Probabilistic Gradient Ascent with Applications to Bipedal Robotic Locomotion

David Budden[1,2], **Josiah Walker**[3], **Madison Flannery**[3] **and Alexandre Mendes**[3]
[1] National ICT Australia (NICTA), Victoria Research Lab
[2] The University of Melbourne, Parkville, VIC 3010, Australia.
[3] The University of Newcastle, Callaghan, NSW 2308, Australia.
david.budden@nicta.com.au

Abstract

Bipedal robotic locomotion is an emerging field within the multi-billion dollar robotics industry, with global initiatives (such as RoboCup, FIRA and the DARPA Robotics Challenge) striving toward the development of robots able to complete complex physical tasks within a human-engineered environment. This paper details the redevelopment of an omni-directional walk engine for the DARwIn-OP, with an improved online optimisation framework developed for 13 of its internal parameters. Applying two well-known optimisation algorithms within this framework yields significant improvement in walk speed and stability. A new non-convex optimisation algorithm (Probabilistic Gradient Ascent) is derived from a reinforcement learning framework and applied to the same task, yielding an average speed improvement of 50.4% and setting a new maximum speed benchmark of 34.1 cm/s.

1 Introduction

1.1 Bipedal Humanoid Locomotion

Mechanical legged locomotion has been motivated historically by the practical advantages exhibited over wheeled locomotion: the ability to move across rough terrain, or to navigate environments featuring discontinuous supports (such as the rungs of a ladder). As a subclass of legged robots, bipedal robots exhibit diverse sociological and commercial advantages. The modern developed world is engineered for humans, by humans, and it is within such environments that robots are required to fulfil their design purpose; whether it be replacing humans in hazardous occupations (the focus of the current DARPA Robotics Challenge [Dietsch, 2012]), or restoring motion to paraplegics through dynamically controlled lower-limb prostheses [Westervelt *et al.*, 2007].

This work focuses on the Robotis DARwIn-OP bipedal robot platform (illustrated in Figure 1), which was adopted by the University of Newcastle's NUbots in 2012 for competing in the RoboCup Humanoid League[1] [Annable *et al.*, 2013]. As a recently developed platform, little published research focuses on improving or benchmarking the locomotion performance of the DARwIn-OP robot. In contrast, many publications focus on the development, improvement and optimisation of walk engines for the Aldebaran NAO [Kulk and Welsh, 2008; 2011a; 2011b], which has been used in the RoboCup Standard Platform League[2] since 2008. This research was considered in selecting an appropriate walk engine for the DARwIn-OP, which is similar in both physical size, degrees-of-freedom and servo layout.

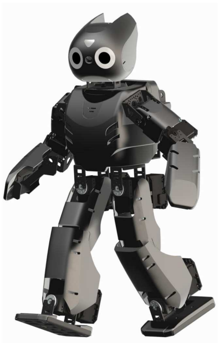


Figure 1: The Robotis DARwIn-OP bipedal robot platform, which was adopted by the University of Newcastle's NUbots in 2012 for competing in the RoboCup Humanoid League.

[1] The RoboCup Humanoid League is an annual international soccer competition between teams of autonomous bipedal robots, with emphasis placed on hardware development. For more information, see `http://www.tzi.de/humanoid/`.

[2] In contrast to the Humanoid League, the RoboCup Standard Platform League requires all teams to adopt an identical robot platform (currently the Aldebaran NAO). For more information, see `http://www.tzi.de/spl/`.

Default Open-Loop

The default walk engine provided with the NAO is a simple, open-loop method. It operates by maintaining the robot's center of mass directly above the area supported by both feet. The resultant gait suffers a number of drawbacks: lack of robustness in stability when exposed to the various surfaces encountered at RoboCup, lack of omnidirectional walking ability, and a low maximum speed of approximately 10 cm/s on the NAO.

Improved Open-Loop

For RoboCup 2008, an improved version of the default NAO walk engine was developed [Kulk and Welsh, 2008]. This version involved the modification of low-level positional controllers to minimise the rigidity of motor joints at various stages of the walk pattern. Improvements in stability and power efficiency were observed, in addition to an increase of maximum walk speed from 10 to 14 cm/s on the NAO[3]. As this method is based on the default NAO walk, it does not allow for omnidirectional walking ability; interpolation between the four basic walk directions is not supported.

Omnidirectional Closed-Loop

For RoboCup 2010, the University of Bremen's B-Human SPL team introduced the first walk engine based on a complete, analytical solution of the NAO's inverse kinematics [Graf *et al.*, 2009]. This omnidirectional walk engine maintains the center of mass above the support area by modeling the robot as an inverted pendulum, with center of mass position and velocity given by

$$\begin{bmatrix} x(t) \\ \dot{x}(t) \end{bmatrix} = \begin{bmatrix} \cosh(kt) & \frac{1}{k}\sinh(kt) \\ k\sinh(kt) & \cosh(kt) \end{bmatrix} \begin{bmatrix} x_0 \\ \dot{x}_0 \end{bmatrix},$$

where $k = \sqrt{\frac{g}{z}}$, g is the gravitational constant and z is the height of the center of mass relative to the ground. This method was able to produce walking speeds of 15, 10 and 9 cm/s, for forward, reverse and coronal walks respectively, with a rotation speed of 35 degrees/s.

1.2 Walk Engine Implementation

The B-Human walk engine was chosen for implementation on the DARwIn-OP platform, and forms the basis of the following optimisation work. Although similar in physical size and servo layout, transferring this code between platforms required two significant steps: the decoupling of hip pitch and yaw, which is actuated by a single servo on the Aldebaran NAO; and the inclusion of an accurate DARwIn-OP mass model [Michel, 1998].

