Co-Evolving Sensor Morphology and Control for a Legged Robot

Honors Thesis

Submitted in Partial Fulfillment of the Requirements

for the Bachelors or Arts

in Computer Science

Author: Pramod J Nathan

Advisor: Prof. Gary B Parker

Connecticut College

May 2004

ABSTRACT

This thesis discusses utilizing genetic algorithms to automatically design a suitable sensor morphology and controller for a given task in categories of environments. The type of sensors, the heading angle and the range of the sensor, and the rules the controller, are co-evolved. The described method enables the system to decipher information from the environment to determine that is relevant to completing a given task while configuring a minimal controller and number of sensors, thus increasing the overall efficiency of the robot. The system is tested in 4 experiments. Three of these experiments are simulated and the fourth test occurs on the actual robot. The thesis shows that using the genetic algorithm is a feasible method for rapid and effective automated design of sensor morphology and controller of autonomous agents.

Table Of Contents:

1. Introduction

This thesis presents an approach to automatically designing the morphology (form and structure) and developing the control program for sensors on a legged robot. The goal of the thesis is to study the success of an agent whose sensor design configuration, type, capabilities and control are automatically designed using a genetic algorithm. The study shows a viable and effective approach to designing robots that are able to consider their own inherent strengths and weaknesses in the environment as they perform their tasks without an explicit human generated blueprint.

1.1 Need for Automated Design

The problem of designing hardware and software systems for robots is difficult and time consuming, requiring experts to work on the design of the system in minute, painstaking detail. Traditionally, robotic systems are built by a team of engineers and experts that are intimately involved in the design process. This design process on robots can be broken down into two sections: morphological (physical hardware or "body") and controller (software that runs the robot, the so-called "brain"). Manually designing a typical machine requires an engineer to adjust and take into consideration tens or hundreds of variables to produce a successful design. Moreover, many variables, such as power consumption and weight of the robot, are dependent on each other. The knowledge of these dependencies, especially the intricate ones, is contingent upon the experience and knowledge of the engineer, which might be insufficient because every new design is a leaning process for the engineer.) This is a painstaking process that might take weeks or months. As described by John Holland, the father of genetic algorithms (GAs), "In one fairly typical case, an engineer working alone took about eight weeks to reach a

satisfactory design." Using an expert system, an application program that solves problems in a particular field using knowledge and analytical rules defined by experts in the field, the "engineer took only two days to find a design with three times the improvement of the manual version." Such computerized systems based in artificial intelligence show their utility in solving design problems. Unfortunately, while expert systems are of significant advantage to designing, they cannot improve a system when multiple simultaneous changes are necessary or when the problem is beyond the expert system's specified domain knowledge. The latter is an especially large stumbling block; since an expert system can only have the knowledge and rules of previously designed systems, which limits their use in the design of new systems. In addition, the system needs a high degree of human intervention and has a low level of autonomous designing capability. However, the shortfalls of the expert system and traditional engineering techniques serve as excellent examples in making us aware of the requirements of designing a system. We need a new design technique that is capable of simultaneously adjusting a high number of variables while taking into account their interdependencies with minimal or no human intervention, as a consequence decreasing human work hours and the overall time needed to produce a successful design.

Following these basic requirements, the GA is the ideal candidate for rapid and successful automatic design. The application of this technique allows for rapid development of designs due to its ability to simultaneously adjust multiple variables and high level of automation.

1.2 Motivation and Inspiration

This thesis focuses on automatically designing the sensor morphology and the controller for a six-legged robot. The study is largely inspired by nature, particularly the sensory systems on animals and the importance of these sensory systems for survival. The primary task of sensors in nature is to decipher the environment in which the animal performs, which is similar to the relationship of artificial autonomous agents and their environments. In nature we see a variety of sensory systems. We as humans have a sense of sight, hearing, touch, smell, and taste; other animals have a different set of senses, which might include magnetic and electrical field sensors, such as a shark that can detect electrical fields, allowing it to catch fish buried in the sand). The more sensors an animal has, the better it can garner information from its environment. However, to be successful in nature an animal must also be energy efficient and hence must be able to select which sensors to have and which to prune. This natural selection of sensors, as theorized by Darwin, is an important feature seen in nature. No animal has every sensory organ. Thus sensors play a key role in deciphering the environment an agent has to perform in, as well as defining which stimuli are advantageous to its survival and which are not.

Sensory systems also have varying degrees of sensitivity and detect vastly different ranges of stimuli: A hawk can see from 20 feet what people can see from 5 feet, giving them 20/5 vision compared to humans having 20/20 vision. While this is another aspect of efficiency (apparently humans do not need 20/5 vision), it also shows that an agent needs to adjust its sensor's capabilities to pick out information relevant to its survival and learn to ignore the others. Other well known examples are a cat's high light

sensitivity and mole poor sense of sight; both attributes are as a result of adaptation to a specific behavior or environment.

Apart from the ranges of sensor capabilities, the effectiveness of these sensors in interpreting the environment is dependent on features such as positioning, the types of sensors, the manner in which the sensors are integrated into the controller of an agent and the limitations of the sensors and agent. Each of these factors, which we will call sensor attributes, have to be integrated in the agent in such a way as to allow the agent to deal with its immediate surroundings while avoiding redundancy in the information it obtains from the environment. Thus an agent is limited by the range of stimuli it can detect, as well as the effectiveness of the sensors that are at its disposal. These sensor attributes are co-related with the agent's capability to learn in the environment.

Here we see two apparently opposing features advantageous to survival: efficiency and the ability to detect and use many stimuli. To efficiently perform a certain task, animals need only a subset of their available sensory organs and capabilities. Similarly, autonomous agents need only a small subset of their sensors for specialized tasks. Thus we have three design considerations for success and efficiency: selection of the sensors, selection of the sensor's attributes, and utilization of the sensors to complete a given task. This study tackles all three above mentioned considerations, using the genetic algorithm to produce an effective design that is efficient and successful.

1.3 Previous work

In the past, researchers have applied the idea of evolution to design and select many sensory attributes. Balakrishnan and Honavar [1] evolved the position and ranges

of sensors but in a highly limited and discrete simulated block environment and the results of the experiment was not transferred to a real robot. Work has also been done in creating adaptive hardware to evolve an array of the same type of sensor [2, 3].

Other research used modular parts to build structures [4] and robots for the purpose of locomotion; this work on buildable structures shows us that the idea of evolving morphologies is a feasible and practical solution for actual hardware design problems.

The field of co-evolution of morphology and the controller has been increasingly researched in recent years. Research done on Tinkerbots [5] used the idea of re-usable sub-procedures called Lindenmayer systems to design scalable complex designs. Similarly, Marbach and Ijspeert[6] present a implementation of co-evolution of a PD controller and morphology using simulated modules to build scalable simulated mobile robots. Yet as pointed out by Pollack et al, [7] these agents do not interact with the environment due to the lack of sensors. This lack of sensors does not allow these agents to evolve beyond a certain complexity due to the lack of constant feedback from the environment.

Research conducted by Lee et al [8], Lund et al.[9], Mark[10], Bugajska and Schultz [11, 12, 13] and Mautner and Belew [14] shows strategies to co-evolve the robot controller and the sensor morphology. Lund et al. [9] studied the co-evolution of placement of sound sensors and controller on an actual Khepra Robot. Mark et al. [9] worked on evolving number and widths of eyes on Braitenberg-like vehicles on a continuous 2 dimensional simulated environment. The studies of Bugajska and Schultz [11, 12] dealt with finding a generalized sensor morphology for any environment

configuration on a flying autonomous micro-air vehicle. Their studies extensively explored applying autonomous navigation to the agent and incorporated anytime learning in their system [13]. Mautner and Belew [14] co-evolved the placement of sensors and effectors as well as the controller of the system in a simulated system. Unlike our study, Bugajska and Schultz [11, 12, 13] dealt with finding a generalized sensor morphology for any environment configuration; in Mautner and Belew [14] the complexity of the environment was constant. Our study deals with different environment configurations of varying complexity. Bugajska and Schultz [11, 12, 13] also have an onboard sensor that could detect distance to goal. Our robot does not have this; instead, the agent must learn to find the goal using cues available in the environment. Moreover, these works evolved sensor morphology using the same type of sensors. The drawback to this is that sensors of the same type have limited detectable stimuli (they can detect only one type of stimuli) and do not provide the control system a wide variety of information about the environment. The agent thus does not have the choice of deciding which stimuli is advantageous for it and utilizing those stimuli, thus reducing the capabilities of the robot as a whole. Finally this study deals with evolving the sensor morphology on a simulated system and transferring this system to a real robot to test for reliability and feasibility of the system.

Lego Mindstorm robots were used by Lund et al [15] in co-evolving the controller and the physical and sensor morphology of the robot. But the search space was limited (825 possibilities) with only a single type of sensor being used with 11 possible positions; there is no such limitation in our study.

One of the first groundbreaking works on co-evolution of morphology and control was conducted by Karl Sims [16]. The study dealt with co-evolution of the neural control and physical morphology of virtual creatures that compete in physically simulated three-dimensional worlds. The study showed interesting and successful strategies and morphologies emerged from the competing co-evolution process. The study differed from ours in that it did not explicitly study the role of the physical environment in the evolution process sensor morphology was only implicitly evolved as a result of the overall change in physical morphology. Lastly, the study was not transferred to a real world environment to test for feasibility since it was a study of an evolutionary strategy and not a study of an application of evolutionary morphology.

1.4 Goals of the Thesis

Pfeifer [17] and Atkins [18] suggested that there is a relationship between the morphology, the controller and the environment of a given system. For a specific task certain stimuli provided by the environment are advantageous while others are not. The task of a sensor is to sense these stimuli. The more complex an environment is, the more complex the sensors need to be in order to obtain and extract relevant stimuli in an environment. At the same time, the agent has to decipher which stimuli are necessary for the completion of a certain task and prune the unnecessary ones. Pruning makes the agent more efficient, as extra sensors put a strain on the energy supply of the agent, and allows for a less complex controller that does not use all the available stimuli.

This thesis shows that evolving sensor morphology allows for the automatic design of an agent that can exploit the environment to complete a given task using stimuli in the environment as cues, decipher relevant information to complete a task, discard

irrelevant information not needed to complete a given task, and decrease the number of sensors it uses for a given task, in a niche environment.

Evolving sensor morphology allows for optimization in terms of power consumption, controller complexity and reaction time. This thesis investigates the use of a GA to evolve the sensor morphology and a controller for actual robots. The aim of the study is to automatically design a successful sensor morphology and controller and show its feasibility on a real robot.

