Subsumption Architecture-Based Social Interactive Robot Arm

Jens van de Water, Maarten H. Lamers
Leiden Institute of Advanced Computer Science, Leiden University, The Netherlands
jensvdwater@planet.nl, lamers@liacs.nl

1. Abstract

In this paper we describe how we tried to give a robot arm social interactive behavior based on its video input via a webcam. Based on Brooks' subsumption architecture, each form of behavior was implemented into a different layer. Not much information is known about this approach for social interactive robots, resulting in most of the paper to explain how my idea was to implement behavior into the robot and what the results were.

2. Introduction

Futuristic films feature many robots as human-like machines. Getting to the point where robots are like humans, by giving them the ability to think, respond and sense, is still far from where we are nowadays. Since it is not easy to comprehend the human mind, creating realistic behavior for robots is difficult to accomplish. One social interactive robot is KISMET [1], a robot programmed to mimic reaction, which is only a small part of what we define as social behavior. Our project focuses on creating realistic basic social behavior, using only a mechanical arm for movement and a sensor for vision. It aims at demonstrating that minimal mechanical and computational complexity can generate behavior that is perceived by onlookers as intended social response.

Much research was done in robotics and artificial intelligence, resulting in different ideas on how to design behavior based robots. A well-known approach towards behavior based artificial intelligence is Brooks' subsumption architecture [2], where the programming is done within several layers, each having their own input, output and priority. The output should not depend on results from other layers and all control the robot individual. Amir and Maynard-Zhang [3], layer-decoupling of tasks has the possibility of achieving more reactive behavior, which is what we are working towards.

Subsumption architecture was often applied to develop robots [4], but most were not programmed for social interaction. MY REAL BABY [5] is an interactive animated doll, apparently implemented using subsumption architecture. Not much information is available about this doll, it was made for commercial purposes, not scientific.

But how to build a socially reactive robot using this method? Stojanov [6] indicates four different levels of mechanical behavior. The first category is the pure reactive system, which is the main functionality for our robot. The next level would be the ability to learn, by adding memory structures. So the first layers being created will only make it react, using a good priority system within the layers. After this, the robot can be extended to add other features, such as memory structures. The other two levels are eclectic architecture and the interactionist approach.

In this paper we answer the question whether it is possible to create a social interactive robot, able to interact naturally with a human, using the subsumption architecture and minimal mechanical and computational complexity.

3. Concept

Using the subsumption architecture, a clear layered concept was needed for making our robot, in which every layer has its own input and output connection to the real world. Our focus is on social interaction via the visual system of the robot, so all input comes from a webcam. The webcam captures a large stream of images, requiring our system to process it fast into useful data. This conversion is being done by the visual preprocessor, which outputs a matrix to the behavioral layers to calculate information from.

The layers themselves implement levels of behavior given to the robot. The human race is very complex, even when only focusing on what we do visually. Most actions we do as response to what we see, we are hardly aware off. There are four major factors which we started with, all normal reactions a human would have:

1. Reflex; When a person or object comes very close at a high speed, for example when assaulted, you would recoil for your own safety.
2. Focus; If you notice any movement, you tend to focus your attention to that object.
3. Distance; When an object has your attention but is distant, you need to come closer to examine it.

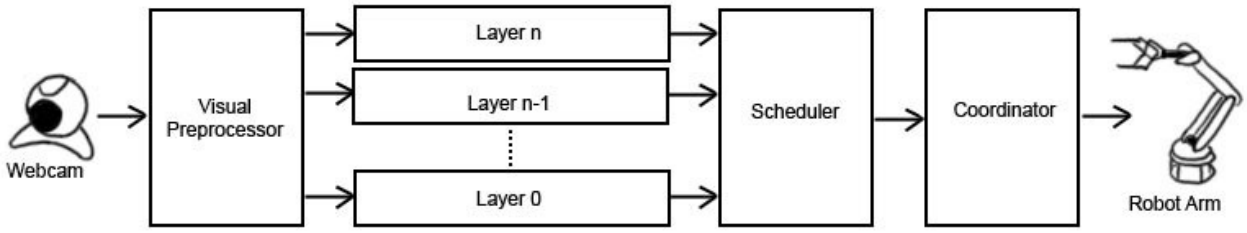

Figure 1: A schematic representation of the system controlling the robotic arm

4. Explore; When there is either no movement around you, or when focusing on the same object for a longer time, you will start looking around in the room for something different that might be interesting to look at.

These factors each define a different behavioral layer for the robot, where each processes its inputs differently.

The idea behind the subsumption architecture is that no layer depends on another and has its own control over the robot's movement. Our program combines the motion output from the four layers and converts it into a single movement for the robot to make.

The robotic arm needs to be coordinated for it to understand which components to move, in what direction and with what speed. The robot arm has six degrees of freedom: the shoulder and wrist are able to rotate and bend, the elbow can only bend, and the hand grasps with its 'fingers'. Controlling this arm is done using a coordinator module. Figure 1 illustrates the overall modular structure of our system.