1.3 Non-Convex Optimisation Techniques

The standard model of a continuous optimisation problem is given by

$$\begin{array}{ll} \text{maximize} & f_0(\boldsymbol{\theta}) \\ \text{subject to} & f_i(\boldsymbol{\theta}) \le b_i, \quad i = 1, \ldots, m, \end{array}$$

where $\boldsymbol{\theta} = (\theta_1, \ldots, \theta_N)$ are the optimisation parameters, $f_0 : \mathbb{R}^N \to \mathbb{R}$ is the objective function (or fitness function), $f_i : \mathbb{R}^N \to \mathbb{R}$, $i = 1, \ldots, m$ are the inequality constraint functions, and the constants $b_1, \ldots, b_m$ are the bounds for the constraints. The parameters are considered optimal if they result in the largest fitness value for all possible parameters satisfying the constraints[4] [Boyd and Vandenberghe, 2004].

A continuous optimisation problem is considered convex if both the objective and constraint functions are convex, which means they satisfy the inequality [Boyd and Vandenberghe, 2004]

$$f_i(\alpha x + \beta y) \le \alpha f_i(x) + \beta f_i(y) \quad \forall\, x, y \in \mathbb{R}^+, \quad \alpha + \beta = 1.$$

As previous research has demonstrated the existence of multiple local optima for locomotion optimisation [Kulk and Welsh, 2011b], this inequality is not fulfilled. Two non-convex optimisation algorithms are therefore considered: Evolutionary Hill Climbing with Line Search [Kulk and Welsh, 2011a; Russell and Norvig, 2010], and Policy Gradient Reinforcement Learning [Kulk and Welsh, 2011a; Sutton *et al.*, 2000].

Evolutionary Hill Climbing with Line Search

The local search family of optimisation algorithms, including hill climbing, are frequently used for real-time applications due to two main advantages: they require little memory (constant asymptotic complexity), and are able converge to reasonable solutions in high-dimensionality state spaces [Russell and Norvig, 2010]. This is accomplished by the arbitrary selection of an initial solution, followed by iterative evaluation and mutation to derive a more optimal solution. This technique, which is only capable of converging to a local optima, can be extended via the process of line search. The gradient of the objective function at the position of the current solution is estimated, and the new solution chosen as a set of parameters that lie on the gradient vector (at some distance proportional to the magnitude of that vector). This method can be implemented via Evolutionary Hill Climbing with Line Search (EHCLS), as defined in Algorithm 1.

[3] A modified version of this walk engine yielded similar walk speeds of approximately 14 cm/s on the DARwIn-OP platform.

[4] As many walk engine parameters capture properties specific to the robot (such as step length), these constrains capture hardware limitations and are handled implicitly.

Algorithm 1 Evolutionary Hill Climbing with Line Search (EHCLS)

```
$\boldsymbol{\theta}_{best} = \boldsymbol{\theta}_{pbest} = \boldsymbol{\theta} = InitialParameters$
$\Delta_{best} = \Delta_{pbest} = 0$
$f_{best} = f_0(\boldsymbol{\theta}_{best})$
loop
    if $f_0(\boldsymbol{\theta}) \geq f_{best}$ then
        $\boldsymbol{\theta}_{pbest} = \boldsymbol{\theta}_{best}$
        $\boldsymbol{\theta}_{best} = \boldsymbol{\theta}$
        $\Delta_{best} = f_0(\boldsymbol{\theta}) - f_{best}$
        $f_{best} = f_0(\boldsymbol{\theta}_{best})$
        $\alpha = \beta \left| \tanh\left(\frac{\Delta_{best}}{\Delta_{pbest}}\right) \right|$
        $\Delta_{pbest} = \Delta_{best}$
    end if
    if $CountSinceImprovement > ResetLimit$ then
        $f_{best} = ResetFraction \cdot f_{best}$
        $\alpha = 0$
    end if
    $\boldsymbol{\theta} = \boldsymbol{\theta}_{best} + \alpha(\boldsymbol{\theta}_{best} - \boldsymbol{\theta}_{pbest}) + (1 - \alpha)\psi \cdot \mathtt{range}(\boldsymbol{\theta})$
end loop
```

In Evolutionary Hill Climbing with Line Search, $\psi = \mathcal{N}\Big(0, \eta \exp\left(\frac{CountSinceImprovment}{ResetLimit} - 1\right)\Big)$ is a Gaussian distribution defining the mutation size, $\mathtt{range}(\boldsymbol{\theta})$ defines the length of the search space for each parameter θ_i and β is some user-defined constant (chosen to be 0.95 following previous walk optimisation research for the Aldebaran NAO [Kulk and Welsh, 2011a]). This implementation contains three parameters that can be tuned to affect convergence: η, which controls the mutation rate; *ResetLimit*, which specifies the reset frequency; and *ResetFraction*, which specifies the reset magnitude [Kulk and Welsh, 2011a].

Policy Gradient Reinforcement Learning

A method of reinforcement learning which considers a parameterised representation of the policy, $\pi_{\boldsymbol{\theta}}$ (where $\boldsymbol{\theta}$ is a d-dimensional vector of parameters), is known as policy search [Russell and Norvig, 2010]. When the expected reward of executing policy $\pi_{\boldsymbol{\theta}}$, $\rho(\boldsymbol{\theta})$, is differentiable, the policy gradient vector $\nabla_{\boldsymbol{\theta}}\rho(\boldsymbol{\theta})$ can be followed to determine a local optima in policy space. This process is known as Policy Gradient Reinforcement Learning (PGRL) [Kulk and Welsh, 2011a; Russell and Norvig, 2010; Sutton *et al.*, 2000].

Many heuristics have been proposed for computationally efficient gradient estimation in high-dimensionality policy spaces [Kulk and Welsh, 2011a; Kohl and Stone, 2004]. PGRL (defined in Algorithm 2) begins with an initial parameterised policy $\boldsymbol{\theta} = \pi_\theta = (\theta_1, \ldots, \theta_d)$, iteratively approximating the policy gradient vector via estimations of the partial derivative of $\rho(\boldsymbol{\theta})$ and improving the policy by moving in the direction of that vector.