Co-evolving sensor morphology and control allows automatic design of an agent that can:

- Exploit the environment to complete a given task using stimuli in the environment as cues
- Decipher relevant information to complete a task
- Discard irrelevant information not needed to complete a task
- Increase the efficiency of the agent (robot) by decreasing the number of sensors used to complete a given task

2. Background Topics

Before moving on to the actual system used to solve the above problem, it is important to grasp the ideas behind the used system. The method implemented in this study relies on two key ideas: the genetic algorithm and reactive control. These concepts are used in the current system to produce a feasible agent that can successfully complete the tasks given to it in a variety of environments.

2.1 Genetic Algorithms

The genetic algorithm (GA) is inspired by evolutionary biology and uses techniques based on Darwinism such as heredity and survival of the fittest. In computer science terms, it is a heuristic based search algorithm that performs a search on a given problem domain.

Let us start with an intuitive idea of the genetic algorithm. Suppose we have a population of animals on an extremely cold planet. To ensure success in such a cold environment some form of insulation might be a good solution (note: it might not be the optimal solution). The basic idea of the genetic algorithm states that given this initial population, the genetic pool contains the solution to survive in this cold environment. The reason the solution is not currently present is because the combination of genes that makes up the solution is split between several individuals. Through natural selection, those individuals that do not have sufficient insulation will die off as will animals with too much insulation, possibly due to lack of mobility. The suitably insulated individuals will survive and pass on their genes to their children thus ensuring the solution to better survival passes on. Essentially, through the process of evolution the animal species will become more adapted to its environment.

Translated to an algorithm the above evolution process (assuming sexual reproduction) would look like the following:

1. From current population:
2. - Check how well each individual is adapted to the current cold
3. - Suitably insulated individuals have higher probability of reproducing

4. - Select two individuals (based on above probability) to reproduce offspring
5. - Produce new offspring (via sexual reproduction and including possibility of mutation)
6. - Do step 4 until new population is generated
7. Using the new population start from step 1

The GA has a similar form:

1. Generate a random population of N individuals/chromosomes using a set *representation*
2. Evaluate the fitness using *fitness function* of each of the chromosomes
3. Until a new population
    a. According to their fitness *select* two parent chromosomes from the current population
    b. Create a new offspring using the two parents by *crossing over* the chromosomes (simulated sexual reproduction).
    c. Using a small probability (mutation rate) *mutate* new offspring
    d. Place new offspring in new population
4. Replace old population with new Population
5. If *end condition* then terminate return best solution
6. else go to step 2

1. Representation

In a computer science, an algorithm finds a solution to a given problem. In a GA a solution is usually represented by a string of 0s and 1s (bits). This representation is known as a chromosome. Each individual has a chromosome of bits, which is initially randomly generated, that represent a solution to the given problem. Hence, each bit usually represents some characteristic of the solution. In our example, the thickness of the insulation could be represented by a number represented in binary (e.g. 1010 could represent 10mm thick insulation).

A set of individuals/chromosomes is called a population.

2. *Fitness function*

In GA terms the ability to survive the cold would be the "fitness" of the individual. This "fitness" is calculated by a "fitness function" which is a measure of the success of the individual. In our cold planet example, this function could simply state, "Individuals that survive the cold are more fit than individuals that die or are weakened." However a better fitness function would give a high fitness to individuals that survive the longest.

The fitness function is one the most difficult part of the algorithm to implement. In the real world, a fitness of an animal is dependent on many factors including the environment it is in and its own abilities. To evaluate its fitness all these factors have to be modeled.

a. Selection

This fitness function is central to the genetic algorithm and allows the program to calculate which individual solutions are more fit than others, thus allowing us to determine which parents to select from the current population. This process, known as selection, usually is implemented by using a stochastic (roulette wheel) selection. Given three fitnesses of 10, 5 and 1, the roulette wheel selection process will make the individual with the fitness of 10 twice as likely to be chosen as a parent and ten times more likely than the individual with the fitness of 1.

b. Crossover

Once the parents are selected a new offspring is produced. The chromosome of the offspring is generated by choosing a random point in the parent's chromosome called a crossover point. The child then consists of everything before this point from the first parent and after this point from the second parent as illustrated below.

Parent 1: 10011 | 100110101 Parent 2: 00110 | 001101110

Child: 10011 001101110

Fig. 1 Crossover

c. Mutation

Mutation takes place once an offspring is generated. The mutation operator randomly changes the offspring generated using some small probability called a mutation rate. This is usually done one bit at a time. Using a low probability a bit is flipped. Although, as in natural evolution, mutation is usually harmful, it is needed to

maintain genetic variety and thus prevent the algorithm from falling into a local maximum. Hence sometimes a mutation can prove beneficial

Elites

Sometimes individuals that have an extremely high fitness are selected to pass to the next population without change. This is done to retain good solutions until better ones are generated.

Using these genetic operators the GA evaluates each individual and produces offspring that improve in fitness, thus getting closer to an optimal solution. The GA is founded on evolution which had proven in effectiveness in nature making it an excellent candidate for the problem of automatic design. The program is terminated once a given condition is satisfied, which in our case, is the number of generations allowed for evolution.

2.2 Reactive Control

A controller is the hardware and software necessary to operate a robot, whose biological counterpart in higher level organisms is the brain. The agent in this project has a reactive controller. Reactive control is the simplest type of control system. Reactive architecture is exactly like it sound: like a human reaction to touching a hot object, the robot reacts directly to the inputs it receives from the sensors without planning or processing.

Reactive Control: Detect ➔ Act

Implementation: Sensor ➔ Reactive Controller ➔ Actuator

Fig. 2 Reactive Control and Implementation on Current System

In a reactive controller, the perceptual world can be broken down into mutually exclusive stimuli. Actions are assigned to each one of these stimuli making up a set of rules that map certain stimuli to certain actions. This controller, though extremely simple, can produce interesting results. The reactive controller, due to its simplicity, does not have an internal representation of the environment but uses the environment as a representation. This allows the relationship between the agent and the environment to be studied with ease.

Such reactive agents are simple to design, fast and cheap, yet, as described by Braitenberg, they can exhibit behaviors which can appear to be aggressive, loving and even decisive [19]. These vehicles are also known as Braitenberg vehicles.

In this study, the reactive control maps one sensor with one action. This can be described as the rule (if sensor A triggered, then perform action X). The specifics of the controller will be described in section 3. Another advantage of the reactive controller is the complexity of the controller can be measured easily by simply counting the number of rules needed to successfully complete a given task. Thus using a reactive control we can quantitatively establish the complexity of the automatically designed controller.

3. Technical Details

This thesis deals with learning a solution to the problem of automatically designing sensor morphology and control using a simulation of an actual system. We then test the automatically generated designs in a real world environment to check for their feasibility.

Learning in a simulated environment is extremely practical when using a GA. A GA requires constant re-evaluation of newly generated individuals. The simulation simplifies the process of re-evaluation by performing the design and evaluation in simulation instead of constantly building actual designs in the real works and testing them. Using a simulation for learning also allows us to rapidly generate solutions to the given design problem without having to deal with physical realities such as wear and tear and short battery lives.

The simulation, which is a simplification of the environment, has drawbacks. It assumes many ideal situations which are not present in the real world. For example a light source can be “ideal” in the simulated world, with its intensity constant for any given distance from the source. This, of course, is not true in real life. Thus a simulation, though is easily implemented for rapid learning, might not produce results transferable to the read world due to idealized models that do not take into consideration the noise of a system. It is therefore necessary to test the generated system on the real world system to check for their effectiveness.

3.1 Hardware System:

3.1.1 Agent and Controller

The ServoBot [19] is a six legged robot (hexapod) with two degrees of freedom per leg, thus requiring 12 servos in all (One servo for up-down motion, the second for back-forth motion). The robot was previously developed by David Braun at Indiana University to study colony and legged robots.

Fig. 3 The Hexapod Robot

The Servobot uses a tripod gait for locomotion. The gait was previously evolved using a cyclic genetic algorithm to learn a near optimal gait for the robot. It is controlled using a Basic Stamp II microprocessor which is onboard the robot. The controller requires 12 pins to control the 12 servos. It has 4 free pins which can be used to interface the controller with other modules, including sensor modules, as will be used in this study.

The tripod gait of the robot is a large source of mechanical noise on the robot. This factor makes it very different from a wheeled robot. The noise increases the uncertainty while sensing. This factor necessitates that any successful sensor design be

robust enough to perform its task while under these noisy conditions. Unlike wheeled robots, which can turn at any angle less than or equal to its maximum turn angle, the legged robot with specific gaits does not have this ability. It has 16 available turn gaits, each of which has a predicted (through measurement) end orientation and position given a start orientation and position (Table 1). The agent's locomotion is non-deterministic, i.e. given a start position and orientation, the end position and orientation after a step has a degree of randomness involved. This non-determinism in the gait is due to the inconsistencies of the robot construction and inherent noise associated with a legged robot interacting with the real world.

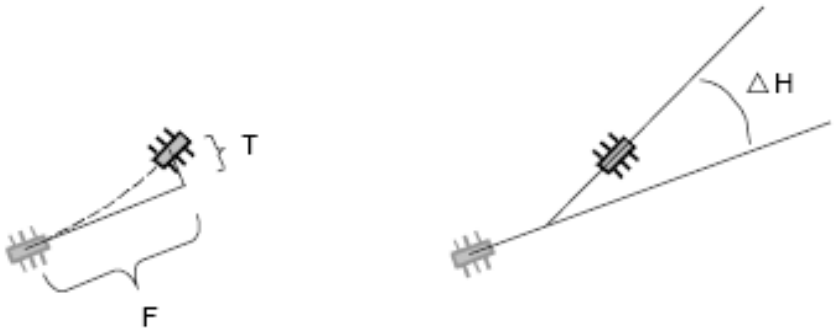


Fig. 4 [19] Gait Cycle Turn Measurements: the diagram explains the turn gait of the robot. Given a start position, F is the distance moved forward, T is the lateral displacement and ΔH is the change of heading from before to after a single gait cycle.

