Bridging the Reality Gap in Evolutionary Robotics

2019

Candidate:	167521
Supervisor:	Dr. Christopher Buckley
Degree:	BSc Computer Science and Artificial Intelligence

University of Sussex – School of Engineering and Informatics

Statement of Originality

This report is submitted as part requirement for the degree of Computer Science with Artificial Intelligence at the University of Sussex. It is the product of my own labour except where indicated in the text. The report may be freely copied and distributed provided the source is acknowledged.

Signed:

Acknowledgements

I would like to thank everyone who has offered me support over the last three years. Namely my brother, sister, mum and dad. My technical supervisor, for sparking my interest in evolutionary robotics as well as offering me guidance with the direction of this project. To my peers for helping me maintain my sanity during the extended sessions in the labs. To my housemates for allowing me to temporarily convert our house in to a robot testing facility. Finally, to my fellow Sussex Snow committee members, for taking the burden of running the university's largest society off my shoulders during the critical final push to submission day.

Abstract

In this paper, by using genotypes that encode network connection weights and biases, we evolve a controller to perform an arbitrary behaviour within an autonomous agent. We then explore the transferability of this optimised controller from a minimal simulation over the 'reality gap' in to a real robot in the form of a LEGO Mindstorm. We investigate any conformities as well as discrepancies between the behaviours either side of the reality gap transfer, comparing the effectiveness of each transferred controller versus one evolved locally.

An additional focus point is also placed on methods that can be used to minimize the deterioration in an optimised models performance as it is transferred in to a real robot, such as the introduction of noise within the minimal simulation in order to more accurately reflect the non-idealistic conditions of the real world. An attempt is also made at achieving our stretch goal of the investigating multiple different behaviours, namely a 'return to home' behaviour in which the agent navigates its way back to its origin subsequent to a random movement pattern, in addition to an agent demonstrating behaviour based on a system of 2-eyed phototaxis. The point of interest here being investigating the hypothesis that visual perception constitutes the largest component of the reality gap.

Table of Contents

1. Introduction

1.1 What is meant by the term 'Reality Gap'

Every one of us has the ability to master complex sensorimotor skills to an aptitude far beyond even the most advanced robotic systems, which is understandable considering we have had decades to refine them. This timescale is of course not practical in the case of computer-based systems, but thanks to the power of simulation combined with modern parallel computing, it does not have to be.

There will always be discrepancies between a real and simulated environment. This difficulty is exacerbated when we consider the fact that advanced deep learning optimization methods excel at exploiting imperfections within simulations in order to thrive in ways that are not possible in the real world. 'Bridging the reality gap' is a phrase used to refer to the robust transfer of a model optimized within a simulation, in to a system that interacts with the same physical world as we do ourselves.

1.2 Motivation

When one thinks of the future, images of sci-fi worlds in which humans coexist commensalistically with their autonomous creations may spring to mind. It has long since served as a setting for an idealised civilisation with a vastly heightened living standard for humans as a result of the variety and complexity of tasks able to be completed by an autonomous agent, particularly those that involve environments that are adversely suited to human survival.

It's no secret that as a species, we are making technological advancements at an alarmingly exponential rate, with approximately 2.5 million years between the discovery of controlled fire [1] and the invention of the wheel [2], then only a few thousand before the telescope [3], and the first computer coming only a couple of hundred years after that [4]. With this difference in each order of magnitude, it should come as no surprise that certain aspects of the stereotypical futuristic utopia described above, are already, or are on the brink of, becoming a reality, and I believe evolutionary robotics to hold huge potential in bridging this gap.

This is where the most alluring aspect of this project lies, the limitless potential or practical applications that can be attributed to it, particularly when you consider that the behaviour we are choosing to evolve is arbitrary. With modern parallel computing, optimization can be carried out in a mere fraction of the time and cost of that would be possible in the real world, but the reality gap is the single biggest obstacle in evolutionary robotics. Quickly and efficiently evolving a complex behaviour within a simulation may serve as a great proof of concept, however its real-world applications are severely limited if the reality gap cannot be effectively bridged.

1.3 Project Aims and Objectives

In order to achieve my key objective of successfully bridging the reality gap, I must first begin within a minimal simulation. Here I will create a neural network controller, which I will evolve to exhibit a given behaviour. Although I will go in to more detail on this in section 4.2, the first of two behaviours I will consider is a 'return to home' behaviour, in which the left and right motor speeds are fed in to two arbitrary leaky-integrator nodes of a continuous time-recurrent neural network during an initial random movement pattern. The bearing along which the agent would have to travel in order to return to its origin would then be returned. The second behaviour I will consider is a two-eyed phototaxis, with the focus of my project being bridging the reality gap, I anticipate a reactive behaviour to a potentially dynamic environment such as this one to yield more interesting results, despite being the more primitive behaviour to evolve within a simulation.

I will discuss the specifics of what evolution strategy will be adopted in section 4.5, but regardless of this, I will use a genotype that encodes both the networks connection weights, as well as its biases. The fitness/cost function upon which these genotypes will be evaluated will of course vary depending on the behaviour in question.

Once an optimised neural network controller has been achieved, the same genotype will have its performance assessed in a real robot. The key aim here is to result in a controller that was efficiently evolved within a minimal simulation, yet is robust enough to exhibit the same behaviour after being transferred across the reality gap. Although I go in to more detail on this in section 3.1, the performance of an optimised controller is expected to deteriorate somewhat after being transferred upwards through the reality gap, however to be deemed as a success it must be able to perform to a greater standard than one evolved locally in a similar time-frame. Furthermore, we aim to investigate methods of minimizing this deterioration in performance such as the introduction of noise within the minimal simulation during the models evolution period, in an attempt to more accurately reflect the non-idealistic conditions of the real world.

2. Professional and Ethical Considerations

Since my project involves no human subjects or participants, it raises no legal, social, political or ethical issues for which I need to seek approval. I will however ensure that all work is carried out with due consideration for the standards set by BCS – The Chartered Institute for IT. Below I have outlined the points within the code of conduct that held the most relevance to my project, and the corresponding steps I have taken or will take in order to ensure my recognised responsibilities are met.

2.1 Public Interest

- 1b) ***"Have due regard for the legitimate rights of Third Parties"***
  I will take great care to avoid any breaches of this by ensuring proper referencing is used throughout all sections of the project, giving credit to any third parties where any material of any form is used that is not of my own creation.

2.2 Professional Competence and Integrity

- 2c) ***"Develop your professional knowledge, skills and competence on a continuing basis, maintaining awareness of technological developments, procedures, and standards that are relevant to your field."***
  My desire to further my knowledge is the driving force behind this project, I'm eager for any opportunity such as this to refine or acquire new skills. My awareness of any relevant developments is maintained by the background reading and research I'm undertaking around the subject, whether it be for work or as a hobbyist.

- 2b) ***"Ensure that you have the knowledge and understanding of Legislation and that you comply with such legislation, in carrying out your professional responsibilities"***
  I have familiarised myself with any applicable laws, statutes and regulations surrounding the project, including those concerned with copyright infringement, and at no point will I overstep any of them.

- 2f) ***"Avoid injuring others, their property, reputation, or employment by false or malicious or negligent action or inaction"***
  This becomes more relevant when dealing with the real robot aspect of the project, and will be fully adhered to by ensuring the test environment is safe, controlled and clear of any potential hazards, and all surroundings will be taken in to consideration at every stage of the testing.

2.3 Duty to Relevant Authority

- 3a) ***"Carry out your professional responsibilities with due care and diligence in accordance with the Relevant Authority's requirements whilst exercising your professional judgement at all times."***
  As my project supervisor and relevant authority, Dr. Christopher Buckley has given me the freedom to take this project in whichever direction I see fit, for which I will of course exercise due care, diligence and my professional judgment at every step. We remain in email correspondence and weekly scheduled meetings to ensure all of his

requirements are met, this is also the reason I produced the attached project brief for him before commencing work on the project.

