Machine Optimisation of Dynamic Gait Parameters for Bipedal Walking

Martin Mason[1], Peter Gibbons[1], Manuel Lopes[1], Francisco Melo[2]

[1]*School of Computing and Mathematics, University of Plymouth Plymouth, PL48AA, UK*

[2]*INESC-ID, Lisbon, Portugal*

Abstract— **This paper describes a compact gait generator that runs on the on-board Atmega microcontroller of the Robotis Bioloid robot. An overview of the parameters that effect dynamic gait is included along with a discussion of how these parameters are implemented in a servo skeleton. The paper reports on the optimisation of the gait parameters using a variant of natural actor-critic method and demonstrates that this learning technique is applicable in a real context. The quality of a parameter set is evaluated based on minimising the time for the robot to travel one meter. The learned gait parameters result in a faster robot than the hand optimised parameters when the learning algorithm has a reasonable initialisation.**

I. Introduction

One of the primary tasks of a bipedal robot is to walk quickly and reliably. One strategy to achieve this is to have a robust dynamic gait that is stable under small perturbing external forces. This has been investigated previously in [1-3]. The gait under investigation has no sensor feedback and thus must be highly stable to deal with the wide variety of environmental conditions that will be encountered. To achieve such a gait it is critical to understand the various variables that influence biped locomotion stability and optimise these variables to suit a variety of different environments.

In this work, the gait is defined utilizing a set of periodic functions to control the foot trajectory. The parameters of these periodic functions can be modified in real time based on the desired speed and orientation of the robot. Since optimal gait parameters are a function of the walking surface and current payload they must be optimised for each new configuration. Manually tuning the gait parameters is extremely time consuming and inappropriate for online use. In addition there are substantial non-linearities in the robot servo skeleton that prevent it from being modelled analytically. A reinforcement learning approach using the fitted natural actor critic algorithm (FNAC) [4] was used to automatically optimise these parameters.

The use of such a learning approach has several advantages. First of all, it provides a general principled method to "program" different gaits for different surfaces and payloads. Secondly, it requires no modelling of either the kinematics/dynamics of the robot or the surface. Instead, the robot uses the outcome of the several trials to learn a robust way of walking. While many possible reinforcement learning algorithms have been applied to bipedal walking, the fitted actor critic method has the advantage that is uses an importance sampling strategy that allows the reuse of data, making the algorithm very efficient in terms of data usage since it uses all sampled data in all iterations of the algorithm unlike most other RL methods. In this practical problem, collecting data is costly and time consuming. The efficient use of data in FNAC is a significant advantage over other existing approaches.

In this paper we introduce a model of critical gait parameters, and present the results of applying FNAC for optimisation of these gait parameters.

A bipedal humanoid platform [5] based on the Bioloid servo skeleton by Robotis (Fig. 1) is used as a test bed to develop a set of parameters to optimize a dynamic gait for competition in Robot Football.

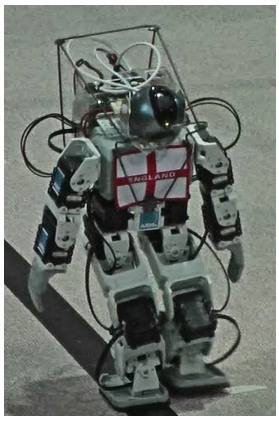

Figure 1: University of Plymouth Biped

II. Overview of Dynamic Gait

A dynamically stable gait is one where the centre of mass of the robot is often outside the support polygon. This introduces a period of instability in the gait cycle that requires the oscillating motion of the legs to be designed appropriately. If the parameterisation of the gait is poor, the walk will be unstable resulting in repeated falls. Prior experience in hand tuning gait parameters [9] has led to dynamically stable gaits on this platform.

1) Inverse Kinematics for Positioning

In order to create a stable gait we first would like to be able to position the robot's foot precisely in space. An inverse kinematic model of the legs of the robot is created by modelling the robots legs as a two link system. The accessible plane of operation for each leg is segmented into a 70x150

grid (Fig. 2) and the joint angles are pre-computed for each of these points and saved as a lookup table.

