Software evolution of a hexapod robot walking gait

Jonathan Currie*, Mark Beckerleg and John Collins

Engineering Research Institute,
AUT University,
34 St Paul Street,
Auckland 1001, New Zealand
E-mail: jonathan.currie@aut.ac.nz
E-mail: Mark.beckerleg@aut.ac.nz
E-mail: John.Collins@aut.ac.nz
*Corresponding author

Abstract: This paper describes the process of evolving a hexapod robot walking gait within a simulated software environment. Initially, a 3D mathematical model of the robot was created using MATLAB, simulating full motion of each robot's six legs. Each leg has three degrees of freedom (DOF), allowing the robot to move in both lateral and rotational directions. The simulation allows the robot's movements to be estimated, and was used with a genetic algorithm (GA) to evolve walking gaits for the robot. The walking gait is described by a chromosome which is an 18 × 9 look up table (LUT) that lists the angular position of all 18 servo motors over a sequence of 9 discrete static positions. Each position in the LUT has 20 discrete states, allowing a flexible range of achievable motions, while maintaining sensible evaluation limits. Successful evolution of these gaits was performed within 700 generations.

Keywords: robotics; hexapod; gait; genetic algorithm; evolution; modelling.

Reference to this paper should be made as follows: Currie, J., Beckerleg, M. and Collins, J. (2010) 'Software evolution of a hexapod robot walking gait', *Int. J. Intelligent Systems Technologies and Applications*, Vol. 8, Nos. 1-4, pp.382–394.

Biographical notes: Jonathan Currie received his BE (Hons) in Electrical and Electronic Engineering in 2009 from AUT University in New Zealand. Now, he is pursuing a PhD in real-time optimisation of refinery processes, with research focus on optimisation strategies and predictive control. Prior to obtaining his degree, he won the New Zealand mechatronics competition and subsequently represented New Zealand at the World Skills mechatronics competition in Finland, 2005.

Mark Beckerleg received his BE (Hons) in Electrical and Electronic Engineering in 1986 from the University of Auckland in New Zealand. He has worked in several fields, ranging from television and oil exploration to research and development of embedded systems. Currently, he is a Senior Lecturer at the Auckland University of Technology, with expertise in embedded systems and HDL programming. His research interests are in evolvable hardware and evolvable software for robotic control and he is completing a PhD in this field.

John Collins received his BSc (Hons) in Physics and Mathematics and PhD in Electronic Engineering from the University of Auckland. He is currently a

senior Lecturer in the School of Engineering at the Auckland University of Technology. He has extensive industrial experience in the development of embedded systems and has worked as a consultant on a number of embedded systems projects. His research interests include robotics and embedded systems design and verification.

1 Introduction

The aim of this research is to evolve a walking gait for a simulated hexapod robot, such that it can walk forward in a straight line. Previous research has been conducted in this area for typically rectangular hexapod robots (Dürr et al., 2004; Mazzapioda and Nolfi, 2006; Parker, 2000), whereas this research will present evolution of a walking gait for a circular hexapod robot, with legs spaced 60° apart.

Designing the walking gait of a multi-legged robot is a complex multi-dimensional control problem (Kohl and Stone, 2004; Pratihar, 2003), with factors including centre of gravity and surface friction to be accounted for. Designing a gait for a hexapod robot requires the coordination and simultaneous movement of all six legs, within a cycle of leg activations (Parker et al., 1997).

The manual configuration of this walking gait is a time consuming process, requiring specialist knowledge of the robot architecture, and can often result in suboptimal performance. An example is the Sony AIBO robot, in which evolved or learned gaits consistently outperformed hand-tuned gaits, in terms of gait speed. The robots were used within Robot Soccer tournaments, and thus speed was one of the deciding factors of the victorious team (Chernova and Veloso, 2004).

Within this research, a genetic algorithm (GA) has been chosen to evolve the walking gait, based on the suitability of a GA for developing locomotion mechanisms (Goldberg, 1989). This is a reasonable assumption based on Evolution Theory (Darwin, 1859), which stated that locomotion mechanisms of many life forms resulted from the process of natural evolution.