Left Turns	Right Turns
{3.7,4.0,24.3}	{5.0,-3.7,-26.7}
{4.8,4.3,18.8}	{6.0,-5.0,-22.2}
{5.3,4.0,16.7}	{7.3,-4.3,-18.8}
{6.5,4.0,14.7}	{8.4,-4.3,-15.8}
{8.4,3.5,11.0}	{9.6,-3.8,-13.5}
{9.4,2.8,8.3}	{10.4,-2.8,-10.3}
{10.1,2.3,6.2}	{11.1,-2.3,-7.0}

Straight: {12.4,-1.0,-2.3}
Reverse: {-12.4,1.0,2.3}

Table 1: Turn Gaits Used in Simulation. The above triplets correspond to {F, T, ΔH} as explained in Figure 4 and is modeled on the actual robot. This model is used by the simulation to define the locomotion of the agent.

The Servobot is equipped with a sensor base attached to the top of the robot that completely covers the robot so that the actuators and effectors of the robot do not affect the sensors. This base is 30cm x 30cm and acts as an easily reconfigurable platform for the sensors. The base can be moved from one robot to another and have any reasonable number of modules attached. Due to its placement on top of the robot, it is not a stable platform as it greatly increases the height of the center of gravity of the robot. This is another factor that amplifies the noise of the system and can prove to be problematic when testing the robot in the real world environment.

3.1.2 Sensors

The sensor base can carry modules of tactile sensors, sound sensors and light sensors, (Infrared and Ultraviolet). These sensors are interfaced with a Basic Stamp II.

Using a Basic Stamp II along with the sensor modules restricts the number of sensors used to 12. The design for each of the sensors is dependent on the sensor morphology generated by the GA.

Figure 5 shows the light sensor constructed on a breadboard. The light sensor can control the length at which it detects input using a simple hardware mechanism. The sensor uses two photo resistors to compare ambient light and directional light. If the directional light is greater than the sensed ambient light the sensor triggers: true. The maximum length at which a light source can be detected is simply a matter of adjusting amount of ambient light shone on the sensor by changing the angle of the rear photoresistor.

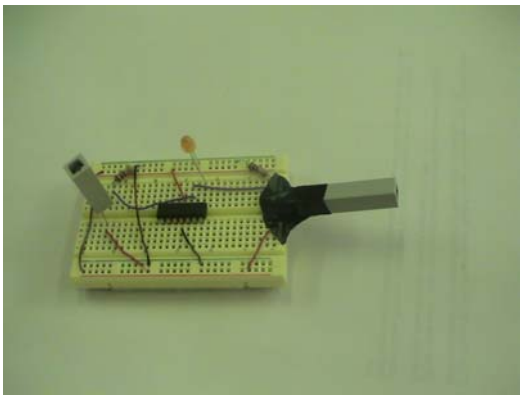


Fig.5 Light Sensor on breadboard

The tactile sensors are simple switches that trigger the feeler it bumps into an obstacle. This sensor is shown in Figure 6. The issues involved in designing this sensor have to do with the legged nature of the agent. Due to a huge amount of mechanical noise involved in walking the tactile sensors are prone to oscillations and produce many false but short lived triggers. The tactile sensors were attached to a 4.7uF capacitor to act as a filter to this noise. If the sensor is falsely triggered due to the oscillations of the robot the noise was filtered out by maintaining a state just prior to its false trigger. If the touch

stimuli as a sustained one though, the sensor would detect it as a true trigger. This scheme while increasing the reliability of the tactile sensor was still affected by false triggers.

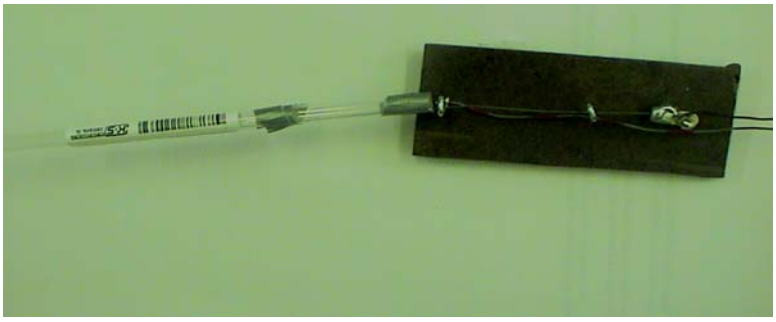


Fig. 6 Tactile Sensor

3.1.3 Environment and Environmental Stimuli

The robot colony space is a 2.5m x 2,5m walled area established in the Connecticut College Robotics Laboratory. It can be equipped with robots and can be configured with any reasonable number of obstacles and light sources. The obstacles are 26cm x 26cm x 5cm blocks. The lights are two omni directional light sources. The obstacles are not high enough to block the light beam. Both the obstacles and the walls can be detected by the tactile sensors placed on the robot.

3.2 Simulation:

3.2.1 Simulated Agent

The simulated robot closely models the ServoBot. Its locomotion characteristics were obtained by measuring the performance of the actual robot. It is simulated to have a 30cm x 30cm reconfigurable platform for the sensors. Its locomotion is non-deterministic in that the measured turn values are used but randomized slightly to model the inherent flaws in the construction of the actual robot.

The controller of the simulated robot is a reactive system that used 13 rules of the form: If (Sensor detects a stimulus A) then (Trigger gait number X), in which each sensor is associated with a specific rule and a single gait. The consequents of these 13 rules are selected by the GA from the 16 available gaits (Table 1). These turn gaits used were from data stored after measuring the 16 gaits on the actual robot. The GA also selects the default gait, which is the gait used when no sensors are triggered.

3.2.2 Simulated Sensors

The sensor base can be configured with 4 tactile sensors, and 8 light sensors (4 infrared sensors and 4 ultraviolet sensors). The sensors are all binary; they either detect a stimulus or do not. Each of the sensors is a reusable module that can interface with the controller. The GA is responsible for evolving the orientation of the sensors and the range of the light sensors. The range at which the light sensors detect a stimulus is implemented in software by setting a cutoff at which the controller no longer detects a stimulus. The GA has 32 equal length choices for the range of each light sensor, whereas the tactile lengths are fixed. In simulation the maximum range of these light sensors is set at 434 cm. This range was chosen since it is slightly more than the hypotenuse of the Robot Colony which is approximately 424 cm. The spread of the sensors was not evolved since no mechanism was in place to adjust this aspect of the sensor. The tactile sensors were placed in the corners of the sensor base and the light sensors were placed on the edges of the sensor base.

In simulation all the sensors were “ideal”. They did not undergo wear and tear, did not trigger falsely, and were not affected by the mechanical noise caused by the robot legged locomotion

The placement, maximum range and spread of the sensors on the sensor base are shown as triangles in Figure 7. (The IR sensor and the UV Sensor range and spread overlap). The GA determined which sensors would be activated, their orientation and (in the case of light sensors) their range. The evolved characteristics allowed the sensory information to be complex enough for the robot to be successful in the real world while making the simplifications necessary to generate a simple simulation.

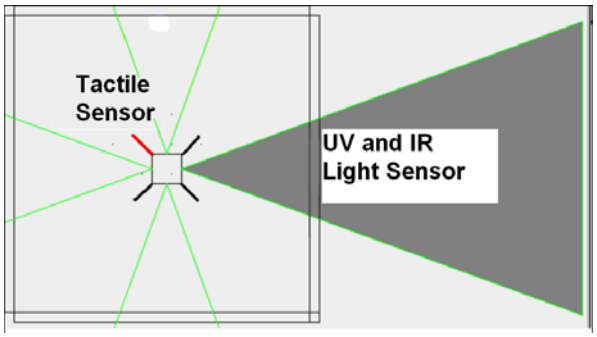


Fig. 7. Sensor Configuration (Range, Spread and Placement of Sensors on Sensor Base

3.2.3 Simulated Environments and Task

The colony space was represented by a 300 X 300 area in simulation. The agent’s task was to traverse from Start Position (40 +/- 10, 150 +/-10) to the Target Position (270, 160). The agent itself had no information about the location of the target point. To navigate it has to learn and use cues from the environment to its advantage.
The Environments (Figure8) used eight 30 x 30 obstacles and two omni directional light sources (IR and UV). In previous experiments a sound source was simulated but due to

high noise levels caused by reflection of sound, it was discarded in favor of the more controllable light stimuli. The robot could detect light over the obstacles and the tactile sensors detected walls and obstacles. The obstacles were configured in 7 different configurations (Figure 12). For each generation the obstacles were moved randomly by a factor equal to or less than +/- 10cm in the X and the Y direction to allow the results to have a level of generality. The obstacle density was 8%. The target was a fixed point marked by one of the light sources.

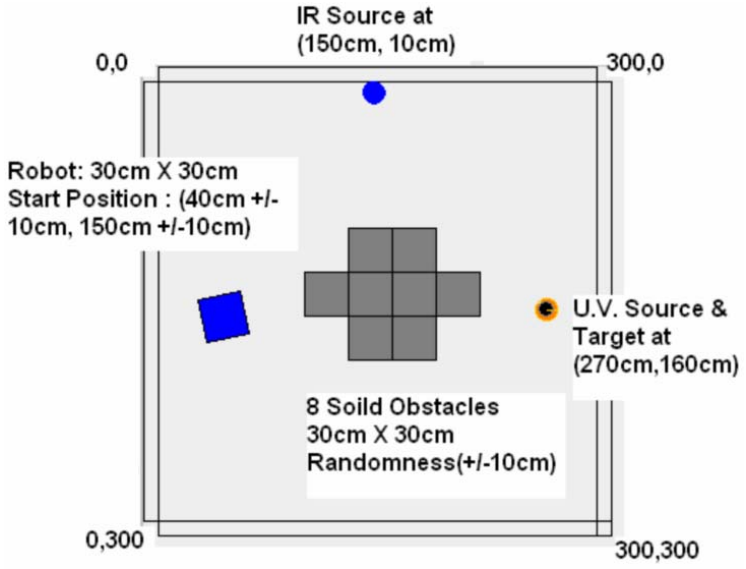


Fig 8. Environment Description (Central Mountain Configuration)

4. Tests and Results

The study is broken down into 4 experiments, the first three being in a simulated environment and the last one on the real robot.

Evolving Sensor Morphology: (Section 4.1)

The initial tests were done in simulation. The first of these experiments tested if

we could evolve a successful sensor morphology, given a fixed controller program and its limitations. The experiment dealt with the evolution of sensor morphology in different niche environments; in particular, the type of sensor, angle of heading and its effect on the controller complexity for the simulated hexapod robot. From this experiment, we show that this automatic design method enables the system to decipher relevant stimuli in an environment, increases the efficiency of the robot and also indirectly alters the controller of the robot to take advantage of the characteristics of a given environment.