The lookup table is sized to occupy 64KBytes and stored in the embedded processor of the robot. When desired foot coordinates are calculated from the gait driving function, the joint angles are retrieved from the lookup table and applied to position the foot at a specified coordinate within the plane. If the hip rotates, the plane of the leg is rotated and the same lookup table for the foot positioning can be maintained.

2) Dynamic Gait Parameters

To establish the dynamic gait, a sinusoid is mapped to the coordinates of the centre of each foot with the parameters of stride amplitude and period.

The lateral (X) and vertical (Y) coordinates of the first foot are calculated according to the following function:

$$X = Stride\_Length * \cos(\frac{2\pi t}{T}) + X\_Offset$$

$$Y = Stride\_Height * \sin(\frac{2\pi t}{T}) + Y\_Offset$$

These two functions create an ellipsoid with a horizontal axis of two times the stride length and a vertical axis of two times the stride Length. The second foot is mapped out of phase by π radians from the first foot. At the current state of development, the foot remains flat during the trajectory, so the value calculated for the hip servo is just mirrored to the planar foot servo.

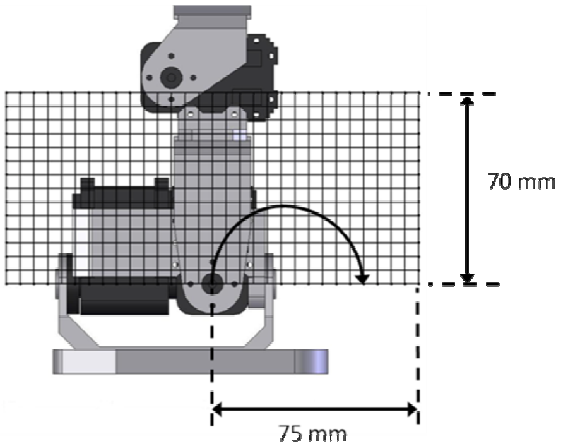

Figure 2: The joint angles for each of the grid points are pre-computed and stored in the onboard microprocessor. The curve indicates the trajectory of the foot as described by the parametric equations above. The horizontal axis of the ellipse is parameterised by the stride length and the vertical axis is parameterised by the stride height.

The servo velocities are calculated by determining the rotation angle of each servo and then requiring that all of the servos take the same amount of time to complete a given action. This basic sinusoidal function serves as a building block for a dynamic gait, but a number of static and dynamic parameters must also be considered.

3) Static Parameters

The following properties are static in that they remain as constants that affect the posture of the robot.

i. Tilt

To maintain a stable dynamic walk, the robot's centre of mass should be positioned slightly forward. Depending on the geometry of the robot and its current mass configuration, the robot should be inclined forward by adding some additional rotation as an offset to the Planar Hip servos.

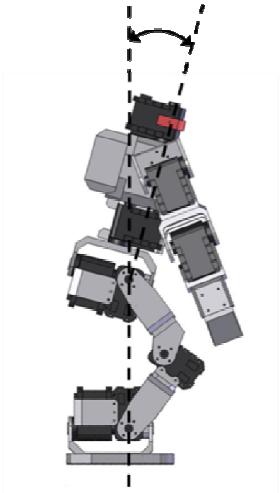

Figure 3: The tilt inclines the centre of mass of the robot to counteract the accelerations during forward movement

ii. Body Offset

This defines how erect the robot stands. When the vertical component of the foot's position is calculated, a constant offset is added to change the overall height of the stance. Since the origin is defined as the foot coordinates with the legs fully straight, without a body offset, the calculated vertical positions will become negative and thus out of bounds for the inverse kinematic model.

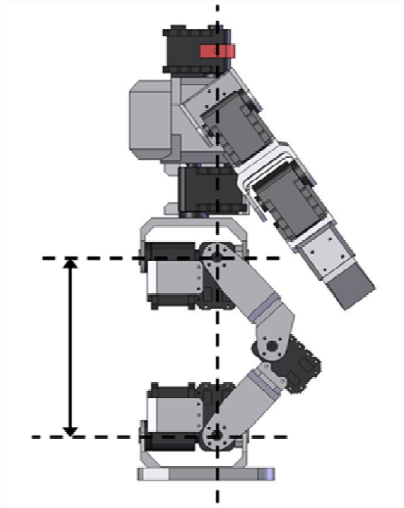

Figure 4: The body offset shifts the origin of the coordinate system through which the foot moves.

iii. Camber:

This adds an offset to the rotational hip servo and a mirroring offset to the rotational foot servo. With a camber of zero, the robot stands with its legs aligned parallel. As a negative camber is added, the legs spread outward providing a force pointing inward which helps to increase stability at the expense of increased turning effort.

