Reinforcement Learning of Bipedal Lateral Behaviour and Stability Control with Ankle-Roll Activation

Bernhard Hengst

School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia
E-mail: bernhardh@cse.unsw.edu.au

Nimble bipeds are expected to execute a rich repertoire of movement behaviours and switch between them seamlessly without falling over. A good example demanding this responsiveness is playing a game of soccer. This paper explores a less well studied approach to biped locomotion using model-based multi-goal reinforcement learning. We apply this approach to learn the side-to-side movement of a small humanoid robot in simulation. By on-line learning one transition function, it is possible to modify the behaviour of the robot varying the cost function. Behaviours include: changing support feet at different frequencies; standing upright; standing on either foot; and switching between these behaviours, subject to the imposed constraints. The behaviours are stable over much of the experienced phase domain. They respond optimally to disturbances and when switching.

Keywords: Machine Learning; Artificial Intelligence; Reinforcement Learning; Robotics; Bipedal Locomotion; System Identification.

1. Introduction

Future bipedal robots are expected to be more agile, exhibiting a variety of behaviours. It is difficult to hand-code these behaviours and to transition seamlessly between them.

In this paper we take a reinforcement learning (RL) approach to learning and controlling behaviour. We apply this approach in simulation, specifically to tackle the problem of lateral (frontal or coronal-plane) behaviour to aid bipedal locomotion. The novelty is in a combination of:

- A random action routine to learn the system state transition function on-line rather than the action-value function $Q(s, a)$ as is more usual in RL.[1] We do not rely on a (linear) inverted pendulum model, or any other model, but perform automatic system identification.

- Using the reward function to craft multiple behaviours such as side-to-side rocking at different frequencies, standing still and balancing on one leg.
- Seamlessly switching between behaviours. The behaviour response to switching is theoretically guaranteed to be optimal.
- Stability of the behaviours to disturbances. The response to any disturbance is an optimal trajectory back to the continuous cycle or state behaviour required to minimise overall cost.

In the rest of this paper we will briefly review related research, present our method and approach, and provide results showing the system behaviour under changing goals, and with disturbances. We conclude with a discussion and planned future work.

2. Related Work

We briefly mention some prominent research on bipedal walking.

Passive-dynamic walkers, originally developed by McGeer[2] based on a "rimless wheel" model, were improved by others and powered to walk on level ground.[3] These machines have no explicit controllers yet exhibit humanlike motions, but are limited in their repertoire of behaviour.

Another research direction is the control of the Zero Moment Point to stay within the support polygon. Exemplified by the work of Kajita, et al,[4] the use of a (linear) inverted pendulum model allows 3D bipedal gaits to be developed. This control-theoretic approach relies on *preview control*[5] - a needed more responsive feed-forward method.

Much research has been devoted to planar bipedal locomotion. An extensive exposition is given by Westervelt et al.[6] Planar research has been extended to 3D.[7] Grizzle reminded researchers that even the simplest bipedal locomotion is challenging, let alone aperiodic walks, non-flat ground, etc.[8] These approaches follow the control system methodology where system identification is manual and specified using differential equations.

Despite some early promises for RL, and while the literature on bipedal walking is extensive, there has been relatively few RL approaches to bipedal locomotion. Exceptions include frontal plane control using an actuated passive walker;[9] point-feet placement;[10–12] temporally extended actions applied to swing foot placement underpinned by semi-MDP theory.[10]

In this paper we propose to exploit the properties of RL to model nonlinear, discrete and continuous dynamics and to learn optimal control for several different behaviours via reward functions.

3. Method

The general method we employ has the following steps:

(1) Construct and constrain the system, specifying joints, links, etc
(2) Define a Markov state and available control actions
(3) Learn the state transition function through exploration
(4) Define cost functions to achieve various behaviours
(5) Learn the optimal action-value Q function for each cost function
(6) Execute the behaviours by switching between optimal policies

We now describe the application of the above method to learn lateral bipedal motion behaviours.

3.1. *Construct and Constrain*

For real or simulated systems, this involves a specification of both the physical and software constraints. In our case we model a 23 DoF Aldebaran Robotics Nao humanoid robot with the Open Dynamics Engine (ODE) physics simulator (see Figure 1).