This first experiment led to a better understanding of the system and the shortfalls of only evolving sensor morphology without the control program. The major shortfall of only designing the sensor morphology was that it relied heavily on its start position and orientation to succeed.

Co-evolving Sensor Morphology and Control(Non-Line of Sight): (Section 4.2)

The above work was expanded to co-evolve the sensor morphology and controller of the robot. In this section the robot to had to design a sensor morphology and controller that could not rely on its starting position or orientation to succeed. In this test the start position and orientation were randomized to a certain degree. This experiment shows that by co-evolving sensor morphology and control we can design autonomous agents that can complete a given task efficiently using stimuli in the environment as cues, while having a level of generality in terms of its start position and orientation. In this part of the study, we evolve the sensor morphology and controller to equip the robot to complete a given navigation task when it starts from a randomized start position and heading, in a wide variety of environments.

The results were surprising in terms of minimization of sensory inputs since one stimulus was completely ignored in every environment.

Co-evolving Sensor Morphology and Control (Line of Sight): (Section 4.3)

The tests in section 4.2 showed surprising results in terms of making minimizing the number of sensors necessary to complete a given task. The automatically designed agents ignored the infrared stimulus even though it was placed in a position that appeared advantageous.

In this test we raised the height of the obstacles so that the agent could not detect a light source behind it (hence needing a line of sight). This test checks the agent's ability to adapt to stimuli changes and is primarily a tool for comparison to the previous test. The results show that co-evolution of controller and morphology produce designs that adapt to changes made in the stimuli of a given environment.

Tests on Actual Robot: (Section 4.4)

The automatically designed controller and sensor morphology were tested in the real world environments to check for its accuracy and success in the real world given the simplifications of the simulation.

4.1 Evolution of Morphology

This section of the simulation deals with evolving sensor morphology on the ServoBot[19] in simulated niche environments. The sensors in this section have pre-defined capabilities of range and coverage.

The parameters that are evolved in this paper are the heading angle of the sensors and which sensors to keep running during the length of the run; the position of the sensors on the sensor based are fixed as shown in Figure 7 and do not change. The heading of the sensors can be rotated 360 and as many as all 12 sensors can be turned off.

Unlike previous research [11, 12 ,13], using pre-defined capabilities of range and coverage of a sensor is an important difference since the sensors available in the market are not reconfigurable in terms of changing their ranges and areas of coverage. We argue that, even with this restriction, the positioning of the sensor on an agent plays a key role in deciding how many sensors are needed for a given task. The location of a sensor also enables the agent to take full advantage of the terrain to complete a given task. Moreover, this system is more practical and is easily transferable to a real robot. Since this study involves evolving sensor morphology, one of the domains the study covers is deciphering the information obtained from the environment to complete a task.

4.1.1. Chromosome

The chromosome used was 120 bits long. The first 12 bits represented which sensors to put on or off. The remaining 108 bits represented the angles at which of the

each of the 12 sensors are placed on board the robot base. The chromosome was divided into 1 set of 12 bits and 12 sets of 9 bits.

100110010010 111001010 100011010… 000110101

Fig 9: The Chromosome. The first 12 bits tell us which sensors to use. In this case the sensors to be used would be number 1, 4, 5, 8 and 11. These correspond to 2 tactile sensors (Sensor 1, 4) two sound sensors (Sensor 5, 8) and one light sensor (Sensor 11)The first 9 string (blue), tells us that the angle of placement should be 98 degrees. 1110010 in binary is 458 in decimal, since this number is greater than 360 the angle of placement should be 458 -360 = 98 degrees

4.1.2. Genetic Operator

During evolution, each set underwent a single point crossover using stochastic (roulette wheel) selection of the parents and had a mutation rate of 4%.

```
If (Agent reached Target)
        Fitness =
                100(2(Number_Sensors_Off) + (Total_Time – Time_to_Target))
Else
        Fitness =
                100(Number_Sensors_Off + Distance_to_Target)
```

Fig. 10: The Fitness Function

The fitness function (Figure 10) is dependant on the success of the agent. If the agent is successful in finding the target, the fitness is based on the number of sensors it has off and the amount of time it took the agent to get to the target. If the agent is unsuccessful in reaching the target, the fitness is dependent on how far away the robot is from the goal as well as the number of sensors it has off. In the first case, to achieve the maximum fitness the robot has to have all its sensors off and reach the goal without any time being used,

which is an impossible scenario in the simulation. A successful agent has the theoretical maximum fitness of 12400 while an unsuccessful agent has a maximum fitness of only 5440. Even though both parts use the number of sensors put off, it is given only half the weight when the agent is unsuccessful.

4.1.3 . Test

We ran 128 individuals for 128 generations for each environment. This was repeated 5 times with random starting populations to check for consistency of the results. The individual with the highest fitness was added to the next generation without change.

The agent was given 100 time intervals to complete the task. A time interval is the time it takes a robot to complete one step. A step starts with the legs in a ready to step position (right front and back legs and left middle leg forward; remaining legs back) and returns to this position after a full step cycle. This time is the same for all 16 step cycles. The time intervals continue to elapse even if the robot collides with an obstacle and tries to continue to move but it takes up time steps. A collision does not stop a test run. The agent can get out of a collision since the agent can shake free of the obstacle and continue its task due to its non-deterministic gait. The run is stopped if the target position is reached or when the agent runs out of time (100 time intervals).

4.1.4. Results

The results in Figure 11 show the average of the elites over 5 runs (Elite: individual with the topmost fitness for a given generation number). The error bars are the standard deviation, showing the consistency of the results over the 5 runs.

In all of the environments, the GA improved the fitness of the agents. Apart from the 7 environment the agent also learnt on a completely random environment. In this environment the obstacle configuration was changed after every generation. In the case of the completely random environment, there is no overall increase in fitness over the number of generations (linear regression shows a small negative slope of 0.0158). This environment showed that if the environment is completely random, the success of the individual too is completely random. The average fitness in the case of the completely random environment was 38.91419 % of the total obtainable fitness. It is interesting to note that environments with a starting fitness of less than 38.9%, namely Single Left Ridge and Double Ridge, had results that were much less consistent in the second half of the generations than the ones that started with fitness above this value. The initial values of the standard deviation in all cases were high due to the random generation of the initial population.

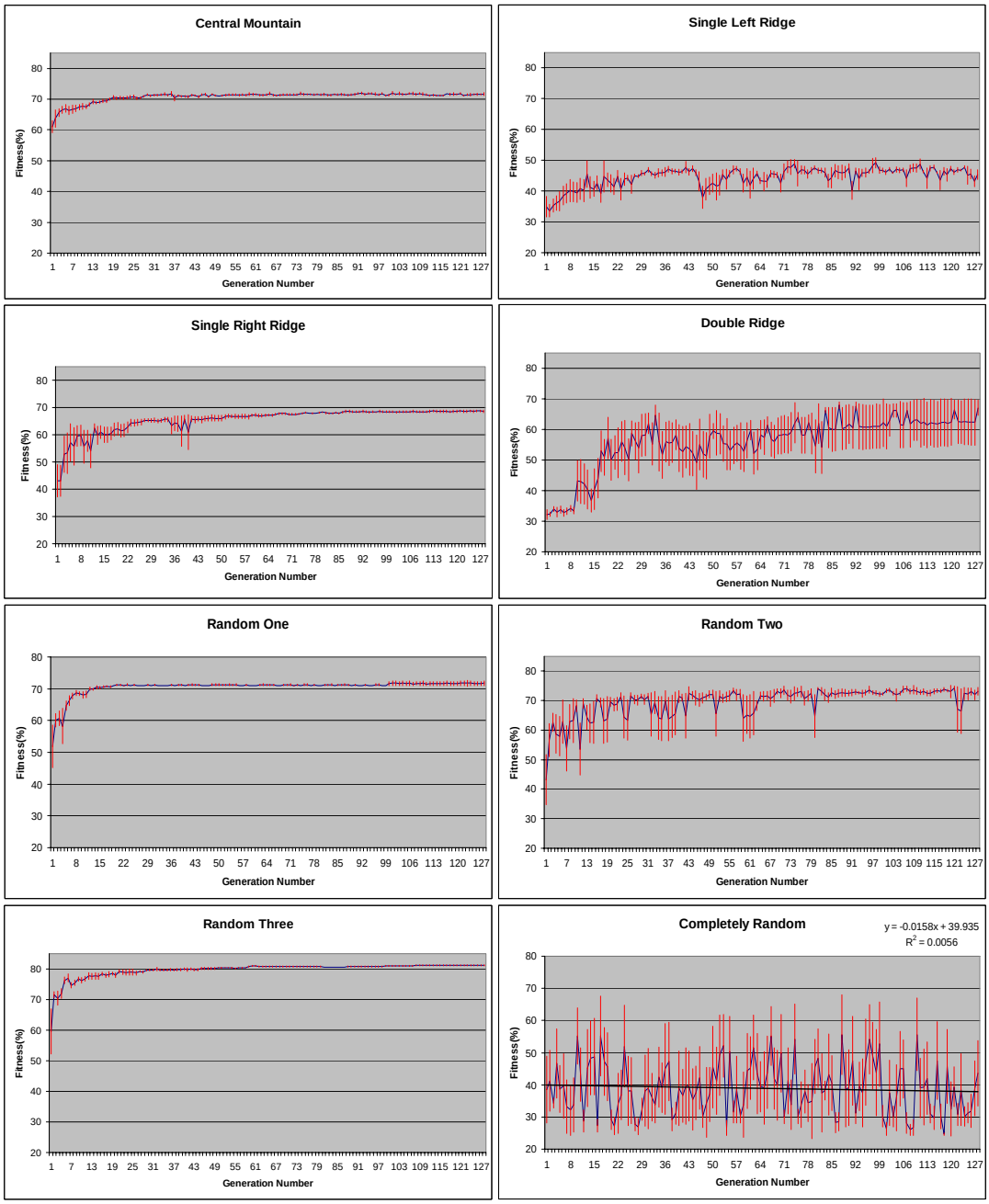
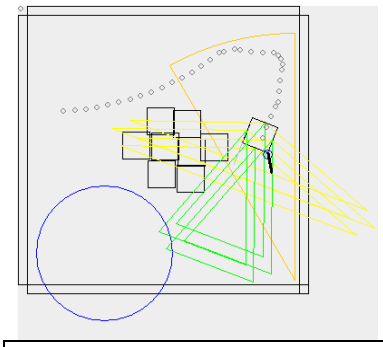
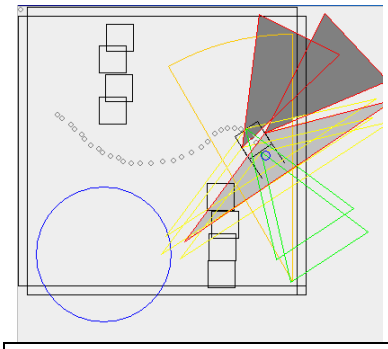


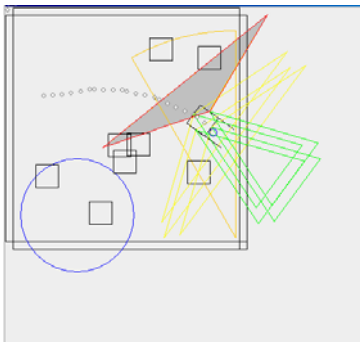
Fig 11: Fitness growth of elite individuals averaged over 5 test runs.



