Development of a Behaviour Based Controlled Lego Vehicle

Roderik Emmerink

January 17, 2008

Abstract

This report discusses the development of Lego vehicle using two behaviour based methodologies. Both lateral control and the subsumption architecture are considered and evaluated as a way to control a Lego vehicle. These two architectures are compared to each other. The author comes to the conclusion that the lateral control is easier to implement and more practical to adjust.

Contents

Chapter 1

Introduction

This report covers the research undertaken in comparing two different behaviour-based architectures for the control of a Lego robot. The Lego robot is supposed to demonstrate the following behaviours:

- obstacle avoidance;
- wandering around in a random pattern;
- line following after a starting point on a S-shaped track has been found;
- collecting food/rewards;
- detecting a stop sign.

The robot is supposed to operate in an artificial environment which is shown in the following figure. The environment contains an S-shaped track, four rectangles that are interpreted as food, a starting point and an end point that are connected to the line on each side.

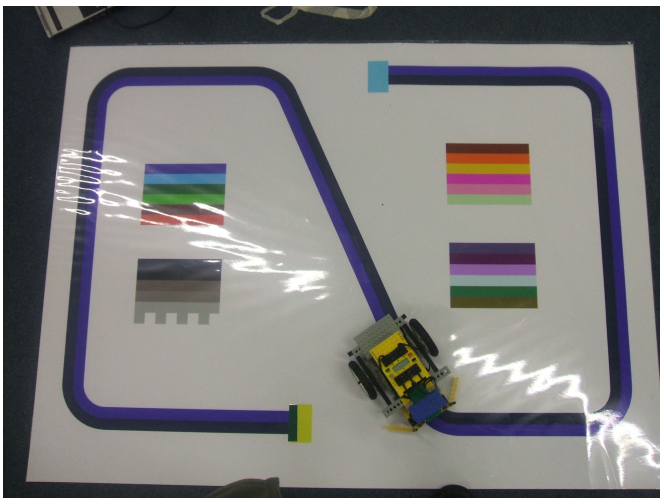

Figure 1.1: The environment for the robot to operate

Chapter 2

Design

Several architectures have been developed for behaviour-based robot control. A few of them are suitable for our purpose. Two of them, the lateral architecture and the subsumption architecture will be used for the experiment.

2.1 Lateral Architecture

A lateral architecture is one where every layer represents a behaviour and where every layer can control the sensors and actuaters, either directly or through a vertically oriented arbiter layer to prevent layers sent different output values at the same time, which will then interfere.

2.1.1 Wandering around

Although the layers do not really have an order in this lateral architecture, the wandering behaviour has been chosen to be implemented in the bottom layer. The second proposed architecture is the subsumption architecture, therefore it might be good to take the priorities into account. There are different opinions on what should be the bottom layer, usually this either obstacle avoidance or wandering. For example Pfeifer and Sheier [3] suggest to use wandering and Amir and Maynar-Reid [1] use obstacle avoidance as the bottom layer. Obstacles influence the basic behaviour of moving around and not the other way around therefore it was decided to take wandering as the bottom layer.

If there is nothing of higher importance the robot should be wandering around, therefore it is decided to start this behaviour directly after initialisation. The wandering can either happen in a structured way or at random. To explore the area in a structured way the robot can follow a line mowing pattern, but since it is not very accurate in its forward movement we need proper feedback to keep on a straight line. Furthermore, it is supposed to be a behaviour based implementation which is usually inspired by nature (can be ants in this case) therefore random movement is chosen (eventually higher layers can structure this behaviour). Experimentally a cycle containing a turn time of in between 400 ms and 900 ms in a random direction followed by a

forward movement of in between 500 ms and 2000 ms has been chosen to implement random wandering.

2.1.2 Obstacle Avoidance

Probably the most important behaviour of mobile robots that require some degree of autonomy is to avoid obstacles. This assignment took place somewhat within the theme of ant behaviour therefore the same methodology as suggested in my report about this topic [2] will be taken for this implementation.

The robot is equipped with two whiskers. When it bounces into an obstacle either on the front left side or the front right side the equivalent whisker sensor gets activated. Then it is necessary to inhibit the wandering behaviour and to move backwards to regain some freedom to move. In order to not bounce into the same obstacle again two strategies may come to mind. One is to make a turn to the opposite direction than the activated whisker, which would result in the highest probability not to bounce into the obstacle again. The other solution is to make a random turn. Although it might be suboptimal in terms of exploration speed it significantly reduces the chance to keep stuck in a U or V-shaped obstacle. After the turn, chosen to be in between 500 ms and 2500 ms, the wandering behaviour takes over again.

