Labboek task 4

Humanoid sumo-challenge

Danny Thie 11052120
Axel Huting 11035072
Kelvin Xie 11044624
Boris Egelie 11071273

June 24, 2016

In de laatste week was het de bedoeling om met een groep aan een eigen project te werken. Wij kozen ervoor om een sumo robot te programmeren die als doel heeft zijn sparring partner (een andere robot) van de sumo-mat te krijgen. Fysieke robots waren niet meer te beschikking hierom maakten wij gebruik van een online simulator voor onze robot: 'sw.theconstructsin.com'.

De code voor de sumo robot was deels gegeven. Nadat de code ons duidelijk was geworden bleek het dat de sumo robot een stukje heeft waarmee hij altijd begint.

Eerst maakt de robot een paar zwaai-bewegingen en loopt vervolgens een stukje de cirkel in. Na dit "openings ritueel" gaat de sumo bot de volgende loop volgen:
1. Als de robot op de grond ligt met zijn gezicht tegen de mat, draait die zich om.
2. Als de robot op de grond ligt met zijn rug tegen de mat, staat die op.
3. Als de tegenstander wordt gezien, beweeg dan richting de tegenstander en trap hem.
4. Als de tegenstander niet wordt gezien, draai dan rond en beweeg het hoofd verticaal.

Alhoewel, deze code is niet compleet en delen zijn niet goed optimaal. Het doel is om deze code zo compleet mogelijk te maken en delen te verbeteren die niet goed werken. Het eerste wat aandacht vereiste was het optimaliseren van het vinden van de tegenstander, in de code was namelijk aangegeven dat dit niet goed werkte.

Wat opviel was dat de sumo robot op zijn tegenstander afloopt zodra die zijn tegenstander ziet, waardoor ze allebei omvallen en vervolgens weer opstaan. Hierna herhaalt dit proces zich en is het willekeurig wie er als eerst uit de ring valt. De fout zat volgens ons in het feit dat de robot niet door heeft wanneer hij zijn tegenstander zou moeten schoppen. Een idee was om de robot te laten stoppen voordat deze tegen zijn tegenstander aan liep. In plaats daarvan zou de robot na een vaste afstand stil staan en in een stabiele houding blijven staan. De robot wacht vervolgens op zijn tegenstander en steekt zijn arm uit om deze als het ware om te duwen. Na dit geïmplementeerd te hebben bleek dit geen effectieve manier te zijn om de tegenstander uit de ring te krijgen. Een betere alternative manier zou een fix van de recognition van de tegenstander nodig hebben.

Hierop hebben we het volgende gevonden:
Met de hulp van een TA werd het snel duidelijk dat de recognition van de tegenstander gefixt kon worden met behulp van OpenCV. Het koste veel moeite om de OpenCV te vinden via de terminal. In de MakeFIle was het mogelijk om code te schrijven die de OpenCV nodig heeft. Naar advies van een TA hebben wij geprobeerd om de code te vinden op het internet. De benodigde code bleek lastig te vinden. Er waren veel voorbeelden te vinden over object herkenning via webcams, of op afbeeldingen. Daarentegen was er niks te vinden dat in een simulatie gebruikt kon worden. Dit probleem zou opgelost kunnen worden door elke tijdstap een frame te laten doorzoeken op een tegenstander, dit bleek echter ook erg lastig. Het duurt lang voordat de simulatie geladen is, waardoor het lang duurt voordat de geschreven code getest kan worden. Dit maakte het moeilijk om in de beperkte tijd deze code werkend te krijgen, dit is ook niet gelukt.

Ervanuitgaande dat wij geslaagd waren in het verbeteren van de visual recognition dan zou de robot nu kunnen herkennen of de tegenstander rechtop staat of op de grond ligt. Dit geeft hem de mogelijkheid om een klein beetje naar voren te lopen zodra zijn tegenstander op de grond ligt. Hiermee kan de robot als volgt worden geprogrameerd: als eerst loopt die een standaard afstand

naar voren. Hierna laat hij zijn tegenstander naar zich toe lopen, totdat die op een goede afstand staat om de duwactie te ondergaan en vervolgens een klein stuk naar voren te lopen. Als dit proces herhaald wordt, zou onze robot uiteindelijk zijn tegenstander van de mat af moeten duwen, aangezien de tegenstander het standaardpatroon van opstaan en lopen zal volgen en daardoor weer in positie zal komen om omgeduwd te worden.Deze tactiek werkt tegen de sparring robot, maar het is twijfelenswaardig of deze ook werkt in de echte competitie. Alhoewel, dit ons niet gelukt is omdat we niet bij de behaviours in de sumo_fighter directory kunnen aanpassen. Alsnog hebben we een manier gevonden om de robot te laten winnen tegen zijn tegenstander.

Het proces ging als volgt. Ten eerste is de standaard afstand die de robot op het begin aflegt verhoogd. Vervolgens maakt de robot een beweging waardoor de tegenstander omvalt, namelijk een handstand. De robot eindigt liggend, dit betekende voor ons dat we moesten implementeren dat de robot vanzelf weer op stond. De robot hoort dit al te doen volgens tweede stap van de while loop (uitgelegd aan het begin), maar aangezien de robot zijn tegenstander nog ziet terwijl die op de grond ligt, probeert die stap vier uit te voeren, in plaats van stap één of twee. Hierop hebben we bedacht om twee extra if-statements te implementeren waarmee de robot zichzelf omhoog werkt en vervolgens door gaat naar stap vier. Hierna wordt de loop herhaald totdat zijn tegenstander uit de ring valt.

Het visuele resultaat is te vinden in de youtube link hieronder:
`https://youtu.be/HbzjMK5RSaY`
De link naar onze code staat in onderstaande google drive link:
`https://drive.google.com/folderview?id=0B_ab-mXSsIQdWXBCUk1DcDF5X3M&usp=drive_web`

Reflecterend op deze week, kregen we het gevoel dat het een erg uitdagend project was. Geen van ons was bekend met C++, maar dit heeft ons juist gestimuleerd om beter hier aan te werken. Verbeterpunten zouden zijn om een beter plan te verzinnen aan het begin van het project, zodat er misverstanden zijn. Bovendien kan er dan gewerkt worden aan de interne communicatie en verloopt de rest van het project soepeler. Verder zijn we tevreden met ons resultaat, echter was het wel jamer dat we een simulator moesten gebruiken in plaats van echte robots. Bij een betere voorbereiding hadden wij waarschijnlijk een ander project gedaan met andere(echte) robots.