Central Mountain	Single Left Ridge	Single Right Ridge
1 Tactile Sensor at 330º	1 Tactile Sensor at 18º 1 Sound Sensor at 123º	1 Tactile Sensor at 333º



Double Ridge	Random One	Random Two
1 Sound Sensor at 359º 2 Light Sensors at 240º and 260º	1 Tactile sensor at 342º	1 Tactile Sensor at 23º 1 Light Sensor at 49º



Random Three:
1 Sound sensor at 195º

Figure 12: Paths taken by elites. The sensors that were used are shown (darkened and filled in for ease of identification) along with the angle they were placed relative to the robot base. The unused sensors all point north relative to the robot base.

In Random Two the relatively high deviations towards the end of the run (generation 122, 123) are due to dips in the fitness. The dips can be attributed to the randomness of the environment as described in Section 2. Due to this randomness, an individual that has performed well in one generation might not do so in the next if the evolved solution is not general enough. The randomness ensured that a solution which was not general enough would not succeed in the following generations. As the agent evolved, individuals that worked with the generality in the environment were generated. In most cases, as the as the population evolved over the number of generations the results became more consistent thus showing that the solutions were robust enough to work in this level of generality.

The paths taken by a sample of the elites during the final run are shown in Figure 12. The final agents that were obtained by this method could decipher relevant stimuli needed to complete a task without being redundant (hence more efficient), using a low number of sensors. It is important to note that the individual agent does not have information on its location in the environment or that of the target's location. It must find the target using characteristics of the environment that are helpful to find this target point, and try to minimize the number of steps used to get to this point.

The success of the agent was heavily dependant on the environment as some graphs show that the solution was more difficult than others. In some environments the path taken to reach the goal was far less than optimal especially in Single Left Ridge. This result was partly due to the agent's build which made the agent drift left during its straight walking gait. Yet most results showed fairly efficient paths taken to the target

point. This difference in path efficiency can be attributed to the agent not having information of the target's location, except though cues in the environment, hence the different configurations of the environment worked to its advantage or disadvantage. This dependence on environment configuration is most apparent in the differences between Single Left Ridge and Single Right Ridge, where we see that due to the ServoBot's tendency to head left, it takes a round-about path in the former environment, but the agent does markedly better in the latter environment.

In the case of Double Ridge, Random Two, and Random Three the paths taken were very close to the shortest paths obtainable, given the gaits of the ServoBot.

In the case of Central Mountain, Single Left Ridge, Single Right Ridge, Random Two, and Random Three the light sensor was not used at all, even though the target was in the area of the light and light is the most apparent way to find the target. However, this is not necessarily the best way to find the target. Using the light stimulus is redundant if the ServoBot could find the target without it and light sensors might not be the best way to avoid obstacles. The Central Mountain run shows this most explicitly. Here the light sensors were not used because it would cause a collision with the Central Mountain. The strategy that was evolved instead made use of its tactile sensor to navigate to the point which minimized the sensor usage and consistently allowed the agent to complete the task.

In terms of complexity of the controller, at most three sensors were used by the final agents for the environments. This corresponds to only three if…then statements being used, as opposed to the original 14. This is a significant difference especially since the controller used on the actual ServoBot is a Basic Stamp II with limited processing

speed and memory. Moreover, the reduction of the sensors used significantly reduces the number of pins used on the controller and decreases the power consumption of the robot.

4.1.5. Discussion of Results

The results of learning with a GA shows that evolving the sensor morphology of an agent is a suitable strategy for designing an agent for niche environments. The evolution also made the robot and controller efficient and pruned the number of rules necessary for completing the task.

The agent completed its task for every environment consistently at the end of the runs without having sensors that explicitly directed it to the target point. Instead it used the stimuli provided by the environment to navigate to this point. This shows that adaptable sensor morphology is beneficial for designing an agent to complete a given task for a given environment. The results essentially show evolved Braitenberg-like vehicles. As is the nature of such reactive systems, the behavior of such an agent is fairly simple but very efficient for simple tasks such as used in this paper. The study shows that the configuration of the environment is a very important factor in deciding which sensors and stimuli to utilize. An important factor in automatic design using evolutionary computation is the ability to decipher and utilize relevant information in the environment. This ability is highly dependent on the environment and task of the evolving agent. The final product of the GA was an agent that was a specific solution to a specific environment. The high priority given for efficiency ensured that the solution would not

be general and the final designs of the agents hinted that as an agent's efficiency increased, the generality of its navigation decreased.

The results also show that, given the limitations of the study so far, this strategy is not suitable to design an agent capable of completing its task in a highly general environment. The evolution of sensor morphology on the legged robot allows the robot to learn its environment and pick out cues needed for completing its task, hence when these cues are random (Random Environment) the success of the robot is also random and was an expected result

There are several shortcomings to evolving only the sensor morphology of the robot without explicitly co-evolving the controller. The agent always had the same heading and position at the start. This allowed the agent to learn an appropriate sequence of actions for the environment. This makes the solution specific to a starting position and heading in the specific environment. The successful agents estimated their own positions based on the distance traveled and its directional heading. The robot essentially developed a well known control method known as dead reckoning used in navigation of ships and aircraft. Dead reckoning estimates the current position of a vehicle based on the distance traveled from a previous position at a certain angle. While in this case the agent does not have a state in which it can store this information, the reactive controller performs dead reckoning by forcing the need for this internal representation on the environment. This also makes the agent highly dependent on the environment to complete given task.

Co-evolving sensor morphology and controller allows the controller to chose which actions to take given a sensor input. In the next section we theorize that using this technique we can randomize the start position and orientation to a certain extent thus decreasing the dependency on the environment and performing other strategies than simple dead reckoning.

4.2 Co-Evolution of Controller and Morphology, Non-line of Sight

In this experiment we wanted the robot to have a sensor morphology and controller that could not rely on its starting position or orientation to succeed. We show that by co-evolving sensor morphology and control we can design autonomous agents that can complete a given task efficiently using stimuli in the environment as cues, while having a level of generality in terms of its start position and orientation. To investigate this method we use a simulation of the robot operating in a real world environment.
In this paper we use a genetic algorithm (GA) to co-evolve the sensor morphology and controller for a simulated robot modeled after an actual legged robot. This is part of a larger study that investigates the use of a GA to evolve the morphology and a controller for actual robots. In this part of the study, we evolve the sensor morphology and controller to equip the robot to complete a given navigation task when it starts from a randomized start position and heading, in a wide variety of environments.

The parameters that are evolved in this paper are the heading angle of the sensors, which sensors to keep running during the length of the run, the range of the light sensors, and the rules of the robot controller. The heading of the sensor can be rotated 360 degrees; all of the sensors can be turned on or off. The GA has 16 rules from which to select the controller for each sensor, and each light sensor has 32 choices of range. The position of the sensors on the base are fixed as shown in Figure 7 and do not change.

4.2.1 Chromosome

The chromosome used was 212 bits long. It was divided into 1 set of 12 bits, 12 sets of 9 bits, 12 sets of 5 bits and 13 sets of 4 bits. The first 12 bits represented which sensors to put on or off, the next 108 bits represent the angles at which each of the 12 sensors are to be placed onboard the robot base, 40 bits represent the range of the light sensors; (The 8 light sensors used 5 bits each and the last 52 bits represented Gaits for each of the 13 rules.

4.2.2. Genetic Operator

During evolution each set of bits underwent a single point crossover using stochastic (roulette wheel) selection of the parents. The mutation rate for the set of rules was set at 1%.. For the other parameters, if the goal was reached by any individual in the generation the mutation rate was set at 1%; otherwise the mutation was set at 6%.

```
If (Agent Reached Target Point)
        Fitness =
                Number of Sensors Turned Off * 50+
                50 * (Total Time –Time Taken to Achieve the Goal)+
                Goal Bonus
Else
        Fitness =
                Distance from Goal to Final End Point * 20
```

Fig. 11. The Fitness Function

The fitness function (Figure 13) is dependant on the success of the agent. If the agent is successful in finding the target, the fitness is based on the number of sensors it has off and the amount of time it took the agent to get to the target. If the agent is unsuccessful in reaching the target, the fitness is dependent on how far away the robot is from the goal. In the first case, to achieve the maximum fitness the robot has to have all its sensors off and reach the goal without any time being used, which is an impossible scenario in the simulation. A successful agent has the theoretical maximum fitness of 15600 while an unsuccessful agent has a maximum fitness of only 8480.

4.2.3 Test

We ran 256 individuals for 512 generations for each environment. This was repeated 5 times with random starting populations to check for consistency of the results. Each individual had 3 chances to achieve its goal. Each time the individual was placed at a random heading and randomly positioned within +/- 10 of (40,150). The individual with the highest fitness was added to the next generation without change.

The agent was given 200 time intervals to complete the task. A time interval is the time it takes a robot to complete one step. A step starts with the legs in a ready to step position (right front and back legs and left middle leg forward; remaining legs back) and returns to this position after a full step cycle. This time is the same for all 16 step cycles. The time intervals continue to elapse even if the robot collides with an obstacle and is stuck. A collision does not stop a test run. The agent can get out of a collision since the agent can work its way free of the obstacle due to its non-deterministic gait. The run is stopped if the target position is reached or when the agent runs out of time (200 time intervals).

4.2.4 Results

The results of the 5 runs on each environment are summarized in Figure 14. These graphs show the average and best individual of the population, over 5 runs, for each generation. The consistency of the results over the 5 runs is represented by the error bars (grey) in the graphs which are the standard deviation over the runs.