A GA follows the principles of natural evolution (Alba and Chicano, 2006; Darwin, 1859). Initially a random population is created, with each individual known as a 'chromosome', and representing a possible walking gait of the robot. The population is then expanded by generating 'offspring', chromosomes which are created from the variables within two parents' look up table (LUT) (genes). Genetic operators perform this procreation process, by combining the genes of each parent into an offspring's LUT. A selection process known as 'survival of the fittest' now places the offspring in direct competition with their parents, by placing the chromosomes into the mathematical model simulation, and returning a 'fitness value'. The best chromosomes are kept, and the worst discarded, and the process continues iteratively until a suitable table of solutions is found (Braunl, 2006).

The robot has been modelled within Matlab, in order to reduce the evaluation period of each GA generated solution. The simulated robot is based on a physical robot constructed from a Lynxmotion kitset as shown in Figure 1, with the electronics purpose built for this research. The robot has six legs, spaced evenly around a circular body. Each leg has three DOF, represented as a pelvic joint, a hip joint and a knee joint.

The objective of this paper is to describe the mathematical model created to simulate the physical robot, and detail the GA used to evolve walking gaits.

Figure 1 Lynxmotion Hexapod Robot BH3-R (see online version for colours)

2 Mathematical model

2.1 Static mathematical model

This is the model built to represent the physical position and orientation of the robot based on the angles of all 18 servo motors. Static refers to modelling the robot in only one position, with no movement taken into account.

Initially, there are two assumptions, this model relies on:

1 the robot cannot balance on two legs, and therefore there must be always at least three legs on the ground

2 the centre of the mass of the robot is always at the origin of the robot's body.

Each leg is divided into four coordinate points, as shown in Figure 2, and these are defined by the geometry of the robot and the angles of the associated servo motors. The calculations to solve the coordinate points take the following form:

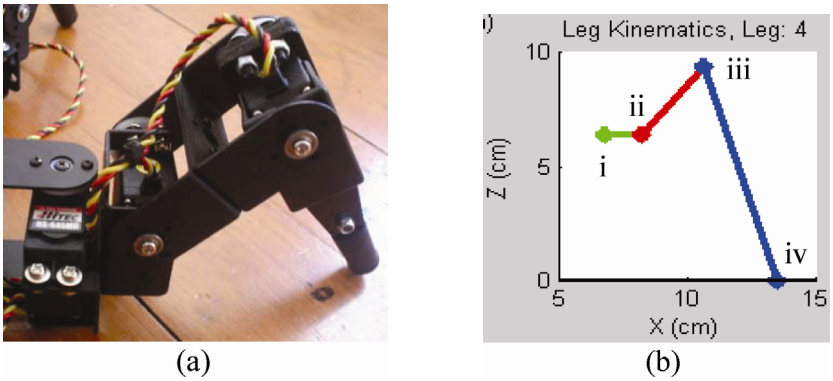
$$X(\text{Pelvis}) = \text{Origin}\,X + \text{Radius} \times \cos(\delta) \tag{1}$$

$$X(\text{Hip}) = X(\text{Pelvis}) + \text{Pelvis} \times \cos(\gamma + \delta) \tag{2}$$

where OriginX is the X coordinate of the centre of the robot body, Radius is the physical radius of the robot's centre body, while δ describes the angle where each leg connects to the robot body, relative to the front of the robot. Pelvis is the physical length of the pelvis of the robot and γ is the angle of the pelvis servo motor.

The equations for the knee and foot follow the same conventions, including the angles of their respective servo motors and body parts.

Figure 2 (a) Actual leg of robot (b) Model leg of robot showing the Pelvis Servo (i), Pelvis stick (green line), Hip Servo (ii), Femur (red line), Knee Servo (iii), Tibia (blue line) and Foot (iv) (see online version for colours)



(a) (b)

This initial part of the model solves for the coordinates of the robot as if the body was sitting on the ground, but the feet are allowed to be at any height (including under the ground). This allows the robot's tilt (φ) and direction of tilt (θ) to be evaluated, based on the three feet that would be supporting the robot. To find the three supporting feet, the coordinates of the three lowest feet are checked to see whether they contain the centre of mass of the robot. Based on assumption (2), it would therefore mean the triangle containing the three coordinate points of each foot, would also contain the centre of the robot. In order to determine this, a 'Point in Triangle' test was required. This is done by computing the barycentric (Fauvel, 1993) coordinates (u, v) of the triangle, and testing for the following:

$$u \geq 0, v \geq 0, u + v \leq 1 \tag{3}$$

When Equation (3) is valid, the centre of the mass is found to be within this triangle. If the lowest three feet do not contain the centre of mass (for example where the lowest three feet are on the same side of the robot), the calculation is iterated using substitute feet until a valid solution is found.

