FIXME - the name of your project

Final Report for CSM6960 Major Project

Author: Stefan Klaus (stk4@aber.ac.uk)
Supervisor: Dr. Myra Wilson (mxw@aber.ac.uk)

18th July 2015
Version: 0.1 (Draft)

This report was submitted as partial fulfilment of a MSc degree in
Intelligent Systems (G496)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature .....

Date .....

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature .....

Date .....

Acknowledgements

I am grateful to...

I’d like to thank...

Abstract

Include an abstract for your project. This should be no more than 300 words.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter 1

Background & Objectives

1.1 Aim of this project

The goal of this project is to create a program that controls a swarm of E-Puck robots in order to map an environment.
The E-Pucks always need to stay in communication, as they share a global map.
The control algorithm is an Artificial Neural Network, a evolutionary algorithm.
The neural network is trained so that it will learn to develop a solution on its own. The benefit of neural networks are that they are extremely versatile. A trained neural network will perform very good even in an different environment. Neural networks also allow for quick reactions of robots with very little processing time once they are fully evolved.

1.2 Information about the project

1.2.1 Choice of development environment and programming language

Since the control algorithm for the robots is an artificial neural network its needs to be trained before it can be tested. Therefore a simulator is needed. The simulator chosen for this project has been created by Elio Tuci(elt7@aber.ac.uk) and Muhanad Hayder Mohammed(mhm4@aber.ac.uk) at Aberystwyth University.
The other candidate was Cyberbotics Webots[1].
The simulator by Elio Tuci was chosen since I worked with it throughout the second semester of this master course and therefore know it well. Other reasons include that simulator is build for the creation of evolutionary algorithms and already posses an implemented genetic algorithm.
At the beginning of the project it was deemed ambitious to create a working genetic algorithm as well as artificial neural network and train it to perform the tasks explained in this chapter.
The programming language chosen for this project is C++ as the simulator is written in it.

[1] `http://goo.gl/BrPK98`

1.2.2 Choice of robot

The E-Puck[2] robot was chosen for this project as it is commercially available and versatile.
The *standard* robot comes with 8 IR proximity sensors placed at different intervals around the robot. It is powered by lithium-ion battery that is easily chargeable. 2 stepper motors allows it too move [10].

Since the project is done in an simulator no direct worries need to be done for battery life, though such could be simulated by calculating battery usage. This is however not done for the sake of this project.

The E-Puck is also useful as there is a wide range of extensions boards available to it, and its bus interface makes it possible and easy to design and add extension boards.
For this project the official Range and Bearing Board is used, more info about that can be found in section 1.3.1 [5].

1.2.3 Choice of environment design

An environment was designed to train the artificial neural network.
The environment represents a large room through which the robot swarm moves. There are multiple obstacles placed throughout the room to train and test the swarms communication and mapping abilities. A representation of the created environment can be found in Appendix B.1 on page 19.

1.3 Analysis

1.3.1 Communications

The Communication capabilities of the E-Puck where analysed. The Standard E-Puck comes with bluetooth communication and posses now WiFi capabilities.
Bluetooth communication for this project has been deemed infeasible as Bluetooth communications can take somewhere around 19.5 ± 4 seconds. A multi robot exploration and mapping project such as this requires almost constant communication, in which case bluetooth connection times of $\tilde{1}9$ seconds are too long.

There are a few proposed ways to implement WiFi communications on the e-puck robot. One of the methods was proposed by Christopher M. Cianci *et al.* [2] is the creation and implementation of a WiFi extension board for the e-puck, enabling communication between ZigBee and other IEEE 802.15.4 compliant transceivers.
There designed communication board is based on the MSP430 Microcontroller[3] and the Chipcon CC2420[4] radio.

[2] `http://www.e-puck.org/index.php`
[3] `http://www.ti.com/product/msp430f169`
[4] http://www.ti.com/product/cc2420

Allowing the e-puck a communication range between 15cm and 5 meters.
However such an board is not commercially available and would need to be custom designed and build, which is outside the spectrum of this project.

For the purposes of this project the use of the official e-puck range and bearing board has been deemed appropriate, as it is would be commercially available.
The range and bearing board is an extension board for the E-Puck which allows for localisation and local communication between E-Pucks using infra-red transmission. The board is powered by its own processor and consists of 12 sets of IR emission/reception modules. The board was first designed and build by Guiłrrez *et al.* [5].