In all of the niche environments, except for Random Two, the GA evolved effective solutions and improved the fitness of the agents. Each of the environments had a different growth rate and became stable after a different number of generations. This shows that success and control learning is highly dependent on the environment in which the agent operates. In Random Two, the agents failed to find the Target point over 512 generations. The fitness of the average individual in this case is approximately 27.29%, while the average fitness of the elites is 43.6%. In the other environments the final

obtained fitness of the average and the elites obtained is markedly higher than that of Random Two.

Figure 15 is a sample from each of the environment types. Observations of the simulated robots in action revealed some of the learned strategies. The evolved strategies were a combination of wall-following and a more direct approach to finding the source of the UV light, hence the target point. Wherever there was a clear path to the Target Point the GA evolved agents that took the direct approach. This can be seen in Double Ridge and Random Three. In both cases, the main sensor is the UV light sensor which is mounted heading forward. Random Three also evolved a tactile sensor on its left flank to take into consideration any change in position of obstacles to its left over the generations. The agents in Central Mountain, Single Left Ridge and Single Right Ridge used a combination of wall following and UV source location. In all three cases the agent used the UV sensor to direct it initially in an appropriate heading. In Central Mountain and Single Left Ridge, the agent then used a wall following strategy to find the Target Point. In Single Right Ridge, the agent used the tactile sensor only to get out of the narrow gap between the wall and the obstacle. It also evolved a tactile sensor pointing inwards so that if it's right turn out of the gap was too close to the obstacle it could detect it and turn away. This is important since the solid obstacles were moved for each generation. The agent then used its UV sensor to find the Target Source.

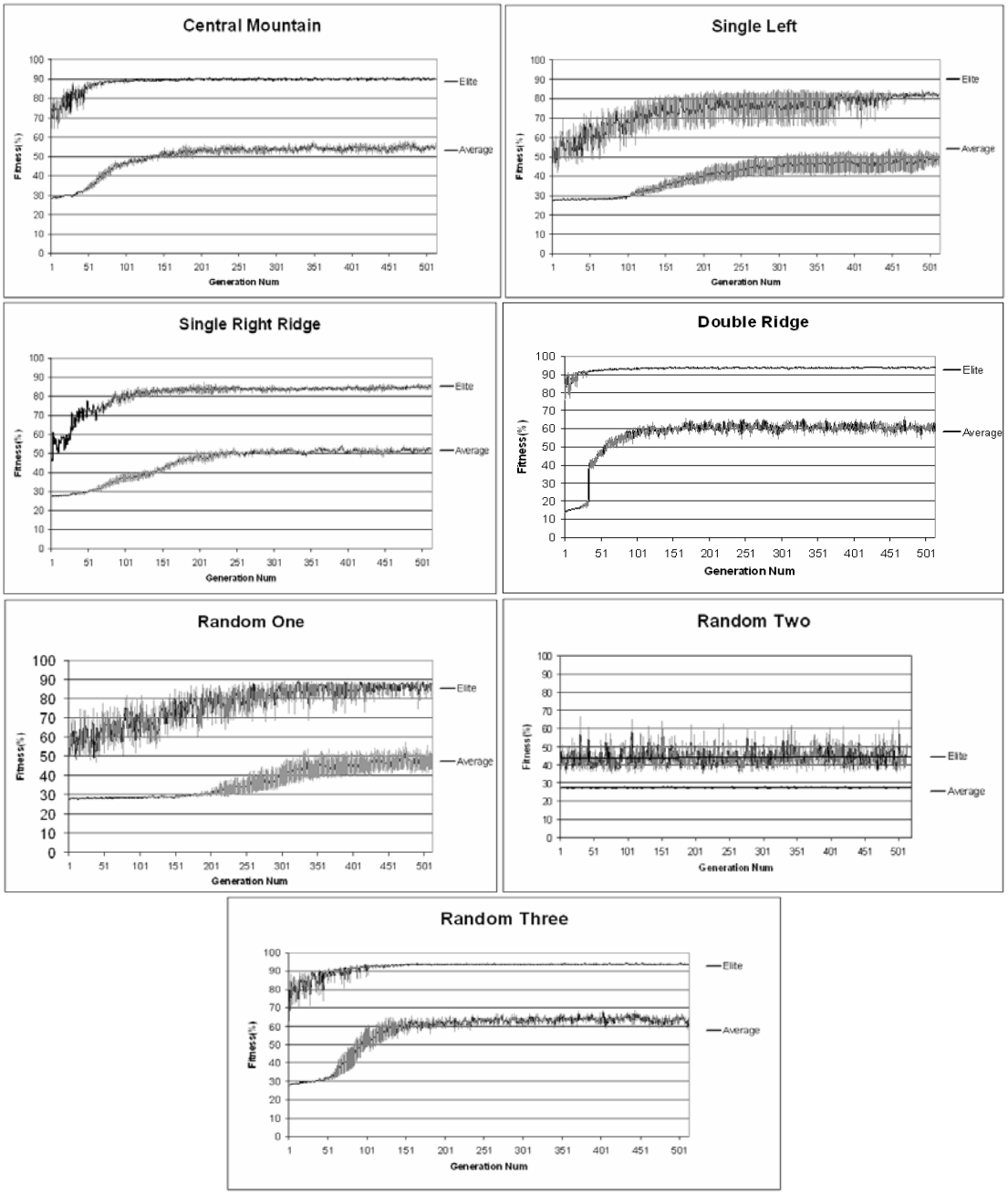
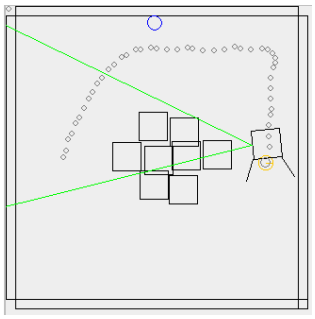
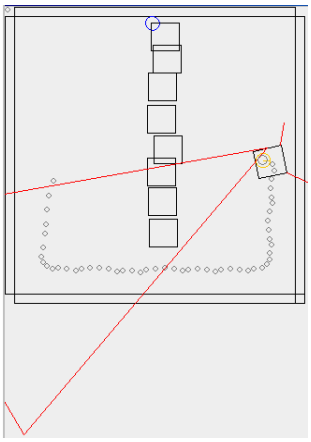
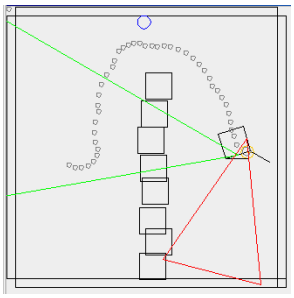


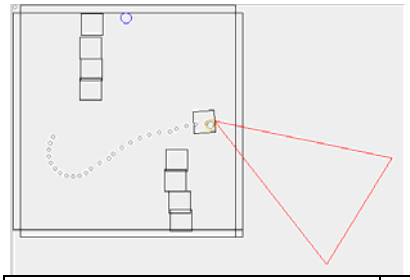
Fig. 12. Elite and Average Fitness growth averaged over 5 Runs

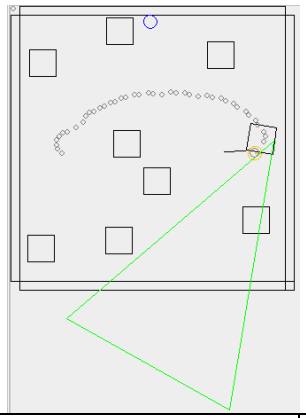


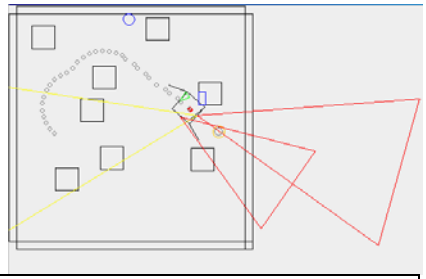




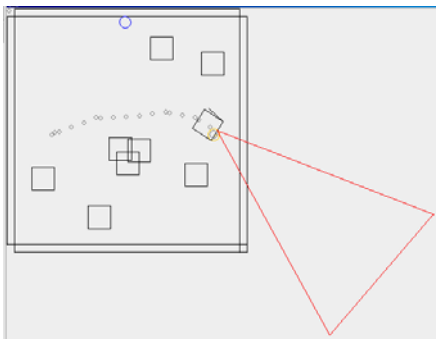
Central Mountain:	Single Left Ridge:	Single Right Ridge:
2 Tactile sensors at 22º and 333º 1 UV sensor at 101º; Length 378 cm 4 Gaits and Rules Used	2 Tactile sensors at 21º and 128º 1 UV sensor at 252º; Length 406cm 4 Gaits and Rules Used	2 Tactile sensors at 287º and 319º 2 UV sensors at 117º and 32º; Length 378cm, 204cm 5 Gaits and Rules Used







Double Ridge:	Random One:	Random Two (UNSUCCESSFUL):
1 Light Sensor Used at 22lº; Length 140cm 2 Gaits and Rules Used	1 Tactile Sensor at 78º 1 UV sensor at 21º; Length 322 cm 3 Gaits and Rules Used	3 Tactile sensors at 18º, 457º, 157º 1IR sensor at 487º; Length 392cm 3 UV sensors at 353º, 334º, 282º; Length 182cm, 294cm, 14cm 8 Gaits and Rules Used



Random Three:
1 Tactile Sensor at 187º 1 UV sensor at 9º; Length 308 cm 3 Gaits and Rules Used

Fig. 13. Paths Taken and Sensors Used by the Robot

In the case of Random Two, the agent could not consistently find the Target Point. This is apparent since the average fitness of the population does not increase nor does the fitness of the elites over a given run. The reason for this failure is the difficulty of this particular environment. The Random Two environment has a configuration of solid obstacles that does not often allow the agent to pass between its gaps. Where there are sufficient gaps, the randomness associated with the placement of the obstacles results in random location of the gaps. The inconsistency of this situation varied the fitness of the individuals sufficiently that there was little or no improvement in acquiring the target over time. This lack of improvement under such an inconsistent situation can be associated with the purely reactive nature of the controller which does not allow for planning of any kind.

It was surprising to find that none of the evolved agents used the IR light as a possible way point to finding its way to the UV light and hence the Target. The penalty of having a sensor on was only 2% of the total fitness, yet none of the successful agents

utilized this stimulus. The IR source's placement was intended to be advantageous. The agent could have used the UV light to navigate around obstacles then get to the target in each environment, especially Single Left Ridge and Single Right Ridge. A closer look at the environment yields an answer to the agent's lack of IR sensor usage. The obstacles are low enough that light can be detected over them. A look at Single Right Ridge shows that the robot can circle in place until its right light sensor detects the target. At this point it will be facing north. It can track north until it detects a wall, moves east and eventually south east to turn away from the wall and will detect the target UV source with its front and slightly right facing sensors.