Once the triangle containing the three supporting feet had been established, the equation of plane containing this triangle was found by solving for the normal to this plane.

Solving a Spherical to Cartesian transformation on the normal vector resulted in the φ and θ of the robot body. The next step was to rotate and lift the robot so the plane connecting the supporting feet was flat, at $Z = 0$. This was achieved by defining a new axis system, defined as three unit vectors:

$$N = \frac{\text{Normal vector}}{\text{Rho}} \tag{4}$$

$$R = \frac{(0,0,1)\,\text{cross normal vector}}{\cos(\text{Phi})} \tag{5}$$

$$S = N \,\text{cross}\, R \tag{6}$$

where N is a unit vector pointing in the direction of the normal, R is a unit vector orthogonal to the normal, and orthogonal to the direction the plane must be rotated to be flat, and S is a unit vector orthogonal to both N and R. This new axis system is individual to the tilt and direction of the robot, and allows the existing coordinates to be translated into a normal Cartesian system, using the following:

$$q = \text{Any}(X, Y, Z) \text{ of the robot} \tag{7}$$

$$r = q \text{ dot } R \tag{8}$$

$$s = q \text{ dot } S \tag{9}$$

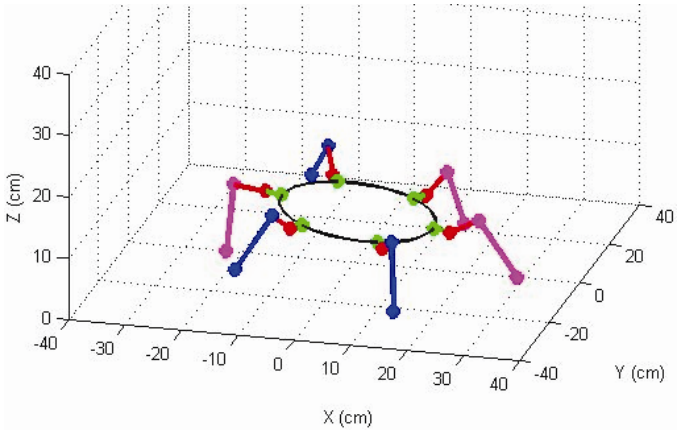
$$t = q \text{ dot } N \tag{10}$$

$$\text{Translated } X = (r \times R) + (s \times -\cos(\theta)) \tag{11}$$

Equations (8)–(10) take the magnitude of the (X, Y, Z) components in relation to the new axis system, while Equation (11) translates the X coordinate to a new position, as if the robot is now standing on the ground. Similar equations translate the Y and Z components. Note that the translation is not performed on the X and Y components if $\varphi = 0°$ (the robot is flat), and instead only on the Z component.

At this stage the robot is now standing on the ground, with the three supporting feet at $Z = 0$, with all coordinate points reflecting the tilt and direction of tilt of the robot body. It was found that due to tilting the robot that one or more of the feet not supporting the robot could end up with $Z < 0$. In order to compensate for this, the model iterates until all feet are at $Z \geq 0$. The resulting model is shown in Figure 3.

Figure 3 Static model of robot, with magenta legs indicating supporting feet (see online version for colours)



2.2 Dynamic mathematical model

To simulate the robot fully, a dynamic model was built around the static model, which would take consecutive static positions of the robot's legs, and make a prediction on the resulting movement. Movement was characterised by the following measurable values:

1 Lateral movement, with respect to X and Y.

2 Heading, the direction the robot is facing.

3 Height, the height of the centre of the robot.

An important assumption was also made:

3 A minimum of two legs must move if the body of the robot is to move (laterally or heading).

Assumption (3) is made to take into account the forces of friction on the feet. This means if one foot moves while the others do not, it is assumed that this foot will slip across the ground, rather than move the robot. In contrast, it is expected that if two or more feet move, then the body of the robot will move, and the feet will stay approximately static.

The model works by assessing two consecutive static positions of the robot. To increase the accuracy of the model, the two consecutive positions are modelled in the same way the controller software on the physical robot can modify the servo angles, by the maximum and minimum change of 4.5°.