3. Related Work

In the early 90's, the University of Sussex played a key role in showing the world what evolutionary robotics had to offer, with Cliff, Husbands and Harvey's 1993 publication [5]. Their work was centred around visually guided robots, and they were amongst the first to genetically evolve the control-network architecture of autonomous robots. After showing promising results, they laid the foundations for a myriad of further works, many of which at the University of Sussex, perhaps including this one.

Floreano and Mondada [6] showed it possible to evolve simple behaviours directly on a real robot, however with it taking 60 hours to train their Khepera robot to navigate a static environment, there is clearly room for improvement. Particularly when you consider their agent was performing nearly at an optimal level after only 50 of the 100 generations. The obvious solution given the context of this paper would be to carry out the evolutionary process within a simulation, however Brooks [7] illustrated a few key reasons which may cause problems for this simulative approach. Unlike a real robot, simulations are conducted in a stable coordinate system, with knowledge of the absolute position of all objects. Simulated sensors are able to return perfect information, void of any uncertainty. In addition, there are of course many physical laws that need be accounted for in the real world, such as friction, inertia, mass etc. In spite of this, Miglino, Nafasi and Taylor [8] built on the work by Floreano et al by using a simulation of a simplified environment to evolve the weights of a recurrent neural network controller which also demonstrated an environment exploration ability. However they were now able to carry out their evolution of 6 times the amount of generations, at 20 times the speed (600 generations in 3 hours). Although the observed fitness values did correlate, the paths taken by their genotypes did differ either side of the reality gap; this is something we will aim to remedy, behavioural consistency between the simulated and non-simulated environments. Thereby ratifying the findings of Nick Jakobi [9], that even complex motor behaviours can be evolved in minimal simulations and impeccably transferred in to reality.

Stefano Nolfi, Orazio Miglino and Domenico Parisi [10] investigated an alternative approach, a hybrid one, in which the evolutionary process is continued post reality gap transfer for a few generations. Although there was a significant drop in the fitness of individuals when transferred on to a real robot, these quickly reached a level akin to those achieved within the simulation after just a few additional generations evolved locally. This quick readjustment period demonstrates that the gap consists largely or parameter fine-tuning, as opposed to any fundamental flaws with the stimulatory evolution strategy.

Nick Jakobi's 1998 publication [11], having been supervised by Dr. Phil Husbands, builds on previous findings through the use of genetic algorithms to evolve neural network controllers, discussing different variants of each, but now shifts the focus to simulations (specifically minimal) and how these transfer over the reality gap. Although the aims of the involved agents are different, there is still a lot I can learn and build on from here. In section 2.2.1, Jakobi explores different control architectures, and which neural networks are suited to producing different behaviours, the behaviours compared include reactive, simple non-reactive as well as those that are dynamically complex non-reactive. In section 4.2 I will expand on the significance of this, any why his findings enforce my decision to use a continuous time-recurrent neural network (CTRNN) for the return to home behaviour, and a simply a linear function of sensory data for the 2-eyed phototaxis.

As explored in Brogan's 1991 publication [12], the feedback loops present in a recurrent neural network will play to our advantage in that they offer robustness to noise, arguably the most major hurdle when transferring a model between simulation and reality. Perhaps the foremost method of tackling this issue, as scrutinized by Jakobi et al [13], is to proportionally model empirically validated noise within the simulation as to more closely mirror those conditions found in the real world. This combined with a real robot designed with resistance to noise in mind can have very positive results, at least with certain robot-environment interaction dynamics. Mondada and Verschure [14] were able to achieve behaviours akin to one another in each environment without the modelling of noise, however performance in the real robot was substantially less robust. There are of course very real limitations to this technique of noise modelling depending on the quantity and nature of said noise, modelled noise is not identical to its real world counterpart, it aims to create a similar degree of inaccuracy even if how it does so is fundamentally different. These differences will result in a growing discrepancy between the environments as more noise is introduced. It may seem apparent why a simulation with insufficient noise can lead to a less successful transfer over the reality gap, but the same is true with too much noise, as the evolved behaviour then becomes dependant on it and its performance then suffers without it.

In the case of a return to home behaviour, I hypothesize noise to be somewhat less of a concern than it would otherwise pose, the key reason for this being that the agent won't be exhibiting reactive behaviour as a result of sensor-read environmental stimulus (as with a visually guided agent), and instead any input will come intrinsically. In terms of the two-eyed phototaxis however, this will rely purely on visual stimulus and I anticipate will suffer far more due to noise as a result once bridged over the reality gap. This would substantiate what is expressed in the Google AI blog [15] regarding visual perception to constitute the widest part of the reality gap.

4. Requirements Analysis

4.1 Primary Objective

The key aim of my project is to successfully bridge the reality gap by transferring a controller that exhibits an arbitrary behaviour, and has been optimised to do so using evolutionary strategies within a simulation, to a real robot in the form of a Lego Mindstorm. However, in order to evaluate this we need to lay the goalposts for what classifies as a successful transfer. It would be optimistic, or even arguably impossible, to create a system that seamlessly transfers between systems to achieve flawless results in both a simulation and reality. This is partially due to the non-existence of a perfect simulation, there will always be discrepancies between an environment that is simulated and one that is real. As explained in section 1.1, this becomes increasingly problematic with more advanced optimization methods due to their tendency to capitalize on any imperfections a simulation may have to succeed in ways that, in actuality, are not possible.

It must be remembered that we are evolving controllers for adaptive systems, and our only way of measuring their success is not by looking at them directly, but at the behaviour produced by them, and this extra degree of separation allows for further discrepancy between a real and simulated environment. For these reasons I will not be looking for a perfect match between the two environments, and will instead classify it as a success so long as the real robot is able to perform its objective to just a satisfactory level. For the purposes of documentation, I can quantify the degree success in a fashion much like that employed by Miglino, Lund, and Nolfi [16]; the fitnesses of the same controller are evaluated and compared between the two environments, with the success of the environmental transition being inversely proportional to the discrepancy between the two fitnesses. I will also appraise the success of a reality gap transfer by comparing the models success to one evolved locally within a similar length of time. Should any transferred model demonstrate the lesser performance, the transfer will of course be deemed a failure.

4.2 Chosen Behaviour

4.2.1 Return to Home

The first behaviour I considered for the purpose of this project is one in which an autonomous agent performs a random movement pattern, taking the left and right motor speeds as inputs in to two nodes of a neural network controller, before outputting the bearing it would have to take in order to return to its place of origin. It is a behaviour akin to path integration, present in nature, most notably perhaps in the ant species *Cataglyphis bicolor*. Insects are significantly less complex than vertebrates in terms of their neurological structure, yet they are still able to exhibit a high degree of behavioural complexity, making them an ideal case study for the field of artificial life when trying to achieve emergent intelligent behaviour.

Not only could this be said to be a complex behaviour, but it is also one that I perceived would bridge the reality gap well, due to the lack of sensory input to the agent. This would allow me to draw contrasts with the behaviour outlined in section 4.2.2, and thereby analyse the challenge of the reality gap in greater depth by breaking it down in to its components of sorts by eliminating the portion of which is pertained to visual stimulus.

