Synthesis of Controllers for Stylized Planar Bipedal Walking

Dana Sharon, and Michiel van de Panne
Department of Computer Science
University of British Columbia
Vancouver, BC, V6T 1Z4, Canada
{dsharon,van}@cs.ubc.ca

Abstract— **We present a method for computing controllers for stable planar-biped walking gaits that follow a particular style. The desired style is specified with a kinematic target trajectory that may or may not be physically realizable. A nearest-neighbor controller representation is used and its free parameters are optimized using a local parameter search technique. The optimization function is constructed by integrating a mass-distance metric over fixed time intervals, which serves to measure the deviation of a simulated motion from a desired target motion. We demonstrate simulated bipedal walks having user-specified styles, walks for bipeds of varying dimensions, walks over terrain of known slopes, and walks that are robust with respect to unobserved terrain variations and modeling errors.**

Index Terms— **biped locomotion, policy search, reward shaping**

I. Introduction

Walking bipeds are high-dimensional non-linear systems, and as such they are challenging to control. There are nevertheless a large variety of published control algorithms, whose diversity stems in part from the many ways in which one can pose walking as a control problem. For example, one can seek to minimize control energy, maximize robustness and stability, mimic a particular theory of gait generation or motor learning, or develop a controller that tackles some combination of these goals.

For many applications, however, it is not sufficient to merely achieve a stable walking motion or one that is energy-optimal. Also needed is the ability to provide a designer with control over the particular *style* of walking motion. This is particularly the case for entertainment robotics, which is currently one of the most important applications for walking control algorithms. In this paper we present a novel method for controlling walks having a desired style and we demonstrate our method for multiple styles of walk, multiple types of biped, and variable terrain. We also show that our method is robust with respect to significant modeling errors.

A. Related Work

Many proposed methods use the zero moment point (ZMP) to define a motion trajectory that is guaranteed to be dynamically stable [1]–[4]. A walking pattern or reference trajectory is typically computed offline that satisfies the necessary ZMP constraints and online tracking is then used to robustly realize the desired trajectory. The reference trajectory may be generated using motion capture along with a dynamic filter [5] that ensures a physically feasible motion. ZMP controllers are popular because they can ensure robot stability in a controlled environment. However, these controllers require accurate models of both the robot and the environment. More significantly, in order to avoid inverse-kinematic singularities, they result in 'bent-knee' motions that do not look natural and that place significant limits on the types of walking styles that can be produced.

The *virtual model control* method [6] introduces a number of virtual components to construct a controller and is appealing for its simplicity. This strategy has been shown to successfully control a bipedal model walking blindly over variable terrain [7]. However, the method involves many manually-tuned control parameters and does not allow for the direct specification of a desired motion style.

Recent work has seen an increasing use of the reinforcement learning (RL) framework [8] to learn control strategies for bipedal walking from delayed rewards. Unfortunately, many model-based RL methods suffer from the curse of dimensionality: the algorithms' time and memory requirements grow exponentially with the dimensionality of the state. Nevertheless, RL has been used to learn walking motions for simple, low dimensional, bipeds [9]–[12]. The approaches are typically shown to work for one model and result in one walking style. Another approach to RL is to apply *policy search* directly without learning a model of the system. A policy or controller can be represented and parameterized in a variety of ways [13], [14] and local or global parameter optimization is then used to search for "good" controllers. The primary difficulty is that the parameter spaces to be searched are typically inordinately large because of their multidimensional nature and they are usually replete with undesirable local minima.

Using prior knowledge about the desired motion can greatly simplify controller synthesis. *Imitation-based learning* or *learning from demonstration* allow for policy search to focus only on the areas of the search space that pertain to the task at hand. Imitation based learning for a 5-link planar biped locomotion is demonstrated in [15], which uses a central-pattern generator (CPG) approach to reproduce one walking style for a specific biped model over flat terrain. In [16], the authors present an imitation-based method for controlling whole body dance motions. The lower body control is ZMP-based and thus shares the general limitations of ZMP-based walking methods.