1.3.2 Control Algorithm

The control algorithm for this project is an artificial neural network assisted by an genetic algorithm.
The background and use of those will be explain in further detail in this section.

1.3.2.1 Artificial Neural Network

The control algorithm used for this project is an artificial neural network(ANN). Artificial neural networks are inspired by the brain.
A biological brain functions by passing electrical signals through nodes, so called neurons. Neurons of the brain can be compared to simple Input/Output connectors which transmit pulse coded analogue information. The relation between a the inputs and outputs can be displayed a simple sigmoid function [7].
To a neuron not all inputs are the same however, different inputs(i.e. outputs from other neurons) have stronger influence on it than others, in neuroscience this is defined as synaptic weights.
This influence can be trained, in the course of a life a neuron *learns* to trust some inputs more than others, the same training is done in an artificial neural network. The synaptic weights are represented by the *weight* value each link between 2 nodes holds. This weight value is calculated and evolved using a genetic algorithm.

1.3.2.2 Genetic Algorithm

Genetic Algorithms(GA) are a evolutionary algorithm inspired by the genetics of living organism. An GA works by having a population of genes, which make out an chromosome. The genes in regard to this GA represent the weights used in the neural network.
A chromosome is constructed as followed, the first 1 gene holds a number representing the genotype length, in other words the number of genes in this chromosome and thereby the number of weights in the neural network. This number is calculated when the neural network is created at the start of the program and passed along to the GA.
The following genes represent each a weight for a specific link between 2 nodes in the neural network. The value is a number between 0 and 1.
The last gene in the chromosome holds the fitness assigned to this chromosome, which is calculated using the fitness function.

An genetic algorithms modifies the genes using operators which mimic their biological counterpart.

The genetic modifier described as *crossover* switches genes in chromosome to vary the genetic pool between generations.
This crossover can be done with either single genes, or groups of genes depending on the implementation, and of course any restrictions based on the nature of the data.

The other operator is mutation, which is the possibility that a random gene switches its value to another random value.
Which genes are chosen for crossover is based on the selection method implemented in the algorithm.
The third major part of an GA is the selection method, the method that is used to choose which chromosomes of the gene pool should carried over into the next generation.
The selection method implemented in this simulator is *Roulette Wheel Selection*. This selection method is part of the *Proportionate Reproduction* scheme this reproduction scheme chooses individuals to be carried over into the next generation based on their objective fitness function [4].

The basic part of the selection process is that the fittest individuals have the highest change to be carried over. This replicated nature in a way that a fitter individual tends to have a higher change of survival and will go forward to the mating pool of the next generation.
However weaker individuals(chromosomes) are not without a probabilistic change to get selected. It is called roulette wheel selection because its graphical representation is similar to a roulette wheel, as figure 1.1 shows [13].

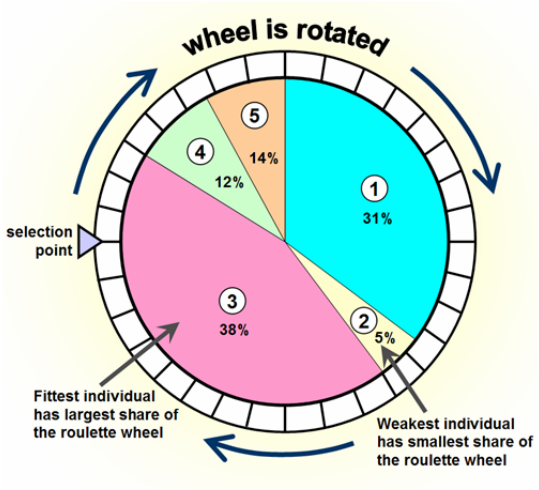

Figure 1.1: Roulette Wheel Selection[5]

As can be seen in figure 1.1 individuals with a higher fitness occupy a large of the overall available area.

[5]Image credit: Newcastle University Engineering and Design Center, accessed 7th of September 2015 `http://goo.gl/uwMSVB`

The number of chromosomes used in this work is 100, of which an elite of 8 is chosen to be taken over into the next generation. As for the genetic operators, crossover has a probability of 0.3 and mutation a probability of 0.05 to occur.