4.2.2 Two-Eyed Phototaxis

A phototaxis is the movement of an organism in response to light, it's a phenomenon that's ever present in the natural world, most notably with simple lifeforms such as plants and plankton for example. In his 1984 publication [17], Valentino Braitenberg describes various thought experiments involving very simple autonomous vehicles manifesting seemingly complex behaviour as a first step towards attempting to understand the composition of intelligent life. It is an approach that is of the mentality that it is actually far more straight-forward to replicate a given behaviour from scratch, than it would seem based on purely observing the output. It's a view that concurs with the attitude proposed by Daniel Dennett in his 1983 publication [18], in which he suggests artificial intelligence should instead take the approach of simulating whole simple systems, as opposed to mimicking isolated aspects of higher cognition such as chess playing.

I felt it to be the more suitable behaviour for the purposes of this project, concerned with bridging the reality gap, since the largest part of the reality gap is commonly considered to be that concerned with visual stimulus [15]. Comparatively, the reality-gap transfer of a return to home behaviour that relies on no external stimulus would prove trivial, and the additional complexity of accomplishing this behaviour to an acceptable standard was subtracting from the focus of the project.

4.3 Control Mechanisms

4.3.1 Return to Home Behaviour

With numerous different neural network architectures available, it is imperative that we pick a suitable one in order to exhibit our chosen behaviours. The continuous time-recurrent neural netwok (CTRNN) model was one first proposed in Beer's 1990 publication [19], but it wasn't until 1995 [20] that he also demonstrated their potential suitability for one of our behaviours in particular due to their ability to approximate trajectories of dynamical systems. With regards to achieving this return to home behaviour, the motor output at any given time is not wholly determined by current input, nor does it involve dynamically complex outputs for motor pattern generation. It is a non-reactive behaviour and Jakobi's [11] review of which neural networks are most suited to this reinforce my decision to use a continuous-time recurrent neural network.

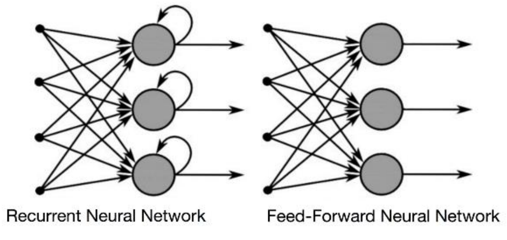

Fig 1.1 and 1.2 – Structure comparison of NN's [27]

Neural networks that allow recurrence, or feedback loops within its connections, allow internal states of sorts to be represented, with these states being partially dependant on prior outputs. This can function in essence as memory and so it is easy to see why this is ideal for my task, where the output (return home bearing) is a function of past inputs (motor speeds) rather than present ones. This is why the neurons within a CTRNN are known as leaky integrators, as this is the name given to a mathematical differential equation that slowly and continuously has its input leaked over time. The state of each node i is modelled by the differential equation below:

$$\tau_i \frac{dy_i}{dt} = -y_i + \sum_{j=1}^{n} w_{ji}\sigma(y_j - \theta_j) + I_i(t)$$

Where $\boldsymbol{\tau_i}$ = Time constant, $\boldsymbol{y_i}$ = activation of node i, $\boldsymbol{w_{ji}}$ = the weight of connection from nodes j (presynaptic) to i (postsynaptic), $\boldsymbol{\theta_j}$= the bias of the presynaptic node, $\boldsymbol{I_i}$ = input to node i.

This function defines the rate at which each neuron leaks its activation level, a visualization of this can be seen in figure 2. Output levels climb over time in accordance with incoming activations before leaking potential at a constant rate defined by its differential equation.

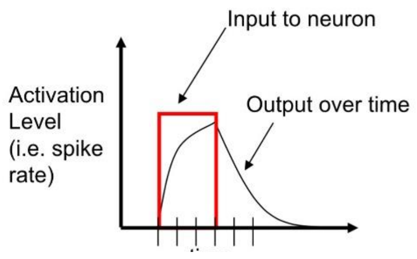

Fig 2 – Leaking activation level of CTRNN node [28]

4.3.1 Two-Eyed Phototaxis

With regards to the 2-eyed phototaxis behaviour described in section 4.2.2, it's an example of a non-temporal simple reactive behaviour to an immediate stimulus that requires no form of memory, simply mapping input to output. We could use a feed forward neural network, the most primitive type in which data can only flow in one direction due to the absence of any feedback loops or cycles, however we can simplify this even further. By basing the motor speeds on a linear function of the sensory data, we can further reduce the likelihood of the controller being a source of error, allowing us to more effectively isolate the reality gap transfer as per our project aims.

4.4 Optimization Methodology – Evolutionary Algorithms

Evolutionary algorithms are an example of the world of AI taking inspiration from the natural world, the theory being that if something as complex as a human being can be evolved biologically over many generations, this can be mimicked artificially as a form of optimization. They take a heuristic approach to solving non-linear optimization problems using agencies such as reproduction, selection, and genetic mutation, which are found to be ubiquitous in nature.

They do have limitations in terms of what information they can store, their lack of inferences regarding the underlying fitness landscape frees them from many potential false assumptions and biases making them well sorted to approximating solutions to a great range of problems. This limited scope of information can be of detriment however, when we consider that only the solutions with the greatest evaluated fitness are kept with the majority being disposed of, yet these could hold valuable information regarding what *not* to do which could be used to shape a fitter next generation. The random element upon which evolution works can also count against it in terms of efficiency, for example, if an agent needs to be evolved to have larger wheels to increase its speed, evolution strategies will attempt all sorts of modifications to any aspect of the agent until it by chance stumbles across the correct one. However alternatives such as gradient descent are not practical here due to the aforementioned lack of inferences about the fitness landscape, this is what we refer to as a 'black box' scenario, in which the gradients are either not available, or simply of no use.

4.4.1 Genetic Algorithms

The most common form of evolutionary algorithm is a genetic algorithm (GA). They begin with the initialization of a random population of any given size; these genotypes are spread throughout the problem space, reducing the chance of a solution getting stuck in a suboptimal local minima. In our case the genotypes will be encoding the NN's connection weights and biases. The terminating condition is customarily only met once the algorithm has either been run for the specified number of generations, there has been no improvement in the last x generations, or when the objective function reaches a pre-defined value. But it's easy to see why the latter would only be suited to certain problems due to the imposed limitation associated with it. The parent selection takes a collection of the fitter genotypes (the specifics of how this is done depends on the genetic algorithm being used). The remainder of the population is culled; these selected few are then duplicated before being mutated in accordance with a crossover/mutation function to refill the population, and the process is repeated, with each loop iteration as 1 generation.

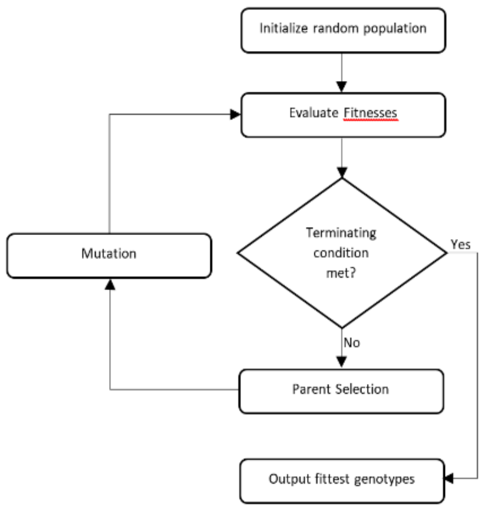

Fig 3 – High level GA description

Due to time constraints I will only be optimizing network starting weights of one fixed network architecture, rather than also varying the amount of layers and nodes. This is because the key focus of my project is not to what degree of accuracy I can have an agent flawlessly and efficiently perform our chosen behaviour. Instead, the emphasis is on consistency between the simulation and real life, successfully bridging the reality gap. As put by Harvey et al, we are using GA's as optimizers rather than adaptive improvers. This disparity is here defined as such

"can a fitness function to be maximised (or a set of them) be fully and unchangeably defined before one starts from scratch ? If the answer is yes, then optimisation techniques are called for. In the case of natural evolution, of course, the answer is no" [21]

This is perhaps where one limitation lies, with the impact my project can have on the field of evolutionary robotics. However this is not to be an issue since the focal point of my project is concerned with bridging the reality gap as opposed to pushing the boundaries with adaptive improvers.