Algorithm 2 Policy Gradient Reinforcement Learning (PGRL)

```
$\boldsymbol{\theta} = InitialParameters$
loop
    Generate $K$ perturbation vectors $(\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_K)$
        $\boldsymbol{\epsilon}_k = (\varepsilon_1, \ldots, \varepsilon_d),\ d = ||\boldsymbol{\theta}||$
        $\varepsilon_i = \mathcal{N}(0, \sigma)$
    Generate $K$ policies $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$
        $\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$
    for $i = 1$ to d do
        $Avg^i_{+\epsilon} \leftarrow$ average $f_0(\boldsymbol{\theta}_k)\ \forall\ \boldsymbol{\theta}_k$ that have a positive perturbation in dimension $i$
        $Avg^i_0 \leftarrow$ average $f_0(\boldsymbol{\theta}_k)\ \forall\ \boldsymbol{\theta}_k$ that have no perturbation in dimension $i$
        $Avg^i_{-\epsilon} \leftarrow$ average $f_0(\boldsymbol{\theta}_k)\ \forall\ \boldsymbol{\theta}_k$ that have a negative perturbation in dimension $i$
        if $Avg^i_0 > Avg^i_{+\epsilon}$ and $Avg^i_0 > Avg^i_{-\epsilon}$ then
            $A_i = 0$
        else
            $A_i = \eta \tanh\left(Avg^i_{+\epsilon} - Avg^i_{-\epsilon}\right) \cdot \mathtt{range}(\boldsymbol{\theta})$
        end if
    end for
    $\boldsymbol{\theta} = \boldsymbol{\theta} + \mathbf{A}$
end loop
```

In Policy Gradient Reinforcement Learning, η controls the step size, σ controls the size of the perturbation applied to each dimension during policy generation and K defines the number of policies generated near $\boldsymbol{\theta}$ at each iteration.

2 Optimisation Framework

For an online optimiser where a single iteration corresponds with several minutes of walk fitness evaluation, optimising over a 50-dimensional parameter space (the number of parameters corresponding with the B-Human walk engine) is intractable. However, as the walk engine models the robot as an inverted pendulum, the oscillation frequency is only sublinearly proportional to the DARwIn-OP's stance height. This allows for the removal of balance PID controller values from the optimisation process, as a stable set of values for one walk should be comparatively stable for any other. As such, only 13 parameters (those corresponding with observable differences in the DARwIn-OP's walk) were chosen for optimisation. These include: `maxSpeedX`, `maxSpeedY`, `maxSpeedYaw`, `standComPositionZ`, `walkRefX`, `walkRefXAtFullSpeedX`, `walkRefY`, `walkRefYAtFullSpeedX`, `walkRefYAtFullSpeedY`, `walkStepDuration`, `walkStepDurationAtFullSpeedX`, `walkStepDurationAtFullSpeedY` and `walkComBodyRotation`.

2.1 Online Optimisation

As no sufficiently accurate simulator models of the DARwIn-OP robot exist, optimisation must be conducted on the physical robot itself. In previous NUbots locomotion research for the Aldebaran NAO [Kulk and Welsh, 2011a; 2011b], the optimisation process involved the robot following a fixed path. The length of each iteration was therefore affected by the quality of the walk, as good quality walks were inherently able to complete the path faster. Conversely, this paper adopts an approach guaranteed to yield iterations of consistent length, while ensuring all types of motion are represented equally. This is accomplished by having the DARwIn-OP execute forward, backward, left, right, clockwise and counter-clockwise walks, each for some fixed time t_n (an implicit method of weighting each motion type, $0 < t_n \leq 5$).

To ensure accurate measurement of the position and orientation of the robot, a ceiling-mounted FireWire camera was used to track the DARwIn-OP's colour-coded "hat" (a method appropriated from the RoboCup Small Size League). The PC interface for this system is demonstrated in Figure 2.

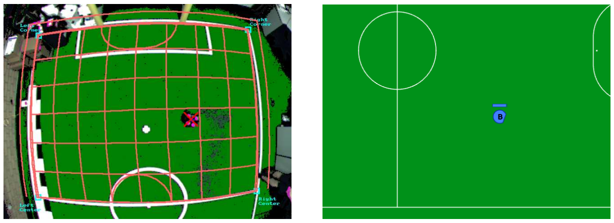


Figure 2: Interface for the ceiling-mounted FireWire camera, demonstrating: a look-up table classified view of the field and robot (with colour-coded "hat"), demonstrating lens distortion correction (left); and a visualisation of the robot's determined field position and orientation (right).

2.2 Fitness Evaluation

For the optimisation of the 2008 NUbots walk engine, three fitness functions based on speed, efficiency and Froude-Number were implemented [Kulk and Welsh, 2011b]. By assuming the existence of a shared global optima, all three functions were combined via the process of safe redundancy [Alamir, 2008]. Although demonstrated as an effective process for the non-omnidirectional open-loop walk engine implemented by the 2008 NUbots team [Kulk and Welsh, 2008], applying this method of fitness evaluation yielded comparatively poor results on the DARwIn-OP (with the optimisation of forward motion favoured at the expense of lateral speed and stability).

To address these issues, the following general fitness function was selected:

$$\mathcal{F} = \gamma \cdot \alpha - \beta^2. \quad (1)$$

This fitness function divides robot motion into three primary components: sagittal (forward and reverse), coronal (left and right) and rotational. The parameters α and β represent "correct" and "incorrect" motion quantities respectively, and γ is a scaling parameter. By quadratically penalising incorrect motion and selecting γ to ensure comparable fitness ranges, favouritism of forward motion is avoided. Specifically, given start field coordinate $p_1 = (x_1, y_1)$, end field coordinate $p_2 = (x_2, y_2)$ and the RoboCup axes conventions:

$$\begin{bmatrix} \Delta_x \\ \Delta_y \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x_2 - x_1 \\ y_2 - y_1 \end{bmatrix}. \quad (2)$$

The set F of individual weighted fitness measurements are then combined by taking the arithmetic mean:

$$\mathcal{F}_\Sigma = \frac{1}{|\mathcal{F}|} \sum_{\mathcal{F} \in \mathcal{F}} \mathcal{F}, \quad (3)$$

where each fitness $\mathcal{F} \in \mathcal{F}$ is a specific implementation of (1).