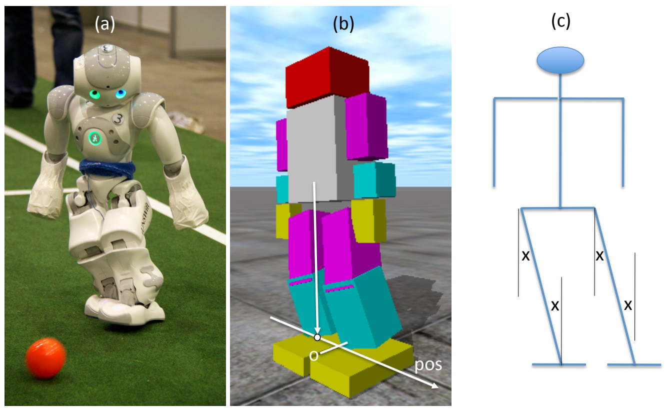

Fig. 1. (a) the Aldebaran Nao Robot, (b) ODE simulation showing the *pos* variable, and (c) the hip and ankle roll - they are equal, except for an additional control action described in Section 3.2.

To aid lateral movements we set the hip and ankle rolls to follow in direct proportion the perpendicular projection of the centre-of-mass (CoM) of the torso unto the line through the feet CoMs with the origin at the half-way point. We call this variable *pos* (see Figure 1). The support foot is determined by the sign of *pos*. The swing foot starts to lift immediately on change of support foot and moves up and down under constant acceleration with a period determined manually.

3.2. *Specify State and Actions*

RL is defined by the tuple $\langle S, A, T, R\rangle$ where S is the set of states, A the set of actions, T a stochastic transition function, and R a stochastic reward or cost function.[1] One would expect that rigid body physics could be modelled by point distributions, but even the ODE simulator generates noise due to the random nature of collision detection.

We define the lateral dynamic state of the Nao by the triple (pos, vel, act), where *pos* is a continuous variable defined in Section 3.1, *vel* is the velocity of *pos*, and *act* is the discrete control action applied at the last time-step to model the motor delay (of the simulator) of 2ms.

The three discrete actions A are $\{-\delta, 0, +\delta\}$, where δ is an additional amount of roll added to the ankle joints. In our application $\delta = 0.045$ radians. These actions may accelerate the robot to the left or right.

3.3. *Learning the Transition Function*

At each time-step we record the transition between states in S given an action from A. Since the sub-state space (pos, vel) is continuous we model the transitions at discrete grid-point values and use radial basis function approximation to generalise the transition function for off-sample states.

We explore and learn the transition function through a random action routine without lifting the swing-foot foot (but see discussion). Actions are generated at random using Equation 1 to adequately explore actions that persist.

$$action = \begin{cases} \text{random action from } A & \text{10\% of the time at random} \\ lastAction & \text{otherwise} \end{cases} \tag{1}$$

A portion for the learned transition function is depicted in Figure 2. Depending on the resolution of the grid, $resPos$ and $resVel$, learning may take several hours on the simulator.

3.4. *Define the Cost Function*

The cost function specifies the immediate rewards in RL (cost being the negative of reward). In our implementation we specify three components:

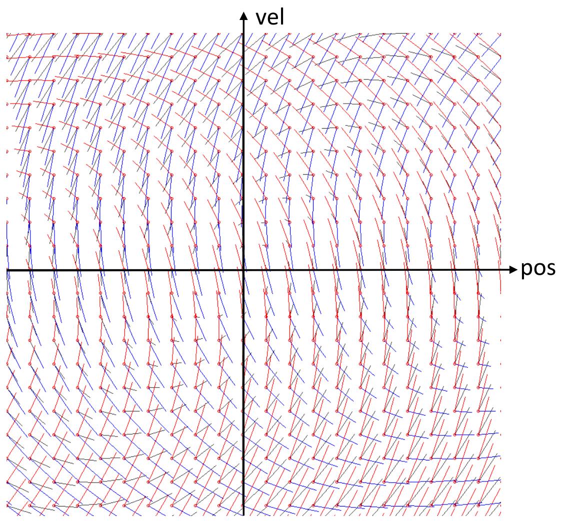

Fig. 2. Part of the transition function centred at $(pos = 0, vel = 0)$ learned on-line showing the next state transition from grid-points for each of the three ankle roll actions.

