DBD04_Learning Robots

Students

F.N.A. Al-Kaylani, K.J.F de Greef, N.M. Nelson

Februari 2011

Contents

1. Introduction

Brains can be very complex; the human brain can contain roughly 15-33 billion neurons that are linked with op to 10.000 synaptic connections each. The communication between the neurons occurs through the axons. These axons (long protoplasmic fibers) carry trains of signal pulses that are able to activate specific recipient cells (Brain).

With the technology nowadays it is possible to measure neural activations that can be our emotions, social interactions, physical behavior and meditation experience. The same principles can be applied in the context of neuro-robots. In this context learning algorithms are used to train and learn a robot to perform certain activities autonomously that are inspired from the human brain.

This report describes a concept inspired from a real life situation where a robot learns from environmental input based on a supervised algorithm and the reinforced learning algorithm (q-learning) to suggest clothing in a shopping context.

2. Concept

The concept is inspired from the shopping context where the main idea is to have a robot that's able to guide people through a clothing store and to suggest clothing that fits them. This idea is suitable in the context of a learning robot where a robot is able to dynamically adapt towards new clothing in a shop, analyze peoples clothing and learn from feedback of people. The robot is able to learn from what a person bought and to link that with the persons own clothing. Information from different users is continuously updated where the robot also can recognize for example the most popular clothing.

A schematic view of the input, memory, output and feedback loop are shown at Figure 1.

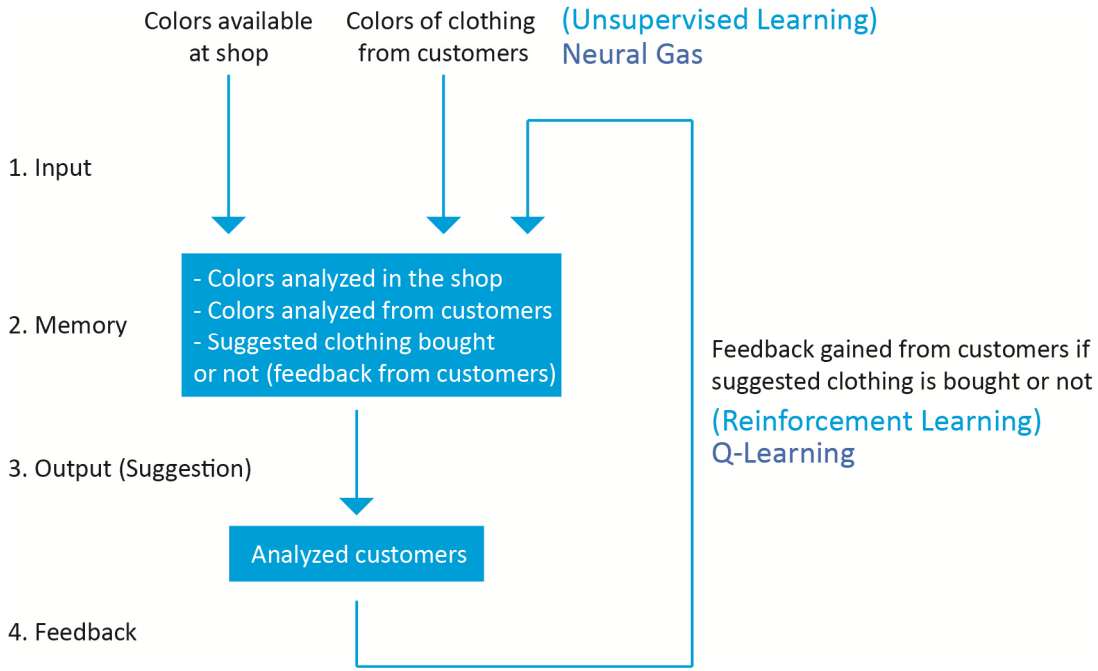
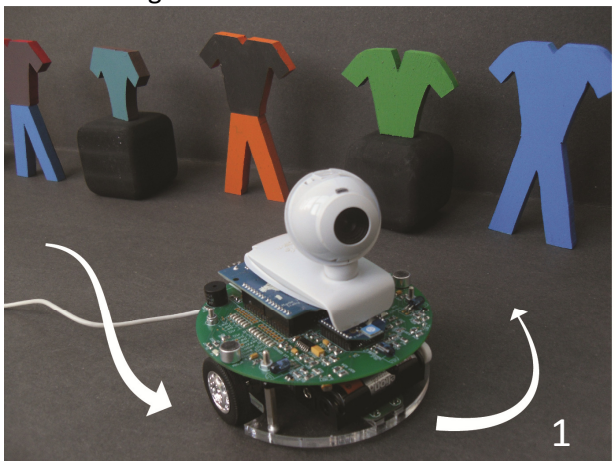


