Fight or Flee

Herprogrammeren van een Aibo ERS-7

Iason de Bondt - 10266224
Ivo Hoolwerf - 10207058
Jorgos Tsovilis - 10246878
Tom van den Bogaart - 10020934

4 juli 2013

Abstract

Voor het project 'Zoeken, Sturen & Bewegen' is er gekozen voor het implementeren van een programma voor de AIBO ERS7. Het volgende is onderzocht: 'is het mogelijk om met de AIBO ERS7 twee verschillende geluidniveaus te onderscheiden door vervolgens een bijbehorende reactie uit te voeren?'. In dit project is een programma ontwikkeld dat er voor zorgt dat de AIBO een aanvalshouding aanneemt en begint te blaffen als er een zacht geluid wordt gehoord. Daarnaast zal de AIBO zich omdraaien en wegrennen bij een hard geluid. Het implementeren van dit programma is gedaan met de open source programmeertaal URBI [2], dit is een implementatie van C++ samen met een aantal ingebouwde functies voor de AIBO ERS7. De resultaten voor het project waren positief, wat betekent dat de AIBO ERS7 reageerde op een geluid van 95 dB (het zachte geluid) door in een aanvalshouding te gaan staan en te gaan blaffen. De AIBO ERS7 kon ook onderscheid maken tussen dit geluid en het geluid van 102 dB (het harde geluid) waar op werd gereageerd door 180 graden te draaien en vervolgens weg te rennen. Naast de beschreven acties voor de AIBO ERS7 zijn er extra methodes geïmplementeerd die ervoor zorgen dat de acties beter verlopen. Het is dus mogelijk om de AIBO ERS7 twee verschillende geluidsniveaus te laten onderscheiden en hierop een bijbehorende actie uit te voeren.

1. Inleiding

De opdracht die de studenten Kunstmatige Intelligentie hebben gekregen voor de laatste week van het project 'Zoeken, Sturen & Bewegen' luidt: *Go, where no one has gone before*. Dit houdt in dat de studenten zelf een robot programmeren, zodat het een bepaalde inventieve en originele actie uit kan voeren met betrekking tot Kunstmatige Intelligentie. Er zijn vrij weinig restricties, zo mogen de studenten zelf uitkiezen met welke robot ze gaan werken of wat de plannen zijn met deze robot.

Er is gekozen voor het herprogrammeren van een AIBO (Artificial Intelligence roBOtdog) van het type ERS7, gefabriceerd door het merk Sony in het jaar 2003. De productie van verschillende AIBO's liep van het jaar 1999 tot 2005, hierna is de productie gestopt [3][4]. Het gekozen idee dat geïmplementeerd moet worden voor het project houdt het volgende in: zodra de AIBO het zachte gemiauw van een kat hoort, moet het een aanvalshouding aannemen en gaan blaffen. Echter, zodra de AIBO een harde brul hoort, moet het zich omdraaien en piepend wegrennen. De onderzoeksvraag voor het project luidt dan ook: 'is het mogelijk om met de AIBO ERS7 twee verschillende geluidniveaus te onderscheiden door vervolgens een bijbehorende reactie uit te voeren?'. Er word verwacht dat deze opdracht uitvoerbaar moet zijn, aangezien de AIBO twee geluid sensoren heeft en er een open source programmeertaal voor de AIBO bestaat, genaamd URBI (Universal Real-Time Behavior Interface). De programmeertaal URBI is een implementatie van C++, waardoor de keuze is gemaakt om gebruik te maken van Windows Visual Studio 2008.

Er wordt begonnen met het beschrijven van de methode die is gebruikt voor dit onderzoek. Ten tweede zal er dieper worden ingegaan op de gevonden resultaten. Tenslotte wordt besproken of het mogelijk is om de AIBO twee verschillende geluidsniveaus te laten onderscheiden om daarbij verschillende reacties uit te voeren.

2. Materiaal en methode

Het benodigde materiaal voor het experiment is te vinden in Appendix A.

Om de AIBO te laten reageren naar behoren, is er begonnen met het implementeren van de microfoon-functie in URBI.

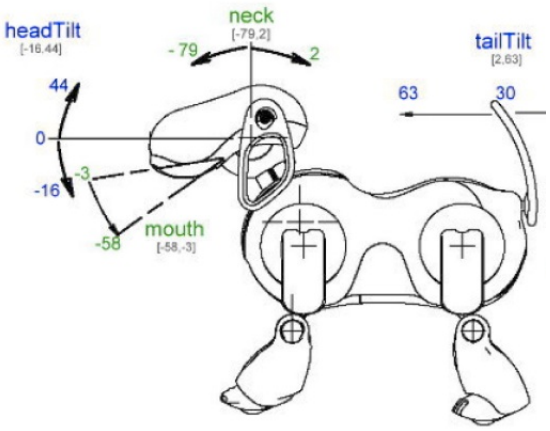
Hierna was het de taak om de robothond twee verschillende acties uit te laten voeren die als reactie zouden gelden op geluid. Er is gekozen voor een simulatie van een boze en een huilende emotie. Om de emoties natuurlijk te simuleren wordt er gebruikt gemaakt van de motoren, de led lampen op het hoofd, en de speakers.