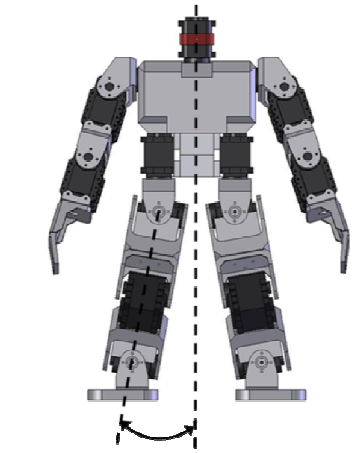

Figure 5: Camber introduces a rotational offset on the hip and ankle joints to increase lateral stability and prevent collisions between servo skeleton elements.

4) Dynamic Parameters

These parameters change as a function of time during the motion of the robot. The overall motion of the legs was discussed previously as a sinusoidal function mapped to the plane defined by the vector from the hip to the knee and the vector from the knee to the ankle as shown.

i. Swing

In order for the robot to move, it has to shift its centre of mass from one foot to the other. In this simple implementation, a linear function is mapped to the rotational hip servos with the same period as the sinusoid and a mirrored function is mapped to the rotational foot servos (Fig. 7). This causes the robot to sway back and forth continually shifting its centre of mass from one foot to the other.

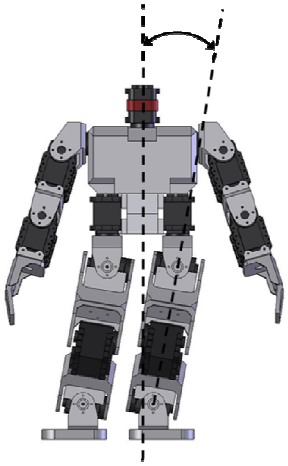

Figure 6: Swing Amplitude controls how far the centre of mass of the robot shifts.

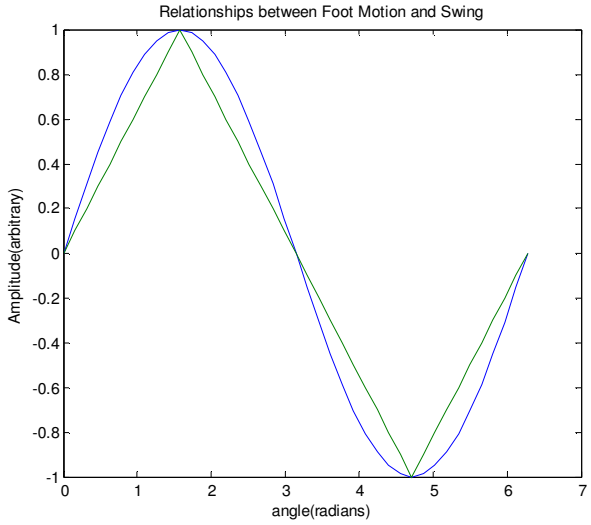

Figure 7: The relationship between the swing of the hips (straight line) and the motion of the feet (curved line).

The following figures (Fig. 8 and 9) summarise the parameters that determine the gait and how they are used in calculation of the servo joint angles.

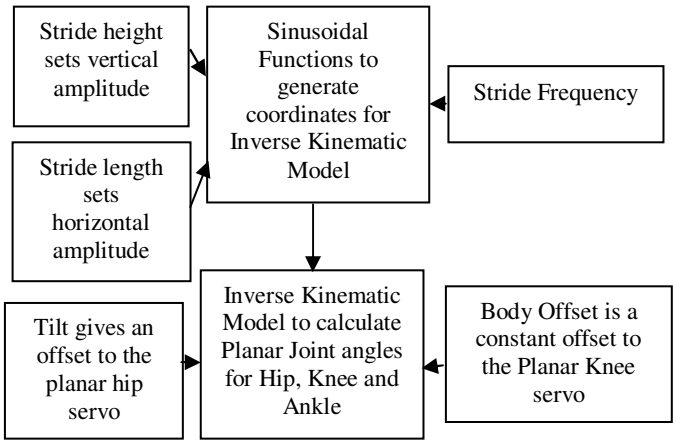

Figure 8: Overview of Planar Servo Calculations.