An incremental routine is setup on the robot controller, allowing the angle change of 4.5° per increment, per servo, iterating until all servo motors have reached their final desired positions. This same incremental routine is thus written into the dynamic model.

The movement of the robot is calculated by each increment, which can extend to over 150 increments for a nine static position LUT. Movement is calculated by the following process:

1 The first static position is programmed so that all servo motors are at 0°, such that the robot stands like in Figure 1, facing forward.

2 For each subsequent static position, defined by the increment routine, the associated servo angles are placed into the static model to determine the coordinates of all parts of each leg and the feet supporting the robot. At this point, the robot height can also be calculated from Equation (4) and:

$$\text{Height} = \text{BaseHeight} + |(X, Y, Z) \text{ dot } N| \tag{12}$$

where (X, Y, Z) are the coordinate points of any supporting foot and BaseHeight = 5.1 cm, and this is also the height of the centre structure of the robot body, as depicted in Figure 1.

3 The next step is to calculate the lateral movement of the robot. If there are feet which have supported the robot over the two consecutive static positions, and the feet have moved, then the robot may have moved as well. To calculate this, the supporting feet displacement vectors are added, to result in a vector which describes the net movement of the robot.

4 Next, the change in heading of the robot (d$\emptyset$) is performed, by calculating the angle between two vectors:

$$\text{Vec}A1 = \tan^{-1}\left[\frac{\text{Position 1 Foot}(Y) - \text{Centre}(Y)}{\text{Position 1 Foot}(X) - \text{Centre}(X)}\right] \tag{13}$$

$$\text{Vec}A2 = \tan^{-1}\left[\frac{\text{Position 2 Foot}(Y) - \text{Centre}(Y)}{\text{Position 2 Foot}(X) - \text{Centre}(X)}\right] \tag{14}$$

$$\text{d}\emptyset = \text{Vec } A1 - \text{Vec } A2 \tag{15}$$

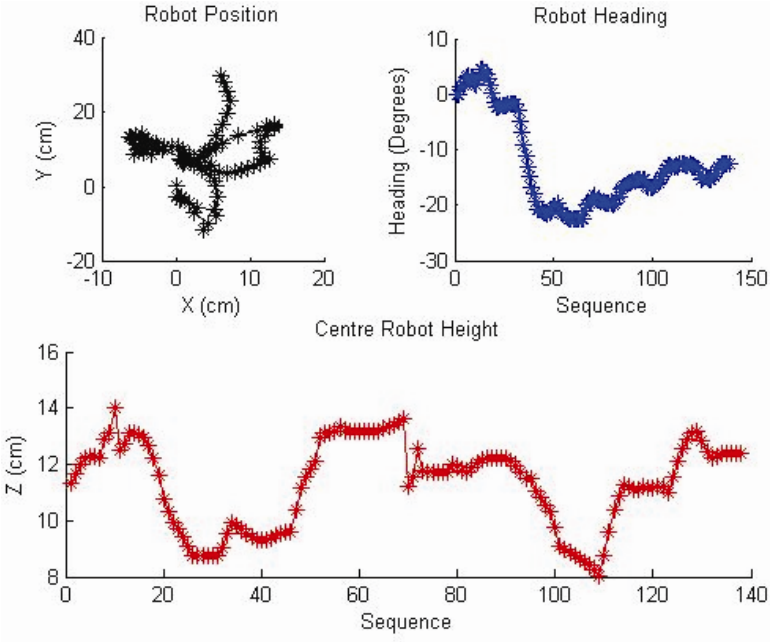
where Position [1, 2] describes the respective static positions, Foot (*X* or *Y*) describes the coordinate of a supporting foot and Centre (*X* or *Y*) describes the coordinate of the centre of the robot.

5 The resulting change in heading of the robot is then defined as the average of d*Ø*, caused by the movement of each supporting foot. The average was chosen to accommodate the random movements associated with a GA created LUT.

6 At this point the robot's height, lateral movement and rotation have been calculated, but the supporting feet have also moved, which may not be the case. For example, if the robot body was to move, based on the displacement vectors of the supporting feet, then the feet must have remained stationary in order for the body to have moved (based on friction). In order to calculate back where the feet should be based on the new heading and position of the robot body, the new positional data are fed back into the static model, and all four leg coordinates for each leg recalculated.