Sagittal Motion

Should provide linear reward for correct sagittal motion and quadratic penalty for any coronal motion:

$$\mathcal{F}_{\mathrm{forward}} = -\mathcal{F}_{\mathrm{reverse}} = 10 \cdot \Delta_{\mathrm{x}} - \Delta_{\mathrm{y}}^2,$$

where Δ_x and Δ_y represent forward and left motions respectively, as defined in (2).

Coronal Motion

Should provide linear reward for correct coronal motion and quadratic penalty for any sagittal motion:

$$\mathcal{F}_{\mathrm{left}} = -\mathcal{F}_{\mathrm{right}} = 10 \cdot \Delta_{\mathrm{y}} - \Delta_{\mathrm{x}}^2.$$

Rotational Motion

The robot should rotate while ideally maintaining zero displacement. As such, rotation should receive a linear reward, whereas any displacement from the starting point should be quadratically penalised. The angle of rotation (measured in radians) is scaled to a similar range to distance travelled (cm):

$$\mathcal{F}_{\mathrm{cw}} = 100 \cdot \theta - \left(\Delta_{\mathrm{x}}^2 + \Delta_{\mathrm{y}}^2\right).$$

2.3 Enforcing Stability

The fitness evaluation method described in Section 2.2 implicitly penalised unstable walks, as a fallen robot will inherently cover less distance over time. However, it is vital that any optimisation algorithm quickly diverges from unstable regions. This is accomplished by a modification of the combination process to penalise any trial during which the robot fell (determined easily from the DARwIn-OP's gyroscope and accelerometer):

$$f_0(\boldsymbol{\theta}) = \mathcal{F}'_\Sigma = \begin{cases} \min\{0, \mathcal{F}_\Sigma\} & \text{if robot has fallen} \\ \mathcal{F}_\Sigma & \text{otherwise} \end{cases} \tag{4}$$

where $\mathcal{F}_\Sigma$ is the fitness function defined in (3).

3 Probabilistic Gradient Ascent

Optimisation of walk parameters is a difficult task, due to both the high dimensionality of the parameter search space, and significant fitness measurement noise resulting from the online optimisation process (see Section 2.1).

This section introduces a new non-convex optimisation algorithm, Probabilistic Gradient Ascent (PGA), which is designed to function effectively within such an environment. Similarly to PGRL [Kulk and Welsh, 2011a; Sutton *et al.*, 2000], PGA is derived from a framework of reinforcement learning; specifically the Policy Improvement with Path Integrals algorithm [Theodorou *et al.*, 2010a], which has been demonstrated as effective for learning optimal actuation policies for robots with up to 50 degrees-of-freedom [Theodorou *et al.*, 2010b].

3.1 Background and Motivation

The Policy Improvement with Path Integrals ($\mathbf{PI^2}$) algorithm was introduced in 2010 by Theodorou *et al.* [Theodorou *et al.*, 2010a; 2010b]. This algorithm provides a new method of probabilistic reinforcement learning derived from a framework of stochastic optimal control, based on the stochastic Hamilton-Jacobi-Bellman (HJB) equations. $\mathbf{PI^2}$ has been demonstrated as effective for a variety of robot locomotion tasks, including the actuation of a 12 degree-of-freedom robot dog [Theodorou *et al.*, 2010a] and 2, 10 and 50 degree-of-freedom robotic arms [Theodorou *et al.*, 2010b].

Given a start time t_i and end time t_N, a trajectory roll-out $\boldsymbol{\tau}_i$ is defined as:

$$\boldsymbol{\tau}_i = \left(\mathbf{x_{t_i}}, \ldots, \mathbf{x_{t_N}}\right), \tag{5}$$

where $\mathbf{x_{t_i}}$ is the system state at time t_i. The iterative implementation of the $\mathbf{PI^2}$ algorithm attempts to minimise the cost function $R(\boldsymbol{\tau}_i)$ for a trajectory $\boldsymbol{\tau}_i$, by repeating Algorithm 3 until convergence.

Algorithm 3 Policy Improvement with Path Integrals ($\mathbf{PI^2}$)

```
$\boldsymbol{\theta} = InitialParameters$
loop
    Generate $K$ roll-outs from start state $x_0$ using parameters $\boldsymbol{\theta} + \varepsilon_t$
    for $k = 1 \ldots K$ do
        $P(\boldsymbol{\tau}_{i,k}) = \frac{e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,k})}}{\sum_{j=1}^{K}\left[e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,j})}\right]}$
        $S(\boldsymbol{\tau}_{i,k}) = \phi_{t_N,k} + \sum_{j=1}^{N-1} q_{t_j,k} + \frac{1}{2}\sum_{j=1+1}^{N-1}$
        $\left(\boldsymbol{\theta} + \mathbf{M_{t_j,k}}\varepsilon_{\mathbf{t_j,k}}\right)^T \mathbf{R}\left(\boldsymbol{\theta} + \mathbf{M_{t_j,k}}\varepsilon_{\mathbf{t_j,k}}\right)$
        $\mathbf{M_{t_j,k}} = \frac{\mathbf{R^{-1}g_{t_j,k}g_{t_j,k}^T}}{\mathbf{g_{t_j,k}^T R^{-1} g_{t_j,k}}}$
    end for
    for $i = 1 \ldots (N-1)$ do
        $\boldsymbol{\delta\theta}_{t_i} = \sum_{K=1}^{K}\left[P(\boldsymbol{\tau}_{i,k})\mathbf{M_{t_j,k}}\varepsilon_{\mathbf{t_j,k}}\right]$
    end for
    $\left[\boldsymbol{\delta\theta}\right]_j = \frac{\sum_{i=0}^{N-1}(N-i)w_{j,t_i}\left[\boldsymbol{\delta\theta}_{t_i}\right]_j}{\sum_{i=0}^{N-1} w_{j,t_i}(N-i)}$
    $\boldsymbol{\theta} = \boldsymbol{\theta} + \boldsymbol{\delta\theta}$
    Create noiseless roll-out to check the trajectory cost $R = \phi_{t_N} + \sum_{i=0}^{N-1} r_{t_i}$
end loop
```