4. Implementation

For our robot, we did all the implementation in the language Max/MSP/Jitter [7]. Its main advantage with regard to our project, is its ability to get data from video input fast and calculate with this information.

4a. Visual Preprocessor

The main purpose of the visual preprocessor is to receive the video input stream and convert it into useful data for the behavioral layers. It is built within three small layers, to be able to give different data depending on what is needed as input for each layer.

First, the input control ensures that the video input is handled correctly, depending on the frame rate and size. The frame rate applied in our system is 5 frames per second. An image is considered an ARGB-matrix, a four-planar matrix, containing values between 0 and 255 for each color's intensity. While the video input is 640x480, the data is stored in a 16x16 ARGB-matrix and is the first output of the visual preprocessor. Main reason being to have faster calculations in the following layers.

For the preprocessor to calculate movement, each two consecutive frames are subtracted, storing it in a third four-planar matrix. In this new matrix, non-zero values indicate where in the frame space movement took place.

The final function of the visual preprocessor is converting an ARPG-matrix into a single-planar matrix. The main reason for this conversion is to simplify later calculations and make heavy movement simpler to detect, the robot basically has a grayscale vision. Converting is done by summing up the values in the three color planars, discarding the alpha-planar which doesn't add information for our robot. These summed up values are first divided by the scalar 16 before being added, otherwise results higher than 255 could appear.

Though the matrix from the direct video input and the subtracted could be used for the behavioral layers, only the final single-planar matrix is being forwarded. This gives us the motion information needed for later calculations in a small and simple 16x16 matrix, without having to discard useless information over and over. If other information is needed after all, it can still be passed to the corresponding layer.

4b. Layer: Reflex

This behavioral layer is relatively simple; its functionality is to add all the values in the motion matrix and give us the result in a single number. When an object, like a hand, comes close to the robot within its visual field at a high velocity, this single number will be very high. No movement at all will result in a low value. When this result becomes too high, the reflective action should be enabled and the robot recoils at a relative high speed,

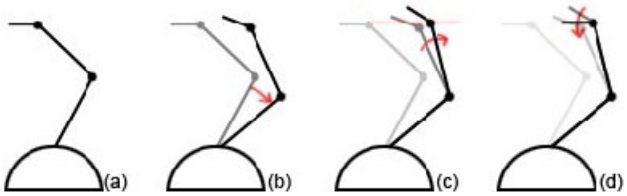

Figure 2: How the robot arm needs to compensate when the shoulder moves; from the original position (a) the shoulder recoils (b), resulting in moving the elbow to compensate the height (c) and the wrist to compensate for the direction(d)

equally to the human reaction. After this reaction, the robot will automatically move forward again to its old position, but at a slower rate. Until the robot is back at its former position before the recoil, no second reflex can take place.

On activating the reflex, the robot shoulder recoils and both the elbow and the wrist change their angle too, compensating the change in height as shown in figure 2.

4c. Layer: Focus

The most important layer of the three, containing several calculations to get the information needed for the robot to focus on a moving object. Its goal is to move the robot arm in the direction where movement was detected as if it is focusing on the moving person or object. It aims to position the motion at the center of the visual field.

To calculate a coordinate for the robot to focus on, several methods exist. Our first implementation was finding the maximum motion value and calculate its coordinates. The problem with this method is that it finds the position with the most movement, which is always around the edge of a moving object and not within. The maximum value of the waving hand in Figure 2 would be at the ring finger, and not at the middle of the hand. For this reason, a mathematical method was used to find the middle of the moving object. Per row and column the total sum is calculated and then multiplied by its row or column number. At the end this value is divided by the sum of all the values in the matrix, giving us a coordinate which has the most movement around itself:

$$Row.coordinate = (Row(0).sum*0 + Row(1).sum*1 + ... + Row(n).sum*n) / Matrix.sum$$

$$Column.coordinate = (Row(0).col*0 + Row(1).col*1 + ... + Row(n).col*n) / Matrix.sum$$

Now the coordinates are known, we can calculate the movement the robot needs to do in order to have the detected motion in the middle of its visual field. By simply subtracting each coordinate by eight, being the centre of the 16x16 matrix, the direction the robot has to move towards is calculated. These two results have a value between minus eight and eight. When the row coordinate has a positive value, the robot arm receives the message to move to the right, and moves to the left when the value is negative. The same method is applied to the column coordinates, where a positive value results in the robot looking down. If the absolute value of the directional result is high, obviously the motion detected was further away from the middle, which results in the robot moving faster in the given direction.

Figure 3: A waving hand is detected as shown in the right picture

4d. Layer: Distance

The purpose of this layer is similar to the function of the focus layer, difference being that focus works in the horizontal and vertical dimensions, whereas the distance layer works with depth. A human has two eyes for an important reason, so it is able to see depth in the world around him. Our robot only contains a single webcam, making it hard to calculate the real depth of an object. Resultingly, this layer has little influence on the interaction of the robot arm, but is still valuable. Human behavior is to come closer to an object that is small, to be able to see it more detailed, and when looking at a big object we tend to move backwards to be able to see it as a whole. This also includes the social distance, meaning when interacting with a person you keep a certain distance and do not shout over a long distance, or touch each other's noses. In essence, this layer scales (zooms in or out) to make the object fit in its visual field.