This sequence is repeated from steps (2)–(6) for each increment. A typical result is shown in Figure 4.

Figure 4 Plot of the three movement parameters over a GA generated LUT (see online version for colours)



3 Genetic algorithm

3.1 Fitness evaluation

In order to analyse the performance of a particular LUT with regards to movement, a single output parameter is required from the simulation, as a measure of performance for

the GA to use. The ultimate goal of this research is for the robot to walk forward in a straight line, and thus the performance criteria are centred on this goal.

Several different fitness calculation methods were tried and these are detailed in Section 4. The final method chosen is based on two weighted requirements, the robot moving forward one metre in a straight line, after a maximum of 100 iterations of the optimised LUT, and the stability and efficiency of the evolved gait.

The target location is directly in front of the robot and the distance is chosen based on direct experimentation with the real robot and walking gaits, but as yet unattained with hand tuning of the gaits. This will account for 75% of the final fitness score.

Stability is measured by using the standard deviation of the body height of the robot, across the simulation of the evolved gait. Efficiency is the number of steps the robot took to walk during the gait. Gaits which move left and right before moving forward are deemed inefficient, and thus will score a lower fitness. Both these factors are weighted appropriately to make up 25% of the final fitness.

The procedure of calculating the fitness is as follows:

1 From the dynamic model, the distance the robot moved, the direction of movement and heading of the robot are passed to a fitness evaluation function.

2 The position of the robot is then calculated for every iteration of the LUT, starting at a single iteration ($n = 0$) and working towards 100 iterations ($n = 99$). For each iteration, the x coordinate of the robot is checked to see if it has remained between ± 30 cm, the vertical boundaries, thus limiting the path of the robot to an approximate straight line. The position after n iterations is calculated as below:

$$(x + iy) = \frac{\left(\text{Rho} \times \text{e}^{i\varnothing}\right) \times \left(1 - \text{e}^{(n+1)} \times i\theta\right)}{1 - \text{e}^{i\theta}} \tag{16}$$

where Rho is the distance the robot moved after one iteration of the LUT, $\varnothing$ is the direction moved and θ is the heading of the robot.

3 If the robot does not deviate outside the vertical boundaries during all 100 iterations, then the fitness is calculated as in the step below.

If the position is outside the vertical boundaries, then the iteration is terminated at its current point, and the fitness calculated at this point:

$$\text{Target } F = 100 - \left(\frac{\sqrt{(0 - x)^2 + (1000 - y)^2}}{10} \right) \tag{17}$$

Equation (17) converts the distance from the terminated (x, y) position to the target position (0, 1,000) to a percentage, which is used as the Target Fitness.

4 The secondary requirements are now calculated; stability and efficiency, as detailed in the following equations:

$$\text{Efficiency} = 80 - \alpha \times 5 \tag{18}$$

$$\text{Stability} = 50 - \beta \times 8 \tag{19}$$

where α is the number of steps taken across a single evaluation of the gait, and β is the standard deviation of the height of the robot. Note a minimum of six steps has been factored in Equation (18).

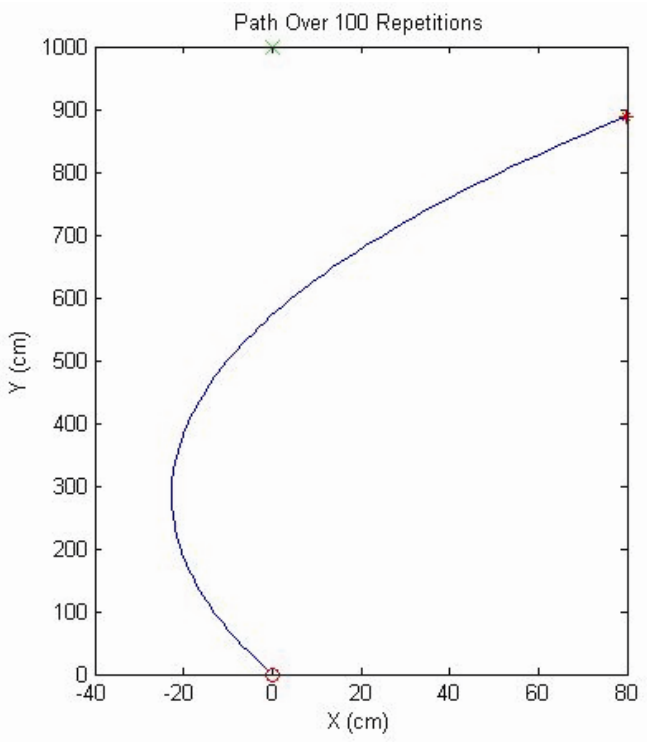
5 Finally the total fitness of the gait can be calculated:

$$\text{Final Fitness} = (\text{Efficiency} + \text{Stability}) \times 0.25 + \text{Target } F \times 0.75 \tag{20}$$