the cost of taking an action; the cost when reaching a desired set of goal states in the (pos, vel) space; and the cost of switching actions (to minimise motor usage). For example, the reward function for the robot to stand still is specified by Equation 2.

$$reward = \begin{cases} 50 & \text{if } |pos| < resPos \text{ and } |vel| < resVel \\ -2 & \text{if } action \neq lastAction \\ -1 & \text{otherwise} \end{cases} \tag{2}$$

Rocking from side-to-side can be similarly specified by rewarding a velocity value at the time of switching the support foot. The magnitude of the velocity will determine the period of the rock. To stand on the left leg we reward states near $(\text{CoM-left-foot}, 0)$.

3.5. *Learning the Optimal Control Policy*

We use standard RL policy iteration (PI) on the discrete Q action-value function.[1] The optimal function, $Q^*(s, a)$, is the future cost of taking action a in state s and following the optimal policy thereafter. The only subtly is the handling of the temporal difference backups as the next state for transitions from the grid-points is unlikely to be another grid point in the continuous state-space. However, it is straight forward to estimate the Q

functions for the next states by calculating a radial-basis weighted average value from neighbouring grid-points.[13] Assuming the function approximator is well behaved, the control policy is guaranteed to be optimal, a result that follows directly from MDP theory.[1] The Q values for the cost functions in Section 3.4 are learned and stored separately. Since the hard work is already done in Section 3.3, PI takes about a minute on a standard PC to learn six different policies: stand still; stand on one of each the two legs; and three sideways rocking motions at different frequencies.

3.6. *Execution*

The control policy is obtained from the optimal Q function, i.e. the optimal action = $\text{argmax}_a Q^*(state, a)$. Control is a matter of calculating the optimal action on-line at each time-step from the stored Q values given the current state and activating the motors accordingly.

4. Results and Discussion

After implementing all the steps in Section 3 we test the various behaviours by switching between them and observing the rendered robot. We also plot the evolution of the *pos* and *vel* state as a time series and with phase portraits.

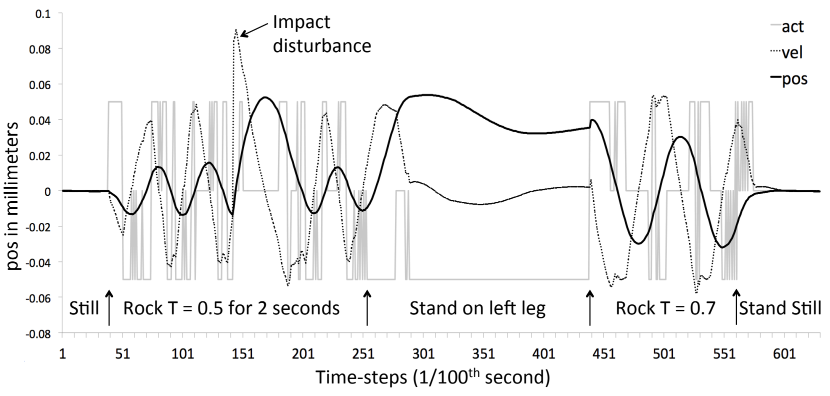

Fig. 3. Time series showing the value of *pos*, *vel* and the *action* for the series of behaviours described in Section 4. Note the quick recovery after the sideway impact disturbance and the smooth transitions between behaviours.

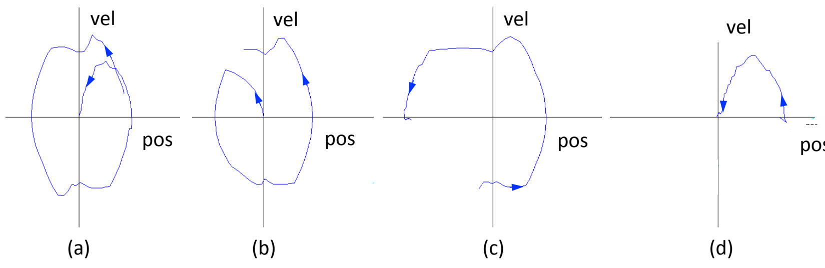

Fig. 4. Example phase portraits, switching between policies: (a) T=0.7sec rock to standing upright, (b) standing upright to T=0.7sec rock, (c) T=0.7sec rock to standing on right leg, (d) standing on left leg to standing upright.

