Software Engineering en Gedistribueerde Applicaties

Eindverslag

Team F.A.C.H.T.

Chris Bovenschen, Harm Dermois,
Alexandra Moraga Pizarro, Tamara Ockhuijsen en Fredo Tan

6104096, 0527963, 6129544, 6060374, 6132421

Universiteit van Amsterdam

24 juni 2011

Inhoudsopgave

1 Inleiding

Dit verslag beschrijft het ontwerpen, implementeren en testen van een gedistribueerde robotapplicatie. De robot die hiervoor wordt gebruikt is de gesimuleerde P2DX, deze heeft twee aangedreven wielen en een zwenkwiel. Het platform voor de robotsimulatie is USARSim. Het doel is om met behulp van de sensoren op de robot in een vrij volle ruimte een pad langs een wand te rijden, botsingen te vermijden, zonodig een wand op te zoeken, uit doodlopende stukken te ontsnappen, bijhouden waar hij is en welk pad hij heeft gevolgd en de omgeving in kaart te brengen.

Het project is op te delen in verschillende fases. Aan het begin is er nagedacht over een ontwerp met een model, een planning en een specificatie van het eindproduct. Hieruit is een voorlopig werkplan ontstaan. In het definitieve werkplan is een uitgebreidere beschrijving van de realisatie van verschillende componenten met bijbehorende tests te vinden samen met de eerste inleverbare versie van de code. Het grootste deel van de tijd heeft gezeten in de implementatie en tests. Als laatst zijn de componenten geïntegreerd tot een geheel, getest en geoptimaliseerd.

Dit verslag behandelt eerst de implementatie van de componenten met de bijbehorende tests en de opzet van het experiment. Daarna worden de resultaten besproken en hieruit volgt een conclusie.

2 Implementatie

De modules zijn verdeeld in low-level, sensor, mid-level en high-level componenten. De listener is een low-level component, de sonar, laser en odometry zijn sensormodules, de wall search, wall follow en collision avoider zijn mid-level componenten en de map maker en path finding zijn high-level componenten. De movements en communicator zijn libraries die door andere modules kunnen worden gebruikt.

De listener is verbonden met de server via een TCP-verbinding en stuurt de data direct door naar de sensormodules. De sensormodules halen van de data die voor hun bestemd zijn de juiste waarden uit de data en sturen deze door naar de mid-level modules en op aanvraag naar de map maker en path finding. Berekende kaarten van de map maker kunnen worden gebruikt voor de path finding. Path finding gebruikt de movements library om het pad te kunnen volgen. Als de robot aan het rijden is, staat de collision avoider te alle tijden aan.

2.1 Low-level

De low-level component verkrijgt rechtstreeks data van de robot, haalt ze uit elkaar en stuurt de data door naar de juiste module.

2.1.1 Listener

De listener zit direct aan de server vast. Hij haalt informatie binnen die de robot verstuurt en stuurt ook commando's naar de robot toe. De binnenkomende data worden meteen doorgestuurd naar de sensormodules. Eerst controleerde de listener de data ook op geldigheid en stuurde deze in het juiste formaat door naar de bijbehorende sensormodules. Echter werd de listener overwelmd door de data, waardoor deze nu slechts een doorgever is geworden.

2.2 Sensormodules

De sensormodules ontvangen data van de listener. Ze kijken of de data voor hun bestemd zijn en anders negeren ze deze. Ze verkrijgen de sensorwaarden uit de data en sturen ze door naar de mid-level modules. De sensormodules kunnen door de high-level modules worden aangeroepen voor de meest recente data. Zodra er ongeldige data binnenkomen, wordt de robot gestopt.

2.2.1 Odometry

Odometry haalt de drie waarden op die nodig zijn voor de bepaling van de positie van de robot. Dit zijn de x, y positie en de theta hoek in verhouding tot de startpositie en de richting van de robot. Deze waarden worden doorgestuurd naar de mid-level modules.

2.2.2 Range scanner

De range scanner bevindt zich op een bepaalde hoogte boven de robot. Hij scant de omgeving horizontaal door middel van het roteren van de laser in een

bepaald interval. De range scanner leest alle 181 laserwaarden uit en stuurt ze door naar de mid-level modules.

2.2.3 Sonar

De sonar leest alle 8 sonarwaarden uit en stuurt ze door naar de mid-level modules. De sonar stuurt een geluidsgolf, waardoor obstakels die op de grond liggen ook te detecteren zijn.

2.3 Mid-level

De mid-level componenten verzorgen het vermijden van botsingen en het zoeken en volgen van een muur.

2.3.1 Collision avoider

De collision avoider kijkt alleen naar de waarden van de sonar, omdat die in tegenstelling tot de laser wel objecten op de grond kan detecteren. Als de kleinste waarde van de sonar kleiner is dan 0,315 stopt de robot met rijden. Bij deze waarde zijn objecten op de grond nog zichtbaar. Zodra deze waarde te klein wordt, kan de sonar het object niet meer zien. Tevens wordt er met behulp van de snelheid uitgerekend na hoeveel seconden de botsing plaats zal vinden als de robot door blijft rijden. Deze berekening wordt gedaan op basis van de snelheid van de wielen, en kan dus in sommige gevallen afwijken.

2.3.2 Wall search

Bij het zoeken naar een muur draait de robot eerst 360 graden om te kijken waar de dichtstbijzijnde muur is. Zodra de kleinste sensorwaarde is gevonden, roteert de robot nog een keer totdat hij in de juiste positie staat. Hij rijdt recht op de muur af totdat hij op een bepaalde afstand van de muur af staat en dan gaat hij over op wall follow.