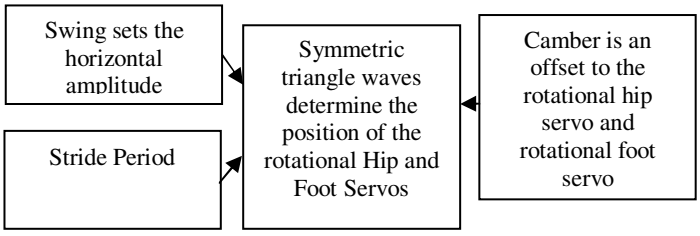

Figure 9: Overview of Rotation Servo Calculations.

III. Gait Optimization:

The goal is to find the parameters of the dynamic gait that produce the fastest possible travel while maintaining robustness in the face of perturbing external forces. The

travel speed was measured directly by timing the robot through a standardised one meter course.

A. Machine Optimisation of Gait Parameters

In the previous section, we described our approach in which the gait is controlled by nine unique parameters. In a learning setting, this means that the robot should determine the "best" choice for these parameters for different floor surfaces and payloads. For learning, a reinforcement learning (RL) approach is adopted in which the robot learns the gait while walking and attempts different parameter sets until one is found that produces the greatest travel speed.

1) Fitted Natural Actor Critic Learning (FNAC):

In the remainder of this section, the fitted natural actor-critic (FNAC) algorithm is described. Reinforcement learning is a class of methods designed to address optimal control problems in the presence of incomplete knowledge about the dynamics of the system to be controlled. Classical RL approaches focused on problems with finite state/control spaces. Two major approaches have been considered in the RL literature for addressing problems with large/infinite state and/or control spaces such as the one addressed in this work. Regression-based methods use sample data collected from the system to estimate some target utility function using regression techniques. The utility function is then used to extract an optimal/near optimal control signal which is used to have the system perform the desired task.

This class of methods is particularly suited to address problems with infinite state-spaces and can take advantage of the numerous regression methods available from the machine learning literature while exhibiting solid convergence properties [6], [7]. Gradient-based methods, on the other hand, are naturally suited to address problems with infinite action-spaces. Such methods consider a parameterised controller and estimate the gradient of the performance with respect to the policy parameters. The parameters are then updated in the direction of this estimated gradient. By construction, gradient-based methods implement an incremental policy optimisation and thus avoid the need for explicit maximisation; it is no surprise that many RL works addressing problems with continuous action spaces thus rely on a gradient-based architecture [8], [9].

The particular RL algorithm used in this paper, FNAC [4] combines the potentially faster convergence of natural gradients [10] and the sound convergence properties of regression algorithms [7]. It also uses importance sampling to allow the reuse of data and thus make the algorithm efficient in terms of data usage.

To describe how the FNAC algorithm can be used in this setting, we denote by p the vector of parameters of the controller, henceforth designated as the control vector. The purpose of the learning algorithm is, therefore, to compute a control vector yielding a high speed gait. In this case, the control vector p takes values in a compact subset of $\mathbb{R}^9$ that is the control space. Finally, for a particular control vector p, let $T_{travel}(p)$ denote the travel time associated with p, i.e. the total time taken by the robot to walk one meter given the control vector p. It is defined that $T_{travel}(p) = \infty$ whenever the robot falls or otherwise fails to reach the 1m target. Let π_θ be a probability distribution over the control space, parameterized by some finite-dimensional vector $\theta \in R^M$ henceforth referred to as a policy.

The value associated with π_θ is defined as

$$\eta(\theta) = \mathbb{E}\left[\frac{1}{T_{travel}(p)}\right] = \int \frac{1}{T_{travel}(p)} \pi_\theta(p) dp$$

and we define the advantage associated with a control vector p as

$$A_\theta(p) = \left[\frac{1}{T_{travel}(p)}\right] - \eta(\theta)$$

We are now in position to describe the FNAC algorithm (see Fig. 10). The algorithm uses a set D of samples obtained from a set of successful and unsuccessful walk attempts, each sample consisting of a pair $(p_i, T_{travel}(p_i))$, where p_i is a sample control vector and $T_{travel}(p_i)$ is the corresponding travel time. For the purposes of the algorithm, it is not important how the samples in D are collected. The samples can be collected before the algorithm is run or they can be collected incrementally, as more iterations of the algorithm are performed.