The combination of switching and durations for the behaviours is infinite, so we show results for an example series of behaviours, namely: from a standing start, execute a rock with period 0.5 seconds for 2 second, then stand on the left leg for 2 seconds, followed by a rock of period 0.7 seconds for 1 second, and back to the standing upright position. During the 2 second rock we impart a sideways impact force (delivered by shooting a cannon-ball at the machine). The results are plotted in Figures 3. Figure 4 shows phase portraits switching between several behaviours. Both Figures show stable behaviour for each policy and when switching.

If the transition function in Section 3.3 is instead learned while lifting the swing foot, the RL will use the replacement of the foot to increase the acceleration of the body. This is a consequence of maximising future reward. In practice this may cause the robot to thump its feet, increasing wear and tear. For this reason we settle for learning without lifting the foot. The downside is that introducing the lifting action subsequent to learning, introduces what appears as a disturbance to the natural cycle and hence the kink in the limit cycles seen in the phase portraits in Figure 4.

The sideways rock was combined with an open loop forward (sagittal) and a sideways gait. Both performed well for small step-sizes, but there is much do to. With a RL sagittal gait under our belt[14] the next step is to combine and synchronise the two orthogonal motions.

Another step is to adapt the simulated results to the real Nao. An advantage of our approach is that the transition function learned on the simulator can be used as a starting point and adjusted on the real robot hopefully with less training time. The actions (in our case ankle rolls) are discrete. A fruitful development would to introduce continuous actions by function fitting and interpolation.

5. Summary

In summary, we have presented a promising approach to tackle more versatile stable biped locomotion, generating multiple behaviours from the one learned transition function, and seamlessly and optimally switching between the behaviours.

References

1. R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction* (MIT Press, Cambridge, Massachusetts, 1998).
2. T. McGeer, Passive dynamic walking, *I. J. Robotic Res.* **9**, 62 (1990).
3. S. Collins, A. Ruina, R. Tedrake and M. Wisse, Efficient bipedal robots based on passive-dynamic walkers, *Science* **307**, 1082 (2005).
4. S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi and H. Hirukawa, Biped walking pattern generation by using preview control of zero-moment point, *ICRA'03* , 1620 (2003).
5. T. B. Sheridan, Three models of preview control, *IEEE Transactions on Human Factors in Electronics* **7**, 91(June 1966).
6. E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi and B. Morris, *Feedback Control of Dynamic Bipedal Robot Locomotion* (CRC Press, Boca Raton, 2007).
7. A. Ames and R. Gregg, Stably extending two-dimensional bipedal walking to three dimensions, *American Control Conference* , 2848 (2007).
8. J. W. Grizzle, C. Chevallereau, A. D. Ames and R. W. Sinnet, 3d bipedal robotic walking: Models feedback control, and open problems, *8th IFAC Symposium on Nonlinear Control Systems* (2010).
9. R. Tedrake, Stochastic policy gradient reinforcement learning on a simple 3d biped, in *Proc. of the 10th Int. Conf. on Intelligent Robots and Systems*, 2004.
10. J. Morimoto, G. Cheng, C. Atkeson and G. Zeglin, A simple reinforcement learning algorithm for biped walking, in *Robotics and Automation, 2004. Proceedings. ICRA '04. 2004 IEEE International Conference on*, april-1 may 2004.
11. S. Wang and J. Braaksma, Reinforcement learning control for biped robot walking on uneven surfaces, in *Proceedings of the 2006 International Joint Conference on Neural Networks*, 2006.
12. J. Morimoto, J. Nakanishi, G. Endo, G. Cheng, C. G. Atkeson and G. Zeglin, Poincaré-map-based reinforcement learning for biped walking., in *ICRA'05*, 2005.
13. B. Hengst, On-line model-based continuous state reinforcement learning using background knowledge, *Twenty-Fifth Australasian Joint Conference on Artificial Intelligence (AI12)* (2012).
14. B. Hengst, M. Lange and B. White, Learning ankle-tilt and foot-placement control for flat-footed bipedal balancing and walking, *11th IEEE-RAS International Conference on Humanoid Robots* (2011).