II. Overview of our Approach

The starting point of our method is the specification of a desired motion style. We require one cycle of the desired gait, which can be specified using keyframes and interpolation or could also come from motion capture data. In this work, the target motion is defined using a set of keyframes and transitions times. The degrees of freedom (DOFs) for the model are then linearly interpolated over time between the keyframes to produce the target motion.

The primary purpose of the target motion is to guide an offline search process to produce a controller that generates balanced, robust motions that are similar to the given motion. It is important to note that the target motion does not serve the role of a traditional reference trajectory, i.e., for online use in making corrections to the walking motion. Our target motion may in fact be physically infeasible and thus it does not have any of the typical constraints demanded of reference trajectories that are used for ZMP-based methods or other local linearization methods.

Our controllers operate using partial-state feedback. As input, they take the current pose of the biped, where we define the *pose* to be the biped's global orientation and its joint angles. As output, the controller produces a target pose that defines a set of target joint angles for proportional-derivative (PD) controllers. Our controllers employ a simple nearest-neighbor internal representation, the details of which we present in Section III.

Controllers have a number of free parameters that govern their input-output mapping. These need to be set so as to produce stable walking gaits of the desired style. This is done using an offline parameter-search optimization process (i.e., direct policy search) the details of which are described in Section IV. As with many large optimization problems, the choice of optimization function and the starting point for the optimization can have a significant influence on the quality of the resulting solutions. We discuss these issues in Section V.

We extensively test our control scheme using several walking styles, a variety of biped models, and on variable terrain. These experiments are documented in Section VI. In Section VII we summarize our contributions and discuss future work.

III. Nearest-Neighbor Controller Representation

Our controller consists of a collection of nodes, $\omega_{1:n}$, that are used to implement a nearest-neighbor control scheme. Each node ω_i has a location, $\zeta_i \in I^m$, and an associated output, $\mu_i \in O^p$, where I^m and O^p define the controller's *input-space* and *output-space*, respectively. The location parameters locate each node in the input space and are used to determine which node is *active*. The *active node*, ω_a, is defined as the nearest-neighbor to the current known state of the dynamically simulated character and is computed as

$$a = argmin_i(||\mathbf{w}^T(\zeta_\mathbf{t} - \zeta_i)||_2). \tag{1}$$

where a is the index of the active node, ζ_i is the location or center pose of the i_{th} node, ζ_t is the current simulated character pose, and $\mathbf{w} \in R^m$ is a vector of weights for the various elements in ζ. Once the active node is found, its output (a target pose) defines the controller's output, μ_a, which is used to drive the individual biped joints.

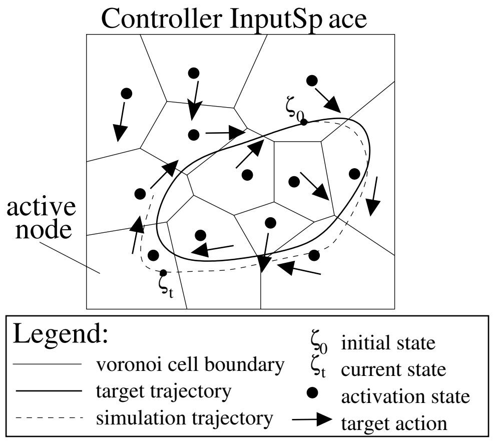

Fig. 1. Abstraction of Controller Representation and Operation

Figure 1 shows an abstraction of our controller representation in operation. The solid loop is a projection of a motion trajectory that we want our character to follow. The solid circles are the node locations in the input space, $\zeta_{1:n}$, and the arrows are an abstraction of the node outputs, $\mu_{1:n}$, which are sets of target joint angles in our case. The node locations serve to partition the input-space into a set of high-dimensional Voronoi cells. The dashed line represents a simulated trajectory as guided by our controller. The biped's position and velocity are initialized using a point on the target trajectory. The illustrated simulation begins with the character in pose ζ_0 and continues on to pose ζ_t, the current state of the simulation. A nearest neighbor search reveals that the lower left node is active (i.e., closest to ζ_t). This node's target pose is thus used at the current simulation time step.