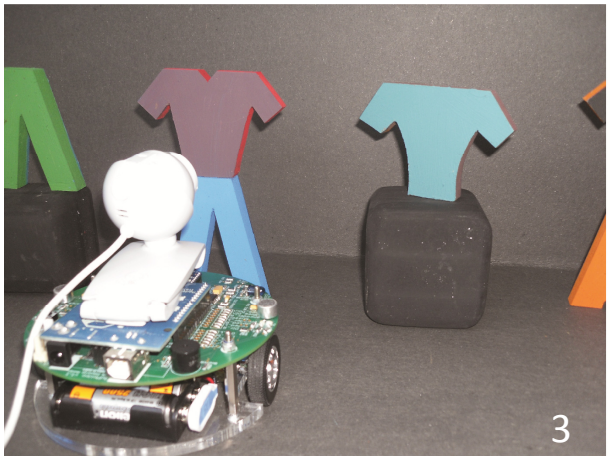
Figure 1, schematic view of interactions of the robot

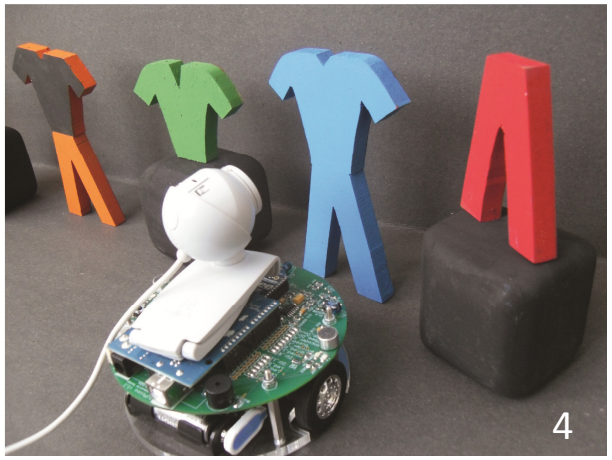
The robot first gets environmental input from the available clothing at the shop. This information is gained from a webcam attached on the admoveo (see Fig. 2). A random function gets the RGB values from several pixels that are used as input for the neurons. This information is then clustered at different classes according to the difference in values and stored in the memory (2). The clothes of customers in the shop will then be grouped to the classes of clothing available in the shop and according to the colors of clothing from customers and the available colors in the shop a suggestion is made by the robot by guiding the customer to different clothing in the shop. The customer then decides to buy the item or not. This type of reinforcement learning (Q-learning) gives the opportunity to the robot to learn from the actions of customers and to compare with its own actions. This makes it possible to improve the suggestions by continuously updating the memory and learning from customers.

2.1 Scenario

The following scenario describes the interaction between the robot and customers in a shop. A detailed description can be found at the next page.







Description scenario:

1. The robot first orientates itself through the shop in order to identify different available colors of clothing. In an optimum situation this will be done with the unsupervised learning algorithm, growing neural gas. The advantage of this algorithm is that the robot automatically can create new classes. This gives the possibility to add for example new clothing in the shop, the algorithm will then identify that there is a new class of clothing if it differs too much from existing classes, and will make a new class. This means that the growing neural gas algorithm continuously adapts to environmental changes.
   Although the unsupervised learning algorithm in this context will be the most suitable it was not applied, instead a supervised learning algorithm was implemented due to time restrictions. With this algorithm several color classes where implemented by capturing different colors and implementing the RGB values in a neural gas algorithm. After training the algorithm, several classes were subtracted that identified different colors. This makes it possible to cluster colors according to the trained program.

2. The next phase, shown at image two, is a customer that wants to enter the shop. At the beginning of the shop the robot will analyze the clothing of the customer by processing the image and translating it into RGB values. It will then classify the values within its classes of colors that were gained at step one. The robot will then be able to make suggestions according to the colors that a customer is wearing.

3 & 4. We can see at the third and fourth image that the robot suggests clothing for the customer. In this example we see that the robot suggests blue clothing for the customer. Off course this is one part of a suggestion system where only color is used. In a more complex situation more factors such as patterns, clothing material and brand could be added to the suggestion system.
After the suggestion is made the customer will either buy or not buy the item. This provides the robot with feedback on its suggestions. The robot will learn from its mistakes and is able to optimize its suggestions. This is done with the Q-learning algorithm (reinforced learning).

See next chapter (3. Prototype) for the implementation of the hardware and software of the robot.