4.4.2 Covariance Matrix Adaptation Evolution Strategy (CMA-ES)

Although originally under a slightly different name, CMAES was first proposed in 2001 by N. Hansen and A Ostermeier [22], I settled for this as my choice of evolutionary algorithm as it is comparatively far more sophisticated when compared to any of the genetic algorithms I have programmed myself. Considering the focus of my project being not on the optimisation, but instead on the reality gap transfer of a readily optimized model, using a script such as the open source one published by Yarpiz [23] would aid me in better achieving this.

As with all evolution strategies, CMAES works on the very high level basis of selecting the parents of the next generation, based on their evaluated fitness according to an objective function. As you would expect, it is well suited to black box scenarios such as this one, as it does not require the existence of, or any assumptions/approximations of the objective function gradients. If the objective function were linear, quadratic or convex then other differential methods such as quasi-newton or gradient descent would outshine CMA-ES in terms of performance. It uses a population-based stochastic approach which aides negate the ruggedness of the function, in combination with non-local sampling to vitiate the 'curse of dimensionality' and reduce the chances of getting stuck in a sub optimal local minima.

The Rastrigin function, pictured right, is defined by as follows:

$$f(x) = An + \sum_{i=1}^{n} [x_i^2 - Acos(2\pi x_i)]$$

Where n equates to the dimensionality of the domain. It is a function that was first put forward by L.A Rastrigin [24] to be used to assess the effectiveness of optimization algorithms. It perfectly illustrates the characteristics I described above for which CMA-ES excels, a rugged function that is neither linear, nor convex-continuous, with a large number of local minima.

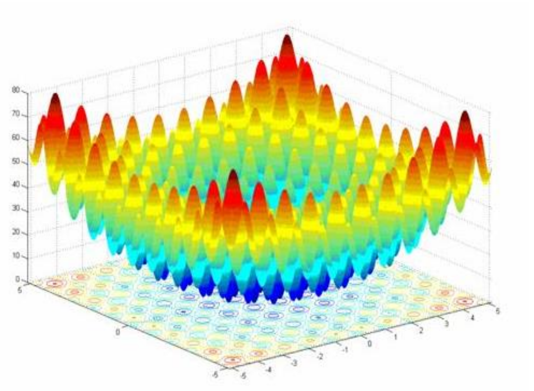

Fig 4 – 3D depiction of Ragistrin function [26]

Normal distributions are pervasive within the natural world, they are the only distribution with a finite variance that is stable, yet achieves maximum entropy amongst its isotropic search points [25]. This is why an n-dimensional (multivariate) normal distribution is used for sampling within CMA-ES. This distribution is uniquely delineated by its mean as well as the covariance matrix. The mean represents the current favourite solution as it lies on the modal value, with the highest density, whilst the covariance matrix dictates the shape of this distributions ellipsoid, and through its adaptation, it is able to reflect the dependencies between the variables.

4.5 Simulation

The first works of my project will be carried out from within a minimal simulation, and the environment for which I have opted for this section of the project is MATLAB. I feel it is very well suited to the task since it is a language I am comfortable with, and in the past have programmed multiple genetic algorithms of my own in (including steady state, tournament selections, truncation, and full microbial). It therefore separates itself from python, the other strong candidate language, as it carries the advantage of giving me the freedom to experiment with, or implement, these GA's as part of the project should I see fit.

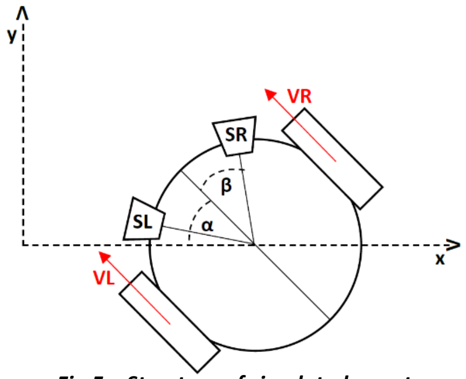

Fig 5 – Structure of simulated agent

In figure 5 above, you can see how the agent is modelled within the minimal simulation. The light intensity at each sensor will be given by $localIntensity = \frac{sourceIntensity}{distToSource^2}$ This reflects the exponential drop-off in detected ambient light intensity with distance from the source. The forwards velocity Vf is given by averaging the velocities of each wheel $Vf = \frac{VL+VR}{2}$. The rotational velocity is given as a function of the difference between the two wheel velocities, $Vr = \frac{VL-VR}{2*agentRadius}$. With these, in addition to the time-step size Dt and the angles illustrated above, we can model the positional updates of the agent and its sensors with each time-step as follows:

Agent

$$xNew = x + Dt * Vf\cos(\alpha)$$
$$yNew = y + Dt * Vf\cos(\alpha)$$
$$\alpha New = \alpha + Dt * Vr$$

Sensor

$$xNew = x + radius * \cos(\alpha + \beta)$$
$$yNew = y + radius * \sin(\alpha + \beta)$$

4.6 Hardware

I had made the decision early on during the stages of my project to base the real-robots portion on a LEGO Mindstorm EV3, a few factors contributed as to why I felt this was the most suitable option. First and foremost is their availability to me versus any alternative, combined with the fact that they offer all the functionality I need. I anticipate that any difficulties I encounter whilst bridging the reality gap will not be as a result of hardware limitations associated with the unit, but instead be resultant of more fundamental issues associated with the task. The ease of assembly is another perk that cannot be overlooked, versus an Arduino-based alternative, allowing me to direct more time towards the focus of the project as opposed to constructing hardware of my own.

As a result of the studies I carried out during the Acquired Intelligence & Adaptive Behaviour module, I am suitably equipped with relevant experience with the functionality of LEGO Mindstorms that I am keen to build on. During these prior studies I experimented with various different hardware configurations and found best success with arrangements as those pictured in figure 6.

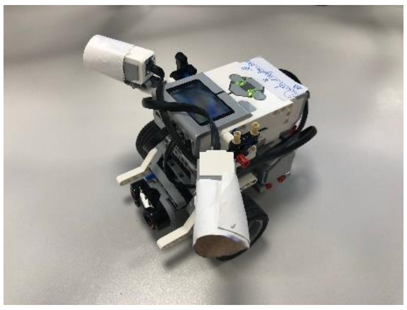
Fig 6.1 – LEGO Mindstorm hardware arrangement (corner view)

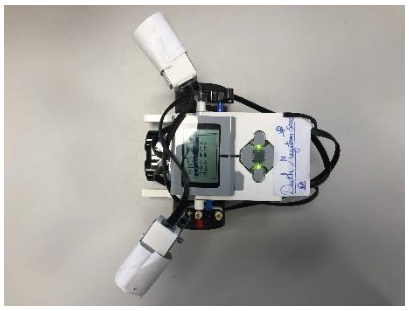
Fig 6.2 – LEGO Mindstorm hardware arrangement (top view)

It is worth noting the tape around the light sensors, this is an example of me designing the hardware with robustness to noise in mind. As I have mentioned in sections 3 and 4.2.2, I foresee the visually guided stimulus, which constitutes the largest part of the reality gap, to be the area with which the transfer struggles the most. Therefore, any possible precaution I can take to aid the mitigation of noise in the form of ambient light would be an advisable step to take.