In $\mathbf{PI^2}$, the cost function $R(\boldsymbol{\tau}_i)$ is defined in terms of the terminal cost $\phi_{t_N} = \phi\left(\mathbf{x_{t_N}}\right)$ and immediate cost $r(t)$ at time t:

$$R(\boldsymbol{\tau}_i) = \phi_{t_N} + \int_{t_i}^{t_N} r(t)\, dt.$$

In the case of walk engine parameter optimisation (and optimisation problems in general), a vector of parameters $\boldsymbol{\theta}$ is simply evaluated in terms of the fitness function $f_0(\boldsymbol{\theta})$ (see Section 1.3). By reducing the problem of minimising the cost $R(\boldsymbol{\tau}_i)$ of time dependant trajectory $\boldsymbol{\tau}_i$ to one of maximising the fitness $f_0(\boldsymbol{\theta})$ of time independent parameters $\boldsymbol{\theta}$, a simpler yet powerful non-convex optimisation algorithm is produced. The following section details such a reduction from $\mathbf{PI^2}$ to the introduced Probabilistic Gradient Ascent algorithm.

3.2 Implementation

Given K trajectory roll-outs generated by the $\mathbf{PI^2}$ algorithm [Theodorou *et al.*, 2010a], the probability $P(\boldsymbol{\tau}_{i,k})$ of the kth roll-out $\boldsymbol{\tau}_{i,k} = \left(\mathbf{x_{t_i}}, \ldots, \mathbf{x_{t_N}}\right)_k$ is defined as:

$$P(\boldsymbol{\tau}_{i,k}) = \frac{e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,k})}}{\sum_{j=1}^{K}\left[e^{-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,j})}\right]}, \tag{6}$$

where λ regulates the sensitivity of the exponentiation of the generalised cost function $S(\boldsymbol{\tau}_{i,k})$. Theodorou *et al.* demonstrated the following equivalence [Theodorou *et al.*, 2010a]:

$$\exp\left(-\frac{1}{\lambda}S(\boldsymbol{\tau}_{i,k})\right) = \exp\left(-h\frac{S(\boldsymbol{\tau}_{i,k}) - \min S(\boldsymbol{\tau}_i)}{\max S(\boldsymbol{\tau}_i) - \min S(\boldsymbol{\tau}_i)}\right),$$

where h is some user-defined constant, and the generalised cost function $S(\boldsymbol{\tau}_{i,k})$ is defined as:

$$S(\boldsymbol{\tau}_{i,k}) = \phi_{t_N,k} + \sum_{j=1}^{N-1} q_{t_j,k} + \frac{1}{2}\sum_{j=1+1}^{N-1}$$

$$\left(\boldsymbol{\theta} + \mathbf{M_{t_j,k}}\varepsilon_{\mathbf{t_j,k}}\right)^T \mathbf{R}\left(\boldsymbol{\theta} + \mathbf{M_{t_j,k}}\varepsilon_{\mathbf{t_j,k}}\right). \qquad (7)$$

By considering the walk optimisation scenario where cost is evaluated only by a single observation at time t_N, it follows from (5) that:

$$\boldsymbol{\tau}_{i,k} = \boldsymbol{\tau}_{N,k} = (\mathbf{x_{t_N}})_{\mathbf{k}},$$

therefore reducing the general cost function (7) to the terminal cost function:

$$S(\boldsymbol{\tau}_k) = \phi_k.$$

Furthermore, as the system is now only observed at time t_N, it is intuitive to represent ϕ_k in terms of the fitness of the system parameters $\boldsymbol{\theta}$:

$$\phi_k \propto f_0(\boldsymbol{\theta}_k). \qquad (8)$$

Substituting (8) into (6) results in the following parameter update equations for Probabilistic Gradient Ascent:

$$\tilde{P}(\boldsymbol{\theta}_k) = \frac{\Lambda_k}{\sum_{j=1}^{K}\Lambda_j}$$

$$\Lambda_k = \exp\left(c\cdot\frac{f_0(\boldsymbol{\theta}_k) - \min f_0(\boldsymbol{\theta})}{\max f_0(\boldsymbol{\theta}) - \min f_0(\boldsymbol{\theta})}\right),$$

where c controls how strongly the parameter update tends toward the maximum observed policy fitness. In this case, each policy $\boldsymbol{\theta}_k$ is produced by permuting the parameters $\boldsymbol{\theta}$ by a vector $\boldsymbol{\epsilon}_k = (\varepsilon_1, \ldots, \varepsilon_d)$, $d = ||\boldsymbol{\theta_k}||$, where each $\varepsilon_i = \mathcal{N}(0, \sigma)$ is Gaussian noise of standard deviation parameterised by step size σ (analogous to the learning rate α present in the EHCLS algorithm [Kulk and Welsh, 2011a; Russell and Norvig, 2010]).

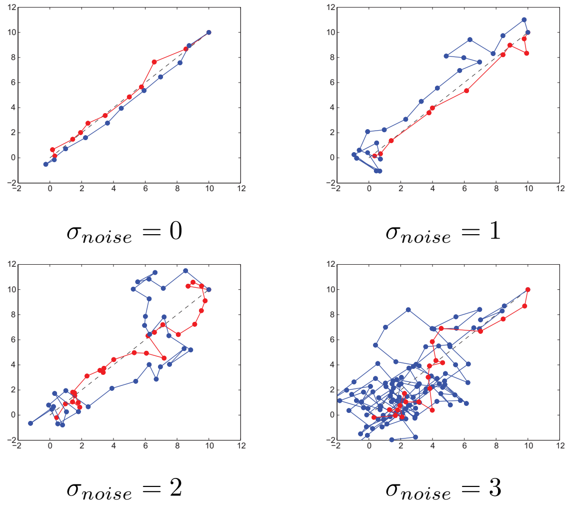
Similarities and differences between the update step of Theodorou *et al.*'s $\mathbf{PI^2}$ algorithm (from which PGA is derived) and gradient descent algorithms have been noted previously [Stulp, 2012]. Instead of estimating a parameter space gradient from a limited number of noisy trials, PGA implements probability-weighted averaging. Specifically, by weighting the lowest score as a 0 baseline, PGA virtually disregards scores relatively close in measured fitness to the low end. This has the effect of filtering out negative outliers, with the strength of this effect parameterised by c.