3. Prototype

3.1 Hardware

For this application the AdMoVeo robot platform is used (Hu & Alers, 2009). The AdMoVeo robot has the following sensors and actuators included:

Sensors:

- Infrared distance sensors
- Light sensors
- Sound sensors
- Line follower
- Wheel encoders (optional)

Actuators:

- Motors
- Sound
- Light

For our concept a webcam is needed that's not included in the AdMoVeo, so a webcam has been mounted on the top of the AdMoVeo (see Fig. 2). The webcam is needed to identify colors from humans and from the robot.

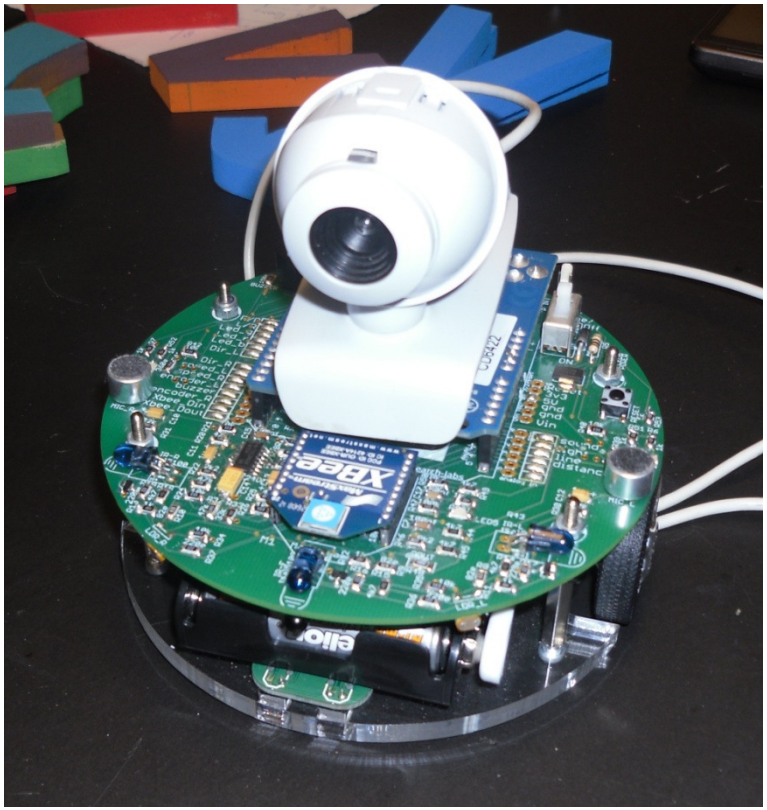


Figure 2, AdMoveo and webcam

3.2 Software

The programming language “processing” is used to control the AdMoVeo. This programming language was combined with the algorithms available from Matlab. The two programs communicate with each other, which makes it possible to get environmental image information from the webcam, to analyze it, implement it in the trained algorithm at Matlab and to control the robot in processing (see Fig. 3). Code can be found in “appendix: code”.