1.3.3 Fitness Function

A fitness function is used to guide the evolution of the neural network and the GA. It is used to rate the performance of a neural network based on a predefined formulae or criterion.
The baseline for the fitness function implemented in this project was proposed by Floreano *et al.* [3].
This fitness function is a good baseline as it prevents the robot from stopping, spinning on the spot, and crashing into obstacles, but still being close enough to an obstacle to get a positive return(obstacle found) from at least 1 sensor reading.
This fitness function was expanded upon as new features where implemented to lead to more complex behaviour of the robot.
The final fitness function is shown and explained in Chapter 2.1.3 on page 11.

1.3.4 Localisation

In real life scenarios GPS information is not always available, especially when mapping the insight of buildings.
For the sake of this project the localisation and rotation readings are taken from the simulator.

While not realistic a time shortage prevented further development of a localisation algorithm. Thought went into to question of localisation and there are a couple of approaches which could be taken in future work to incorporate them.
One of the approaches is to use odometry, in which the robots position and location is calculated using the knowledge of how many steps the stepper motors did between readings and the wheel diameter of the robots wheels. Knowing how many *steps* equal a full rotation, in case of the E-Pucks motors this is 20 steps[6], and the diameter of the robots wheels, around 41mm[7], allows to calculate what distance the robot has driven in a straight line.

The rotation of a robot can be calculated using the same data combined with the knowledge of the robots wheelbase.
However odometry is not a perfect localisation method. Uncertainty about for example the robots wheel diameter, or a wrongly calibrated stepper motor can throw of the location and rotation calculation completely. In real world applications, or simulators which simulate real world properties such as friction between the wheels and the floor, can cause additional problems.
This *error* in the calculation and movement get bigger overtime unless the localisation and rotation values stored in the robots memory are reset at certain intervals. In order to be able to reset it however exact knowledge of the location in the world is needed, not something possible in all environments.
Therefore odometry can be at best be seen as an estimate of the robots location and rotation.

[6] e-puck.org webside, accessed 8th of September, 2015,, `http://goo.gl/YpQ2nf`
[7] e-puck.org webside, accessed 8th of September, 2015, `http://goo.gl/YpQ2nf`

Another possible approach is to locate a robot using another robots sensor, such as the IR sensors on the range and bearing board.
However without prior knowledge of where the *searching* robot is in the world it is impossible to calculate the location of the *searched after* robot, only it distance and bearing from the *searching* robot.
This knowledge might be enough for some applications, however there are also limits to the localisation possibilities using this approach. IR sensors beams widen over distance, meaning the error of an bearings reading increases over distance. Therefore there is a a limit to over how large distances a robots location can be calculated using this.

Thoughts went also in to combining both the odometry calculations with the range and bearing information of the robot, however a shortage of time let to that no method was implemented in the system.

1.3.5 Mapping

To map the environment the E-Pucks IR sensors are used.
Using the knowledge of the robots position as well as the sensor read out it is possible to calculate the position of an obstacle in regard to the robot.

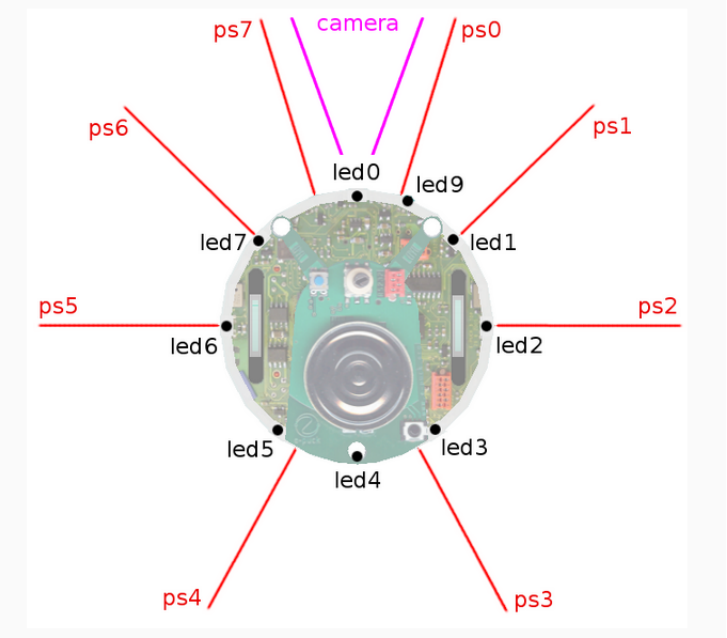

Figure 1.2: E-Puck sensor placement

Figure 1.2 shows the placement of the robots IR sensors, labelled *ps0* through *ps7*.
With the knowledge of where a particular sensor is placed on the robot combined with the knowledge of the robots position, rotation and the return of the IR reading, it is possible to calculate the placement of a obstacle in the environment.

