - 50000 epochs - 1 richting**		
Positie	**Richting**	**Tijd**	**Tijd**	**Verschil**	**Aanrijdingen**	**Tijd**	**Verschil**	**Aanrijdingen**	**Tijd**	**Verschil**	**Aanrijdingen**
1	Achteruit	61,26	66,69	5,43	1	146,31	85,05	3	80,48	19,22	1
1	Vooruit	53,78	51,09	-2,69	0	47,91	-5,87	0	52,98	-0,80	0
2	Achteruit	67,50	216,87	149,37	7	68,54	1,04	1	69,85	2,35	1
2	Vooruit	49,88	42,96	-6,92	0	44,18	-5,70	0	47,76	-2,12	0
3	Achteruit	69,22	109,28	40,06	3	66,54	-2,68	0	85,40	16,18	2
3	Vooruit	44,96	41,11	-3,85	0	37,06	-7,90	0	57,20	12,24	1
4	Achteruit	69,85	82,04	12,19	1	73,49	3,64	1	89,69	19,84	2
4	Vooruit	49,82	39,00	-10,82	0	36,23	-13,59	0	46,45	-3,37	1
5	Achteruit	77,79	109,56	31,77	2	82,46	4,67	1	135,77	57,98	3
5	Vooruit	36,69	31,33	-5,36	0	33,08	-3,61	0	86,43	49,74	1
6	Achteruit	87,71	124,11	36,40	2	106,83	19,12	1	114,54	26,83	2
6	Vooruit	29,27	30,89	1,62	0	30,47	1,20	0	29,21	-0,06	0
7	Achteruit	50,25	128,38	78,13	2	158,97	108,72	2	108,34	58,09	1
7	Vooruit	26,36	25,69	-0,67	0	26,40	0,04	0	24,47	-1,89	0
8	Achteruit	38,35	378,88	340,53	6	112,36	74,01	0	67,22	28,87	1
8	Vooruit	20,37	19,65	-0,72	0	23,37	3,00	0	19,67	-0,70	0
9	Achteruit	37,95	171,39	133,44	3	276,48	238,53	6	64,26	26,31	0
9	Vooruit	16,15	14,06	-2,09	0	14,54	-1,61	0	14,66	-1,49	0
10	Achteruit	24,00	258,92	234,92	4	317,17	293,17	7	74,11	50,11	1
10	Vooruit	11,67	9,63	-2,04	0	9,09	-2,58	0	9,90	-1,77	0
11	Achteruit	49,30	311,74	262,44	5	451,68	402,38	11	174,24	124,94	2
11	Vooruit	25,17	66,10	40,93	1	109,39	84,22	1	122,52	97,35	2
12	Achteruit	47,41	224,14	176,73	3	412,04	364,63	9	54,88	7,47	1
12	Vooruit	29,31	161,35	132,04	3	112,52	83,21	1	28,45	-0,86	0
13	Achteruit	79,07	205,04	125,97	3	226,07	147,00	4	135,88	56,81	0
13	Vooruit	32,89	59,51	26,62	0	33,98	1,09	0	32,11	-0,78	0

Tabel A.2: De uitgevoerde metingen op de drie configuraties - vervolg

Start		**Demo**	**Direct Leren - 50000 epochs**			**Direct Leren - 132000 epochs**			**Direct Leren - 50000 epochs - 1 richting**		
Positie	**Richting**	**Tijd**	**Tijd**	**Verschil**	**Aanrijdingen**	**Tijd**	**Verschil**	**Aanrijdingen**	**Tijd**	**Verschil**	**Aanrijdingen**
14	Achteruit	71,12	366,16	295,04	4	355,54	284,42	8	151,93	80,81	1
14	Vooruit	39,01	178,45	139,44	3	38,77	-0,24	0	38,85	-0,16	0
15	Achteruit	67,79	286,19	218,40	5	723,91	656,12	20	123,79	56,00	0
15	Vooruit	44,62	68,55	23,93	0	49,48	4,86	0	42,87	-1,75	0
16	Achteruit	62,37	261,71	199,34	4	152,12	89,75	4	119,27	56,90	2
16	Vooruit	49,44	74,39	24,95	0	216,72	167,28	3	47,78	-1,66	0
17	Achteruit	79,87	370,13	290,26	7	755,86	675,99	17	122,77	42,90	1
17	Vooruit	57,03	95,45	38,42	1	120,69	63,66	2	53,78	-3,25	0
18	Achteruit	77,78	220,64	142,86	4	324,54	246,76	8	107,33	29,55	0
18	Vooruit	61,44	181,14	119,70	3	77,78	16,34	2	59,54	-1,90	0
19	Achteruit	83,18	218,19	135,01	6	97,66	14,48	2	104,30	21,12	0
19	Vooruit	66,12	293,84	227,72	7	185,83	119,71	4	60,83	-5,29	0
20	Achteruit	88,78	290,02	201,24	3	498,20	409,42	11	102,34	13,56	0
20	Vooruit	68,72	215,54	146,82	4	138,20	69,48	4	66,49	-2,23	0
21	Achteruit	96,47	206,72	110,25	4	260,97	164,50	7	112,12	15,65	2
21	Vooruit	72,32	196,32	124,00	2	853,77	781,45	23	76,91	4,59	1
22	Achteruit	86,37	306,42	220,05	7	419,28	332,91	15	100,09	13,72	1
22	Vooruit	73,47	212,15	138,68	4	103,00	29,53	2	83,21	9,74	1
23	Achteruit	84,88	274,76	189,88	6	338,37	253,49	11	90,00	5,12	0
23	Vooruit	77,19	243,21	166,02	3	663,69	586,50	18	78,59	1,40	0
24	Achteruit	89,99	111,26	21,27	0	233,23	143,24	5	91,45	1,46	1
24	Vooruit	88,31	229,91	141,60	4	211,42	123,11	4	79,13	-9,18	0
25	Achteruit	89,98	245,68	155,70	4	353,89	263,91	5	89,14	-0,84	1
25	Vooruit	87,88	340,23	252,35	4	279,05	191,17	8	83,23	-4,65	0
Totaal		**2950,11**	**8466,47**	**5516,36**	**135**	**10509,13**	**7559,02**	**231**	**3912,21**	**962,1**	**33**