We note that the control representation can also be thought of as a type of hybrid controller because of the way it partitions the input space into discrete regions. Our particular usage of the control representation also resembles that of a cyclic finite state machine where the finite-state transitions are triggered by crossing the hyperplanes separating the Voronoi cells. The optimization process can be thought of as adjusting the positioning of these hyperplanes as well as the outputs assigned to the cells.

IV. Controller Optimization

The node locations and node outputs are free parameters of the representation and are thus not known in advance. The values of these parameters are determined through an optimization that minimizes a cost function[1] $C(\Theta)$, where Θ denotes the vector of all the free parameters of the controller, $[\zeta_{1:n}, \mu_{1:n}]$. The cost function reflects how well a given controller follows the desired trajectory over a given time interval. If designed with care, it can lead to a well-shaped optimization landscape with meaningful local minima. We defer the details of the cost function and the general notion of *shaping* to Section V, but the important implication is that local (greedy) hill-climbing techniques can be used instead of needing to consider global search.

We employ a simple deterministic search (SDS) algorithm, also known as *coordinate search*, to find the optimal controller parameters. A formal analysis of the convergence properties of this class of algorithms can be found in [17]. The core SDS algorithm is summarized in Figure 2. In the absence of known derivative or gradient information we explore the corresponding parameter space by making systematic evaluations of $C(\Theta \pm \Delta\Theta_i)$ for each free parameter i in Θ. SDS uses a (local) hill climbing approach by updating Θ_{opt} if it yields a better cost function value.

We iteratively reduce the size of parameter perturbations, α. Using this coarse-to-fine method allows us to first explore larger neighborhoods in the parameter space and then refine our search once we get closer to our local minimum.

```
Input:

    starting point $\Theta$
    perturbation step size $\alpha$
    decay factor $\beta$
    $e_{best} \leftarrow C(\Theta)$

Algorithm:

    repeat
        for each element $i$ in $\Theta$ do
            $\Delta\Theta \leftarrow \alpha E$ {$E \triangleq$ the Kronecker delta, $\delta_i$}
            $e_{new} \leftarrow C(\Theta + \Delta\Theta)$
            if $e_{new} < e_{best}$ then
                $\Theta \leftarrow \Theta + \Delta\Theta$
                $e_{best} \leftarrow e_{new}$
            else
                $e_{new} \leftarrow C(\Theta - \Delta\Theta)$
                if$e_{new} < e_{best}$ then
                    $\Theta \leftarrow \Theta - \Delta\Theta$
                    $e_{best} \leftarrow e_{new}$
                end if
            end if
        end for
        $\alpha \leftarrow \beta\alpha$
    until converged

Output:
    $\Theta_{opt} \leftarrow \Theta$
```

Fig. 2. Simple Deterministic Search algorithm (SDS)

V. Shaping the Optimization

The success of direct search techniques such as ours is largely dependent on the existence of meaningful and well-defined local optima in the search space. The choice of controller representation and of the cost function can both significantly affect the structure of the local optima.

We use *shaping* to refer to all factors that impact upon the ability of local hill climbing to find meaningful solutions. This includes, but is not limited to, reward shaping [18]. Below we summarize three general forms of shaping that play an important role in our work.

A. Initialization

In local search algorithms such as ours, a successful optimization may largely depend on the parameter initialization. We initialize controller nodes to appropriately-spaced positions along the cyclical target motion, as shown in Figure 3(a). The number of nodes is chosen based on experience and generally depends on the complexity of the motion trajectory we wish to mimic. Both center poses, ζ_i, and target poses, μ_i, lie directly on the trajectory. We assign $\mu_i = \zeta_{i+1}$, i.e., each node's target pose will, in the absence of external forces, drive the character towards the center pose of the next node. Such an initialization is intuitive and it helps shape the controller optimization by focusing on those areas of the pose-space that are of interest, namely, those surrounding the desired trajectory. The dashed line in Figure 3(a) is an abstraction of a simulated trajectory and indicates that the initial guess at the values of the controller parameters is suboptimal.