Bij het implementeren van de boze emotie wordt de AIBO zodanig gesimuleerd dat de robot een aanvalshouding aanneemt, de led lampen rood kleuren en er een blaffend geluid afspeelt. Om het blaffen van de AIBO natuurlijk over te laten komen is ervoor gekozen de mond open en dicht te laten gaan tijdens het afspelen van het geluid.

In de uitvoering van de huilende emotie zal de AIBO zich 180 graden omdraaien en wegrennen, de led lampen blauw kleuren en zal de robot een geluid afspelen van een piepende hond. Daarnaast zal de staart in een zo laag mogelijke stand worden gezet om de werkelijkheid waarin een hond huilt te benaderen.

Tenslotte wordt er een functie ingebouwd, wat zal voorkomen dat de AIBO bij het wegrennen in zijn 'vlucht reactie' niet tegen een object aanloopt. Er zal hier gebruik gemaakt worden van de sensor in de neus.

Om dit experiment te repliceren, is het vereist om de code genaamd *micro.u* op de geheugenkaart te zetten, en in het bestand *URBI.INI* de regel `load("micro.u");` toe te voegen.



Figuur 1: Illustratie van verscheidene motoren met bijbehorende waardes van de AIBO ERS7 [1]

3. Resultaten

De gestelde onderzoeksvraag is getest met twee verschillende geluiden, een zacht en een hard geluid. Verschillend in het niveau van het volume, respectievelijk 95 en 102 decibel[1]. Om dit geluid op te vangen is er gebruik gemaakt van de functie `micro.volume`, zoals te zien is in Appendix B regels 56-68. Deze regels code realiseren dat bij een geluid tussen de 3000 en 4500 (eenheid is niet bekend) de functie `fight()` begint en boven de 4500 functie `flee()` wordt aangeroepen.

De functie `fight()` staat op regels 48-53 en zorgt ervoor dat het bestand *dogoff.wav* wordt afgespeeld, dit is het blaffen van de AIBO. De led lampen op het hoofd vertonen een boze emotie met behulp van `setAngryEyes()`, zie regels 4-13. Daarnaast neemt de robot een aanvalshouding aan met `robot.attack()`, zie regels 70-85. Om het blaffen van de AIBO natuurlijk te simuleren is er voor gekozen de mond open en dicht te laten gaan tijdens het afspelen van het geluid. Dit wordt gerealiseerd door de waarde van de mond te laten gedragen als een sinus functie die binnen het minimum -58 en maximum -3 schommelt (zie figuur 1 en regel 83). Vervolgens gaat de robot weer terug in neutrale positie met behulp van functie `robot.standback()`.

De functie `flee()` is te vinden op regels 26-45 in Appendix B en is meer uitgebreid. Het begint met het configureren van de led lampen zodat de AIBO een zielige expressie uitstraalt met behulp van `setSadEyes()`, zie regels 15-24. Tegelijkertijd wordt het bestand *dogg.wav* afgespeeld, wat ervoor zorgt dat de robot een huilend geluid maakt. Het oorspronkelijke idee om de staart tussen de benen te laten hangen kon door de gebrekkige mogelijke standen niet gerealiseerd worden. Wel staat de staart zo ver mogelijk naar beneden (zie figuur 1 in combinatie met regels 32-33). Vervolgens keert de robot 180 graden om en rent weg van zijn huidige positie (zie regel 34).

Als de afstand tussen het hoofd van de robot en een object minder dan 20 centimeter is, draait de robot 90 graden met de klok mee en zal het verder lopen. Als dit gedaan is, zal de robot een neutrale positie innemen door `robot.standback()` uit te voeren.
Een extra functie `wait()` (zie regel 62, 66) is toegevoegd om te voorkomen dat de AIBO de functies `fight()` en `flee()` parallel uitvoert. Dit is een extra preventie, zodat de AIBO niet tijdens het uitvoeren van de `fight()` functie zijn eigen geblaf opvangt en daar voor wegrent.

[1]Meetwaarden verkregen door herhaaldelijk testen met de app *Decibel 10th* voor *iPhone*

4. Conclusie en Discussie

De resultaten in beschouwing genomen kan geconcludeerd worden dat de verwachtingen van het experiment correct waren. De AIBO ERS7 is, na het implementeren van de bijgeleverde code, inderdaad in staat om twee verschillende geluidsniveaus te onderscheiden door vervolgens een bijbehorende reactie uit te voeren.