The robots all share a single global map, which is updated with the position of all robots, as well as the position of any encountered obstacles, at each iteration.
The map it self a standard occupancy grid map: a 2 dimensional grid where each cell can have 1 of 3 possible values: 0 = unexplored, 1 = obstacle, 2 = robot position.

1.4 Process

The life cycle model used for this project is ”Feature Driven Development” (FDD) as it seems the more appropriate for this project than other models, such as extreme programming or test driven development.
The reason FDD is more appropriate is that this is a single person project, as well as the requirements allowed for easy distinguish in which order features need to be implemented.
For example: the controller(neural network) needs to be implemented to be able to train it. Once the it is implemented and the robots are able to move based on its control, the communication between the robots can be implemented. Only once this is done and tested the mapping algorithm can be started to be developed, as the features build on each other.

The milestones of the project rather small and incremental ”upgrades” on each other. For example the fitness function went from ”move in certain direction” to ”move in a certain direction and avoid obstacles” to much later ”move throughout the environment, don’t spin at the same spot, avoid crashing into obstacles but be close enough to map them and stay in communication range with other robots”.

Chapter 2

Design

2.1 Overall Architecture

2.1.1 Control Algorithm

In this section the design of the control algorithm will be explained.

2.1.2 Artificial Neural Network

The robot swarm is controlled by an artificial neural network.
The neural network is consistent of 8 inputs, 3 hidden nodes, and 4 outputs.
There are bias nodes connected to the hidden and output layer, both of which are always set to 1.
A representation of the network is shown in figure 2.2. The ANN is a multilayer feed forward network, meaning all layers nodes are connected to all nodes in the following layer and that data is only passed forward in the network, never back as would be the case using a different type of neural network, like a backpropagation algorithm.

The inputs are taken from the E-Pucks 8 IR proximity sensors. The hidden layer consists of 3 hidden nodes, which give more computational depth to the network.
From the 4 outputs of the neural the speed of the 2 stepper motors of the e-puck are calculated. This is done by calculating the difference between them.
The weights of the neural network are generated by a Genetic algorithm, which is part of the used simulator. The genes created by the GA are a value between 0 and 1, however the neural network algorithm scales them to be be a value between -5 and 5.
The reason for scaling is explained in section 2.1.2.2 of this chapter.

The bias is needed to be able to shift the entire sigmoid function along the x axis.

[1] Image credit Stackoverflow, accessed 8th of September, 2015 `http://goo.gl/Vktx0X`

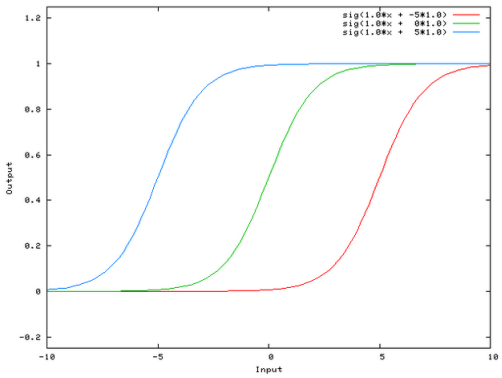

Figure 2.1: Representation of a shifting sigmoid function[1]

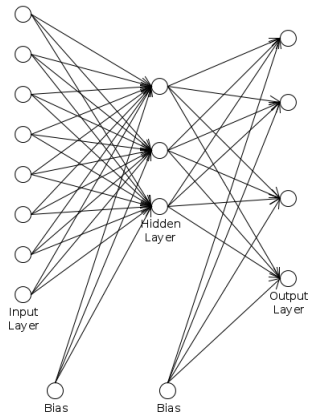

Figure 2.2: Representation of the Neural Network

2.1.2.1 Input Layer

The inputs to the ANN are returned from the E-Pucks 8 IR sensors and are a value between 0 and 4096.

As can be in the in figure 2.3 the returned IR sensors value rises drastically after the robot comes closer to an obstacle than 3 centimetres.