A perfect fitness of 100% would result in the robot stopping at exactly (0, 1,000), with a six step tripod gait being evolved that maintains ultimate stability.

The resulting position across 100 iterations can then be plotted, and an example is shown in Figure 5.

Figure 5 Plot of the robot's position over 100 iterations of a LUT (see online version for colours)



3.2 Genetic algorithm

Within this research, the GA is implemented as follows:

1 An initial random population of 100 LUTs is created using Matlab's random number generation, and using an initial seed created from the current computer time. This allows different LUTs to be generated every time the GA is run.

2 These initial chromosomes are then simulated within the mathematical model, and the fitness value of each is returned.

3 The population is now expanded by creating 100 exact copies of the parents. Each offspring is then procreated using 'Two Point Crossover', a method which chooses two random points within a parent's LUT, and copies the contents of LUT between these points into the offspring's LUT. This has the effect of creating an offspring which contains genes from both parent chromosomes.

4 To add diversity to the population, five offspring are randomly chosen to have ten random genes mutated, which is equivalent to a 0.31% mutation rate. This allows the population to maintain diversification, so to avoid becoming trapped at local optima of fitness.

5 The offspring chromosomes are now simulated within the mathematical model, and a fitness value of each is returned.

6 At this point, there are now 200 chromosomes, and this must be reduced to a stable population size of 100. This is performed using 'Pair Based Tournament Selection', a process which finds the best individual pair of chromosomes; one parent and one offspring. The chromosome with the best fitness is kept, and the worst is discarded. This is performed on 100 pairs, thus reducing the population size to 100.

7 In order to allow variance in pair combinations, the resulting order of the population is now randomised.

8 The best performing chromosome from the selection process is always kept, and its fitness value compared to the desired fitness value. If the best fitness value meets the termination criteria, then the GA is finished, otherwise the GA iterates again from step (3).

The resulting best performing chromosome is now an optimised LUT, which results in an acceptable walking gait of the robot. Within this research, there will be multiple acceptable walking gaits for this robot, and the GA has produced several variations.

4 Results

During the process of this research, the fitness evaluation strategy had to be refined many times over. The first strategy was simply to get the robot to walk as far as possible while maintaining a constant forward facing heading. The result of this was the simulated robot learned to walk diagonally, while maintaining a straight heading.

The second strategy included extrapolating the position of the robot, after n iterations of the LUT, and comparing the resulting position to a target location, 1000 cm in front of the robot. Under this situation, the robot learned to walk in an arc to the target location, thus also defeating the straight line requirement.

The third strategy including adding vertical boundaries to the extrapolation, with the expectation of limiting the path of the robot. Although the robot would now approach the target position in a straighter path, the evolved gait was very inefficient.

The final strategy, as detailed in Section 3, has resulted in the successful evolution of walking gait which results in the simulated robot walking forward in an approximate straight line. Evolution of this gait occurred after 684 generations, and an evolution period of 39 hour Figure 6.

The best gait evolved had a final fitness of 61.06%, and followed an approximate straight path as shown in Figure 7.

Figure 6 Fitness across four GA evaluations (see online version for colours)