At each iteration k of the FNAC algorithm, the data in D is processed by the critic component of the algorithm. This component estimates the value $\eta(\theta_k)$ associated with the current policy, $\pi_{\theta k}$, by solving the regression problem

$$\hat{\eta}_k = \underset{v}{\operatorname{argmin}} \sum_i \frac{\pi_{\theta k}(p_i)}{\pi_0(p_i)} \left(\frac{1}{T_{travel}(p)} - v\right)^2$$

where π_0 is the policy used to obtain the samples in the dataset D. This estimate is then used to compute a linear approximation of the advantage function, $A_{\theta k}$. In other words, given an estimate $\hat{\eta}_k$ of $\eta(\theta_k)$ the advantage function $A_{\theta k}$ is approximated by solving the following regression problem:

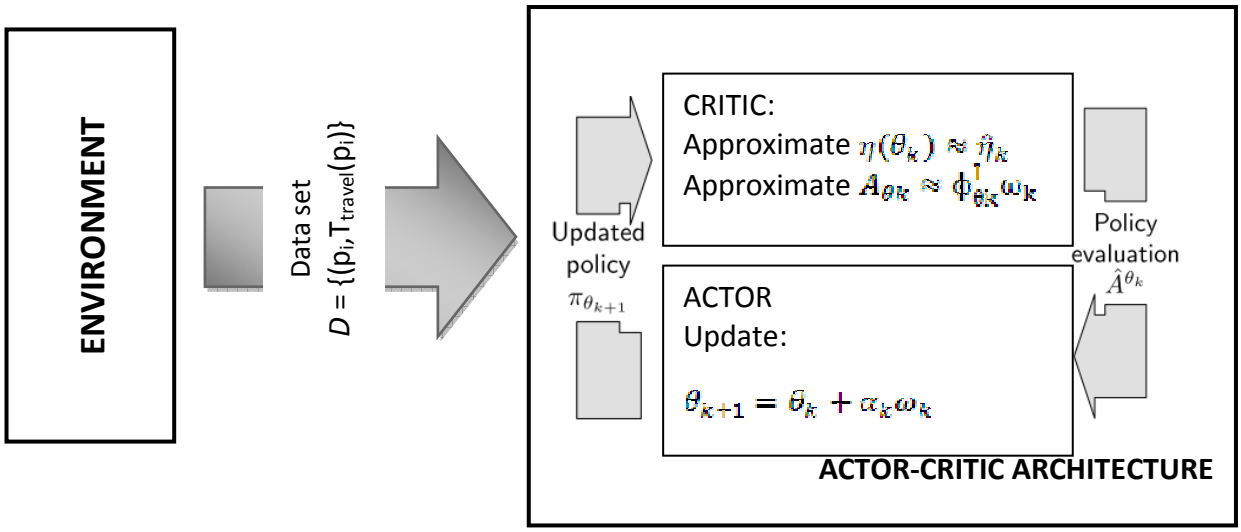

Figure 10: Schematic Representation of the Finite Actor Critic Architecture.

$$\omega_k = \text{argmin}_\omega \sum_i \left(\frac{1}{T_{travel}(p)} - \hat{\eta}_k - \nabla^{\mathrm{T}}_{\theta k} \log\left(\pi_{\theta k}(p_i)\right)\omega \right)^2$$

where T denotes the transpose operator. The above regression problem can easily be solved by setting

$$\boldsymbol{M} = \sum_i \nabla_{\theta k} \log\left(\pi_{\theta k}(p_i)\right) \nabla^{\mathrm{T}}_{\theta k} \log\left(\pi_{\theta k}(p_i)\right)$$

and

$$\boldsymbol{b} = \sum_i \nabla_{\theta k} \log\left(\pi_{\theta k}(p_i)\right) \left(\frac{1}{T_{travel}(p)} - \hat{\eta}_k \right)$$

from where it is found that $\omega_k = \mathbf{M}^{-1}\, \boldsymbol{b}$. The details of the derivations above fall out of the scope of this paper, since they would require the introduction of a significant amount of new notation and nomenclature. Detailed derivations of more general forms of the above expressions are given in [4]. The actor component of the FNAC algorithm simply implements the natural gradient update which, given the parameterised policy at iteration k, $\pi_{\theta k}$, updates the parameter vector θ_k as

$$\theta_{k+1} = \theta_k + \alpha_k \omega_k$$

where ω_k is the linear coefficient vector corresponding to the approximated advantage function $A_{\theta k}$ computed by the critic.

