11 Incremental Evolution of an Omni-directional Biped for Rugged Terrain

Eric Vaughan, Ezequiel Di Paolo and Inman Harvey

11.1 Introduction

Natural Evolution has produced humans that can walk and talk, without any explicit design process; Darwinian evolution has taken the role of the Blind Watchmaker. The process took some 4 billion years overall, with inconceivably immense resources and plenty of dead ends. It was not aimed, we may assume, at the end goal of walking, talking humans; there are certainly plenty of viable species that neither walk nor talk.

A human designer, aiming to replace at least part of the explicit design process by an Evolutionary Robotics (ER) methodology based on Darwinian evolution, will have comparatively tiny resources and a limited timeframe. There are many possible motives for using ER (Harvey et al 2005), and the one presented here can be called an engineering motivation: to design a mechanism for a specific application. The use of ER will only be justified to the extent that it produces better results than can be expected through conventional design methods. The human engineer will have a focused goal, and will want to apply every trick that can be found to speed up the evolutionary process. There will typically be a continuing interplay between the role of 'blind' ER and the vision of the engineer; we present here one case study of how this can work out.

The engineering goal here is the design of efficient and robust machines for bipedal walking in any direction on both flat and irregular surfaces. Bipedal robots have the potential to replace or assist humans in the terrains that they use, including rugged surfaces outdoors and steps and stairs indoors. It turns out that it is challenging to replicate human walking. One reason human locomotion is so efficient is that it leverages passive dynamics to reduce energy consumption and uses the elastic nature of tendons to store and release energy; these considerations have been missing from traditional robot design. Here we present one part of a larger body of work undertaken by the first author in doctoral research on the development of bipedal walking (Vaughan, 2007). The results presented include the successful coordination and control of many more degrees of physical freedom (up to 35) than are typically tackled by conventional design methods; this endorses the effectiveness of the ER methodology. Going beyond the domain of bipedal walking, we consider that the ongoing interplay between engineering vision and ER methods in incremental design, as illustrated in this case study, may have valuable lessons for a wider audience.

11.2 Empowering ER through incremental design

Natural evolution has clearly been incremental, at both micro and macro scales, with future generations altering and extending the design achievements of earlier ones. Human design methods are likewise often incremental. The Wright brothers started by adding elements of active control to kites, so as to produce unpowered dynamic flying machines

– gliders. Only then did they go on to adding power, and increasing the sophistication of the controls. Our current aircraft can trace their ancestry through continued incremental improvements from those early days.

This present body of work takes inspiration from the Wright brothers, but applied to walking. McGeer (1990) showed that a simple set of jointed legs, in proportions similar to human legs with their knees, could walk down a slope with no power other than that provided by gravity, and no control other than that provided by the pendulum dynamics. This Passive Dynamic Walker (PDW) can play the part of the Wrights' glider, to which power and increasingly sophisticated levels of control is to be added. This implies an incremental design pathway that can be well suited to ER. Ways in which an evolutionary algorithm can be applied to such an incremental process, and the use of incremental methods in ER, were proposed in Harvey (1992), Harvey et al. (1993), and a body of work following from this. Prior work applying incremental evolution specifically to walking robots includes Kodjabachian and Meyer (1998).

Brooks' (1991) Subsumption Architecture also advocates an incremental design approach. Though it is explicitly inspired by the incremental aspects of natural evolution, design by hand is used throughout. The emphasis is on building complete robots that initially have simple behaviours, and then adding extra functionality to enable extra layers of behaviour and more sophistication. At each successive stage, the robot has to function successfully in the real world at its currently expected level, and only after this is achieved will the next stage be added. We can consider several potential advantages to this incremental approach, over and beyond the fact that it follows good engineering principles of iterative development and testing.

Firstly, it breaks down what may be one very large design problem into many smaller ones, each of them individually more tractable. Since limited resources may well make cracking the big problem in one go unlikely or impossible within the available timescale, the achievement of some intermediate stepping stones can be a better result. Secondly, intuitions supported by anecdotal evidence (and meriting more principled investigation) suggest that it may well often consume less resources to achieve an ambitious end goal via intermediate stepping stones than it would to attempt it in one go, even if the engineer learnt nothing new during the process. Thirdly, as will be indicated in the examples below, it is likely that the engineer will indeed learn of significant new factors during the intermediate stages, and this can lead to recognition of stumbling blocks, improvements in choosing what the next stepping stone might be, and the possibility of making available new and appropriate resources for the next stage of design. The ER example here illustrates an ongoing collaboration between the engineer and the evolutionary process; often the former is providing the broad brush strokes outlining the direction the next design stage should take, whilst the latter is providing the essential detail by juggling the parameters of a highly complex system so as to coordinate the different parts. Here the coordination needs to be between neural and physical dynamics. Walking involves real time dynamics, so it is natural that the control system being involved should also cope with real time dynamics. This consideration influenced the choice of Continuous Time Recurrent Neural Networks (Beer, 1995), described in more detail below.

Most forms of evolutionary algorithm will handle incremental evolution satisfactorily, so the details of the GA used here are not significant. One observation to note is that, at each successive stage of evolution, the population will be based on that which succeeded at a preceding stage, and hence will tend to be always quite genetically converged rather than initially randomised. This has some implications for the evolutionary dynamics (Harvey, 2001).

11.3 Staged evolutionary design of walkers

In previous work (Vaughan et al 2004a, Vaughan 2007), passive dynamics were explored in physical simulations using a three Staged Evolutionary Design (SED). In the first stage a ten degrees of freedom machine was created with hips and ankles that could walk down a gentle slope un-powered by optimizing the physical properties of the body. In the second stage a simple neural network was hand-designed and coupled with a central pattern generator. In the final stage sensor-input was added to the network and the slope was lowered to a flat surface over many generations. This machine showed that efficient walking attractors can be generated in the body itself and it is possible to vary their range from sloped surfaces to flat ones by adding simple stabilizing neural networks. In following work (Vaughan et al 2004b, Vaughan 2007), some of the weaknesses of this model were addressed, specifically its lack of a weighted torso and inability to walk backward. Using a more complex network and prior knowledge the sloped platform stage was bypassed and the machine learned to walk directly on a flat surface.

In this chapter a more sophisticated 3-D bipedal machine is developed that can walk unsupported in any direction on both flat and irregular surfaces. First simple models, some basic principles of walking and balance are discussed and explored. This is followed by a description of the body and networks used. The machine is developed in three stages demonstrating the power of the incremental methodology.