Algorithm 4 Probabilistic Gradient Ascent (PGA)

```
$\boldsymbol{\theta} = InitialParameters$
loop
    Generate $K$ perturbation vectors $(\boldsymbol{\epsilon}_1, \ldots, \boldsymbol{\epsilon}_K)$
        $\boldsymbol{\epsilon}_k = (\varepsilon_1, \ldots, \varepsilon_d),\ d = ||\boldsymbol{\theta}||$
        $\varepsilon_i = \mathcal{N}(\prime, \sigma)$
    Generate $K$ policies $(\boldsymbol{\theta}_1, \ldots, \boldsymbol{\theta}_K)$
        $\boldsymbol{\theta}_k = \boldsymbol{\theta} + \boldsymbol{\epsilon}_k$
    for $k = 1 \ldots K$ do
        $\Lambda_k = \exp\left(c \cdot \frac{f_0(\boldsymbol{\theta}_k) - \min f_0(\boldsymbol{\theta})}{\max f_0(\boldsymbol{\theta}) - \min f_0(\boldsymbol{\theta})}\right)$
    end for
    $\tilde{P}(\boldsymbol{\theta}_k) = \frac{\Lambda_k}{\sum_{j=1}^{K}\Lambda_j}$
    $\boldsymbol{\theta} = \boldsymbol{\theta} + \sum_{k=1}^{K}\boldsymbol{\epsilon}_k\tilde{P}(\boldsymbol{\theta}_k)$
end loop
```

3.3 Visual Demonstration

Figure 3 compares the updates of PGA (red) and PGRL (blue) algorithms for a simple optimisation problem (fitness function $f_0(x, y) = -\sqrt{x^2 + y^2}$). At each step, both algorithms generate ten policies using the same perturbation vector of Gaussian noise ($\sigma_{pert} = 1$), with Gaussian noise of increasing standard deviation σ_{noise} introduced to each fitness measurement.



$\sigma_{noise} = 0$ $\sigma_{noise} = 1$

$\sigma_{noise} = 2$ $\sigma_{noise} = 3$

Figure 3: Probabilistic Gradient Ascent (PGA, red) and Policy Gradient Reinforcement Learning (PGRL, blue) compared for the simple optimisation task described in Section 3.3. Increasing levels of Gaussian noise are added to each fitness measurement. As noise standard deviation σ increases from 0 to 3 in these examples, PGA is more effective at disregarding outliers due to its probability-weighting update step.

4 Experimental Results

4.1 Fitness Metric Validation

Before assessing the quality of presented PGA algorithm, it is necessary to validate the chosen fitness metric using a well-established optimisation algorithm. Specifically, it is critical that measured improvements in fitness correspond with physical improvements in walk speed and stability.

As defined in (4), the fitness values for sagittal, coronal and rotational motion are combined into a single fitness function $f_0(\boldsymbol{\theta}) = \mathcal{F}'_\Sigma$. Using the EHCLS algorithm (defined in Section 1.3) the fitness improved significantly over a period of ten hours (not including breaks and battery replacement time) from an initial value of 85 to a maximum value of 216; an increase by over 150%. These results are presented in the top plot of Figure 4. The black line represents each fitness measurement over time (demonstrating significant measurement noise), with the red line indicating the current maximum fitness.

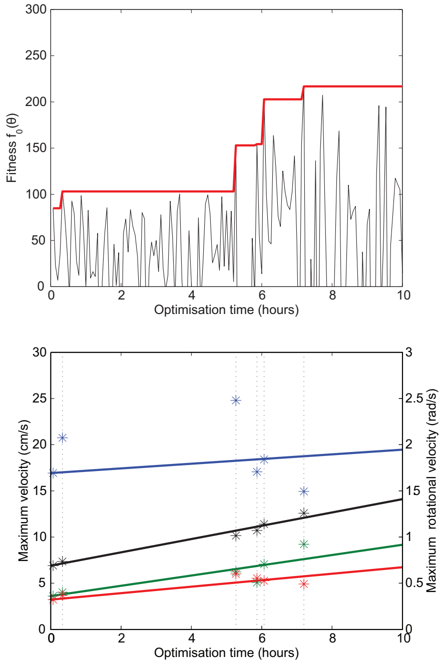


Figure 4: Walk engine optimisation results using the Evolutionary Hill Climbing with Line Search algorithm. The top plot demonstrates the fitness function $f_0(\boldsymbol{\theta})$ over time (black), along with the current maximum fitness (red). For each improvement in overall fitness, a dotted line is displayed in the bottom plot. The four points along each line indicate the forward (blue), reverse (green), coronal (red) and rotational (black) maximum speeds, with a regression line fitted to demonstrate the upward trend in each set.

The top plot of Figure 4 demonstrates clear improvement in all lateral motion types, which follows intuitively from the *a priori* knowledge that the initial walk parameters strongly favoured forward motion. Although a slight upward trend in maximum forward velocity is observed over time, is is less significant than for the remaining results, and the highest-scoring walk parameters actually yield an overall decrease by 2 cm/s. An average velocity increase of 68.7% is evident across the four motion types.

These results support the selection of the online optimisation framework and fitness function $f_0(\boldsymbol{\theta}) = \mathcal{F}'_\Sigma$ as appropriate for effectively improving omnidirectional locomotion performance.