2.1.2.2 Hidden Layer & Sigmoid Function

In the hidden layer the sum of all inputs to a node is multiplied by the weights to that node and than fed to sigmoid function.

[2]Image credit: Webots User Guide `http://goo.gl/kyCINM`

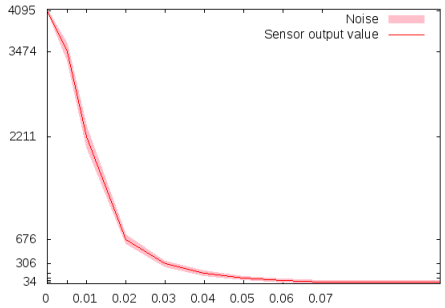

Figure 2.3: IR sensor response against distance[2]

A sigmoid function refers to a mathematical function that has an ”S”(sigmoid) shape.
A sigmoid, or activation function, is an abstract representation of a neuron firing(activating) in the brain. There are a number of different approaches to activation functions, the simplest is a simple binary step function, with only 2 stages: *on* or *off*.
The activation function in this neural network is a sigmoid function which is given by:

$$S(x) = \frac{1}{1 + e^{-x}} \tag{1}$$

Where e represents *Euler's number* which is 2.71828[...] and x represents the input to the function, in this case the sum of all inputs multiplied by the weights.
The output of the sigmoid function is a number between 0 and 1.

Sufficient scaling of inputs is important as it increases the *range* of the activation functions calculation.
Figure 2.4 shows a graphical representation of this sigmoid function. The red lines represent the range on which an activation can happen if the inputs are between 0 and 1. Figure 2.5 however shows the area of activation if the inputs are between -5 and 5. The graph shows that scaled inputs have a higher activation range which leads to better performing networks.

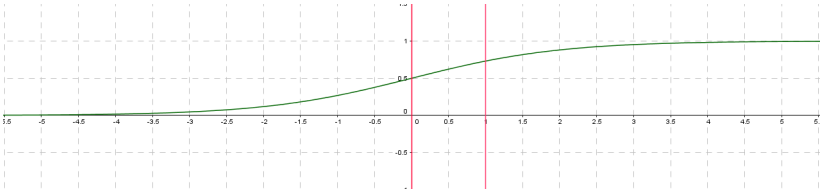

Figure 2.4: Activation range for unscaled inputs

2.1.2.3 Output Layer

In the outputlayer the outputs from the hiddenlayer are multiplied with their respective weights, and than passed to a sigmoid function again.
The output of the layer are than send back to the experiment class.

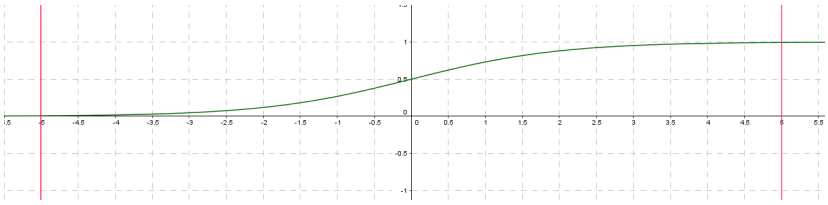

Figure 2.5: Activation range for scaled inputs

The velocity which will be set for each wheel is calculated by the difference between the 2 of the nodes. The *first* 2 nodes are used to calculate the velocity for the left wheel, the *last* 2 nodes for the right wheel.

2.1.3 Fitness Function

The fitness function is used to guide the evolution towards to the best solution.
The fitness function used in this project is as follows:

$$f = mean(V_l, V_r) \times (1 - \sqrt{|V_l - V_r|}) \times (1 - S_{ir}) \times coms \times pos_z \tag{2}$$

Where V_l and V_r represent the velocity of the left and right wheel, S_ir represents the highest IR sensor reading of the that iteration, $coms$ represent the robots distance to a another robot in the swarm. pos_z represents the z position of the robot.
This is a modified version of a fitness function proposed by Floreano *et al.* [3].
Their fitness function, as is shown in equation 3, covers the basic movement rules for a robot.

$$f = mean(V_l, V_r) \times (1 - \sqrt{|V_l - V_r|}) \times (1 - S_{ir}) \tag{3}$$