In stage one a simple planar machine is developed based on previous work that can walk forward and backwards on a flat surface. The addition of neural circuits to a passive dynamic walker in the planar case has been recently explored by Manoonpong et al. (2007) using synaptic plasticity to achieve adaptive control, but with significant differences from the approach described here. Through observation improvements are added to increase the machine's performance and to allow it to walk on rugged terrain. In stage two a lateral control system is added allowing the machine to walk unrestrained in three dimensions. The machine is tested on flat and rugged surfaces, and shows the ability to walk at different speeds and make dynamic quick movements in response to the environment. In stage three, we examine and discuss some implications of walking with ankles and flat feet. At the end of the chapter preliminary experiments with a spine, passive arms, and extra hip joints are discussed. These are difficult problems that have not been studied previously but can be approached using the SED methodology.

11.4 Walking revisited

Following the incremental strategy, the same problem of robust walking is approached again but at each stage taking experience from earlier work (Vaughan et al 2004a, 2004b, Vaughan 2007). Previous models are scaled up to a machine with 35 degrees of freedom with a flexible spine. At this point it is beneficial to revisit what has been learned from previous work. The whole idea of walking is revisited in order to come up with a strategy for scaling up to a machine that could challenge trajectory-based machines such as Honda's Asimo. Three concepts in walking are discussed: *foot placement*, *foot passing*, and *weight balancing*.

Foot placement. Walking can be thought of as controlled falling where the legs consistently break the fall of body mass on each step. Generally this is controlled by foot placement. The larger the angle between the hip and the leg when the foot strikes the ground, the greater the decrease in the body's velocity. Smaller angles can act to increase velocity by failing to reduce the machines fall (Raibert, 1986). This provides a basic mechanism for controlling a machine's velocity not just forward and backward but sideways as well.

Foot passing. To apply foot placement to a machine's gait each leg needs a way for the foot to pass the other without striking the ground. A simple artificial solution would be to use a telescoping leg. When the machine falls forward its rear leg naturally loses contact with the ground. Upon detecting this the leg contracts, swings past the other leg, and extends. When the leg finally strikes the ground the rear leg loses contact and the cycle is repeated with the other leg. If the angle of the hip is correct when the foot strikes the ground the machine can maintain a desired velocity. Not only does the hip angle need to be correct but the leg must swing to that hip angle in just the right amount of time. This implies that any control system used to rotate the hip or knee must have good control over joint velocity. For a more human-like gait the telescoping leg model can be replaced with knees as shown in Figure 11.1. It is important to note that this creates a virtual angle at the hip. On a telescoping leg the hip angle will not change as the leg is contracted, but when a knee is added, the hip angle must increase as the knee bends. A dotted line is used in Figure 11.1 to denote the virtual angle that should be used when calculating foot placement.

[Figure 11.1 here]

Weight balancing. In previous work (Vaughan et al 2004b) a *stance* mode (see 5.2 for a detailed explanation of modes) was implemented making a positive connection between a gyroscope that detected the orientation of the waist around the *x* axis and the desired hip velocity. This *stance* mode balanced a weighted torso placed above the hips. The idea of balancing weight above a machine's hips is often the focal point of bipedal research. At its simplest the torso can be thought of as an inverted pendulum (Raibert, 1986). Linear feed back from orientation sensors can be used to rotate the hip and balance the torso dynamically. If the torso begins to tip, the hip is rotated to capture its weight.

11.5 Stage one: Walking forward and backward

The purpose of stage one is to develop a planar machine similar to the one explored in previous work but built upon a more flexible control system. Movement is constrained to the sagittal plane (*x* and *y* axis). The degree of freedom in the hip that allows the leg to rotate to the side (around the *x* axis) is locked (Figure 11.2). A pre-designed control system is encoded into a genotype and seeded into a population of machines that is later evolved with a genetic algorithm (GA). To simplify the problem the use of flat spring loaded feet is added in stage two.

[Figure 11.2 here]

11.5.1 The body

The body used in this chapter initially has 6 degrees of freedom: two in each hip and one in each knee. Later in Stage 2 ankles and feet are added increasing the number of degrees to 10 (Figure 11.2). The simulation is done with the Open Dynamics Engine (ODE) and the parameters of both the body and control system are evolved using a GA (see details below). Body parameters are evolved from the following ranges: thigh's mass (kg) [2,4], thigh's length (m) [2,4], shin's mass (kg) [1,3], shin's length (m) [0.3, 0.4]. The torso's mass is 30 kg and its length 0.7m. The range of motion for all joints on the machine were between -π/2 and π/2.

A simple muscle model is used that supports foot placement and energy conservation through passive dynamics. Each degree of freedom has three parameters:

target angle, *desired velocity*, and *maximum torque*. The first two allow the joint to move to a target angle smoothly within a specified time as required by foot placement. The last parameter places a limitation on how much torque can be used to reach the target giving evolution a mechanism to take advantage of the natural dynamics of the body.

11.5.2 Modes

We refer to dynamic patterns playing a functional role in walking as modes. These are implemented through individual neural circuits (similar to reflex circuits in animal walking). In previous work just two modes were used for each leg in the walking cycle: *swing* and *stance*. A winner-take-all circuit was used to switch between each mode depending on whether the foot was on or off the ground. In the more complex machine proposed here the *swing* mode could become overly complex. It must contract the leg, swing it to the proper location, and then extend it in just the right amount of time. It must keep track of the virtual hip angle and respond dynamically to changes in forward of backward velocity. The approach of this chapter is to hand design a simple network that can be improved though evolution. However, this mode would appear to require a non-trivial, non-linear solution. One approach is to break the *swing* mode up into several modes that have simpler solutions as done by Raibert (1986); Pratt and Pratt (1999). Logically these are: *contract*, *swing*, and *extend*. Figure 11.3 shows the transitions between modes for each leg. Modes are implemented as networks with *sensors*, two *hidden layers*, *effectors*, and a *switching neuron*, (Figure 11.4).

[Figure 11.3 here]

[Figure 11.4 here]

Sensors. A list of sensors and their description can be found in Table 11.1. For angle sensors such as *Hip X*, *Hip Y*, and *Knee* a single sensor neuron is used. Angles are mapped linearly onto the sensor neuron’s activation with negative angles corresponding to activations below 0.5 and positive angles above 0.5. For velocity sensors neuron pairs are used: *forward/backward* and *left/right*. When the machine is falling forward the *forward velocity* neuron's activation increases over 0.5 while the *backward neuron's* activation is maintained at 0.5. When falling backward the opposite is true. While this could have been encoded in one neuron the ability to get the velocity regardless of its sign helps to simplify the design.

[Table 11.1 here]

Hidden layers. There are two hidden layers (Figure 11.4). The first hidden layer is primarily used to control movement through the sagittal plane, although there are some exceptions. The second layer is used in stage two to control movements laterally outside the sagittal plane.