Naast de onderzoeksvraag die met de gegeven resultaten beantwoord is, is er een opzet gemaakt voor een mogelijk vervolgonderzoek. Dit is het navigeren van een AIBO in een kunstmatig doolhof [3]. In het oorspronkelijke programma is de functie *flee* uitgebreid zodat de AIBO bij het wegrennen een obstakel detecteert voordat het een botsing maakt, om vervolgens zich om te draaien en een ander pad te kiezen om zijn weg te voltooien. Dit is gedaan met behulp van een sensor die de afstand kan berekenen. De uitdaging in dit vervolgonderzoek is nu niet meer het detecteren van objecten, maar het bedenken van een *path planning* algoritme zodat de AIBO zonder obstakels te raken zich door een kunstmatig doolhof kan manoeuvreren.

Referenties

[1] Xavier Blanc. *Applying MDE for AIBO*. nd. URL: `http://www.powershow.com/view1/1f8d9e-ZTJmZ/Applying_MDE_for_AIBO_powerpoint_ppt_presentation`.

[2] Gostai. *The urbiscript language is a dynamic script language, focusing on prototype-oriented programming and object programming.* Jul 2013. URL: `http://www.urbiforge.org/`.

[3] Kyle Lawton en Elizabeth Shrecengost. "The Sony AIBO: Using IR for Maze Navigation". In: *Journal of the Pennsylvania Governor's School for the Sciences* 23 (2005), p. 203–215.

[4] Eric A. Taub. *For Sony's Robotic Aibo, It's the Last Year of the Dog.* Jan 2006. URL: `http://www.nytimes.com/2006/01/30/technology/30aibo.html?_r=0`.

Appendices

A. Materiaal

- Sony Aibo ERS-7
- Memory Card met URBI geïnstalleerd
- Memory Card reader
- Aibo charging station
- Computer (Operating System niet belangrijk) met tekstverwerker.

B. Broncode

Bestand micro.u:

```
group AngryEyes {ledF13,ledF14};
group SadEyes {ledF3,ledF4,ledF5,ledF6,ledF7,ledF8};

function setAngryEyes() // Procedure for the eyes
  {
     AngryEyes.val=1;
     modeR.val=1;
     ledBFC.val=1;
     wait(10s);
     AngryEyes.val=0;
     modeR.val=0;
     ledBFC.val=0;
 };

function setSadEyes() // Procedure for the eyes
  {
     SadEyes.val=1;
     modeB.val=1;
     ledBFC.val=1;
     wait(10s);
     SadEyes.val=0;
     modeB.val=0;
     ledBFC.val=0;
  };

// Actions when fleeing.
Y=0;
function flee()
  {
    setSadEyes(),
    loopn (6) { speaker.play("dogg.wav") } &
    tailPan.val = -59 time:2000 &
    tailTilt.val = 2,
    robot.turn(3500) | stopif(distanceNear.val < 20) robot.walk(15000);
    at(distanceNear.val < 20)
      {
        if (Y != 1)
          {
            Y=1 | robot.turn(2000) | stopif(distanceNear.val < 20) robot.walk(5000);
          };
      };
    wait(8s);
    robot.standback();
```

```
44.     Y=0;
45.   };
46.
47. // Actions for fighting.
48. function fight()
49.   {
50.     loopn (16) { speaker.play("dogoff.wav") } &
51.     setAngryEyes(),
52.     robot.attack();
53.     robot.standback();
54.   };
55.
56. // Determining what action it takes, depending on sound input.
57. X=0;
58. at(micro.volume > 3000 && X != 1)
59.   {
60.     if(micro.volume <= 4500)
61.     {
62.        X=1 | fight() | wait(8s) | X=0;
63.     };
64.     if (micro.volume > 4500)
65.     {
66.         X=1 | flee() | wait(10s) | X=0;
67.       };
68.   };
69.
70. // Attack stance
71. function robot.attack()
72. {
73.   head.val = 0 time:1000;
74.   headTilt.val = 44 time:1000&
75.   neck.val = 2 time:1000 &
76.   tail.val = 0 time:2000 &
77.   legF1.val = 80 time:2000 &
78.   legF2.val = 18 time:2000 &
79.   legF3.val = 20 time:2000 &
80.   legH1.val = -25 time:2000 &
81.   legH2.val = 10 time:2000 &
82.   legH3.val = 0 time:2000 &
83.   timeout(10s) mouth.val = -26 sin:1s ampli:22;
84.   mouth.val = -3 time:500;
85. };
86.
87. // Makes the robot go back to normal position.
88. function robot.standback()
89. {
90.   head.val = 0 time:1000;
91.   neck.val = 0 time:1000 &
92.   tail.val = 0 time:2000 &
93.   legH1.val = 0 time:2000 &
94.   legH2.val = 10 time:2000 &
95.   legH3.val = 0 time:2000 &
96.   legF1.val = 0 time:2000 &
97.   legF2.val = 0 time:2000 &
98.   legF3.val = 0 time:2000
99. };
```