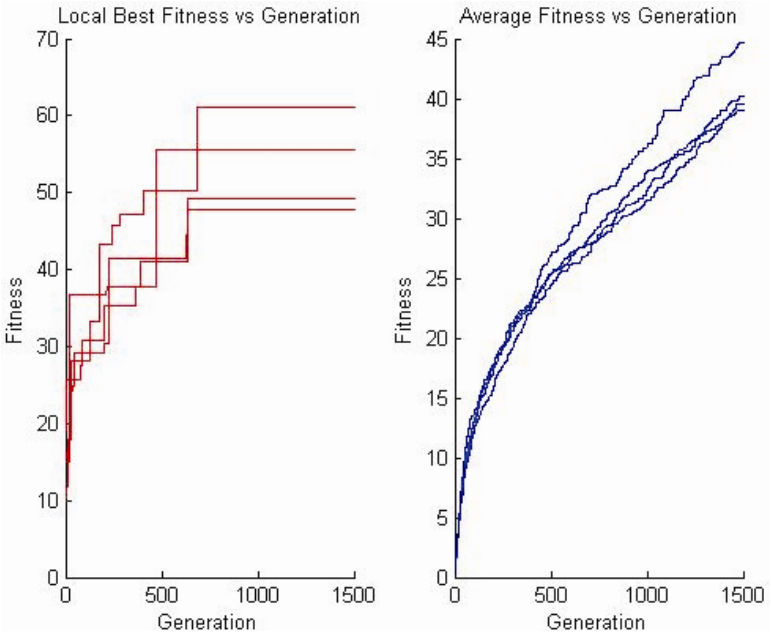
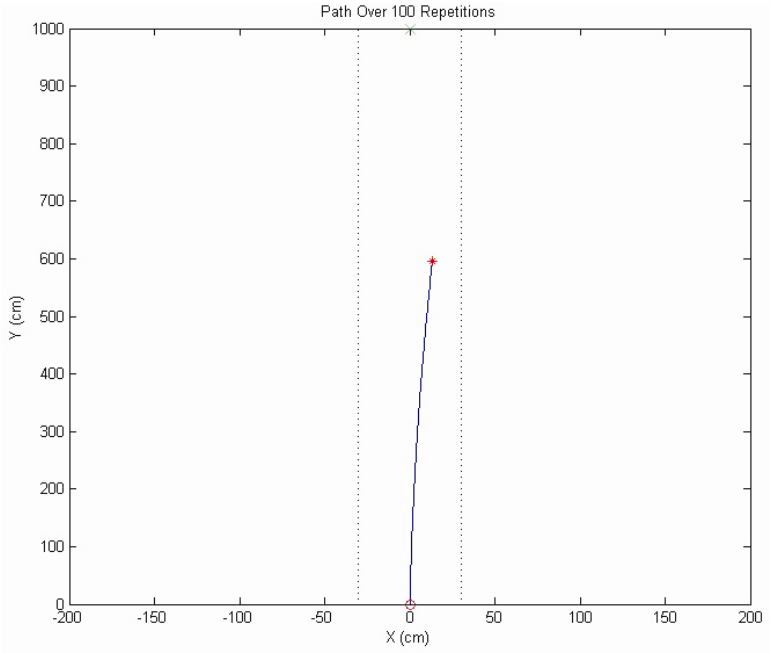


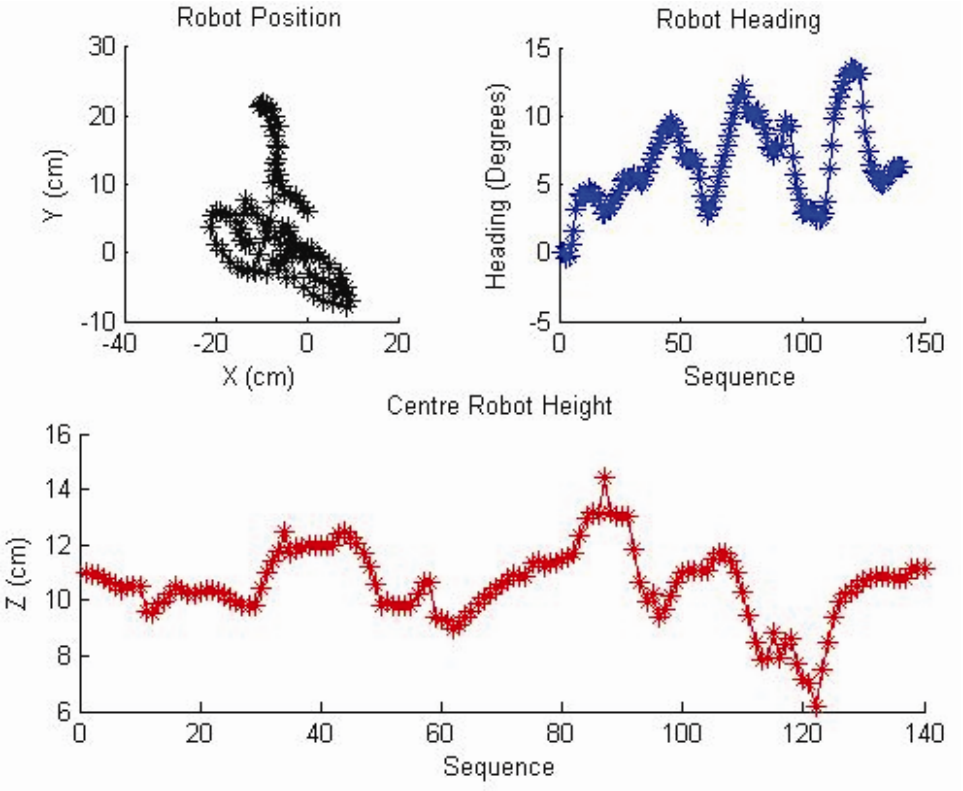
Figure 7 Path of the best evolved gait across 100 iterations (see online version for colours)