IV. EXPERIMENTAL EVALUATION

A custom firmware for the Bioloid processor was developed that contains the inverse kinematics for the Bioloid servo skeleton and all of the functions detailed above. A graphical front end passes variables to the processor through a wireless serial connection. The front end allows the user to dynamically alter all of the parameters of the model and quickly implement new parameter values in addition to allowing you to drive the robot remotely.

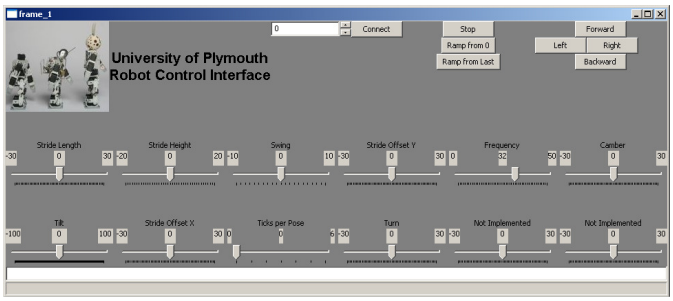

Figure 11: Graphical interface allows quick adjustment of gait parameters.

The performance of the FNAC learning method is evaluated using the Bioloid servo skeleton [5]. As seen in Section 1, the servo skeleton used in the experiments has nine parameters that describe the gait. The experiments proceeded as follows. At every trial a set of gait parameters is loaded into the robot. The initial position, configuration of the robot, and walking surface is always roughly the same. The robot is then initialised with the set of gait parameters and begins to walk toward the target line. The walk is considered a success if the robot does not fall and remains within a one meter wide lane during the walk. The time required to travel one meter is measured. Table 1 shows the constraints that were placed on the parameters based on the physical limitations of the servo skeleton.

Table 1. Gait parameter constraints.

Gait Parameter	Minimum	Maximum
Stride Height	0 mm	20 mm
Stride Length	0 mm	30 mm
Swing	0 mm	15 mm
X Offset	-30 mm	30 mm
Y Offset	0	40 mm
Camber	-10 degrees	30 degrees
Period	250 ms	1000 ms

Recall from Section III that the robot learns the parameters of a policy π_θ, from a dataset D of pairs (p_i , T_{travel} (p_i)). The initial sample vectors p_i were derived from a dataset of gait parameters generated from a hand tuned exploration of part of the parameter space detailed in [11]. These p_i vectors were drawn within the above ranges and the corresponding travel time measured after running the controller. These samples were processed by the algorithm and, as the policy π_θ improved, new samples were generated and used for subsequent learning. The learned policy π_θ consisted of a (one-dimensional) Gaussian distribution for each of the nine elements of the control vector, parameterised by the corresponding mean and variance. The goal of the algorithm is to find the parameters for the resulting 9-dimensional Gaussian distribution that minimise the average travel time.

The first several iterations of the algorithm resulted in failures. The first successful traverse was in trial 5. After this first success, the robot used the FNAC algorithm to compute a new policy after 20 trials. The control vector selected at each trial was chosen using the learned policy π_θ with probability $1 - \epsilon$ and drawn uniformly with probability ϵ. In this case ϵ was set to 0.1. For each sampled vector, the corresponding travel time was measured from the moment the robot was allowed to move, not when movement was observed. Each trial was observed to end when the robot completely crossed the one meter line, left its lane or fell over. A failure may occur due to either the robot being significantly out of balance, resonances in the gait system or a large bias in one direction causing the robot to veer significantly. Figure 12 shows a step of a successful gait and a step from an unsuccessful gait.