In this equation the values are normalized to fit:

$$0 \leq V \leq 1$$
$$0 \leq \Delta V \leq 1$$
$$0 \leq S_{ir} \leq 1$$

Where V is the velocity of a wheel, here 0 would mean full speed backwards and 1 full speed forwards. Stand still is represented by a value of 0.5.
ΔV represents the absolute difference between the left and right wheel velocities.
The IR reading S_{ir} is normalized to a number between 0 and 1, where 1 would mean the robot is very close to an object, 0 would mean the sensor has no return at all, i.e. no object is in range. However the simulator introduces noise into system, similar to how there would be noise in a real world application. Therefore this value will never be exactly 0.
The standard values of the velocity are all in a range from -1 to 1 and have been normalized to 0 to 1 in order to ensure that the fitness function works properly [3].
The rationality behind the different *segments* of the fitness function is a follows: $mean(V_l, V_r)$ is the average between the velocities, this ensures that the robot does not stand still as stopping would prevent the fitness value of increasing. This is also means that the fitness is increasing when

the robot is turning.
$1 - \sqrt{|V_l - V_r|}$ motivates the robots to move in a straight line rather than driving in large circles.
$1 - S_{ir}$ does reward the robot for not crashing into walls as the normalized function does return 1 when the robot is close to colliding with an obstacle, this would mean this sub-calculation would return 0.
$coms$ represents the the distance between it and another robot. The higher the value the better, this prevents the robots from driving close to each other, and gets them to drive at the full extent of their communication range. If the robots pass a certain distance this value is set to 0.