Effectors. These are neurons that connect to each of the three powered joints: The hip around the *x* axis, the hip around the *y* axis, and the knee around the *y* axis. Each joint is controlled by three neurons.

Tar. The target angle the joint should rotate to. Its value can be anywhere between 0 and 1. If the value is 0 the joint will strive to rotate to -*range* and if it is 1 it will try to rotate to +*range*. Where *range* is the maximum amount of movement of the joint in radians.

Vel. The desired velocity *v* that the joint should rotate at until it reaches its desired angle. This value can be between 0 and 1. Actual velocity (*av*) is calculated as: $av = (v - 0.5)*2$.

Tq. The maximum torque (*mt*) that can be applied to reach the desired velocity. If *mt* is 0 the joint becomes unpowered regardless of velocity or target.

Switching neuron. Each mode network may connect to all incoming sensors as well as signals coming from other mode networks. The activity of the switching neuron (*S*) in each mode circuit can be increased or decreased by neurons in the same circuit or in others. Only one of the four modes in a leg can be active at any time so the mode with the strongest switching neuron activity is enabled and all other modes are inhibited (i.e., their effectors shut down).

Activation functions. Continuous-time recurrent neural networks are used to implement the different mode networks (Beer, 1995). The activation of neuron *i* (y_i) is given by:

$$\tau_i \dot{y}_i = -y_i + \sum_j w_{ij} Act(y_j + b_j) + I_i$$

where τ_i is the decay time constant, w_{ij} is the weight of the connection between neuron j and i, b_j is a bias term, $Act()$ is the activation function, and I_i is an external input.

Five types of activation functions are used.

Type 1 $$\sigma(s) = \frac{1}{1+e^{-s}}$$

Type 2 $$\alpha(s) = |\sigma(s) - 0.5| + 0.5$$

Type 3 $$\eta(s) = \begin{cases} 0 & s + 0.5 < 0 \\ 1 & s + 0.5 > 1 \\ s + 0.5 & otherwise \end{cases}$$

Type 4 $$\beta_1(s) = \begin{cases} 1 & s \geq 0.5 \\ 0 & s < 0.5 \end{cases}$$

Type 5 $$\beta_2(s) = \begin{cases} 1 & s \geq 0.5 \\ 0.5 & s < 0.5 \end{cases}$$

Computationally, most of these activation functions could be replaced in practice by small networks of neurons using only the sigmoid function, σ(s). However, the use of the different functions greatly simplifies the hand design of seed networks. As a graphic convention in the figures below describing neural circuits, neurons with activation function of Type 1 are represented by plain circles, Type 2 by squares, Type 3 by cubes, Type 4 by triangles and Type 5 by diamonds. Excitatory connections are represented by full lines, inhibitory connections by dashed lines, bias neurons by double circles and modulatory connections as arrows linking a neuron and another connection.

11.5.3 Support

The *support* mode becomes active when the foot is touching the ground. It attempts to balance the upper torso of the biped by rotating the hip either forward or backward. A simple algorithm is:

- Keep the knee straight
- If the torso is pitching forward power the leg backwards until the torso becomes upright. If the torso is pitching backward power the leg forward until the torso becomes upright.

The support network (see Figure 11.5) has four tasks: keeping the knee straight, keeping the torso upright, storing the angle of the leg, and ensuring that when the leg leaves the ground the *contract* mode is triggered.

Keeping the knee straight. The knee is kept straight by giving the (*B*) neuron a positive bias and excitatory connections to the knee angle target, velocity, and torque. This causes the knee to gently push into the kneecap.

Keeping torso upright. Connections from the forward (6) and backward (5) pitch sensors increase the hip's velocity and torque. The more the torso falls forward or backward the more speed and torque can be used to bring it up right again. A positive connection from (5) and a negative connection from (6) set the hip's desired target angle toward the direction the leg needs to rotate.

Storing the hip angle. Why store the hip angle? Once the leg loses traction with the ground the leg switches to the *contract* mode. In this state the leg must keep the leg supported at the angle at which it left the ground and retract the leg. This requires some way to store the last hip location. A single positive recurrent self-connection on neuron (*M*) is used to store the current angle by adding a negative and positive connection to neuron (*C*). Whatever value (*C*) receives will automatically be stored in neuron (*N*).

Triggering the contract mode. Once a foot loses traction its touch sensor inhibits the switching neuron and then *contract*, *swing*, and *extend* have to compete to see which mode gains control. Each one needs to be active at different stages of the walking gait. One solution is to train the connections from a mode's input layer to its switching neuron such that it becomes active at the right time. However, these functions may not be linear requiring the addition of an additional hidden layer between the inputs and the switching neuron. A simpler approach is to notice that the *support* mode always precedes the *contract* mode. It is much easier to allow the *support* mode to tip the vote through a positive connection to *contract's* switching neuron. Neuron (*E*) is given a positive bias and connected to the *contract* network's (*Z*) neuron. The (*Z*) neuron has a larger time constant (τ) than other neurons so while *support* is active, it charges up. When *support* does finally lose, (*Z*) is still be excited momentarily resulting in a higher activity in *contract's* switch neuron.

[Figure 11.5 here]

11.5.4 Contract

The *contract* mode’s task is to contract the leg while maintaining the current hip angle. When the leg is fully contracted to the desired leg height it triggers the *swing* mode. The *contract* network is shown in Figure 11.6.

Contraction speed. One important factor in walking is the speed at which the leg moves. This is not only how fast the leg swings forward but how fast it is contracted, swung, and extended. The total time it takes for all three to occur is critical to good foot placement. Too slow and the leg will not reach the target in time and the machine will stumble. Too fast and unnecessary energy is used that could have been saved. The speed of leg contraction should be proportional to the speed the machine is moving forward or backward. This is the function of neuron (*C*), which is excited by the forward (*7*) and backward (*8*) velocity sensors. It in turn excites the knee velocity/torque and the hip velocity/torque. The faster the machine falls the greater the strength and velocity of contraction.

Contraction height. Contraction height is specified by neuron (*D*) whose positive bias pulls the knee up and the foot in at the velocity specified by neuron (*C*). Future stages could use this neuron to increase leg height on more rugged terrain.

Virtual hip angle. The leg must contract but keep whatever angle it had when it was in the *support* mode. Neuron (*B*) adjusts the leg angle by adding the last angle stored in *support's* (*M*) neuron.

Triggering swing. The *swing* mode is triggered by thresholded neuron (*E*) when the leg is fully contracted. Neuron (*E*) receives the difference between the current knee angle (*3*) and the desired contraction height (*D*). When the sum of the two is greater than 0, neuron (*E*) reaches its threshold and fires, inhibiting *contract* and exciting the *swing* mode.