5. Results

5.1 Minimal Simulation

5.1.1 Return to Home Behaviour

During the implementation of the 'return to home' behaviour, I encountered many difficulties that I deem worthy of discussion here. The first issue I encountered when evolving the weights of a neural network controller to perform this behaviour was due to the degree of randomness involved. The evolution strategy would discover a set of weights, that purely by chance, for its given initial random movement pattern, would perform incredibly well. This became clear when the same genotype was tested again with a different random movement pattern. This issue was easily solved by modifying the fitness function to test each genotype over several different random movement patterns, taking its fitness as an average of the discrepancies between the actual and predicted home bearings, being sure to account for wrap-around error calculation. For example the evaluated error for a returned bearing of 10 degrees, in a situation where the actual home bearing lies at 350 degrees, must result in an error of 20 as opposed to 340. (please note the use of degrees here is purely for ease of human visualisation, where in practice the units are in fact radians)

This however led to the next issue, of a sort that I had foreseen and discussed briefly in section 1.1. As I increased the iterations over which each genotype was averaged, to ensure none of them would falsely succeed out of chance, it now meant the controllers evolved to predict as close to the median of the possible output range in order to minimize the average error. This was to be corrected by raising the cost function to a power, as this exploit discovered by the evolution strategy is one that relied on a linear cost function. By raising it to a power any poorer performing genotypes are penalized exponentially more so, supposedly invalidating that tactic.

After making the changes above, I ran in to the next issue. As the optimization ran for more and more iterations, the predicted outputs would tend from the centre to each of the extremes, beginning as it had done previously by estimating π (directly between the 0 – 2π output range). However by the end, for any given genotype the predicted home bearing would be returned as either 0 or 2π with no in-between. I anticipate this to be because using a tanh activation function results in an output of -1 to 1, I had then simply scaled this to 0 to 2π, the issue being that this is inclusive of both boundaries. With the wrap around error calculation, this became the equivalent tactic of its prior one of reducing the average error by predicting the median, however slightly more effective due to the cost function being raised by a power, meaning predicting on each boundary would be penalized marginally less so than right in the middle would.

As a result of these difficulties combined with numerous others, the focus of my project was being shifted from bridging the reality gap to simply modelling this behaviour within a minimal simulation. Due to time constraints as well as apparent limitations in my expertise I was left with no option but to redirect my efforts towards the pursuit of the focal point of my project solely with the 2-eyed phototaxis behaviour. I do not see this to be a detrimental decision to the project as I anticipate the transfer of this behaviour to be by far the more trivial of the two due to its lack of reactive behaviour to external inputs, however it remains something I do wish to investigate further in the future.

5.1.2 Two-Eyed Phototaxis

I designed two separate objective functions for this behaviour, the first which I named pTaxisDist (As it was concerned purely with distance), ran for a fixed amount of time and aimed to minimize the agents distance from the light source by the end of this allotted time.

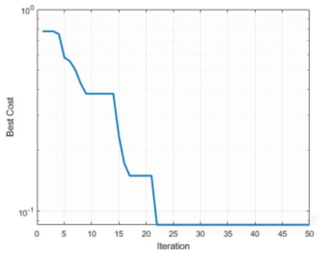

Fig 7.1 – CMA-ES optimization

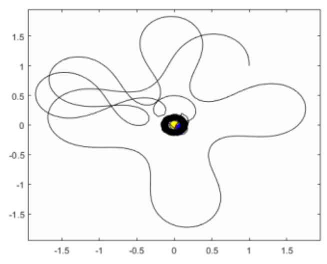

Fig 7.2 – Path of evolved agent

In figure 7.1 you can see how the CMA-ES script minimizes the pTaxisDist objective function for a population with each generation to produce a fully optimised genotype. Figure 7.2 shows the behaviour of this optimised genotype within the minimal simulation, and when observing the paths taken by this genotype, as well as many other optimised ones, it is apparent that should the objective be to simply navigate towards the light, this is not an optimal performance when we factor in time. Although it got to the light source and remained there until the specified run time of 40 seconds was exceeded, it took a very inefficient route to do so, which is understandable considering that objective function takes no account of time until that which it is allotted, ends.

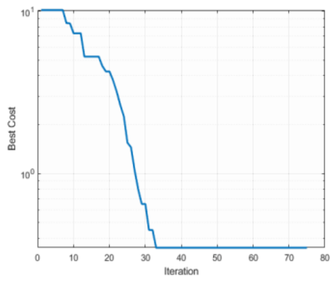

Fig 8.1 – CMA-ES optimization (pTaxisTime)

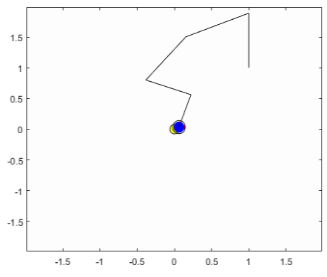

Fig 8.2 – Path of evolved

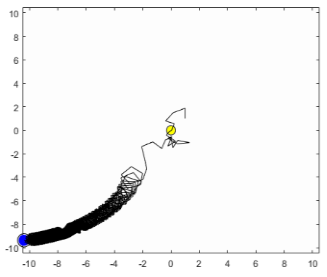

Fig 8.3 – Continued path of evolved agent

This is what led me to design a second objective function, named pTaxisTime (as it is concerned purely about time), which instead rewards genotypes purely on how fast they are able to reach the light source. Figure 8.1 shows the minimization of this objective function by CMA-ES, and in figure 8.2 you can see the performance of the winning genotype as seen by the optimization script, reaching its goal of the light source in only 0.35 seconds. At first glance one may assume this to be a grand success, however it is far too erratic to be able to call it that by any means. To prove this I allowed the simulation to continue running for a full 40 seconds (as was done by the agent evolved using pTaxisDist in figure 7.2) rather than terminating when as it hits the light source by chance. This is what you see in figure 8.3, in which the agent ends up 13.99 metres away from its supposed destination, despite passing through it only 0.35 seconds in. If I allowed the CMA-ES script to run for an extra 25 iterations, I was able to achieve genotypes that were erratic to an even greater extreme, reaching the light source after just 0.2 seconds, yet after 40 it would be a total of 133m away. Any genotypes evolved based on this objective function would be far too volatile to transfer over the reality gap to exhibit a similar behaviour with any kind of success. The failure of this second objective function is not dissimilar to the sort I first mentioned in section 1.1, in which advanced optimization techniques are proficient at exploiting loopholes of sorts to cheat simulations to succeed in ways that wouldn't be viable in the real world.

By simply reducing the time limit that pTaxisDist was permitted to run for during its evolutionary phase, I was able to achieve a middle ground of sorts between the two objective functions. Please see figure 8 for a visualisation of the path taken by an agent evolved using pTaxisDist, after reducing the time limit from 40 to 20. After seeing positive effects from reducing the time limit, one may be tempted to push this further, however we have already demonstrated with the pTaxisTime cost function that there is a clear trade off between how fast and agent will reach its goal, and how volatile its performance becomes in doing so. I am sure you will agree that figure 9 exhibits a good balance of robustness and efficiency, resultantly it will be the controllers that were evolved based on pTaxisDist with this reduced time limitation that I shall attempt to bridge the reality gap with and transfer on to a LEGO Mindstorm. Please see section 5.1.3 for the pseudocode of each objective function.