[1]We use the terms *cost function* and *reward function* interchangeably.

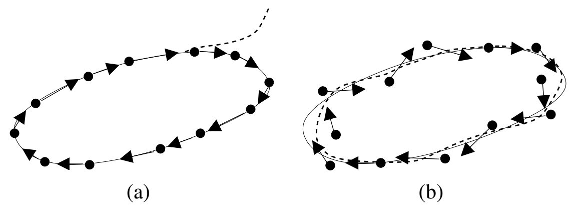

Fig. 3. (a) Controller parameters are initialized to points along the target trajectory. (b) Optimized controller parameters.

Figure 3(b) illustrates an abstraction of the same controller after optimization, showing a change in the node locations and the node actions. The dashed line represents a trajectory as guided by the controller and it approximately follows the desired trajectory over the entire cycle.

B. Style-Based Cost Function

The choice of an appropriate cost function is important and challenging. Our controller has two potentially conflicting goals, that of following a desired motion trajectory and that of maintaining a balanced walk. This second goal can be implicitly taken into account by measuring the ability to mimic the target trajectory over longer time intervals. Actions that result in falling motions will naturally cause larger deviations from the desired motion.

Our cost function, $C(\Theta)$, integrates the deviation, $\delta(t)$, of a physics-based walk simulation from the target walk cycle. The starting state of the dynamic simulation is initialized to coincide with a starting point on the kinematic target motion. At any time t, the error, $\delta(t)$, between the dynamically simulated character and the kinematic target walk cycle is defined using a mass-distance metric integrated over each link of the biped model:

$$\delta(t) = \sum_{i=1}^{M} \int_L \rho_i (x_i(t) - x'_i(t))^2 dx, \quad (2)$$

where M is the number of character links, L is the length of a link, $x_i(t)$ is a point on the i_{th} link at time t for the dynamically-simulated biped, $x'_i(t)$ is the location of the same point at the same time instant for the target walk cycle, and ρ_i is the associated density for the i_{th} link.

In practice, we approximate the continuous mass-distance integral by representing each link using two point masses, one placed at each end of the link. Each point mass is assigned half the weight of the link. Integrating this error over time gives us the cost function:

$$C(\Theta) = \int_0^{T_{sim}} \delta(t) dt \quad (3)$$

where T_{sim} is the duration of the simulation.

C. *Bootstrapping*

We optimize over a sequence of optimization *episodes*, where each episode has a progressively longer simulation time, T_{sim}. Typically we begin with $T_{sim} = 0.2s$ and finish with $T_{sim} = 2s$. The optimized controller resulting from each episode is used as the initial guess for the next episode. When a target trajectory is physically infeasible, the controller will adapt to deviate from the target trajectory in the short term in order to better follow the target trajectory over a longer time interval.

VI. Results

We test our control method using multiple biped models, multiple walk styles, several terrain slopes, and several different types of modeling errors. The animations shown will continue to walk indefinitely, as defined by a $70s$ test simulation. The equations of motion are derived using a reduced-coordinate Newton-Euler formulation. Animations of our results can be found at `www.cs.ubc.ca/~dsharon/icra05.html`.

A. *Character Variety*

We apply our control strategy to three planar biped models as shown in Figure 4. The human character in Figure 4(a) and the robot (mechbot) in Figure 4(c) both have 7 links and 6 pin-joints. The more complex human model in Figure 4(b) has 16 links and 15 joints. A torso-servo acts on the hip of the current stance leg in order to drive the torso to follow the global orientation as given by the kinematic target motion. We implement joint angle limits for all our controllers using exponential springs. We employ stiff linear springs-and-dampers as our ground contact model. Table I gives mass and link parameters for the 7-link biped model. Torque limits for the hip, knee, and ankle joints are set to 370Nm. Refer to [19] for joint stiffnesses and for the specifications of the other two models.