[Figure 11.6 here]

11.5.5 Swing

The purpose of the *swing* mode is to generate proper foot placement and move the leg such that when extended it will break the machine's fall and reduce its speed. Once the leg has reached its position it triggers the *extend* mode. The *swing* network is shown in Figure 11.7.

As discussed earlier, the speed at which a leg moves is critical for proper foot placement. The leg must swing forward or backward proportionally to the machine's velocity. The faster it falls the faster the leg needs to move to catch the machine's weight.

Foot placement is also proportional to the velocity of the machine. Both of these mechanisms together produce a balanced walking gait.

The swing algorithm uses six neurons: (*A*), (*B*), (*C*), (*D*), (*E*), (*J*), and (*M*).

Neuron (*A*) stores the virtual hip angle (Factor three) in (*M*) by subtracting the desired contraction height (*D*) from the actual hip angle (*2*).

Neuron (*B*) computes the velocity and torque of the hip as it swings forward or backward. Factor one is implemented by positive and negative connections from forward (*7*) and backward (*8*) velocities. Factor two by adding in the previous leg position stored by the *support* mode.

The knee is kept bent at its contraction height due to the lack of any connections to its velocity combined with (*C*) constantly exciting the torque.

Neuron (*D*) is the contraction height. It has connections to the hip target to keep the leg contracted throughout the swing phase.

Neuron (*E*) computes the absolute error between the target hip angle and the actual hip angle. A large error keeps (*J*) below its threshold, when the error is small enough (*J*) fires and triggers the *extend* mode.

In previous work (Vaughan et al. 2004a) a machine was evolved that could be controlled easily by gently pushing it at the speed and direction required. If no force was applied the machine would stand still. Such control mechanisms are very desirable when building machines for carrying loads, as they do not require a driver. Instead they can be gently pushed or pulled by a rope in the desired direction. If a more complex control system is needed it can be incrementally added later. In this chapter subsumptive control of such a mechanism is explored by adding a special neuron (*K*) to modify the default behaviour. As discussed earlier, foot placement can be used to directly control the velocity of a walker. Too large a step and the walker will slow down; too small a step and the machine will continue to accelerate as it falls. Neuron (*K*) can adjust the machine's velocity simply by connecting it to the *Hip Y's* target neuron. As (*K*) is increased the machine tends to walk forward as it fails to catch its centre of mass (COM) in time. If gently pushed backwards it will overcompensate and take too big a step backwards causing it eventually walk forwards again. When (*K*) is inhibited the opposite happens and the walker tends to walk backward. The actual speed of the walking gait depends on (*K*) allowing basic control over acceleration and deceleration of the machine.

[Figure 11.7 here]

11.5.6 Extend

The *extend* network's job is the opposite of the *contract* network. It must straighten the leg while keeping it fixed at the desired angle stored by the *swing* network. Once the foot touches the ground the *support* network is automatically triggered.

Only two neurons are required: (*C*) and (*B*) (Figure 11.8). Neuron (*C*) has a bias and supplies a constant velocity and torque to the knee and hip to keep them firm when the machine is not moving. When the machine has velocity it is added to (*C*). The faster the machine falls the faster and stronger the leg extends out to catch its fall.

[Figure 11.8 here]

11.5.7 Experiments in flat surfaces

A geographically distributed GA (Husbands, 1994) was used with 25 individuals. Each generation, 20% of the genes were selected according to fitness and changed using creep mutation to populate the next generation. The mutation rate was 0.02.

Machines were evaluated by testing their ability to walk both forward and backward. Two test cases were used. In the first test case the (*K*) neurons in each *swing* leg were set to a random negative value causing the machine to walk backwards. In the second case a random positive value was used causing the machine to walk forward. Five trials were used for each test case and then the averages for each case were multiplied together.

The following fitness function was used:

$$fitness = time * min(rot) * min(vel) * min(energy)$$

where $min(t) = 1/(1 + t)$, *time* is the amount of time the machine walked (maximum is 30 seconds), *rot* is the absolute average error between 0 and the body's pitch angle, *vel* is the absolute average error between the desired velocity (specified by the (*K*) neuron) and the actual velocity, and *energy* is the average torque used by all actuators.

An evaluation was started by placing both legs on the ground and stimulating the switching neuron for *support* on one leg while stimulating *contract* on the other. Once a leg lost contact with the ground all stimulation was removed.

11.5.8 Observations and improvements

The hand-designed networks did well even before any evolutionary processes were applied with an average fitness of 0.4. After evolving for 1320 generations it improved by 200% percent to a fitness of 0.8.

The machine walked well on a flat surface but in real world environments this is rarely the case. On less smooth surfaces the machine's natural walking dynamics must be able to minimize disturbances in the torso in the same way shock absorbers work for a car. A bumpy ground surface was simulated using a mesh of flat triangles. The mesh was a grid of 70x70 squares each split into two triangles with random heights. The maximum height of the surface was 3 cm. The same population used on the flat surface were tested on the rugged surface. Initially their average fitness fell from 0.8 down to 0.12. After 400 generations its best average fitness was only 0.24.

Although the machine's performance is excellent on flat surfaces, when placed on bumpy uneven surfaces the machine loses balance falling either forward or backward. Through careful observation the cause was isolated to four different situations. The first two were early foot strike when stepping on a small incline and late foot strike when stepping into a small depression. The third was when the machine lost forward momentum and came to a complete stop with both legs side by side. Eventually it began to fall either forward or backward while both feet continued to stay on the ground. There was no triangle between each foot and the torso so neither leg could be lifted off the ground as the torso moved. Unable to switch to the *contract* mode the walking gait stopped completely and the machine fell. The fourth situation was in the event both legs accidentally lost traction with the ground. In this situation both legs would enter the *contract* mode simultaneously and the machine would fall to its knees.

In early foot strike, the machine steps onto an incline and the leg strikes the ground before it is fully extended. When the leg does finally extend, it is during *support* which causes it to push itself backwards (Figure 11.9). The simplest remedy for this behaviour is to modify *support* to reduce the velocity at which it can straighten the leg.

[Figure 11.9 here]

In late foot strike the foot extends fully but does not strike immediately due to a small depression on the ground. As the foot moves into the depression the machine's center of mass falls too far forward causing it to fall (Figure 11.10). The easiest remedy is to wait until the leg is fully extended then if the foot still hasn't struck the ground, bend the knee on opposite the leg. This will push the extended leg farther down until it strikes the ground preventing the machine's mass from falling too far forward. To modify the modes, knee angle (*3*) causes neuron (*A*) to fire when it reaches full extension. Neuron (*A*) in turn excites neuron (*F*), which due to its longer time constant fires a few moments later.