For this layer, first the amount of consecutive empty rows and columns on both sides of the motion matrix is calculated. For example, when the sum of the first and last columns have minimum values, the counter is heightened by one. Then the second and penultimate are checked for movement, in case this is detected, no more columns are checked and the value of the counter is forwarded.

When this value is bigger than two for the columns and the rows, the object is apparently too small within the visual field of the robot. The reaction would be to move the robot arm forward to get closer to this object. When both are smaller than two, the object is big and might not even fit within the matrix, resulting in a backwards retraction. When there is not much movement, the results would always be higher than two, resulting in a constant zooming in. We prevented this by adding the requirement that there has to be enough movement in the image to enable motion of the robot.

Equally to the reflex function, this layer activates its movements by changing the angle of the shoulder, elbow and wrist (see Figure 2). When enabled, these axes rotate with a constant factor, whereas the focus layer is using a variable value in order to control the robot's movement.

4e. Layer: Explore

Goal of this layer is to snap out of the focus and look around, but only when there is no movement detected by the robot. It then starts to explore the room around him, trying to find movement elsewhere. Notice how it is already split into two parts, likewise to how the robot is programmed.

The explore layer starts in the focused state, which means it isn't moving around and pays attention to a person or moving object. He loses interest though when all movement is gone, and start its 'boredom timer'. For every frame in which no motion is detected, its counter gets raised, and when the limit is reached, it switches to the explorative state. Movement resets this counter though, resulting in the exploration to start only after 5 consecutive seconds without motion.

This second state, the explorative state, simply means looking around randomly and searching for movement. When this is detected, it stops exploring and goes back to the focused state.

4f. Scheduler

The output of the behavioral layers are send to the scheduler. In here, the values get split into five groups, one for each axes. Since multiple layers control single axes, the scheduler gives priority to the highest value, i.e. the biggest motion. The final function in the scheduler is to add the values of each axes into a command for the robot arm itself.

The moment the coordinator gives a signal to the scheduler, a command can be send to the robot. These commands are send in a specific order:

1. At the start the 'homing' position is saved, this is the angle of each specific axes.
2. The commands from the scheduler change the values of the positions for each axis. After five commands have taken place, a new position is generated.
3. The final command in the scheduler is to force the robot to move towards the newly created position.

4g. Coordinator

The coordinator forwards its commands to the robot arm and awaits the robot's signal with the message it is ready for the next command. This signal is forwarded to the scheduler. In addition, the text messages generated by the robot are printed and can be used as feedback to the user.

5. Discussion

Unfortunately we were unable to test the robot's level of social interaction, which were caused by two major problems.

As explained before, the scheduler generates five new coordinates for the robot, a new coordinate for each of the axes, followed by the move command. This results in six commands have to be send to the robot for simply moving around, with the given input from the behavioral layers. Merely five of these sequences can be executed per second, including the webcam input, making the robot's responses slow compared to human behavior. Besides, all of the movements are done at an equally high speed. When the reaction requires a heavy movement, the robot will be at the right coordinate in time when it has a high pace. For short distances, the tempo is too high and results in unnatural vibrations, shaking the table the robot arm was placed upon.

Another weak link in our program is lacking background subtraction. Motion is detected by subtracting the last two frames from the webcam. When the robot moves, including the webcam, the background is a little different compared to the previous frame and is thus seen as motion. This distortion results in none of the behavioral layers to work properly. Nonetheless the robot does follow a heavily waving hand, recoil when you try to punch it, or start looking around when it doesn't see any movement for too long.

One might wonder whether we had any positive results at all. The robot is fully build up with the subsumption architecture, containing seven independent layers. The behavioral layers can be enabled and disabled without causing any conflicts with the other layers.

[MIS IK HIER NOG WAT???]

6. References

[1] C. Breazeal, "Designing Sociable Robots", MIT Press, Cambridge (USA), 2002, Chapter 1

[2] R.A. Brooks, "Intelligence without representation", MIT Artificial Intelligence Laboratory, Cambridge (USA), 1987

[3] E. Amir, P. Maynard-Zhang, "Logic-Based Subsumption Architecture", Artificial Intelligence 153 (1-2), 2004, pp 167-237.

[4] R.A. Brooks, "How to Build Complete Creatures Rather than Isolated Cognitive Simulators", MIT Artificial Intelligence Laboratory, Cambridge (USA), 1991

[5] L. Nicks, "The Robot – The Life Story of a Technology", Greenwood Press, Westport (USA), 2007

[6] G. Stojanov, "Petitagé: A Case Study in Developmental Robotics", In Balkenius, et al (Eds.),

[7] “Max/MSP/Jitter”, Cycling ’74, San Francisco (USA)