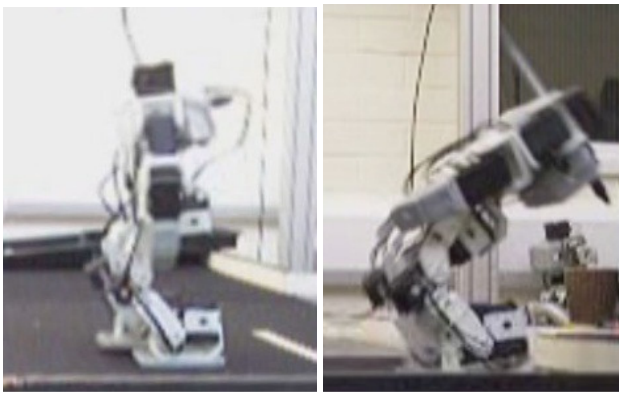

Figure 12: A successful gait is shown on the left and an unsuccessful gait is shown on the right.

The reward was calculated for each successive run, based on the travel time, as T_{max}- T_{travel}. If the run was a failure due to the robot falling over the reward was defined at -10. If the run failed due to the robot veering to one side or the other, the reward was defined as -1000. Figure 13 shows the evolution of the reward. The reward rises on average once a successful control vector is found with some fluctuations due to exploration. In particular, the fluctuations happened during exploration trials.

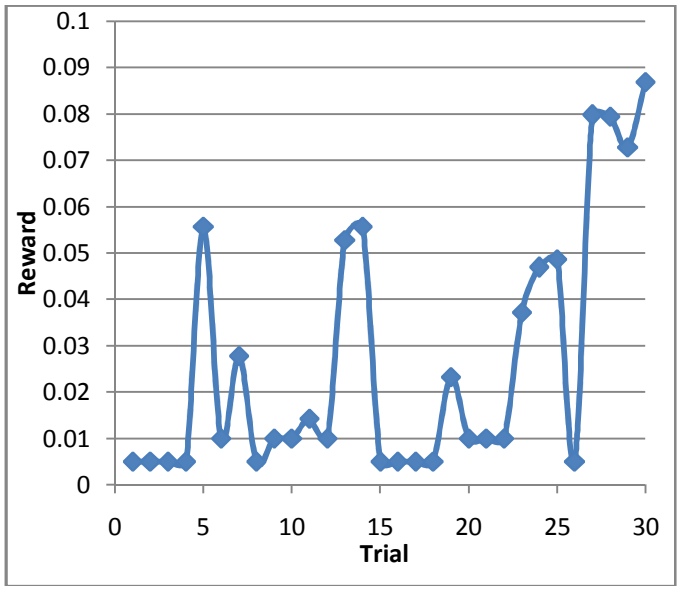

Figure 13: Evolution of the reward. Larger values on the plot correspond to a more successful walk.

Since the optimal policy is not known, the error can only be estimated based on the best control vector. Fig. 14 shows the evolution of the difference between the parameters of the policy at each iteration and the best control vector (the "error"). As can be seen from the plots in Figure 14, the algorithm does approach the best control vector (the "error" does decrease in absolute value). However, since the algorithm takes into account every sample observed so far, it will output a policy closer to the "mean" of all successful control vectors.

Figure 14: The error associated with three of the parameters is shown. The stride length error decreased rapidly and then was relatively constant. The Y offset error oscillated significantly over the course of the experiment. The Tilt error was relatively constant and only decreased at the end of the experiment.

Figure 15 below shows the sum of all the errors over the course of the experiment. Some of the parameters have a much larger effect on the gait stability so a small change in one parameter has a much larger effect then a large change in a different parameter.

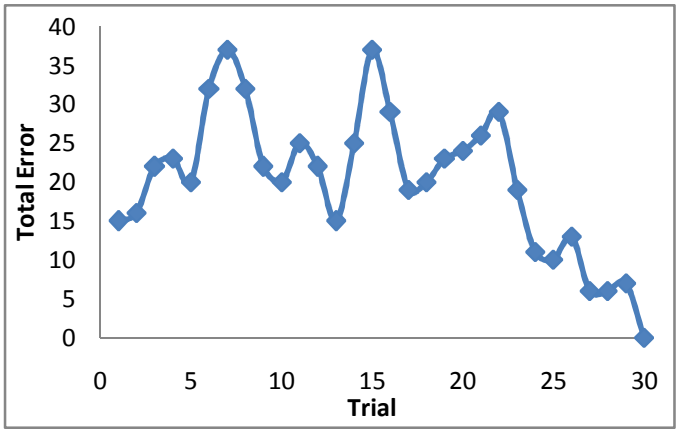

Figure 15: Total error as a function of trial.