2.2 Some detailed design

2.2.1 Even more detail

2.3 User Interface

2.4 Other relevant sections

Chapter 3

Implementation

Chapter 4

Testing

4.1 Overall Approach to Testing

4.2 Automated Testing

4.2.1 Unit Tests

4.2.2 User Interface Testing

4.2.3 Stress Testing

4.2.4 Other types of testing

4.3 Integration Testing

4.4 User Testing

Chapter 5

Evaluation

Appendices

Appendix A

Third-Party Code and Libraries

Appendix B

Code samples

2.1 Environment Design

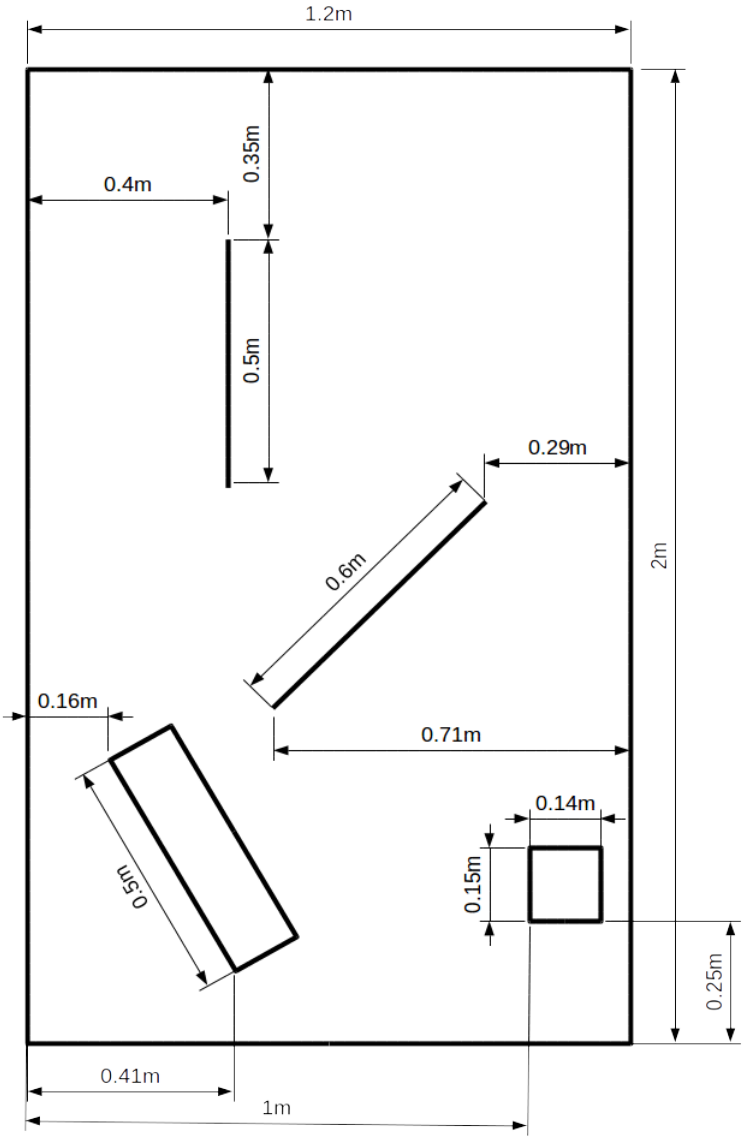

Figure B.1: Environment Design

Annotated Bibliography

[1] A. Birk and S. Carpin, "Merging Occupancy Grid Maps From Multiple Robots," *Proceedings of the IEEE*, vol. 94, no. 7, pp. 1384–1397, July 2006. [Online]. Available: http://dx.doi.org/10.1109/jproc.2006.876965

[2] C. Cianci, X. Raemy, J. Pugh, and A. Martinoli, "Communication in a Swarm of Miniature Robots: The e-Puck as an Educational Tool for Swarm Robotics," in *Swarm Robotics*, ser. Lecture Notes in Computer Science, E. Şahin, W. Spears, and A. Winfield, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, vol. 4433, ch. 7, pp. 103–115. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-71541-2_7

[3] D. Floreano and F. Mondada, "Evolution of homing navigation in a real mobile robot," *Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on*, vol. 26, no. 3, pp. 396–407, Jun 1996.

[4] D. E. Goldberg and K. Deb, "A comparative analysis of selection schemes used in genetic algorithms," *Foundations of genetic algorithms*, vol. 1, pp. 69–93, 1991.

[5] A. Gutiérrez, A. Campo, M. Dorigo, D. Amor, L. Magdalena, and F. Monasterio-Huelin, "An Open Localization and Local Communication Embodied Sensor," *Sensors*, vol. 8, no. 11, pp. 7545–7563, Nov. 2008. [Online]. Available: http://dx.doi.org/10.3390/s8117545

[6] A. Gutiérrez, A. Campo, M. Dorigo, D. Amor, L. Magdalena, and F. Monasterio-Huelin, "An open localization and local communication embodied sensor," *Sensors*, vol. 8, no. 11, pp. 7545–7563, 2008.

[7] J. Hopfield, "Artificial neural networks," *Circuits and Devices Magazine, IEEE*, vol. 4, no. 5, pp. 3–10, Sept 1988.

[8] R. Kala, A. Shukla, and R. Tiwari, "Robotic path planning using evolutionary momentum-based exploration," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 23, no. 4, pp. 469–495, July 2011. [Online]. Available: http://dx.doi.org/10.1080/0952813x.2010.490963

[9] Q. Meng and M. H. Lee, "Active Exploration in Building Hierarchical Neural Networks for Robotics," in *Instrumentation and Measurement Technology Conference, 2006. IMTC 2006. Proceedings of the IEEE*. IEEE, Apr. 2006, pp. 2095–2100. [Online]. Available: http://dx.doi.org/10.1109/imtc.2006.328464

[10] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.-C. Zufferey, D. Floreano, and A. Martinoli, "The e-puck, a robot designed for education in

engineering," in *Proceedings of the 9th conference on autonomous robot systems and competitions*, vol. 1, no. LIS-CONF-2009-004. IPCB: Instituto Politécnico de Castelo Branco, 2009, pp. 59–65.

[11] S. Thrun, "Learning Occupancy Grid Maps with Forward Sensor Models," *Autonomous Robots*, vol. 15, no. 2, pp. 111–127, Sept. 2003. [Online]. Available: http://dx.doi.org/10.1023/a:1025584807625

[12] K. M. Wurm, A. Hornung, M. Bennewitz, C. Stachniss, and W. Burgard, "OctoMap: A probabilistic, flexible, and compact 3D map representation for robotic systems," in *Proc. of the ICRA 2010 workshop on best practice in 3D perception and modeling for mobile manipulation*, vol. 2, 2010.

[13] J. Zhong, X. Hu, M. Gu, and J. Zhang, "Comparison of performance between different selection strategies on simple genetic algorithms," in *Computational Intelligence for Modelling, Control and Automation, 2005 and International Conference on Intelligent Agents, Web Technologies and Internet Commerce, International Conference on*, vol. 2, Nov 2005, pp. 1115–1121.