4.2 Evaluation of Optimisation Algorithms

Figure 5 compares the maximum velocities obtained for forward (a), reverse (b), coronal (c) and rotational (d) motion, for PGRL (brown), EHCLS (orange) and PGA (yellow) optimisation algorithms. The black bars represent the parameters resulting from the previous NUbots optimisation framework [Kulk and Welsh, 2011a; 2011b] (which were used by the NUbots at RoboCup 2012, and formed the initial parameters for each optimisation experiment). PGA demonstrates the best overall performance against this simplified metric, yielding maximum forward, reverse and coronal velocities of 34.1, 9.6, and 9.5 cm/s respectively, with a rotational velocity of 1.38 rad/s.

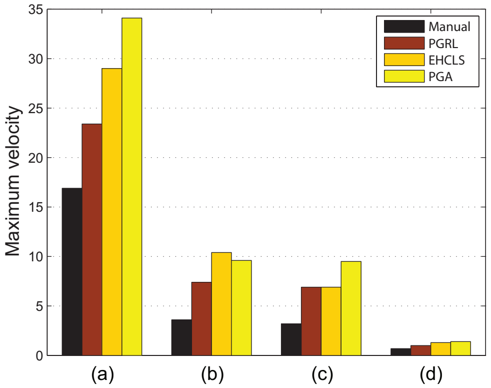


Figure 5: Maximum velocities obtained for: a) forward (cm/s), b) reverse (cm/s), c) coronal (cm/s) and d) rotational (rad/s) motion types. The walk parameters used by the NUbots at RoboCup 2012 are compared against the best resulting from three optimisation algorithms: Policy Gradient Reinforcement Learning, Evolutionary Hill Climbing with Line Search and Probabilistic Gradient Ascent (PGA). PGA (introduced in this paper) demonstrates the best overall performance, yielding a maximum forward walk velocity of 34.1 cm/s.

5 Discussion

As defined in Section 3.2, PGA is parameterised by the user-defined value c, which controls how strongly the parameter update tends toward the maximum observed policy fitness (for that iteration). As this value has an equivalent effect to h from the $\mathbf{PI^2}$ algorithm [Theodorou *et al.*, 2010a], its value was chosen to be $c = 10$ (in keeping with Theodorou *et. al.*, who set $h = 10$ for their reinforcement learning experimentation [Theodorou *et al.*, 2010a]). Parameters for EHCLS and PGRL were chosen to be consistent with previous NUbots walk optimisation research [Kulk and Welsh, 2011a; 2011b].

Within the RoboCup community, it is commonplace for teams to benchmark the performance of bipedal robot locomotion purely in terms of maximum velocities obtained when walking in a particular direction. Although only implicitly capturing stability (see the fitness function defined in (4)), these results capture a critical requirement for a soccer-playing robot (its ability to intercept the ball faster than the opponent), and have therefore been presented in Figure 5 to demonstrate the relative optimisation performance of PGA versus EHCLS and PRGL. It should be noted that these maximum speeds need not belong to a single parameter vector $\boldsymbol{\theta}$.

As the most regularly encountered RoboCup bipedal locomotion statistic, the maximum forward velocity of 34.1 cm/s obtained by PGA optimisation represents the following improvements over previous published results:

- An improvement of 250% over the original NUbots DARwIn-OP walk engine, adapted from the Aldebaran NAO [Kulk and Welsh, 2008]
- An improvement of 101% over the initial walk engine parameters resulting from the previous NUbots optimisation framework (developed for the Aldebaran NAO and a non-omnidirectional walk engine [Kulk and Welsh, 2011a; 2011b])
- An improvement of 42% over the maximum DARwIn-OP walk velocity specified in technical documentation [Ha *et al.*, 2011]
- An improvement of 18% over the best set of walk parameters produced by two well known non-convex optimisation algorithms (Evolutionary Hill Climbing with Line Search and Policy Gradient Reinforcement Learning
- An improvement of 10% over the best Aldebaran NAO walk velocity published by walk engine developers B-Human [Graf and Röfer, 2012]

Although not explicitly required by this evaluation metric, this 34.1 cm/s is particularly stable. Video footage is available online at `http://www.davidbudden.com/research/walk-optimisation/`.

6 Conclusion

Developing effective methods of bipedal humanoid locomotion is critically important in modern robotics for many reasons. Beyond the obvious advantages of a fast and stable walk, minimising coronal oscillations reduces motion blur in image processing. Motion blur degrades colour classification performance [Budden *et al.*, 2013; Budden and Mendes, 2013] and therefore the accuracy of object recognition algorithms [Budden *et al.*, 2012]. This reduces the ability of the robot to both self-localise within its environment [Budden and Prokopenko, 2013] and execute complex behaviours and strategies [Prokopenko *et al.*, 2013].

A major observed issue in the optimisation of bipedal robotic locomotion is inherent noise in fitness measurement and discontinuities introduced by falls. Online optimisation is a very time consuming process, with a single iteration taking minutes to complete (and consequently decreasing motor life). This limitation prevents multiple trials of each vector of walk parameters as a viable option, requiring any chosen optimisation algorithm to be robust against such fitness uncertainty. To address this issue, a new non-convex optimisation algorithm (Probabilistic Gradient Ascent, or PGA) was developed from a reinforcement learning framework. This algorithm replaces traditional gradient estimation with probability-weighted averaging, allowing for better performance in these scenarios.

In addition to enhancing the DARwIn-OP's capabilities with a modified version of B-Human's omnidirectional walk engine [Graf *et al.*, 2009], the documented maximum speed of the DARwIn-OP was exceeded by 20% by an improved optimisation framework utilising two well-known algorithms. Applying PGA yielded further improvement, with the highest fitness parameters yielding an average improvement of 50.4% across four motion types.