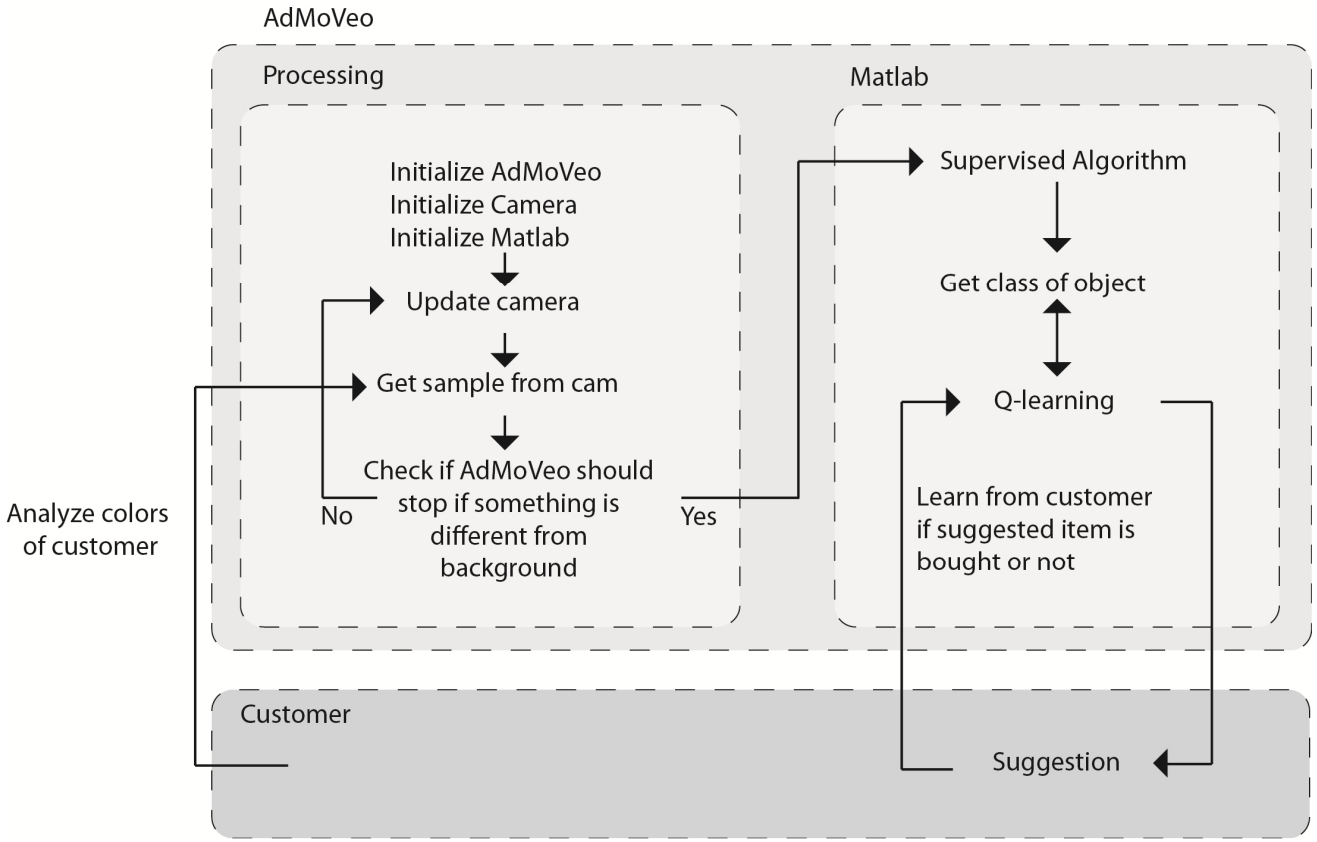


Figure 3, schematic view of communication between programs

4. Evaluation

At the faculty we aim at the design and creation of intelligent systems, products and services. We learned that an intelligent product consists of adaptive capabilities for specific situations, context of use and user's needs. During the module we found out that these adaptive capabilities can be reached trough making a product or system learn. Learning can give a product or a system capacity and knowledge which was not yet present at the design stage. Learning is very useful when a fixed structure cannot be found at the design stage. The desired structure or information can be established in a later stage, trough interaction with the environment.
We believe that we accomplished in this sense. We came up with a concept which is applicable in everyday situations and can be of use for specific user groups. In the situation of our concept it was not yet clear in the design stage which colors would be worn with what colors. Therefore the learning is very useful. This is especially the case in fashion, since fashion constantly changes. If our concept would be able to learn the latest fashion colors from customers it would not have to be re-programmed every now and then.

In our concept two different types of learning are implemented, namely Unsupervised Learning (Neural Gas) and Reinforcement Learning (Q Learning). We succeeded in getting the Neural Gas working. We trained the robot to recognize different colors which it could cluster. Due to time issues we were unable to implement Q Learning. Though we did get a good understanding of what these two different types of learning mean and how they can be used.

During the course of this module we had most trouble connecting MATLAB with processing. Processing is capable of quick video processing and controlling the AdMoVeo, whereas MATLAB is very useful for writing algorithms for the actual learning part of the robot. We found a way on how to make communication possible between processing and MATLAB. We will upload our method to the wiki site so follow up students can quickly get started with communicating between processing and MATLAB, thus increasing their time to actually work on the learning part of the robots.

References

Brain. (sd). Opgeroepen op 2 15, 2011, van Wikipedia: http://en.wikipedia.org/wiki/Brain

Hu, J., & Alers, S. (2009). AdMoVeo: an educational robotic platform for learning behavior programming., (p. 2). Taipei.