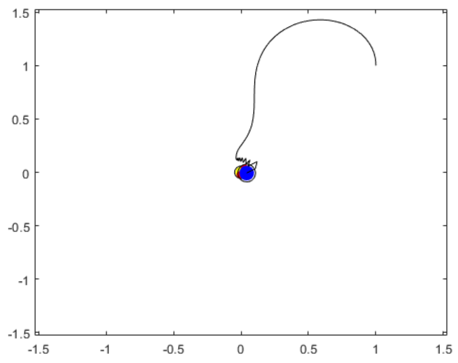

Fig 9 – Path of evolved agent with reduced T (pTaxisDist)

5.1.3 Pseudocode

<u>pTaxisDist(geno)</u>

input: geno – An array of floats $(w_1, w_2, w_3, w_4, b_1, b_2)$ encoding the weights and biases of the genotype.

output: distFromLight – A float representing the agents final distance from the light source.

```
While time < timeLimit
    LMotorSpeed → leftSensor * $w_1$ + rightSensor * $w_3$ + $b_1$
    RMotorSpeed → leftSensor * $w_2$ + rightSensor * $w_4$ + $b_2$
    time → time + 1
End While
distFromLight → $\sqrt{(\text{finalX})^2 + (\text{finalY})^2}$
Return distFromLight
```

<u>pTaxisTime(geno)</u>

input: geno – An array of floats $(w_1, w_2, w_3, w_4, b_1, b_2)$ encoding the weights and biases of the genotype.

output: time – A float representing the time taken for the agent to reach the light source

```
reachedLight → False
While time < timeLimit and reachedLight = False
    LMotorSpeed → leftSensor * $w_1$ + rightSensor * $w_3$ + $b_1$
    RMotorSpeed → leftSensor * $w_2$ + rightSensor * $w_4$ + $b_2$
    distFromLight → $\sqrt{(\text{finalX})^2 + (\text{finalY})^2}$
    If distFromLight < 0.1
        reachedLight → true
    End if
End While
Return time
```

5.2 Real Robot

After directly translating the MATLAB code from my minimal simulation in to ROBOTC, the C programming language upon which the LEGO Mindstorm runs, the first issue that became apparent prior to any modification was that the robot would hardly move. This is hardly surprising once you acknowledge that the Mindstorm motors have an input range of -127 to 127, whereas those being used within the minimal simulation approximated the range -1 to 1. Subsequent to appropriately scaling these values, the next obstacle was the network biases bearing little to no influence whatsoever on the behaviour of the agent, however this was simply another question of scaling values, this time those of the light sensor readings. They were again far too high versus those used within the simulation and as a result, diluted them to a point of redundancy.

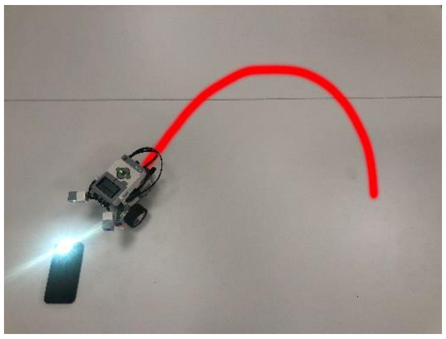
Fig 10.1 – Path of LEGO Mindstorm

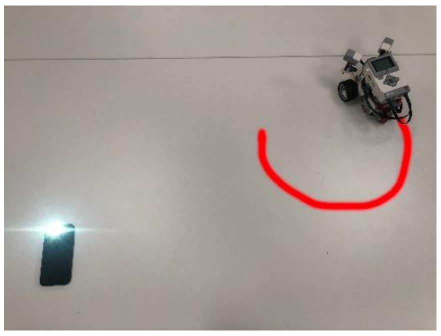
Fig 10.2 – Path w/ altered initial pos

Once these described adaptations had been made, I aimed to replicate the behaviour exhibited in Figure 8 using the same genotype. After my first test (figure 10.1) I was incredibly pleased with the outcome, although by no means a mirror image of figure 9, it was certainly exhibiting the same behaviour. However once I repeated the experiment with different initial conditions, the agent began moving as illustrated in figure 10.2, spiralling away from the light source. It is worth noting that the environment in figure 10 has been illuminated for the purposes of photography, and the light source laid down flat as to be visible to the camera.

I returned to the minimal simulation to investigate why this might have been the case, after changing the initial bearing to more closely mirror that which I used in fig 10.2, the results can be seen on the right in figure 11. With this the issue immediately became clear, the genotype being used had been evolved based on a singular initial position and orientation, it essentially lacked the ability to turn right but had previously had no need to.

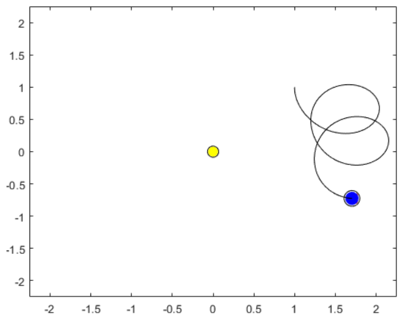

Fig 11 – Path w/ altered intial bearing

In order to remedy this, I returned to the minimal simulation and created a third objective function, which I named pTaxisDistAvg. It is essentially a duplicate of pTaxisDist, with the key difference being that it is nested within a loop that is executed 4 times, incrementing the initial bearing by 90 degree intervals each time. The returned cost is then an average of each agents final distance from the light source. It must be noted that I increased the runtime back up to 40 seconds, as without this the increased complexity resulted in a volatile solution. In figure 12 you can see the resulting performance of a CMA-ES optimised genotype using this cost function, note that the path taken now looks almost identical regardless of the initial bearing, with the only discrepancy being how far round one of those initial loops the agent begins. This does still seem like a very irregular movement pattern to have evolved, however with the focus of my project being aimed at the consistency of behaviour between simulation and real robot, as opposed to simply evolving a flawless performance, it did not concern me as an issue. I in fact viewed it as the contrary, a robust transfer of a more irregular behaviour should be seen as a more successful traverse of the reality gap than a more primitive one, such as heading directly for the light, due to the added complexity of the behaviour.

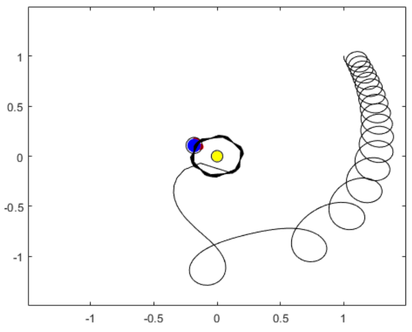

Fig 12 – Agent evolved with pTaxisDistAvg

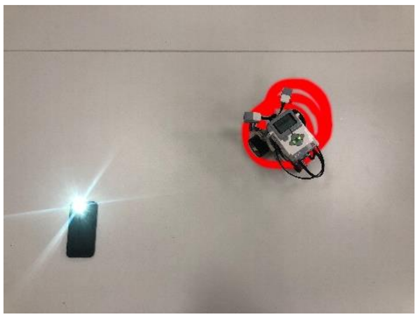
Fig 13.1 – Genotype of fig 10 in real robot

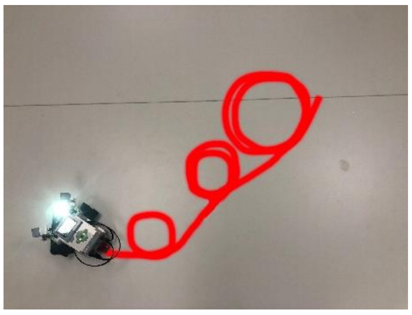
Fig 13.2 – w/ Light sensor influence increased

When transferring the evolved controller responsible for the path in figure 12 across the reality gap in to the LEGO Mindstorm, it simply circled on the spot for the most part, making occasional and seemingly irregular adjustments to the diameter and centre location of the circle (see fig 13.1). For the purpose of experimentation, I moved the light source, and it did indeed influence these path adjustments, until a point at which the light was close enough, the agent broke free from its circular motion path and headed directly at the light source. This demonstrated the discrepancy between the influence of the light sensors between the simulation and of those in the real robot, and so naturally the next step was to reduce the amount by which the sensor readings in the real robot had been scaled down. The results of this can be seen in figure 13.2, and again, it's clearly no mirror image of figure 11 however they do share enough in common for us to conclude that this behaviour has been bridged across the reality gap. Again, I must stress that for the benefit of the camera, the images in figure 13 have been illuminated far beyond the conditions that were present during the time of testing, in order to reduce noise giving the agent the best possible chance of success. But this is where our next challenge lies, now that we have achieved the core focus of our project, we attempt to take it further by transferring a system that offers robustness to noise.