During the recent redevelopment of the pre-2012 NUbots vision system [Nicklin *et al.*, 2011], PGA was utilised in the selection of internal system parameters, yielding significant improvement in object recognition performance. It is anticipated that the applications of PGA will extend well beyond the optimisation of bipedal locomotion, to a wider range of optimisation problems exhibiting significant uncertainty in performance evaluation.

Acknowledgement

NICTA is funded by the Australian Government as represented by the Department of Broadband, Communications and the Digital Economy and the Australian Research Council through the ICT Centre of Excellence program.

References

[Alamir, 2008] M. Alamir. On useful redundancy in dynamic inverse problems related optimization. In *Preprints of the 17th IFAC World Congress*, 2008.

[Annable *et al.*, 2013] B. Annable, D. Budden, S. Calland, S. Chalup, S. K Fenn, M. Flannery, J. Fountain, R. King, A. Mendes, M. Metcalfe, S. Nicklin, P. Turner, and J. Walker. The NUbots team description paper 2013. 2013.

[Boyd and Vandenberghe, 2004] S. Boyd and L. Vandenberghe. *Convex optimization*. Cambridge University Press, 2004.

[Budden and Mendes, 2013] D. Budden and A. Mendes. Unsupervised recognition of salient colour for real-time image processing. In *RoboCup 2013: Robot Soccer World Cup XVII*. Springer, 2013.

[Budden and Prokopenko, 2013] D. Budden and M. Prokopenko. Improved particle filtering for pseudo-uniform belief distributions in robot localisation. In *RoboCup 2013: Robot Soccer World Cup XVII*. Springer, 2013.

[Budden *et al.*, 2012] D. Budden, S. Fenn, J. Walker, and A. Mendes. A novel approach to ball detection for humanoid robot soccer. In *Advances in Artificial Intelligence (LNAI 7691)*. Springer, 2012.

[Budden *et al.*, 2013] D. Budden, S. Fenn, A. Mendes, and S. Chalup. Evaluation of colour models for computer vision using cluster validation techniques. In *RoboCup 2012: Robot Soccer World Cup XVI (LNAI 7500)*. Springer, 2013.

[Dietsch, 2012] J. Dietsch. DARPA entices roboticists to take the next step. *Robotics & Automation Magazine*, 19(3):9–10, 2012.

[Graf and Röfer, 2012] C. Graf and T. Röfer. A center of mass observing 3D-LIPM gait for the RoboCup standard platform league humanoid. *RoboCup 2011: Robot Soccer World Cup XV*, pages 102–113, 2012.

[Graf *et al.*, 2009] C. Graf, A. Härtl, T. Röfer, and T. Laue. A robust closed-loop gait for the standard platform league humanoid. In *Proceedings of the Fourth Workshop on Humanoid Soccer Robots*, pages 30–37. IEEE, 2009.

[Ha *et al.*, 2011] I. Ha, Y. Tamura, H. Asama, J. Han, and D.W. Hong. Development of open humanoid platform DARwIn-OP. In *SICE Annual Conference (SICE), 2011 Proceedings of*, pages 2178–2181. IEEE, 2011.

[Kohl and Stone, 2004] N. Kohl and P. Stone. Policy gradient reinforcement learning for fast quadrupedal locomotion. In *Robotics and Automation, IEEE Conference on*, volume 3, pages 2619–2624. IEEE, 2004.

[Kulk and Welsh, 2008] J. Kulk and J. Welsh. A low power walk for the NAO robot. In *Proceedings of the 2008 Australasian Conference on Robotics & Automation (ACRA-2008), J. Kim and R. Mahony, Eds*, 2008.

[Kulk and Welsh, 2011a] J. Kulk and J. Welsh. Evaluation of walk optimisation techniques for the NAO robot. In *Humanoid Robots, 11th IEEE-RAS International Conference on*, pages 306–311. IEEE, 2011.

[Kulk and Welsh, 2011b] J. Kulk and J. Welsh. Using redundant fitness functions to improve optimisers for humanoid robot walking. In *Humanoid Robots (Humanoids), 2011 11th IEEE-RAS International Conference on*, pages 312–317. IEEE, 2011.

[Michel, 1998] O. Michel. Webots: Symbiosis between virtual and real mobile robots. In *Virtual Worlds*, pages 254–263. Springer, 1998.

[Nicklin *et al.*, 2011] S. Nicklin, S. Bhatia, D. Budden, R. King, J. Kulk, J. Walker, A. Wong, and S. Chalup. The nubots' team description for 2011, 2011.

[Prokopenko *et al.*, 2013] M. Prokopenko, O. Obst, P. Wang, D. Budden, and O. Cliff. Gliders2013: Tactical analysis with information dynamics. In *RoboCup 2013 Symposium and Competitions: Team Description Papers, Eindhoven*, 2013.

[Russell and Norvig, 2010] S. Russell and P. Norvig. *Artificial intelligence: A modern approach*. Pearson Prentice Hall, 2010.

[Stulp, 2012] F. Stulp. Adaptive exploration for continual reinforcement learning. In *Intelligent Robots and Systems (IROS), 2012 IEEE/RSJ International Conference on*, pages 1631–1636. IEEE, 2012.

[Sutton *et al.*, 2000] R. Sutton, D. McAllester, S. Singh, and Y. Mansour. Policy gradient methods for reinforcement learning with function approximation. *Advances in Neural Information Processing Systems*, 12(22), 2000.

[Theodorou *et al.*, 2010a] E. Theodorou, J. Buchli, and S. Schaal. A generalized path integral control approach to reinforcement learning. *The Journal of Machine Learning Research*, 11:3137–3181, 2010.

[Theodorou *et al.*, 2010b] E. Theodorou, J. Buchli, and S. Schaal. Reinforcement learning of motor skills in high dimensions: A path integral approach. In *Robotics and Automation (ICRA), 2010 IEEE International Conference on*, pages 2397–2403. IEEE, 2010.

[Westervelt *et al.*, 2007] E. Westervelt, J. Grizzle, C. Chevallereau, J. Choi, and B. Morris. *Feedback control of dynamic bipedal robot locomotion*. CRC Press, 2007.