Appendix: Code

Processing

```
import processing.serial.*;                //processing serial connection for the adMoveo
import nl.tue.id.creapro.admoveo.*;   //adMoveo library
import JMyron.*;                           //jMyron library for controlling a webcam
import matlabcontrol.*;                    //matlabcontrol library for communication with Matlab

int cycle = 0;
int cycleSize = 10;
int nrSamples = 20;                        //number of color samples to take
int[] readRect = {170,300,110,300};      //area where to measure (x,y,width,height) must be between 640,480
int[] backArray = {161, 129, 127};       //array with the RGB values of the background
int backLimit = 20;                        //how much the sampled color has to differ from the background for action

boolean drive = true;
boolean getSample = true;
double[] colorArray = new double [nrSamples*3];

RemoteMatlabProxyFactory factory ;
RemoteMatlabProxy proxy;
AdMoVeo admoveo;
JMyron m;

void setup() {
 //ini adMoveo at COM16
 admoveo = new AdMoVeo(this, "COM16");

 //ini Cam with JMyron at 640x480
 size(640,480);
 m = new JMyron();
 m.start(width,height);
 m.findGlobs(0);
 loadPixels();

 //ini MATLAB Create a factory, launch a proxy & MATLAB, display confirmation in MATLAB
 try {
  factory = new RemoteMatlabProxyFactory();
  proxy = factory.getProxy();
  proxy.eval("disp('Processing and Matlab are now connected')");
 }
 //catch errors if Matlab does not respond
```

```
  catch (MatlabConnectionException me){}
  catch (MatlabInvocationException mie){}
  delay(10000);
}

void draw() {
  updateCamera();              //update the camera image
  drawSampleArea();            //draw a rectangle on the area where samples are taken
  getSample();                 //get a video RGB sample
  calcStop();                  //calculate if the Admoveo should stop

  //if getSample is true, stop the adMoveo and send the last sample to the test function in Matlab
  Object testResult = 0;
  if(drive == false && getSample == true){
    moveAdmoveo();
    Object[] oTestArray = {colorArray};
    try{
      testResult = proxy.returningFeval("test", oTestArray);
    }
    catch(MatlabInvocationException mie){}
    println(testResult);
    drive = true;
    getSample = false;
  }else if(drive == true){
    moveAdmoveo();
  }

  if(getSample == false){
    cycle ++;
    if(cycle == cycleSize){
      getSample = true;
      cycle = 0;
    }
  }

  delay(100);
}

//update the camera view and draw it on the stage
void updateCamera() {
  m.update();
  m.imageCopy(pixels);
  updatePixels();
```

```
}

//draw lines around the sample area on stage
void drawSampleArea() {
 strokeWeight(2);
 line(readRect[0],readRect[2],(readRect[0]+readRect[1]),readRect[2]);
 line((readRect[0]+readRect[1]),readRect[2],(readRect[0]+readRect[1]),(readRect[2]+readRect[3]));
 line((readRect[0]+readRect[1]),(readRect[2]+readRect[3]),readRect[0],(readRect[2]+readRect[3]));
 line(readRect[0],(readRect[2]+readRect[3]),readRect[0],readRect[2]);
}

//take nrSamples times a sample from the camera image and store them in the colorArray
void getSample() {
 int[] img = m.image();                                  //get the normal image of the camera
 for(int i=0; i<nrSamples; i++) {
  int x = int((random(readRect[1])))+readRect[0];        //random coordinatse within sample area
  int y = int((random(readRect[3])))+readRect[2];

  ellipse(x, y, 2, 2);                                   //draw the sample coordinate

  colorArray[i*3] = int(red(img[y*width+x]));            //store the samples in the colorArray
  colorArray[(i*3)+1] = int(green(img[y*width+x]));
  colorArray[(i*3)+2] = int(blue(img[y*width+x]));
 }
}

//calculate if the average color from the samples differs enough from the background color to stop the adMoveo
void calcStop() {
 //calc the average color of all samples
 int[] avgArray = {0, 0, 0};
 for(int i=0; i<60; i++) {
  if(i < 20) {avgArray[0] += (int)colorArray[i];}
  if(i < 40) {avgArray[1] += (int)colorArray[i];}
  if(i < 60) {avgArray[2] += (int)colorArray[i];}
 }
 for(int i=0; i<3; i++) {
  avgArray[i] = avgArray[i]/20;
 }

 //if the average color differs from the background color by at least backLimit stop the adMoveo
 if(abs(backArray[0]-avgArray[0]) > backLimit && abs(backArray[1]-avgArray[1]) > backLimit && abs(backArray[2]-avgArray[2]) > backLimit) {
  drive = false;
 }
```

```
}

//either more or stop the adMoveo
void moveAdmoveo(){
  if(drive == false){
    admoveo.getLeftMotor().setPower(0);
    admoveo.getRightMotor().setPower(0);
  }else if(drive == true){
    admoveo.getLeftMotor().setPower(200);
    admoveo.getRightMotor().setPower(200);
  }
}

//get camera settings if the stage is clicked (will crash on Mac)
void mousePressed() {
  m.settings();
}
```