The experiment was repeated with an additional initial set of parameters derived by randomly sampling the parameter space. After 20 unsuccessful trials, it appeared that the FNAC algorithm was not successful at randomly exploring a large parameter space.

V. Conclusion and Further Work

The learned best set of parameters resulted in a travel time of 11.1 ± 0.1 seconds. Using the identical robot with the best hand tuned parameters resulted in a travel time of 13.8 ± 0.1 seconds. At the initialisation of the learning process the robot took over 30 seconds to complete the course. The learned parameters showed a significant improvement from the initialisation and a small but repeatable improvement over the hand optimised parameters. The FNAC algorithm was successful in optimising the gait parameters given a reasonable initialisation.

In addition, the FNAC algorithm generated stable gaits that used substantially different sets of parameters than those found for the hand optimised gait. This suggests that the technique should generalise to different surfaces and environmental conditions.

Further work is required in the project including testing how well this technique generalises to different surfaces, robot configurations and payloads. During the study, the arms were held fixed parallel to the sides of the biped. Arm oscillations can provide additional stabilisation [11] and need to be investigated. The trajectories through the parameter space that allow for stable transitions between different locomotion speeds and rotations need to be investigated. With the tools developed, these additional investigations can be performed in a methodical way and should lead to increased optimisation of the biped gait.

References

[1] Mayer N, Ogino M, Guerra, R, Boedecker, J, Fuke S, Toyama H, Watanabe A, Masui K, Asada M "JEAP Team Description" RoboCup Symposium, 2008.

[2] S. Behnke, Online trajectory generation for omnidirectional biped walking, in: Proc. IEEE International Conference on Robotics and Automation, 15-19 May 2006, pp. 1597-1603.

[3] Eric R. Westervelt, Jessy W. Grizzle, Christine Chevallereau, Jun-Ho Choi, and Benjamin Morris, Feedback Control of Dynamic Bipedal Robot Locomotion, Taylor & Francis/CRC Press, June 2007.

[4] F. Melo and M. Lopes, "Fitted natural actor-critic: A new algorithm for continuous state-action MDPs," in Proc. European Conf. Machine Learning, 2008, pp. 66–81.

[5] Wolf J, Vicente A, Gibbons P, Gardiner N, Tilbury J, Bugmann G and Culverhouse P (2009) in Proc. FIRA RoboWorld Congress 2009, Incheon, Korea, August 16-20, 2009 (Springer Progress in Robotics: ISBN 978-3-642-03985-0) . pp. 25-33.

[6] R. Munos and C. Szepesvári, "Finite-time bounds for sampling-based fitted value iteration," J. Machine Learning Research, vol. (submitted), 2007.

[7] A. Antos, R. Munos, and C. Szepesvári, "Fitted Q-iteration in continuous action-space MDPs," in Adv. Neural Information Proc. Systems,vol. 20, 2007.

[8] J. Peters, S. Vijayakumar, and S. Schaal, "Natural Actor-Critic," in Proc. 16th European Conf. Machine Learning, 2005, pp. 280–291.

[9] A. Lazaric, M. Restelli, and A. Bonarini, "Reinforcement learning in continuous action spaces through sequential Monte Carlo methods," in Adv. Neural Information Proc. Systems, vol. 20, 2007.

[10] S. Kakade, "A natural policy gradient," in Adv. Neural Information Proc. Systems, vol. 14, 2001, pp. 1531–1538.

[11] Gibbons, P, Mason, M. Vicente, A., Bugmann, G. and Culverhouse, P. Optimization of Dynamic Gait for Bipedal Robots (2009), Proceedings of 4th Workshop of Humanoid Soccer Robots IEEE Humanoids 2009.

[12] Shibukawa M, Sugitani K, Kasamatsu K, Suzuki S, Ninomija S (2001) "The Relationship Between Arm Movement and Walking Stability in Bipedal Walking" Accessed online Oct 1 2009. URL handle.dtic.mil/100.2/ADA409793.