Neuron (*F*) triggers the knee neurons in the opposite legs network to contract momentarily.

[Figure 11.10 here]

Correcting the third situation when the walking gait is stalled as both legs come together requires a different approach. This is a fundamental problem in walking. What happens when a person is standing and they suddenly begin to fall backwards? They naturally lean to one side, pick up the opposite leg, and take a step back. This can be looked at as an additional mode called *stand* whose purpose it to contract the opposing leg if the machine begins to fall while its legs are together (Figure 11.11). This situation also happens at the beginning of our experiments when the machine is first placed on the ground and pushed.

Currently this is solved artificially by placing one leg in *support* mode and one in *contract* mode at the very beginning of the experiment. This can easily be replaced by a *stand* mode allowing the machine to initiate the first step itself.

[Figure 11.11 here]

11.5.8 Stand

The *stand* mode inherits all the functionality of the of the *support* mode while adding some additional features. Five additional neurons are used: (*Y*), (*G*), (*H*), (*I*), and (*J*), (Figure 11.12).

The (*Y*) neuron responds to the difference between each legs virtual angle. The closer they are to each other the smaller it becomes. In turn, it inhibits (*Z*), which stimulates the switching neuron. The result is that the legs must be very close to each other in order for the *stand* mode to become active.

Neuron (*I*) is excited if the machine begins to fall forward or backward, if it falls too much (*J*) will fire contracting the opposite legs knee. As both legs are in *stand* only one (*I*) - (*J*) circuit should activate or both legs would simultaneously contract and the machine will fall. Normally this would be determined by weight. If the centre of mass (COM) is over the left foot it is natural to contract the right one and *vice versa*. However, the machine in stage one is planar and is supported laterally such that the COM is equally distributed between each leg. In stage two when lateral support is added, *Hip X angle* can be used to decide which leg the COM is over. To remedy this in planar experiments the concept of handedness must be introduced. A small random initial bias is given to each (*H*) neuron so when the *stand* mode is entered the neuron with the greatest bias will become active.

[Figure 11.12 here]

Neuron (*G*) has a threshold causing it to fire only if the network's (*H*) is the winner of the competition. It acts as a simple switch turning on and off the connection between (*I*) and (*J*).

The fourth situation that destabilizes the machine on irregular surfaces is when both legs lose traction simultaneously. When this happens it is important that one of the legs continues to be in the *support* mode until the machine touches the ground again. An easy solution is to create positive lateral connections from *contract*, *swing*, or *extend* on one leg to the *support*'s switching neuron on the other. The *support's* switching neurons should be more excited when a mode other than *support* on the other leg is active. A specialized neuron with a threshold activation function (type 5) is used that only fires if the mode is active.

Overall performance improvement. The four modifications were made and a new population of 25 machines was evolved under the bumpy surface condition. After the first generation the population's average fitness was 0.2, very close to the best average of the unmodified population of 0.24. The fitness quickly grew until machines appeared that could walk the entire evaluation period. After 260 generations the average began to flatten out at 0.54.

11.6 Stage two: Lateral control

The goal of stage one was to develop a planar machine that could walk on irregular surfaces. In this second stage the virtual boom is removed and the machine is allowed to walk unsupported. The machine must now work to keep its torso from falling or rolling to the left or right outside of the sagittal plane.

Two simple walking principles can greatly simplify the understanding of lateral balance: lateral foot placement and weight shifting. Lateral foot placement is fundamentally the same concept used in the sagittal plane. If the machine detects it is falling to the right it must also move its foot to the right to catch its weight as it falls and *vice versa*.

If a person is standing and wishes to take a step with their right foot they must first have their COM over their left. If they don't they will have to step out to the right to stop their lateral momentum. This weight shift allows a person to step in a straight line forward. This can be done mechanically with a damped spring in the hips. For example if the machine is taking a step forward with its right foot the COM will begin to move toward the right foot. When the right leg eventually touches the ground the COM compresses into the spring and balances over the right hip. When the left leg contracts this potential energy is released and the COM is pushed back toward the left hip. This creates a kind of throw-and-catch game with the COM between the legs.

11.6.1 Support

Lateral support along the *y* axis is a similar problem to that of the sagittal plane, it must keep the torso upright as the machine begins to fall. This can be accomplished by four additional neurons: (*F*), (*A*), (*G*), and (*T*) (Figure 11.13). The (*F*) neuron takes input from the roll neurons (*11, 12*) and adjusts the target lateral hip angle around the *x* axis to keep the torso upright. Torque and velocity are controlled by (*A*), which takes the absolute value of (*F*).

Unlike support moving along the sagittal plane, support along the *y* axis must also control the shifting of the COM from one leg to the other. At its simplest, if the *support* mode detects that the opposite leg is not in *support* (contracting) it should begin to push its leg to the outside moving the COM over the contracting leg. When the opposite leg does finally touch the ground the COM will now be resting above it. This can be done with just two neurons (*G*) and (*T*). Neuron (*T*) has a threshold and fires only when the *support* mode is active. It in turn inhibits the (*G*) neuron in the opposite leg. Neuron (*G*) has a small bias and an inhibitory connection to (*F*). If both (*T*)'s are firing then both legs are down weakening the bias of both (*G*)'s. In this case (*G*) has no effect on the hip angle. If the opposite leg moves into *contract* its (*T*) will stop firing and (*G*)'s bias will cause it to excite (*F*) moving the leg to the outside. The result is the COM begins to move over the newly contracting leg.

[Figure 11.13 here]

When a leg enters the *support* mode the COM will be over it. The task of this mode is to capture this mass and then release it when the opposite leg contracts. This can be accomplished with a mechanical spring (Vaughan et al 2004a). Catching of the COM only happens as the COM moves from the inside leg towards outside, requiring this spring to be enabled only when the hip X angle (*1*) is negative. A spring can be approximated by making an inhibitory connection from the hip X angle (*1*) to the (*F*) neuron. An additional inhibitory connection disables this connection when (*1*) is a negative angle. This creates a spring that only engages when the COM is falling toward the outside of the leg. It is up to the other leg to capture the COM if it falls to the inside of the leg.

11.6.2 Contract

The task of the *contract* network is to keep the leg laterally stable while the leg is lifted. If the leg is angled out to the left or right, it slowly brings the leg back toward the centre. Ideally the leg should be angled slightly underneath the torso, reducing the distance the COM needs to be shifted on each step.

A single neuron (*I*) is used to provide a small bias that sets the desired hip angle, velocity, and torque (Figure 11.14). Its connection to the hips velocity is relatively small so the machine doesn't jerk its leg but moves it slowly under the hip.