4.2.5. Discussion of Results

The method used in this research to co-evolve the senor morphology and controller for a hexapod robot was successful in simulation. Sensor morphologies and controllers were evolved that were specialized to handle types of environments whereas randomizing the robot's start position and orientation showed that the result is a generalized solution for a given environment. In all but one of the test environments the evolved robots improved throughout the evolution and produced reasonable sensor configurations with and associated rules.

This co-evolution strategy designed robots and their controllers that were successful in achieving the target and pruning the unnecessary sensors so that it was done efficiently. This can be clearly seen in Figure 14 since almost all of the successful agents in the latter half of the evolution were among 80% to 90% fit.

The agents evolved were highly specialized in that they were not successful in the other environments. Their sensor configuration and control were extremely varied and unique for the different environments. This can be attributed to the reactive nature of the controller and a fitness function that rewarded efficiency in sensor usage. The GA evolved successful and in some cases familiar control strategies, like wall-following.

Compared to the only evolution of sensor morphology, co-evolving sensor morphology and control proved to be a successful strategy in moving towards a more generalized agent. The agent was not dependent on its start position and orientation. The agents evolved other navigation strategies such as wall following instead of simple dead reckoning.

One of the more interesting parts of the results was that the infra red (IR) stimulus was not used at all to successfully navigate to the target point. It was intended that this information would be useful. However the GA developed solutions that did not require the extra sensor type. The reason behind this is that the IR stimulus is redundant given the location of the target spot and the proximity of the UV stimulus to it. The need to be efficient in terms of number of sensors used caused the IR sensors to remain unused.

This design information saves significant time and money since these sensors would not have to be purchased or built for the robot. In this way this co-evolution strategy is beneficial before actual building as it allows pruning of unnecessary modules that would not improve the performance and make the robot more expensive than necessary. Moreover it also means that the sensors do not have to be UV or IR which are relatively expensive and time consuming to build than visible spectrum light sensors

To test how adaptable the system is to a change in the environment's stimuli the next "Line of Sight" experiment is performed. In this simulation we perform the same experiments but increase the obstacles height such that it cannot sense light behind them.

4.3 Co-Evolution of Controller and Morphology, Line of Sight

The line of sight tests were performed to assess the adaptability of the design process given a change in the environment. The previous experiment showed that the IR sensor was not used. It was reasoned that the IR stimulus was redundant due to the relatively important location of the UV sensor. To change this redundancy, the walls were increased in height so that a sensor could only detect a stimulus if it was in its "Line of Sight", thus increasing the complexity of the environment.

4.3.1 Tests

This experiment merely tests the adaptability of the co-evolution strategy and is not meant to be as exhaustive as the previous experiments. Thus the setup of the experiment including the genetic operators and fitness functions are unchanged. Moreover the Random One, Random Two and Random Three environments are not tested in this experiment as they are randomly generated at the start of an test and hence will not be comparable to the environments in the non-line of sight experiments. Hence there are four tests in this section

4.3.2 Results

The results in Figure 16 show the average of the elites over 5 runs. The error bars are the standard deviation, showing the consistency of the results over the 5 runs.

The fitness of the agents in each environment is comparable to the Non-line of sight experiments. This shows that the agents are just as effective in this slightly different environment and have been able to adapt to the changes forced on them.

The GA evolved effective solutions and improved the fitness of the agents in each tested environment. The growth rates of the environments were comparable to their non-line of sight counterparts with each environment having a different growth rate and stabilization generation.

In Figure 17 one can see an elite agent from each of the environments. The learnt strategies were similar to the non-line of sight results. In fact a side-by-side comparison shows marker similarities in the paths used to get to the goal. Yet in this test every agent used the IR and UV sensor.

In the case of Central Mountain the agent appears to have used a wall following strategy. Upon closer inspection we can see that it uses its IR sensor to find a wall then follow it till gets to the wall. This agent uses two IR sensors for different actions. One IR sensor (pointing north) helps the agent to initially find the wall. The other IR sensor (pointing south) is then used to keep the agent away from the IR source so that it can find the target point. The tactile sensors were used to perform wall following

In Double Ridge the agent starts off in a tight left circle. When the IR sensor triggers the gait changes to a straightened one till it hits the UV source. One the UV source is found the robot follows it to the target spot.

In Single Left Ridge the IR sensors are only used in case robot is facing the wrong way, if it is it turns around, if not then robot uses wall following to get to past the Ridge and uses its UV sensor and tactile sensor to guide it to the target.

Single Right Ridge used its IR sensor to go toward the IR source, its bump sensor to turn away from the wall and then its UV sensor to head towards the light.

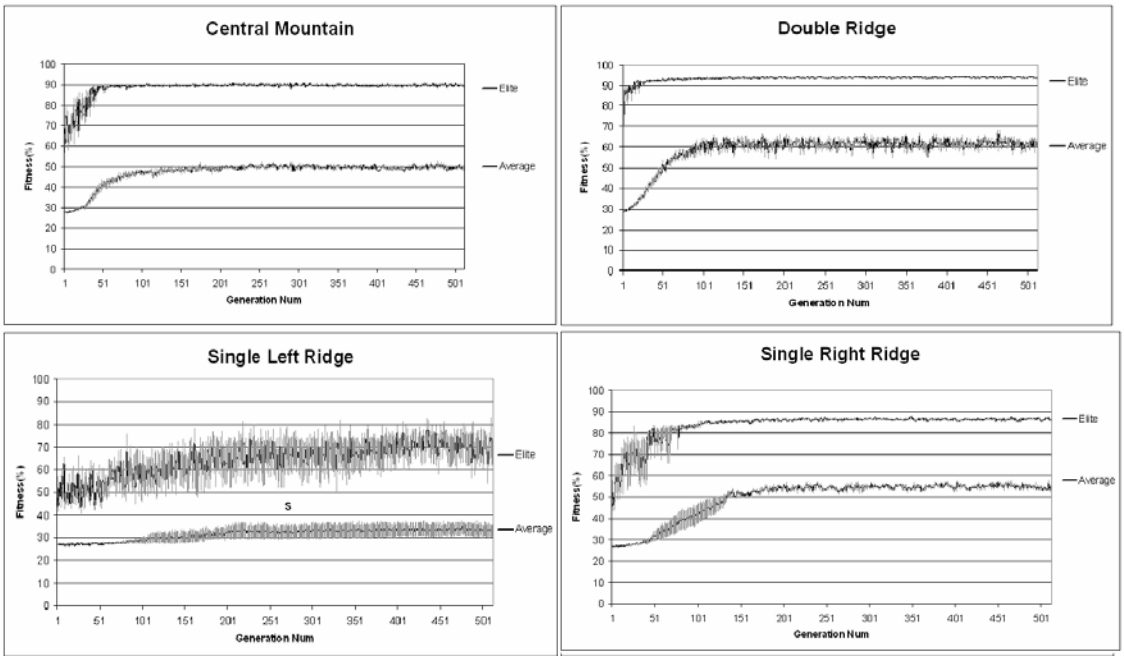


Fig. 16. Elite and Average Fitness growth averaged over 5 Runs

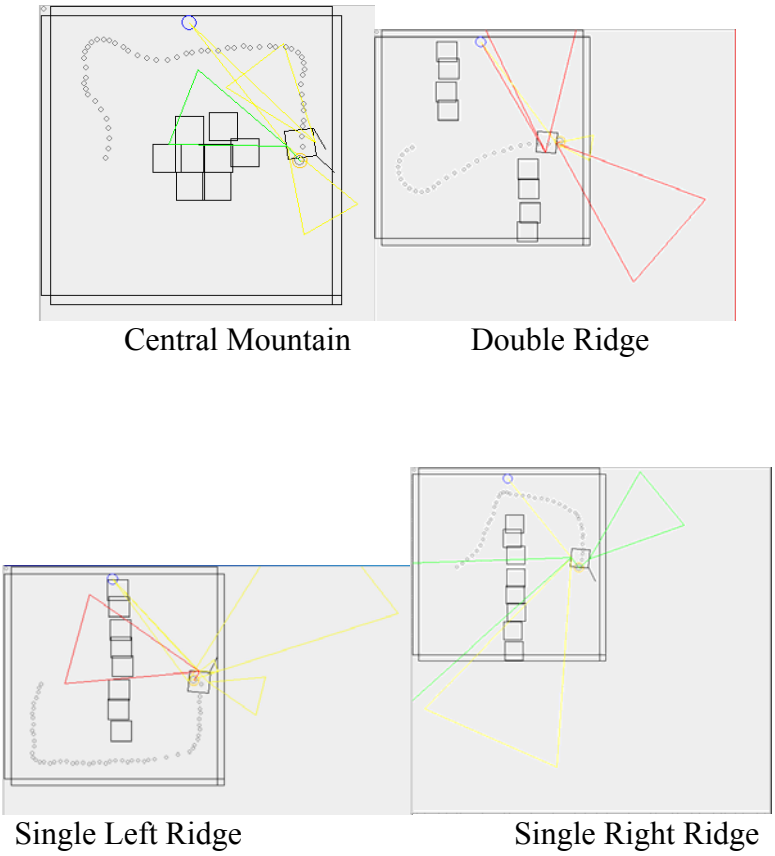


Fig. 17 Paths Taken and Sensors Used by the Robot

4.3.3 Discussion of results

The tests show that same co-evolutionary strategy was successful in adapting to the changed environment. The fitness of the individuals were comparable to the non-line of sight tests, as were the efficiency of the paths taken

The strategy was not as successful in generating high efficiency in terms of sensor usage in this environment as it was in the non-line of sight experiment. This sheds light on the reactive nature of the agent; it shows that, the greater the complexity of the environment the greater the number of sensors needed to solve the task. Moreover, since

each sensor is mapped to an action, the more complex a task the more complex the actions need to be to complete it.

4.4 Tests on Robot

The co-evolution strategy showed its success in navigating though an environment in simulation. The simulation though is an extremely ideal world: no wear and tear occurs, there is little noise involved (albeit, it could be modeled but increasing the task of simulation greatly) and the stimuli and the sensors are ideal.