2.1.3 Line following

By analogy of ants that follow pheromone tracks the requirement to follow a line was given. When the robot finds the starting point it is supposed to follow an S-shaped track which is build up of a dark blue and black line. The robot is equipped with a light sensor that returns a value that represents shades of gray that are directly below it. When the starting point has been found (measured twice for redundancy purpose) the robot turns so till its sensor is above the line. When the robot loses the line (sensing white), because of a corner or because it is not moving straight, it stops its forward movement and tries to find the line again. This is done by turning left a short time and turning right double that time afterwards, this procedure is terminated after a short delay (so it is not exactly on the edge of the line). Some extra behaviour is later added to decrease the chance of losing the line. This is done by taking into account that the left side of the line is black and the right side of the line is blue. Therefore it was added to slightly reduce the speed of the wheel of the opposite side. Changing the speed by means of the power did hardly make any difference therefore it was chosen not to power one wheel for some time slots.

2.1.4 Collecting food

As an analogy for food there are 4 spots marked with a line pattern. The ant (lego vehicle) will make a stop and a sound to indicate that it had found the food. The line following behaviour is inhibited when the robot looks for food because the behaviours do interfere otherwise. One or more distinguishing colors are taken from these spots for the detection of the food. The robot

makes the sound and waits for half a second after which it will continue its search for food. It is not necessary to detect all of the line pattern, because it will only decrease the chance of detecting the food during its random walk and finding one or two distinguishing colours should already enough information to know that it is not any other spot on the map (in theory).

2.1.5 Detecting a stop sign

At the end of the line a stop spot can be found. Whenever this colour is detected for a short period it will make the ant die (as if it was poisonned), this means that the motors are stopped and all the layers/behaviours become inactive. For testing purpose the ant will reincarnate after a short moment.

2.2 Subsumption Architecture

The subsumption architecture, suggested by Brooks, has some similarities with the lateral architecture as described in last section. It is also based on different layers which represent mainly the different behaviours that are undertaken by the robot. On the other hand the subsumption architecture is more limited. It can suppress underlying layers but not layers that are above the layer. The subsumption architecture is hierarchical and needs to rely on lower layers for the low level control (e.g. motor control). Because of the partly sequential and partly parallel character of the behaviours it was too hard to implement the subsumption architecture itself. As an alternative a completely sequential architecture can be implemented, however some behaviours, obstacle avoidance for example, should be present in every state (sequence) which also makes that the lateral solution described in last chapter performs better. Nevertheless a sequential solution is easier to debug and layers will not interfere with each other which can be a major advance in robots that have to follow procedures in a controlled environment, but this is not suitable for our ant analogy.

Chapter 3

Implementation

For the implementation two platforms were considered. These were LEGO Bricx and the AIBO robot dog. The AIBO seemed to be the most suitable of the two since it has a colour camera which can be used to find the coloured area's that denote the food and other spots in the artificial environment. Nevertheless I was quite unexperienced with AIBO development and making and understanding software for the dog took me several weeks. Since I could never accomplish good results in time I chose to switch to the LEGO robot.

Although the default LEGO robots were quite suitable for this purpose I chose to build a wheeled robot. The robot has two wheels on its sides and one free sphere for stability. It uses gears to obtain an appropriate speed for the wheels so that the whole speed range might be of use for the control of the robot. The robot is equipped with a light sensor that can perceive shades of grey and two whiskers that can detect collisions with obstacles.

The software is implemented using NQC (Not Quite C) which made it possible to execute parallel processes by means of tasks. Every layer is implemented by means of a task. The sensor value of the light sensor is read as a raw value in order to gain the highest sensitivity.