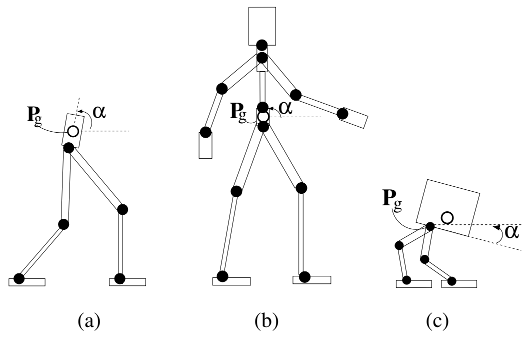

Fig. 4. Biped Models: (a) 7-link human. (b) 16-link human. (c) 7-link mechbot.

segment	mass (kg)	length (m)	inertia (kgm^2/s^2)
torso	40	0.2	0.1333
thigh	7	0.46	0.1234
shin	4	0.44	0.0645
foot	2	0.22	0.00807

TABLE I
Link Parameters for the 7-Link Human

Figures 5(a), 5(c), and 5(e) show user-specified keyframes that define target motion trajectories for each biped model. Figures 5(b), 5(d), and 5(f) show simulation results using our optimized controllers. The controllers have 12 nodes each. All the walks are left-right symmetric, both in terms of the keyframes defining the target motion, as well as the controller nodes. Thus, only 6 nodes require optimization. For the 16-link character, only the hip, knee, and ankle motions were optimized. This is because we assume that the lower body can react appropriately to the predictable upper body movements. We thus have a total of 6 nodes $\times$ 12 parameters per node = 72 free parameters for each of our biped models.

B. *Style*

Figure 6 demonstrates our method's ability to produce walks with significantly different styles and stride durations. Figures 6(a) and 6(c) show the keyframes defining two desired walking styles that are both significantly different

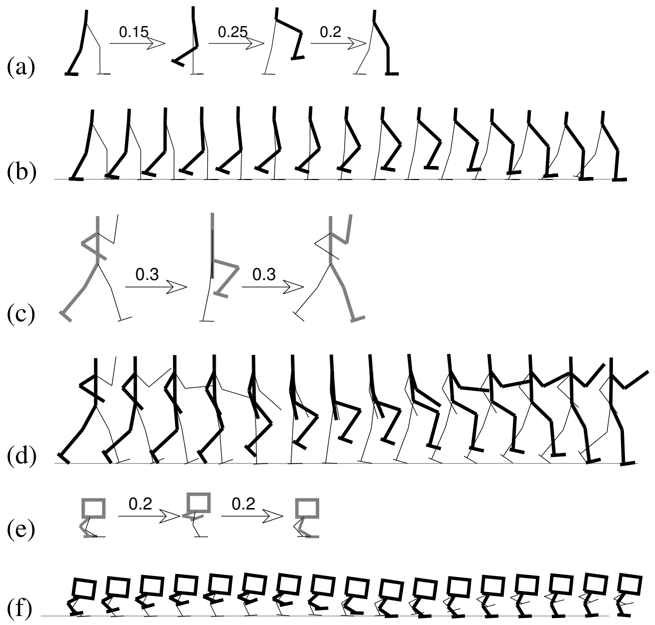

Fig. 5. Character Controllers: keyframes (a,c,e) and controller driven simulations (b, d, f) for each character.

from the basic walk given in Figure 5(a). The first style has a prolonged double stance period and an exaggerated lift of the knee in the fourth keyframe. The second novel style is very unconventional in that it has an extra backward swing in the third keyframe and thus has an alternating forward-backward-forward swing of the swing leg before footstrike. In this case there is a potential ambiguity in using the pose alone to determine the current active node, given that similar poses are reached at multiple points in the walk cycle. We deal with this issue by introducing a restriction on the sequence in which nodes can be activated. Figures 6(b) and 6(d) show the walks that result from our controllers. Note that the style is mimicked closely. Both of these controllers were defined using 24 nodes. With left-right symmetry enforced, only 12 nodes require optimization. Each node has 12 free parameters, resulting in a total of 144 free parameters.