[Figure 11.14 here]

11.6.3 Swing and Extend

During the swing phase it is possible for the machine to begin to fall to the left or the right. Modifications must be made to reduce velocities along the *y* axis when this happens. A single neuron (*H*) is used to track the error between the desired lateral velocity (always 0) and the current lateral velocity (Figure 11.15). Neuron (*H*) receives inputs from the right (*10*) and left (*9*) velocity sensors and though (*G*), adjusts the desired hip target, velocity, and torque. The (*G*) neuron from the opposite leg's *support* network provides critical error correction information. Neuron (*G*) as discussed earlier in the *support* network, becomes excited when the COM shifts over the opposite leg. If (*G*) is active then the machine will be falling toward the opposite leg at a velocity proportional to (*G*). By adding this information to the (*H*) neuron we keep the leg from using foot placement to correct the shift velocity. This lateral velocity is a natural part of the gait and should not cause the legs to compensate for it. The *extend* network is identical to that of *swing.*

[Figure 11.15 here]

11.6.4 Stand

The *stand* mode becomes active when both legs are together and both feet are touching the ground. At this point the COM should be over one of its feet in case it needs to take a step in the future. In the planar machine the (*H*) neuron competed with the other leg's (*H*) neuron to decide which leg would be in *support* mode and which would become *contract*. In this stage the (*H*) neurons can receive input from the Hip X angle (*1*) forcing the leg that is supporting more of the COM to win. This winning state gently shifts the COM completely over its leg by slowly bending its knee. To keep the opposite leg relaxed during the weight shift (*G*) disables the connection between (*F*) and (*A*) when it is not firing. The resulting machine always keeps its weight over one of its feet.

11.6.5 Experiments

Experiments on a flat surface were conducted in the same manner as in stage one with a few minor changes. The forces normally applied to torso to prevent any movement other than along the sagittal plane (a virtual boom) were removed. An addition was made to the fitness function to minimize lateral movement from side to side. This improves stability by selecting individuals who shift their weight only when needed. The mutation rate was lowered to 0.01 after observation showed lateral stability was more sensitive to these kinds of changes. Fitness is defined as:

$$fitness = time * min(rot) * min(vel) * min(energy) * min(lvel)$$

where $min(t) = 1/(1 + t)$, *time* is the amount of time the machine walked (maximum is 30 seconds), *rot* is the absolute average error between 0 and pitch/roll/yaw angles of the torso, *vel* is the absolute average error between the desired velocity and the actual velocity, *energy* is the average torque used by all actuators, and *lvel* is the absolute average error between 0 and the velocity along the y axis.

A population was evolved for 190 generations on a flat surface. The average fitness in the first generation was 0.16 but over the next 80 generations rose to 0.6 and then flattened out over the next 110 generations.

As in stage one, when machines from the flat population above were placed on an irregular surface their fitness was poor averaging around 0.1. In this stage observations were made directly before attempting to continue evolution on a bumpy surface.

11.6.6. Observations and improvements

If a machine is built using just foot placement and weight shifting, it is capable of walking quite well on a flat surface. However, as in the first stage when placed on an irregular surface the machine often lost its balance. One reason could be that all of the innovations to handle *early foot* and *late foot strike* had been evolved while on a flat surface. The other observed cause was *foot tangling*. If the right foot of the machine stepped into a gully it caused the machine to shift too much weight to the right. To compensate the machine tried to move the left foot towards the right only to get it tangled up with the right leg (Figure 11.16).

[Figure 11.16 here]

The easiest way to modify the current machine is first to develop a small symmetric neural network that inhibits or disables a neuron if foot passing is allowed given the virtual leg angles and forward and backward velocity. Once computed it can be fed as an input to any state that requires it. The centre of the entire walking network is an ideal location for this new leg crossing network as it has access to all state sensors allowing a new sensor to be added (*13*). The basic method to keep the legs from getting tangled is:

1. *Side* = *left* or *right*
2. If the machine is moving forward while falling toward the *Side* and the *Side* leg is in front of the opposite leg, disable the opposite leg's ability to move toward the *Side*.
3. If the machine is moving backward while falling to the *Side* and the *Side* leg is in back of the opposite leg, disable the opposite leg's ability to move toward the *Side*.

11.6.7 Foot tangle network

The neural controller for foot tangle can be seen in (Figure 11.17). To determine whether the machine can move a leg laterally the following information must be considered:

1. What direction is the machine walking? Forward or backward?
2. What are the virtual leg angles?
3. What is the difference between these angles?

Neurons (*A*) and (*C*) create a winner take all circuit to determine which way the machine is walking. Neuron (*A*) takes input from the forward velocity sensor and (*C*) from the backward one. Due to slow time constants (*A*) and (*C*) are resistant to quick fluctuations in velocity. If (*A*) becomes active the machine is moving forward, if (*C*), backward. Neuron (*B*) estimates the virtual angle of each leg and (*D*) and (*F*) calculate the difference between the leg angles and fire when one leg is in front of the other. Neuron (*D*) triggers (*E*) when walking backwards and crossing is not allowed. Neuron (*F*) triggers (*E*) when walking forward and crossing is not allowed. An additional sensor (*13*) is used to propagate this information to each mode network.

[Figure 11.17 here]

On the *swing* and *extend* modes connections are made that disable the connections between the left velocity sensor when crossing is not possible (Figure 11.18, right). Due to the bilateral nature of the network the left and right sensors are swapped on the left side of the body. As a result each leg is prevented from moving inward under the body where it could get tangled up with the other leg when sensor (*13*) has a high activation.

Figure 11.18 shows the improvement in fitness after including the tangle network in the population seed. When the best machine is observed it shows quick movements when necessary both forward and laterally in response to environment disturbance. Figure 11.19 shows it quickly stepping in front of its right leg to try and capture the COM. Figure 11.20 illustrates how the machine can adjust its stride dynamically when moving over rugged terrain.

[Figure 11.18 here]

[Figure 11.19 here]

[Figure 11.20 here]

11.7 Spines, Hips, Arms and Toes

It is possible to build more complex walking machines using the techniques explored in this chapter. Preliminary experiments were done with jointed spines, rotational hip joints, arms, and toes. Each new control system was added on top of earlier successful ones. To add a spine, the body orientation sensors were moved to the head and a simple control system was used to balance the spine on top of the legs. Arms were added that passively swing from side to side on each step. An extra degree of freedom allowed the hip joint to rotate around the *z* axis and a spring loaded toe increased traction. Even with these radical changes evolution managed to integrate them into a natural walking gait. Figure 11.21 shows a machine with 35 degrees of freedom that can walk indefinitely. While this machine was constrained to a flat surface due to limitation of the simulation, it does show that such incremental methods can scale up to similar complexities currently explored by trajectory-based machines. This machine was called "Spine Walker".