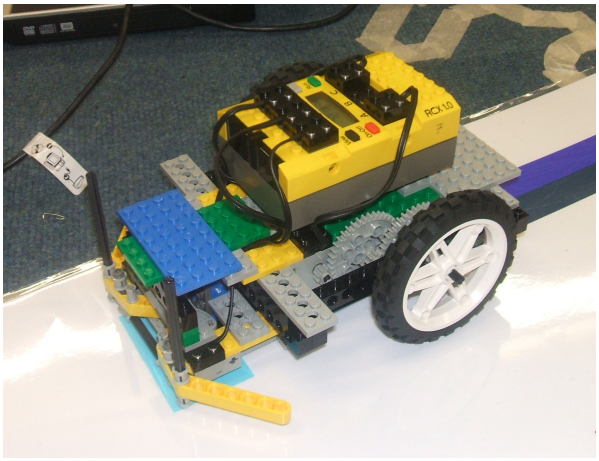

Figure 3.1: Embodiment of the LEGO Vehicle

Chapter 4

Testing and Results

Testing is undertaken by observation of the robot movement, by giving sound signals when the robot is in a certain state and by using a datalog that can be read out for logging intensity values from the light sensor at certain moments. All behaviours are tested seperately and in conjunction with other layers (as a whole).

4.1 Wandering around

Wandering around works as implemented.

4.2 Obstacle Avoidance

The obstacle avoidance works perfectly, when someone put his shoe (or any other obstacle) in front of the robot it touches one of the whiskers and the robot responds. When the robots moves into a U-shaped obstacle it will manage to move out of it. When both of the whiskers are touched together it will not make a difference for the response. Only when the obstacle does not touch the whiskers (like a pen in front of it 4 cm above the floor) then the robot will not detect the obstacle. In this environment that situation will not likely appear.

4.3 Line following

The line following layer on its own worked very well. Although the robot had a slight tendency to go to the right it corrected this when necessary. Nevertheless the turns to the left needed more corrections and took therefore slightly longer to take. Optimizing the robot to stay in the middle of the line by trying to make a distinction between the dark blue on the left and the black on the right did not work out. Although it was a good idea in its essence, due to different reflections the sensor returned identical values for blue and black from time to time which made it adjust in the opposite direction. When all layers were active at once there were also interferences with food collections.

Sometimes it accidentally detects food when reading the blue, black, white values of the line and therefore it exits the line following behaviour. The only way to overcome this is to disable the food detection while following the line. Incorrect colour values are often read (follows from logging these values in a datalog). This is due to difference of lighting conditions at different positions in the environment as well as reflections. To overcome this problem the active light sensor is covered somewhat to reduce the effects of the external lighting. Another reason that incorrect values are measured is the problem that it the resolution is just 1 sample at a time. When the robot is just above the edge of two different colours it measure an intermediate value that can for example match the stop condition. To overcome this problem the colours are usually read twice with a short time interval in between, only when it is in the correct range twice it will go into a new state. The lighting conditions can also be different becuase not all experiments are taken at the same time. To overcome this problem all necessary values are calibrated at the beginning of every experiment. A sample value is taken by touching a whisker. Although callibration sounds like a good solution, it did not improve the results. Some colour values are just to near to each other and because of callibration the range of accepted values for one colour had to be slightly decreased to prevent overlap of ranges. Different values appeared to be nearer to each other one time than the other time while callibrating. For this experiment, especially to make use of the two colours on the track, it is hardly adviced to use a light sensor that can either perceive colour and/or one that has a much better sensitivity.

4.4 Collecting food

Food collection worked with some of the food areas. Sometimes it detects food accidentally, mostly when passing the S-shaped line. The same problems concerning colour sensing as discussed in last section troubled the detection of food. Besides that it could take a huge amount of time for the robot to find the food since there was no way to predict where it could be found. A higher resolution of the sensor, e.g. by using a small camera, can overcome this problem.

4.5 Detecting a stop sign

Detecting the stop sign was not a problem. When the blue spot was sensed the robot stopped all its behaviours.

Chapter 5

Conclusion

It can be concluded that the the lateral architecture can be favoured when (mostly in theory) compared to the subsumption architecture. Furthermore it can be concluded that the robot does not perform optimal due to difficulties with colour sensing.

Bibliography

[1] Eyal Amir and Pedrito Maynard-Reid II. Logic-based subsumption architecture. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 147–152, 1999.

[2] Roderik Frederik Emmerink. Obstacle avoidance techniques and their appropriateness for imitating ant behaviour. 2008.

[3] Rolf Pfeifer and Christian Scheier. *Understanding Intelligence*. MIT Press, Cambridge, Mass., 1999.

Chapter 6

Appendix

A cd with the NQC code used for this experiment comes with this report.