Although the overall gait resulted in a straight walking path, plotting the movement incrementally showed the gait was typically inefficient. This can be seen in Figure 8.

The plot of robot position shows the robot dwelling around the centre point, walking forward, then appearing to fall backwards, which can be seen in the robot height plot and corresponding points in the position plot. The stability of the robot was good (approximately level at 11 cm) until the fall happened at sequence iteration 100 and 9. The plot then shows the robot standing back up to level by the end of the gait.

Figure 8 Plot of the movement of the simulated robot (see online version for colours)



5 Conclusion

The result of this research is that a walking gait has been successfully evolved, which guides the simulated robot in a approximate straight line, however the efficiency of the gait was poor. This is because the fitness evaluation was biased towards forming a gait with a straight line path rather than efficiency of motion.

The GA was run multiple times on multiple PCs with varying rates of successful gait evolution. These fitness values were comparable to the best achieved, however the direction of the path formed larger arcs rather than the desired straight line, although improved efficiency and stability of the gait were exhibited.

References

Alba, E. and Chicano, F. (2006) 'Genetic algorithms', in *Metaheuristic Procedures for Training Neutral Networks*, Vol. 36, pp.109–137, US: Springer.

Braunl, T. (2006) 'Genetic algorithms', in *Embedded Robotics*: Berlin, Heidelberg: Springer, pp.291–305.

Chernova, S. and Veloso, M. (2004) 'An evolutionary approach to gait learning for four-legged robots', *Intelligent Robots and Systems, 2004 (IROS 2004), Proceedings 2004 IEEE/RSJ International Conference on*, pp.2562–2567.

Darwin, C. (1859) *Origin of Species*. London, UK: John Murray.

Dürr, V., Schmitz, J. and Cruse, H. (2004) 'Behaviour-based modelling of hexapod locomotion: linking biology and technical application', *Arthropod Structure and Development,* Vol. 33, pp.237–250.

Fauvel, J. (1993) *Mobius and his Band: Mathematics and Astronomy in Nineteenth-Century Germany,* USA: Oxford University Press, August.

Goldberg, D.E. (1989) *Genetic Algorithms in Search, Optimisation and Machine Learning*. MA: Addison-Wesley.

Kohl, N. and Stone, P. (2004) 'Policy gradient reinforcement learning for fast quadrupedal locomotion', in *Robotics and Automation, 2004. Proceedings ICRA '04. 2004 IEEE International Conference on*, pp.2619–2624.

Mazzapioda, M. and Nolfi, S. (2006) 'Synchronization and gait adaptation in evolving hexapod robots', *Lecture Notes in Computer Science*, Vol. 4095/2006, Berlin, Heidelberg: Springer, pp.113–125.

Parker, G., Braun, D. and Cyliax, I. (1997) 'Evolving hexapod gaits using a cyclic genetic algorithm', *Proceedings of IASTED International Conference on Artificial Intelligence and Soft Computing (ASC'97)*, pp.141–144.

Parker, G.B. (2000) 'Evolving leg cycles to produce hexapod gaits', in *The World Automation Congress WAC*, Vol. 10, pp.250–255.

Pratihar, D. (2003) 'Evolutionary robotics – a review', *Sadhana,* Vol. 28, pp.999–1009.