2.3.3 Wall follow

De muur wordt gevolgd die door wall search is gevonden. Als de muur zich meer aan de linkerkant van de robot bevindt, draait hij naar rechts en als de muur zich meer aan de rechterkant van de robot bevindt, draait hij naar links. De kleinste laserwaarde bevindt zich nu aan de rechter- of linkerkant. Hij blijft rechtdoor rijden als de kleinste laserwaarde niet kleiner en groter wordt dan een ingestelde waarde. Als hij te ver van de muur afwijkt, stuurt hij bij richting de muur. Als hij te dicht bij de muur komt, stuurt hij bij van de muur af. Als er geen muur meer wordt waargenomen, wordt de wall_continued functie binnen de wall search aangeroepen. Deze functie kijkt of er in de buurt nog een muur is te vinden. Als dit zo is, gaat hij daarheen. Als er zich in de buurt nog een andere muur bevindt, zoekt wall search naar de nieuwe dichtstbijzijnde muur. Elke keer wanneer de logica klaar is, wordt er nieuwe informatie opgevraagd van de sensormodules.

2.4 High-level

De high-level componenten houden bij waar de robot is geweest en kunnen hem naar een ander punt laten rijden. Dit zijn de modules die meer tijd nodig hebben dan de mid-level modules.

2.4.1 Map maker

De map maker maakt een kaart van de constante stroom van laser en odometry waarden die hij binnenkrijgt. Voor het maken van een kaart wordt de Simultaneous Localization And Mapping (SLAM) techniek gebruikt. De kaart heeft de vorm van een matrix. De bibliotheek die wordt gebruikt om deze map te maken is tinySLAM. Dit is een zeer simpel algoritme dat niet meer dan 200 regels code bevat. Het enige wat het elke keer doet, is de gevonden lijnen tekenen. Dit algoritme heeft nog heel wat uitbreidingen, maar alleen de basis van dit algoritme wordt gebruikt. De originele code is geschreven in C, maar met behulp van een gevonden versie in Python en een paar aanpassingen is deze bruikbaar geworden.

2.4.2 Path finding

Path finding kan alleen gebruikt worden binnen de al ontdekte gebieden en kan dus alleen een route plannen binnen een bekend domein. Hierin probeert hij een zo kort mogelijk pad te vinden naar de bestemming. Er wordt gebruik gemaakt van het A ster zoekalgoritme om het efficiëntste pad te vinden tussen twee punten.

2.5 Libraries

Alle modules kunnen gebruik maken van de beschikbare libraries. Ze kunnen deze libraries importeren en hun functies gebruiken.

2.5.1 Communicator

De communicator zorgt voor de communicatie tussen de modules en maakt een geïntegreerd geheel van de losse modules. De communicatie verloopt via een communicatieprotocol.

2.5.2 Movements

Dit is een library waarin alle bewegingen zijn gedefinieerd. Deze library wordt gebruikt voor de modules die de robot willen besturen. In de movements staan functies gedefinieerd voor het vooruit, naar links, naar rechts en achteruit rijden. Tevens is het mogelijk om naar links en rechts te roteren, een spoor achter te laten, de camera te zien vanuit de robot en de robot te laten stoppen met rijden.

3 Tests

Voor elke module is er een aparte test geschreven om er zeker van te zijn dat elke module op zichzelf goed werkt. Elke module wordt direct aan de simulator vastgezet en getest met de directe data die binnenkomen.

3.1 Listener

De listener wordt getest door data te verkrijgen en deze per type bericht in een nieuwe array uit te printen.

3.2 Odometry

De odometry wordt getest door 100 regels data te verkrijgen van de simulator en hiervan de odometriewaarden te printen. De odometriewaarden worden getest op geldigheid.

3.3 Range scanner

De range scanner wordt getest door 100 regels data te verkrijgen van de simulator en hiervan de 181 laserwaarden te printen. De laserwaarden worden getest op geldigheid.

3.4 Sonar

De sonar wordt getest door 100 regels data te verkrijgen en hiervan de 8 sonarwaarden te printen. De sonarwaarden worden getest op geldigheid.

3.5 Collision avoider

De collision avoider wordt getest door de robot vanuit verschillende startposities rond te laten rijden zonder dat er botsingen ontstaan.

3.6 Wall combo

De wall combo is geschreven om te testen of de wall search goed werkt in samenwerking met de wall follow. Deze test simuleert de werking in een gedistribueerd systeem.

3.7 Wall follow

De wall follow wordt getest door de robot vanuit verschillende startposities een muur te laten volgen. Het is de bedoeling dat deze module wordt aangeroepen wanneer er al een muur is gevonden. Dan kan de robot de muur naar links en naar rechts toe volgen. Binnen deze gebruikte methode zijn stompe bochten ook mogelijk.

3.8 Wall search

De wall search wordt getest door de robot vanuit verschillende startposities de dichstbijzijnde muur te zoeken.

3.9 Map maker

De map maker wordt getest door hem een output bestand met laser- en odometriewaarden van de wall combo te laten lezen en de kaart ervan te tekenen.

3.10 Path finding

De path finding wordt getest door een punt op te geven waar de robot naartoe moet rijden. Hij moet hierbij botsingen ontwijken en naar het punt toe rijden op een zo efficiënt mogelijke manier.

4 Experiment

De robot wordt geïnitialiseerd op een bepaalde plaats. Hij zoekt een muur en blijft deze muur volgen. Zodra hij de opdracht krijgt om naar een punt toe te rijden, wordt path finding ingeschakeld. De map maker werkt continu de kaart bij en de collision avoider zorgt ervoor dat er geen botsingen plaatsvinden. Als hij geen nieuwe opdracht krijgt om naar een ander punt toe te rijden, gaat hij weer een muur zoeken en deze volgen.

5 Resultaten

6 Conclusie

7 Discussie

8 Bronvermelding

9 Appendix