As shown in figures 10 and 13, note that the best success was achieved without the slight hardware modifications made in section 4.6, in a noise-free environment they offered no benefit, and instead limited the agent's peripheral vision, resulting in more snappy responses to the stimulus and a less smooth path.

In section 4.1 as part of our primary objective, we also specified that any transferred controller must outshine one evolved locally within a similar timeframe. Since within the simulation, each of the 90 (population size) genotypes was tested over a period of 40 seconds in each of the 4 directions, and evolved over 100 generations, that would equate to 16.7 days of real-world time (40 seconds x 90 x 4 x 100 = 1.44 x 10^6), discounting any inevitable delays such as reset times. This was all able to be simulated in just 35.113 seconds, that's over 41 thousand times faster, taking just 0.00244% (to 3 s.f) of the time, these figures speak for themselves in terms of reinforcing the motivation for this project as to why it is certainly one worth pursuing.

5.3 A Robust System

When testing our transferred controller in a well-lit room, the performance deteriorated noticeably. Due to the presence of light in all directions, it no longer circled towards the target source as it had done previously. Instead, it took a convoluted route, a wide arc tending towards the source whilst constantly swivelling left and right. Once the agent neared the source of light, it did perform the occasional loop, reminiscent of what characterised its performance in the darker environment, however although it would get close, it did not once reach it within the 40 second run time. In contrast, prior to the noisy environment, it was able to reach the source in an average of 11 seconds. A redeeming factor that can't be overlooked however, was the preservation of the evolved ability to begin in any orientation, figure 14 illustrates an example in which the agent began pointing away from the source, with the 'X' marking the point at which it reverted from moving backwards, to forwards.

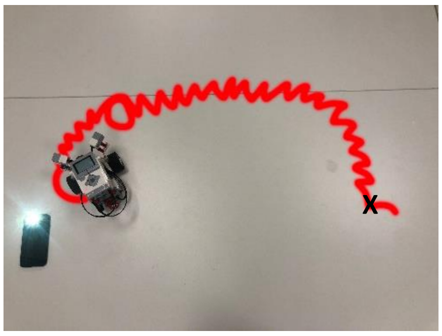

Fig 14 – Path taken in well-lit environment

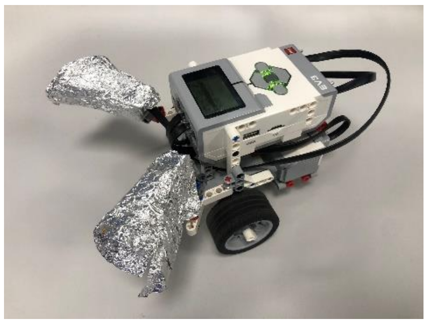

Fig 15 – Hardware modification

My first step in attempting to create a more robust system was to make the slight hardware modifications visible in figure 15. Although the tube-shaped tape proved ineffective during testing in a noise free environment, reflective tin foil cones now provided some much-needed directionality to the agent's stimulus.

I then introduced noise in the form of random normally-distributed variation in the light sensors within the minimal simulation during the evolutionary phase, in the hope that an agent evolved with the presence of noise, would cope with it better in the real world. Figure 16.1 demonstrates the performance of the resulting genotype in a noise-free minimal simulation. Once bridged across the reality gap it performed far more consistently than the previous controller, tracing an almost identical path every time, but on no occasion did it quite reach the light source, instead spiralling on the spot just next to it (fig 16.2). This however does not represent an issue with the reality gap traverse, as the same behaviour was exhibited in figure 16.1, and our concern lies with the consistency between behaviours, as opposed to the specifics of what they are. One limitation during my testing here was my access to an omnidirectional light as per that of the simulation; the agent settled in to a displacement-free rotation just on the edge of the lamps directional range.

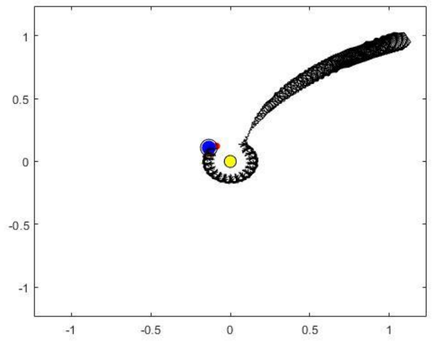

Fig 16.1 – Path of noise(randn)-evolved agent [Sim]

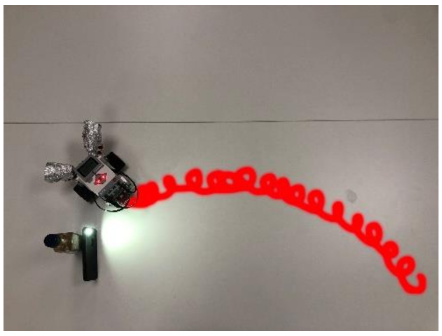
Fig 16.2 – Path of noise(randn)-evolved agent [Real]

I was surprised with the level of success that this had achieved, as I had concerns that the noise was not empirically validated as the findings of Jakobi et al [13] showed necessary. The noise in our scenario takes the form of ambient light of a constant intensity as opposed to varying in all directions in accordance with a normal distribution, and so I hypothesised a more effective way to mirror this to be to scale down the detected light intensities in the simulation during the CMA-ES optimisation. Such to reduce the contrast that becomes relied upon between light intensities when aiming at the source versus not, as the key impact of our well-illuminated testing environment is the abating of this disparity.

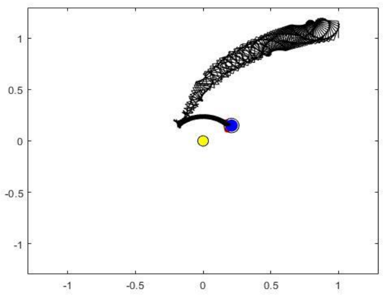
Fig 17.1 – Path of noise(light scaling)-evolved agent [Sim]

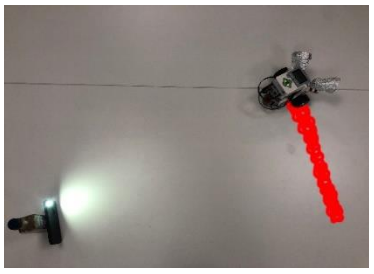
Fig 17.2 – Path of noise(light scaling)-evolved agent [Real]

It's evident from figure 17.1 that this approach has negatively impacted the simulated performance of navigating towards the light, but as we have stressed previously, we are more concerned with the controller's transferability as opposed to its isolated performance. When testing it in a LEGO Mindstorm in our noisy environment however, its performance only deteriorated further (fig 17.2), over the 40 second period it would complete in the range of 33 to 35 full rotations, whilst achieving a fairly consistent displacement of 42cm in a direction relative to its initial orientation, as opposed to the light source. The light would bear no effect unless I manually held it right up to the agent, since the detected intensity drops off exponentially as the distance between agent and light source increases. As a result of lessening the influence of the light sensors, they were now not acute enough to account for this drop off and differentiate the light source from the surroundings from any sort of reasonable starting distance.

With the above results, I realised the error in my earlier thinking. It became clear why far greater success was had with the genotype that was evolved with noise applied in the form of normally distributed variance added to the light sensor readings. By applying this variance over 36,000 (90 popSize x 100 generations x 4 times each) iterations purely during the CMA-ES, it applied a Gaussian smoothing effect of sorts to the light sensor readings upon which the evolved behaviour was based. Thereby achieving the desired outcome of reducing the sharp contrast between received values when aiming at the light source, compared to when not, these softer boundaries much more accurately replicate the conditions of this environment. Being more acute to this is exactly what is needed in environment with higher levels of ambient light.

