Real Robots 1-Eyed Phototaxis

Contents

[Introduction 1](#_Toc8126549)

[Method 1](#_Toc8126550)

[Results 1](#_Toc8126551)

[Discussion 1](#_Toc8126552)

# Introduction

The purpose of this investigation is to develop an agent run on a lego mindstorm EV3. The agent should be capable of moving towards a light with only one light sensor. Through this investigation I intend to explore problems that arise with hard coding behaviour into a robot as opposed using an evolutionary approach.

# Method

On start, the initial problem to solve is find the direction of the strongest light source. Intuitively the only way to do this is have the agent spin in a circle to find the value of the strongest light source in its environment then move in that direction. With the lego mindstroms, using only one light sensor, it’s impossible for me to guarantee that the bot has spun in a full 360 circle without overshooting it. This overshooting is the method I employed, due to factors such as differing floor frictions, there is no one amount of spinning that guarantees exactly 360 spin. I spun the robot for enough time to do a 400-degree spin (4200 milliseconds) in order to counter this problem. This phase is called ‘checking’ and run through the ‘findStrongest()’ method.

Once the bot has detected the strongest light source around it, the bot starts turning till it encounters this light level again then proceeds to move forward until that light level starts reducing. The light level reducing is measured with a small buffer threshold in order to encourage movement with a very minor penalty to accuracy. This phase is called the ‘seek’ phase and is implemented using the ‘seek()’ method.

These 2 steps are repeated infinitely, ensuring the robot is constantly determining the strongest light source then seeking it. This does mean that the bot requires a roughly 400 degree turn as well as a second turn to find the light source again.

The success of each implementation was determined visually under 2 test environments;

1. The robot was set at a random distance and angle with 1 strong light source in which it was expected to seek out and move towards.
2. The robot was set in a random position and distance with a slowly moving light source it was expected to follow.

# Results

During the initial phase of the robot identifying the strongest light direction, a vital potential error in this is if the light sources around it change and it gets stuck in an infinite loop trying to find the original light strength direction it first encountered. I managed to limit this potential error my ensuring that, one the strongest light level has been detected. When the robot starts turning to find this light level again, there is a cap on the amount of turning it does before searching again.

I found that when moving towards a light source, the initial buffer can be a catch 22, if the buffer is too small or non-existent then the bot could potentially not find it again due to small light differences. Whereas, if the buffer is too big, then the robot veers slightly to the side of the light source. I was able to optimise this buffer threshold though to minimise the negatives, so another solution was not needed.

My biggest hurdles were time taken to test implementations/ changes and no fixed practice area with a stead and consistent light source. Due to varied light levels throughout the day and large testing times compared to simulation, progress was slowed based upon the difficulty and time taken to test each version.

The final implementation was able to pass both test. The first test with a static light source was passed to a more satisfactory standard whereas the searching and seeking time meant that a slow moving light source could be harder to find and relied on correct timing and light movement for the bot to seek it out correctly.

# Discussion

While developing my hand designed implementation, I had to deal with non-perfect sensor accuracies. Non perfect sensors forced me to implement solutions such as thresholds. A genetic algorithm approach would automatically take non-perfect sensor accuracies into account (Poikselka, Vallivaara, & Roning, 2015). This problem would scale with amount of sensors used and complexity of the environment.

Using a genetic algorithm approach to solving a problem using robotics allows for the initial phase of the solution to be generated using simulation (Poikselka, Vallivaara, & Roning, 2015). The biggest difference between my projects in hand coding light following robots and using an evolutionary approach, is the amount of time taken to test a solution. In simulation the tests can be automated and processed incredibly fast. Whereas, with real robots the tests are time consuming and hard to automate. This observation is in line with previous studies done using the same NXT platform (Poikselka, Vallivaara, & Roning, 2015). While converting simulation to reality comes with its own problems, this could still play a vital role in the early development of a genetic algorithm solution using a real robot.

While my intuition based trial and error approach to design was not optimal, it meant less iterations where needed than using a genetic algorithm approach (Lew, Horton, & Sherriff, 2010). This greatly reduced the cost of testing time. However, the problem environment I was designing the robot for was an incredibly simple environment. It is likely that evolutionary approaches scale better into more complex environments.

For me to change the behaviour of the robot required redesigning and creating a new implementation. This adds an extra layer of cost to the process for small changes. Whereas the behaviour of a genetic algorithm developed agent can be completely changed by re running the experiment with a slightly modified fitness function (Lew, Horton, & Sherriff, 2010).

It is worth mentioning that while this investigation was performed exclusively on the NXT Lego robot platform using RobotC, results may vary when using a different platform (Lew, Horton, & Sherriff, 2010). A different platform such as LeJos firmware with different features such as networking capabilities could vastly change the issues encountered or potentially improve others.

In summary, while hand coding and designing the light seeking robot worked well in simple environment, it becomes increasingly more complex as the problem does. This became apparent through the performance gap between the 2 different tests I employed. An evolutionary approach could scale better into these more complex problem spaces. If an evolutionary approach is able to automate each generation then the overall time cost to perform the project could be greatly reduced (Lew, Horton, & Sherriff, 2010) and provide a better solution to developing robotic solutions.

# Bibliography

Lew, N. W., Horton, T. B., & Sherriff, M. S. (2010). Using LEGO MINDSTORMS NXT and LEJOS in an Advanced Software Engineering Courses. *IEEE Conference on Software Engineering Education and Training.* Pittsburgh: IEEE.

Poikselka, K., Vallivaara, I., & Roning, J. (2015). Evolutionary Robotics on Lego NXT Platform. *International Conference on Tools with Artificial Intelligence (ICTAI).* Vietri sul Mare: IEEE.