[Figure 11.21 here]

The design of Spine Walker is based around ideas proposed by Raibert (1986) and Pratt and Pratt (1999). Like Raibert and Pratt's work this machine was built using hand-wired dynamic equations instead of neural circuits. When both approaches were compared, the neural network based approach used throughout this chapter appeared to be more evolvable. Although the more complex spine based machine did not use the neural circuits previously described, it still was built using staged evolutionary design and is another good illustration of our methodology. In this section some of the ideas revealed by work on the Spine Walker are discussed and how they could be transferred to the neural network based approach presented in earlier sections

The Spine Walker was developed in 4 main stages:

Stage 1. The machine was evolved to walk on a flat surface.

Stage 2. An extra degree of freedom was added to the hip.

Stage 3. A flexible toe joint was added.

Stage 4. A flexible spine was added.

Stage 5. A head and arms were added.

Hips with 3 degrees of freedom. Spine Walker has one more degree of freedom in the hip than the other machines in this chapter. This allows it to rotate the hip around the *z* axis the way humans are able to do. The control system for this is quite simple. In the *swing* mode the opposite leg's hip rotates around the *z* axis to bring the hip farther forward (Figure 11.22). This results in a more natural looking walk. In terms of the current circuit model, this could be done by creating connections from the *swing* mode of one leg to the *support* mode of the other. When the leg lifts off the ground and enters *swing* it can rotate the opposite leg's hip forward.

Flexible toe joint. A spring loaded toe joint is added to each foot to make the machine walk with a more human looking gait. This change was observed to increase stability by increasing traction with the ground as the heel was lifting. The increased traction reduced the chance of the body twisting (yaw) too much and losing its balance (Figure 11.22, right).

Spine with 9 degrees or freedom. The second stage is to add a spine on top of the hips that supported a weighted torso. The goal of the spine is to use an inclinometer in the torso to balance the torso on top of the hips. It uses a similar balancing algorithm to the one used by the *support* mode earlier in this chapter. Its only input is that of the inclinometer and its output the direction the spine should bend around 3 axes to support it. The control system bends the spine so it isolates movements in the hips so they do not affect the torso. This was taken from the observation that when people walk their upper torso tends to be relatively still compared to their hips. Once the spine is in place the population was evolved until the machine could walk without falling (Figure 11.22).

[Figure 11.22 here]

Head and Arms. The final stage is to add a head and passive arms to the machine. This proved relatively trivial. The arms are damped and allowed to swing freely. When a leg enters the *swing* mode it causes the shoulder on the opposite arm to swing forward slightly leading to a natural looking smooth gait. This could be done in our neural model by creating connections between the *swing* network and motor neurons in the opposite arm (Figure 11.23).

[Figure 11.23 here]

11.8 Conclusions.

The machine developed in this chapter was built upon experience gained from previous work combining evolutionary design of body and controllers with engineering insights about the problem domain. The basic concepts of walking were re-examined and simplified to foot placement, foot passing, and torso balancing. In foot placement simple linear relationships were shown between velocity and hip angle. To support foot passing the walking gait was broken down into five networks: *stand*, *support*, *contract*, *swing*, and *extend*. Each mode was hand-designed using simple neural circuits. A larger network was constructed that could modify these modes allowing only one active circuit per leg to be active at any given time. Through contraction and extension each foot could pass in front of the other. The *support* mode allowed the leg whose foot was touching the ground to dynamically support a weighted torso. If the torso began to tilt the hip was powered to capture its centre of mass.

The process of incremental design though seeding networks and stages continued to produce integrated networks with complex behaviour. In stage one the machine was first tested on a flat surface and then on an irregular one. Though observation four situations were found that reduced its fitness on rugged surfaces: early foot strike, late foot strike, loss of momentum when feet come together, and when both feet lost traction with the ground completely. The analysis of these issues led to network modifications to remove the problems. In stage two the virtual boom that constrained the machine to the sagittal plane was removed. This required modification of the five modes. As in stage one the machine was tested on both flat and rugged surfaces. On flat surfaces the machine could walk quite well even before being evolved. On rugged surfaces the machine tended to tangle its legs together when trying to regain lateral balance. A tangle network was designed that inhibited leg movement that might cause the legs to collide increasing fitness. This suggests that ankles are not necessary for basic walking but they can prevent twisting and inject energy into the gait. When walking normally the leg movements were slow and smooth but when destabilized due to a rugged surface the machine made quick movements to recover.

Preliminary work explores more complex machines including a jointed spine, arms, and extra hip joints. As before, each stage is built on top of earlier ones by adding additional simple control systems that evolution integrated into the whole. The result is a machine with 35 degrees of freedom that can walk infinitely on a flat surface.

Overall these machines show the power of incremental design using evolution. They can walk in any direction on smooth and rugged surfaces. They can walk slowly at 0.1 *m/s* or fast at 0.3 *m/s*. They can be scaled up to control many degrees of freedom mimicking human like movement using spines and arms. These machines suggest that passive dynamic walkers can be scaled up to a level of human like complexity that until recently were only explored by trajectory-based approaches. They walk with natural walking gaits in 3D space and can switch behaviours depending on the state of their environment. Like trajectory based machines when pushed they can recapture their centre of mass even on rugged surfaces in a controlled way and in the presence of a large number of degrees of freedom. However, these machines have the advantage of greater efficiency, speed, and simplicity. The control systems are relatively simple and could be implemented with cheap embedded controllers. They are fast, efficient, robust, and can perform different behaviours making them more practical for legged vehicles and lifelike toys.

To build a physical machine based on these simulations two issues need to be addressed: the need for low impedance actuators, and any mismatches between reality and simulation. There are solutions for the first issue, such as Series Elastic Actuators developed at MIT by Robinson et al. (1999), or the Programmable Spring developed at Sussex by Bigge (2010). For the second issue, there are known methods for 'crossing the reality gap' (Jakobi, 1998), and there are indications mentioned above, where at some new incremental stages walking was achieved even before further evolution, that the outcome of this design process is surprisingly robust, and hence more likely to be robust enough to cross the reality gap.

The techniques used here have achieved results beyond the former state of the art. The design insights brought out here are specific to bipedal walking machines, but it is hoped that this case study may also provide useful insights and lessons for people trying to use an incremental ER methodology in other domains.

References.

Beer, R. D. (1995). A dynamical systems perspective on agent-environment interaction. *Artificial Intelligence*, 72:173-215.

Bigge, W. T. (2010). *The programmable spring: towards physical emulators of mechanical systems*. Doctoral thesis, University of Sussex.

Brooks, R. A. (1991). Intelligence without Representation. *Artificial Intelligence*, 47:139-159.