6. Conclusion

In terms of our primary focus of simply bridging an evolved behaviour across the reality gap, this was certainly achieved in the controlled environment with very limited noise, as to closely replicate the conditions present in the idealised simulation within which the genotypes were evolved. The performance demonstrated in figure 13.2 (section 5.2) was not only highly repeatable from a great variety of initial positions and bearings, but the value of simulation was also proven, as per our criteria to far outshine anything that would have been tractable to evolve locally within a real robot.

When furthering our investigation to explore, firstly, the *effects* of a more realistic noisy environment upon these agents, our results corroborated with those of Mondada and Verschure [14], albeit to a lesser extent of success. The genotypes in question were evolved in the absence of noise, and when faced with the reality gap, key characteristics of the behaviours were preserved, however the systems offered very little robustness to the well-illuminated environment. If nothing more, these results serve as a proof of concept to support the findings in the aforementioned paper, as well as more accurately outline where the true challenge of the reality gap lies, bridging it robustly for a system to function under more realistic circumstances.

During the scrutinization of methods to achieve a more robust transfer, our results substantiated those of Jakobi et al [13], in which a combination of empirically validated noise introduced in to the simulation, combined with a robot designed with resistance to noise in mind, had desirable results. As demonstrated here, this does then depend heavily on the method with which the noise is modelled, varying the light sensors input in accordance with normally distributed random values yielded considerably more positive results than simply scaling the values down. Which leads me on to a possible point for further investigation; how versatile is this method of noise modelling? What other agent-environment interaction scenarios can this method of noise modelling be applied to?

Although a good level of success relating to our primary objective was achieved, the shortfall of our work here lies within the strict limitations that had to be associated with any 'robust' transferred behaviour. In order to succeed in bridging the reality gap whilst accounting for noise, we had very little say in the specifics of the behaviour we were able to do this with. As a result, we were creating something purely replicable as opposed to a well-defined behaviour of our choosing. However now that we have proven the potential of simulation based optimisation, combined with the plausibility of bridging the reality gap, this is a point from which I would love to pick up from again as part of future research; the bridging of robust, well-defined behaviours from simulation into dynamic real-life situations, applying everything we have learned thus far. I would also love to broaden my understanding of the complexities of the reality gap in exploring behaviours based on other senses. The GoogleAI blog [15] states that visual perception constitutes the widest part of the gap, this being the case we should be able to implement audio-stimulated behaviours, for example, with respectable degrees of success, provided we can suitably mirror the necessary mechanics within a minimal simulation.

7. Bibliography

[1] S. R. James, “Hominid Use of Fire in the Lower and Middle Pleistocene: A Review of the Evidence,” *University of Chicago Press,* pp. 1-26, 1989.

[2] D. A. Anthony, “The Horse, the Wheel, and Lnaguage: How Bronze-Age Riders From the Eurasioan Steppes Shaped the Modern World,” *Princeton University Press.,* p. 67, 2007.

[3] “Telescope History,” NASA, [Online]. Available: https://www.nasa.gov/audience/forstudents/9-12/features/telescope_feature_912.html. [Accessed February 2019].

[4] D. S. Halacy, Charles Babbage, Father of the Computer, Crowell-Collier Press, 1970.

[5] D. Cliff, I. Harvey and P. Husbands, “Evolving Visually Guided Robots,” in *SAB92*, 1992.

[6] F. M. Floreano D., “Automatic Creation of an Autonomous Agent: Genetic Evolutions of a Neural-Network Driven Robot,” in *From Animals to Animats 3: Proceedings of Third Conference on Simulation of Adaptive Behaviour*, 1994.

[7] R. A. Brooks, “Artificial Life and Real Robots,” in *In Toward a Practice of Autonomous Systems: Proceedings of the First European Conference on Artificial Life*, Cambridge, 1992.

[8] K. N. C. E. T. Orazio Miglino, “Selection for Wandering Behaviour in a Small Robot,” University of California, Los Angeles, 1994.

[9] N. Jakobi, “Running Across the Reality Gap: Octopod Locomotion Evolved in a Minimal Simulation,” in *Lecture Notes in Computer Science: EvoRobots*, Berlin, Springer, 1998, pp. 39-58.

[10] O. M. a. D. P. Stefano Nolfi, “Phenotypic plasticity in evolving neural networks,” C.N.R, Rome, 1994.

[11] J. N., “Minimal Simulations for Evolutionary Robotics,” University of Sussex, 1998.

[12] W. L. Brogan, Modern Control Theory (3rd ed.), Prentice-Hall, 1991.

[13] H. P. H. I. Jokobi N., Noise and the Reality Gap: The Use of Simulation in Evolutionary Robotics., 1995.

[14] P. V. F. Mondada, “Modeling System-Environment Interaction: The Complementary Roles of Simulations and Real World Artifacts,” in *Proceedings of Second European Conference on Artificial Life*, Brussels, 1993.

[15] S. L. Konstantinos Bousmalis, “Closing the Simulation-to-Reality Gap for Deep Robotic Learning,” Google AI, 30 October 2018. [Online]. Available: https://ai.googleblog.com/2017/10/closing-simulation-to-reality-gap-for.html. [Accessed 3 March 2019].

[16] O. L. H. a. N. S. Miglino, “Evolving Mobile Robots in Simulated and Real Environments,” *Artificial Life,* pp. 417-434, 1995.

[17] V. Braitenberg, Vehicles: Experiments in Synthetic Psychology, MIT Press, 1986.

[18] D. C. Dennett, “Why Not the Whole Iguana?,” *Behvioural and Brain Sciences,* vol. 1, no. 1, pp. 103-104, 1978.

[19] R. D. Beer, Intelligence as adaptive behaviour: An experiment in computational neuroethology., Cleveland: Case Western Reserve University, 1990.

[20] R. D. Beer, “On the dynamics of small continuous-time recurrent neural networks,” *Adaptive Behaviour,* vol. 3, pp. 469-509, 1995.

[21] I. H. P. C. D. T. A. a. J. N. Harvey, “Evolutionary Robotics: The Sussex Approach,” *Robotics and Autonomous Systems,* pp. 205-224, 1997.

[22] A. O. Nikolaus Hansen, “Completely Derandomized Self-Adaptation in Evolution Strategies,” *Evolutionary Computation,* vol. 9, no. 2, pp. 159-195, 2001.

[23] Yarpiz, “CMA-ES In Matlab,” Yarpiz, 6 September 2015. [Online]. Available: http://yarpiz.com/235/ypea108-cma-es. [Accessed 5 February 2019].

[24] L. Rastrigin, Systems of Extreme Control, Moscow, 1974.

[25] N. H. Anne Auger, “Tutorial CMA-ES — Evolution Strategies and,” 6 July 2013. [Online]. Available: http://www.cmap.polytechnique.fr/~nikolaus.hansen/gecco2013-CMA-ES-tutorial.pdf. [Accessed 20 4 2019].

[26] A.-R. Hedar, “Global Optimization Test Problems,” [Online]. Available: http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO_files/Page2607.htm. [Accessed 21 March 2019].

[27] N. Donges, “Recurrent Neural Networks and LSTM,” 25 February 2018. [Online]. Available: https://towardsdatascience.com/recurrent-neural-networks-and-lstm-4b601dd822a5. [Accessed 15 March 2019].

[28] D. K. Stanley, “SlidePlayer,” 25 October 2006. [Online]. Available: https://slideplayer.com/slide/4642416/. [Accessed 30 March 2019].