The real world environment is starkly different. Building is a painstaking process involving wear and tear, non ideal environments, unforeseen noise sources, imperfect stimuli and unreliable sensors.

A successful design needs to be robust enough to succeed in the real world given these problems. Hence the automatically generated sensor morphology and controller are tested on an actual robot in the Colony Robotics Space provided at the Connecticut College Robotics Laboratory.

The colony space is shown in Fig in the Random Three Environment setup

Fig 18 The Robot Colony in Random Two Environment Configuration with the Servobot and Light Stimulus

The differences in the simulated environment and the real environment are stark. The light source is not ideal, the surface of the colony is carpeted making the gaits of the Servobot uncertain and the tactile sensors are prone to noise due to the walking motion of the robot.

The designs created automatically by the co-evolution of sensor morphology and controller need to be robust enough to overcome these problems and still successfully navigate through the environments.

4.4.1 Tests

The tests described in this section test the design of the Non-line of Sight agents in the Robot Colony. Each evolved sensor morphology along with it controller was transferred to a real robot as can be seen from examples in Figure 18. The agent was then

placed in the robot colony to commence the navigation task. Random Two, the failed strategy in the Non-line of Sight experiments, was not tested as it had failed in the simulation itself proving its infeasibility.

Thus the environments that were tested are Central Mountain, Single Left Ridge, Single Right Ridge, Double Ridge, Random One and Random Two.

In the simulation the agent was given a maximum of 200 time intervals to complete the task. A time interval was defined as the time it takes the Servobot to complete one step. In the test this measure was not needed, instead the Servobot had maximum of 200 steps to complete the task. As in the simulation the obstacle placement in the environment had a degree of randomness (+/- 10cm) associated with it as did the start heading and location. Lastly, just as in the simulation, a collision does not stop a test run, since the Servobot can get out of a collision since the agent can work its way free of the obstacle due to its non-deterministic gait.

4.4.2 Results

All designs, except the Single Left Ridge, automatically generated using co-evolution of sensor morphology and control proved successful in completing the navigation task. When the paths are compared to the paths taken in simulation the results are very similar as seen in the time lapse photos of the Central Mountain and its simulated counterpart in Figure 19.

Fig 19 Comparison of Actual Performance to Simulated Path

The results for Double Ridge, Single Right Ridge, Random One and Random Three are in the CD-ROM provided which is part of this thesis. This includes photos and movies of the successful tests on the actual robot.

In all niche environments except for Single Left Ridge the GA evolved successful designs that were robust enough to transfer to the real world. The agent's failure to complete the task in the Single Left Ridge Environment showed that the success was also dependent on the simplicity of the design. In the case of Single Left Ridge the final design evolved required 4 sensors and 5 rules. This was the most number of sensors required by any agent all of the niche environments. The dependence of the reactive controller on the reliability of the sensors means that the actions of the agent will be increasingly non-deterministic as the sensor input increases. Moreover, as the number of sensor inputs increase, the number of noisy actions increase making it difficult for the robot to navigate through the given environment

The other designs, although subject to the similar noise in the environment were simpler in terms of controller complexity (i.e. number of rules used) and sensors used.

4.4.3 Discussion of Results

The test results show that the co-evolution of sensor morphology and controller for the ServoBot is a viable option for designing robust system to perform specific tasks in niche environments.

Even though the sensor morphologies and controllers were evolved in a specific environment with a comparatively very low noise level, the designs produced were robust enough to perform well in the highly noisy real world environment. This particularly of note since mechanical noise and environmental noise were not factored into the GA during the co-evolution process. As mentioned above, efficiency in terms of sensors used seemed to be a large factor in allowing the agent design to be successful in the real world. The other major factor in the good performance of the robot is the randomness introduced into the simulation during the GA learning. The agents designed in an environment of uncertainty had to be more robust and thus were able to successfully complete their tasks in the highly noisy real world environment.

Just as in the simulation the agents were specialized for their environments and could not navigate other environments due to their reactive controllers.

5. Conclusions

This thesis was a study of the use of the genetic algorithm to co-evolve sensor morphology and control for a legged robot. The results from the simulation and tests show that this approach to design provides:

1. An automated approach to the design process.
2. Rapid development of many designs while taking into consideration dependencies between many variables.

3. A agent that can
    a. Exploit the environment to complete its task.
    b. Decipher relevant information to complete its task and discard irrelevant information not needed to complete a task.
4. A effective design that uses a low level of input while completing a task in an efficient manner.
5. Learned morphologies and controllers can be transferred to the real world and successfully complete the task despite noisy sensor data.

The automated approach to the design process is a new and effective method for rapid development. The GA, due to its design and dependency on the fitness function, inherently takes into consideration many tens or hundreds of variables whose dependencies do not have to be explicitly defined. Since these dependencies are inherent to evaluation of the fitness function in a GA, they are implicitly taken into consideration and adjusted. Due to this implicit adjustment, and degree of randomness added to the system allows the GA to design agents that are robust enough to be successful in such an environment (although this environment cannot be completely random as shown in the experiment in section 4.1).

An agent that is successful in such a degree of randomness has a high likelihood of being effective in the real world environment which is inherently unpredictable.

In terms of designing autonomous agents, the co-evolutionary approach has shown impressive results in terms of its usage of the environment to complete a given task. The efficiency and the successful functioning of the design give the agent a

"mindless intelligence", in which the agent itself does not cognitively discard irrelevant information but rather the evolutionary process creates an agent that exhibits a degree of understanding of its environment and its relationship to it and the task at hand.

6. Future Work

This evolutionary approach creates a successful but simple agent that cannot think or plan or show complex behavior. This is primarily due to the reactive controller of the robot. For autonomous agents to be consistently successful in real world applications the intelligent agent also needs cognitive and planning abilities.

In future work it will be necessary to include a controller that takes over from the reactive controller if a failure occurs due to noisy inputs. This controller could be behavior based, in which the evolved reactive controller makes up the lowest behavior in the agent. If this basic behavior fails, other higher level planner or navigation methods can take over help solve the problem. These higher level behaviors can also evolved using previously evolved reactive behaviors as a basis in an incremental evolutionary approach

REFERENCES

[1] K. Balakrisha and V. Honavar, "On Sensor Evolution in Robotics," Proceedings of the First Annual Conference on Genetic Programming, Stanford University, USA, 1996.

[2] M. Fend, M. Yokoi, R. Pfeifer, "Optimal Morphology of a Biologically-Inspired Whisker Array on an Obstacle-Avoiding Robot," Proceedings of the 7th European Conference on Artificial Life, ECAL Dortmund, Germany, 2003.

[3] L. Lichtensteiger, "Towards Optimal Sensor Morphology for Specific Tasks: Evolution of an Artificial Compound Eye for Estimating Time to Contact," In Sensor Fusion and Decentralized Control in Robotic Systems III, Proceedings of SPIE Vol. 4196, pp. 138-146, 2000.

[4] P. Funes and J. Pollack, "Computer Evolution of Buildable Objects" Proceedings of the Fourth European Conference on Artificial Life, Cambridge, USA, 1997.

[5] G. Hornby, H. Lipson, and J. Pollack, "Evolution of Generative Design Systems for Modular Physical Robots," Proceedings of the IEEE International Conference on Robotics and Automation, pages 4146–4151, 2001.

[6] D. Marbach and A.J. Ijspeert. "Co-evolution of configuration and Control for Homogenous Modular Robots", In F. Groen et al., editor, Proceedings of the Eighth Conference on Intelligent Autonomous Systems (IAS8), pp.712-719. IOS Press, 2004.

[7] J. Pollack, H. Lipson, G. Hornby, and P. Funes, "Three Generations of Automatically Designed Robots," *Artificial Life* Vol. 7, Issue 3, pages 215 – 223, 2001.

[8] W.Lee., J. Hallam, H.H. Lund.: A hybrid gp/ga approach for co-evolving controllers and robot bodies to achieve fitness-specifed tasks. In: Proceedings of IEEE 3rd International Conference on Evolutionary Computation, IEEE Press (1996) 384 389: lee96hybrid.pdf

[9] H.H. Lund , J. Hallam, Lee. W, "Evolving Robot Morphology" IEEE International Conference on Evolutionary Computation, Indianapolis, USA1997.

[10] H. Mark, D. Polani, T. Uthmann. "A Framework for sensor evolution on a population of Braitenberg vehicle-like agents" In Christoph Adami, Richard K. Belew, Hiroaki Kitano, and Charles E. Taylor, editors, Artificial Life VI, pages 428--432, Cambridge, Mass, 1998. MIT Press.

[11] M. Bugajska, and A. Schultz, "Coevolution of Form and Function in the Design of Micro Air Vehicles," Proceedings of the NASA/DoD Conference on Evolvable Hardware, Alexandria, VA, USA, 2002.

[12] M. Bugajska. and A. Schultz "Co-Evolution of Form and Function in the Design of Autonomous Agents: Micro Air Vehicle Project." GECCO-2000 Workshop on Evolution of Sensors in Nature, Hardware, and Simulation, Las Vegas, NV; July 8, 2000

[13] M. Bugajska. and A. Schultz "Anytime Co-evolution of Form and Function" In Proceedings of 2003 Congress on Evolutionary Computaton (CEC-2003), Canberra, Australia, December 8 - 12, 2003

[14] C. Mautner and R. Belew, "Evolving Robot Morphology and Control," Proceedings of Artificial Life and Robotics, Oita, 1999.

[15] H.H. Lund "Co-evolving Controller Morphology with LEGO robots" In Hara and Pfeifer (eds.) Morpho-functional Machines, Springer-Verlag, Heidelberg.

[16] K. Sims Evolving "3D Morphology and Behavior by Competition" 1994. Artificial Life Vol 1, Issue 4, pages 353-372. Cambridge, Mass, 1994 MIT press.

[17] R. Pfeifer, "Building 'Fungus Eaters': Design Principles of Autonomous Agents," Proceedings of the Forth International Conference on Simulation of Adaptive Behavior, Cambridge, USA, 1996.

[18] M. Atkins and P. Cohen, "Monitoring Strategies of Embedded Agents: Experiments and Analysis," Proceedings from Animals to Animats 4, Cambridge, USA, 1996.

[20] G. Parker, "Evolving Cyclic Control for a Hexapod Performing Area Coverage," Proceedings of the IEEE International Symposium on Computational Intelligence in Robotics and Automation, Banff, Canada, 2001.

[20] V. Braitenberg, Vehicles, Experiments is Synthetic Psychology, The MIT Press, Cambridge, Mass., 1984.