Harvey, I. (1992). Species Adaptation Genetic Algorithms: a basis for a continuing SAGA. In Varela, F. J. and Bourgine, P. (eds) *Towards a Practice of Autonomous Systems: Proceedings 1st Eur. Conf. on Artificial Life*. MIT Press, Cambridge MA.

Harvey, I., Husbands, P. and Cliff, D. (1993). Issues in Evolutionary Robotics. In Meyer, J.-A., Roitblat, H., and Wilson, S. (eds), *From Animals to Animats 2: Proceedings 2nd Intl. Conf. on Simulation of Adaptive Behaviour (SAB92).* MIT Press, Cambridge MA.

Harvey, I. (2001). Artificial Evolution: A Continuing SAGA. In Gomi, T., (ed), *Evolutionary Robotics: From Intelligent Robots to Artificial Life.* Springer-Verlag LNCS 2217.

Harvey, I., Di Paolo, E., Wood, R., Quinn, M. and Tuci, E. (2005). Evolutionary Robotics: A new scientific tool for studying cognition. *Artificial Life*, 11(1-2):79-98.

Husbands, P. (1994). Distributed coevolutionary genetic algorithms for multi-criteria and multi-constraint optimisation. In Fogarty, T., (ed), *Evolutionary Computing, AISB Workshop Selected Papers*, volume 865 (LNCS), pages 150–165. Springer-Verlag.

Jakobi, N. (1998). *Minimal Simulations for Evolutionary Robotics*. Doctoral thesis, University of Sussex.

Kodjabachian, J. and Meyer, J.-A. (1998). Evolution and development of neural networks controlling locomotion, gradient following and obstacle avoidance in artificial insects. *IEEE Trans. Neural Networks*, 9:796-812.

Manoonpong, P., Geng, T., Kulvicius, T., Porr, B. and Wörgötter, F. (2007). Adaptive, Fast Walking in a Biped Robot under Neuronal Control and Learning. *PLoS Comput. Biol.,* 3(7): e134. doi:10.1371/journal.pcbi.

McGeer, T. (1990). Passive walking with knees. In *Proceedings of the IEEE Conference on Robotics and Automation,* volume 2, pages 1640-1645.

Raibert, M. H. (1986). *Legged Robots That Balance*. Cambridge, MA: MIT Press.

Robinson, D. W., Pratt, J. E., Paluska, D. J., and Pratt, G. A. (1999). Series Elastic Actuator development for a biomimetic robot. *IEEE/ASME International Conference on Advanced Intelligent Mechatronics*.

Pratt, J. and Pratt, G. (1999). Exploiting natural dynamics in the control of a 3d bipedal walking simulation. In *Proceedings of the International Conference on Climbing and Walking Robots (CLAWAR99)*, Portsmouth, UK.

Vaughan, E., Di Paolo, E., and Harvey, I. (2004a). The evolution of control and adaptation in a 3d powered passive dynamic walker. In *Proceedings of the Ninth International Conference on the Simulation and Synthesis of Living Systems, ALIFE'9* Boston. Cambridge, MA: MIT Press.

Vaughan, E., Di Paolo, E., and Harvey, I. (2004b). The tango of a load balancing biped. In *Proceedings of CLAWAR 2004, 7th International Conference on Climbing and Walking Robots*.

Vaughan, E. (2007) *The Evolution of the Omni-directional Bipedal Robot*. Doctoral thesis, University of Sussex.

Tables

Table 11.1

The sensor neurons supplied to each mode circuit.

Sensor Type	Description
Hip X Angle	The current angle of the hip joint as it rotates around the x axis.
Hip Y Angle	The current angle of the hip joint as it rotates around the y axis.
Knee Angle	The current angle of the knee joint as it rotates around the y axis.
Foot Touch	Becomes 1 when the foot is touching the ground, 0 otherwise.
Forward Pitch	The angle of machine's torso as it tilts forward.
Backward Pitch	The angle of machine's torso as it tilts backward.
Right Roll	The angle of machine's torso as it tilts right.
Left Roll	The angle of machine's torso as it tilts left.
Forward Velocity	Velocity of the machine's torso as if moves forward.
Backward Velocity	Velocity of the machine's torso as if moves backward.
Right Velocity	Velocity of the machine's torso as if moves right.
Left Velocity	Velocity of the machine's torso as if moves left

Figure Captions

Figure 11.1

Walking gait with jointed knees. The dotted line denotes the virtual hip angle used for foot placement.

Figure 11.2

Body of walking machine with axes of movement.

Figure 11.3

Mode diagram. Each leg has its own mode independent of the other.

Figure 11.4

Neural network architecture for implementing each mode.

Figure 11.5

Network circuit design for *support* mode.

Figure 11.6

Network circuit design for the *contract* mode.

Figure 11.7

Network circuit design for the *swing* mode.

Figure 11.8

Network circuit design for the *extend* mode.

Figure 11.9

Early foot-strike (top), strategy for overcoming early foot-strike (bottom).

Figure 11.10

Late foot-strike (top), strategy for overcoming late foot-strike (bottom).

Figure 11.11

Addition of the *stand* mode.

Figure 11.12

Network design for the *stand* mode. This mode inherits all the connections of the *support* network while adding functionality. Inherited connections are not shown.

Figure 11.13

The *support* network for lateral movement outside of the sagittal plane.

Figure 11.14

The *contract* network for lateral support.

Figure 11.15

The *swing* network for lateral movement outside the sagittal plane.

Figure 11.16

Foot tangling: while walking on an irregular surface the machine begins to fall to its left. To compensate it swings its right leg directly into the left knocking its hip out of its socket.

Figure 11.17

Implementation of the *tangle* network (left); modifications to the *swing* and *extend* networks (right).

Figure 11.18

Fitness after modifications to prevent foot tangling.

Figure 11.19

A machine stepping in front of its right leg to try to capture its COM.

Figure 11.20

Front view of machine recovering as it steps in a gully on 5 cm surface.

Figure 11.21

Preliminary walker with spine, head, ankles, and arms. The machine had 35 degrees of freedom: 2 in each ankle, 1 in each knee, 3 in each hip, 9 in the spine, 3 in each shoulder, 1 in each elbow, 2 in each wrist, and 1 in each toe. The toe joint used a simple linear spring that provided additional traction as the heel lifted off the ground. This tended to prevent and twisting of the body as it took each step.

Figure 11.22

Hip Movement of Spine Walker. The upper three frames show how the hips can rotate around the z axis by the addition of an extra degree of freedom. The lower three frames show how the hips can move from side to side due to a gentle flexing of the spine.

Figure 11.23

Spine Walker walking on a flat surface.