C. Terrain

We test our method on the 7-link human biped walking uphill. This controller is created and optimized specifically for the slope presented to it. Figure 7(a) shows the keyframes used to define a desired walk up a hill. The hill has a slope of 10 degrees. As before, the controller has 12 nodes and a total of 72 parameters to be optimized. Figure 7(b) shows a dynamic simulation of our controller. Only one leg is drawn for clarity.

D. Robustness

We test our controller for robustness with respect to errors in character model parameters and disturbances from the environment. All the walks shown in Figure 8 were produced using the controller illustrated in Figure 5(a) and (b).

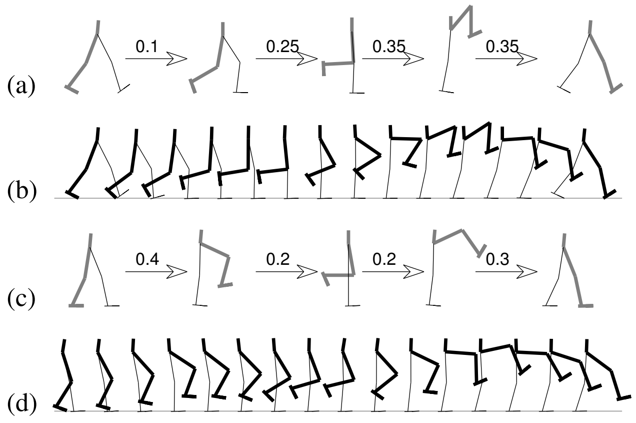

Fig. 6. (a) Keyframes and (b) resulting simulation of a stylized walk with a high leg lift. (c) Keyframes and (d) resulting simulation of a stylized walk with an extra back-swing of the swing leg.

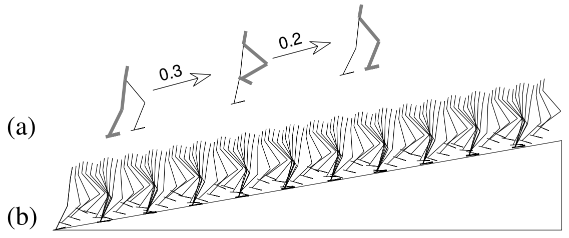

Fig. 7. Walk up a hill with 10 degree slope.

Figure 8(a) shows a walk result for a biped model having a significantly reduced torso mass. Figure 8(b) and 8(c) show remarkable robustness with respect to significant link length changes. Figure 8(d) shows a walk over unanticipated bumpy terrain. The character is walking blindly in that it continuously expects flat ground and must recover at each step. Note that the controller is not re-optimized for these new situations, but rather it is robust to the applied changes.

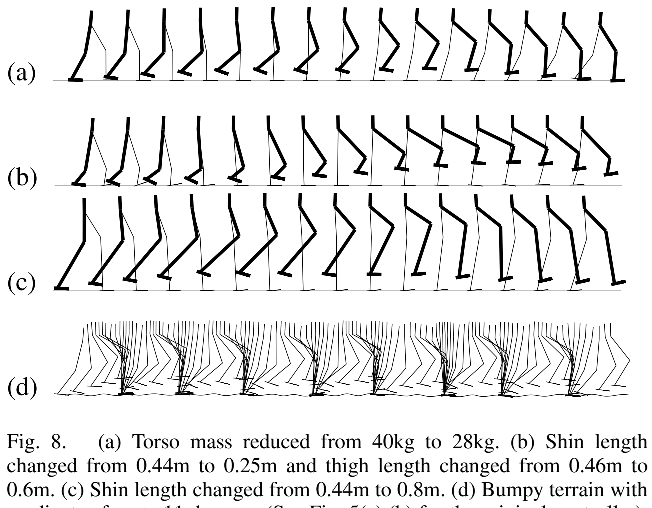

Fig. 8. (a) Torso mass reduced from 40kg to 28kg. (b) Shin length changed from 0.44m to 0.25m and thigh length changed from 0.46m to 0.6m. (c) Shin length changed from 0.44m to 0.8m. (d) Bumpy terrain with gradients of up to 11 degrees. (See Fig. 5(a),(b) for the original controller).

Figure 9 shows a typical example of the performance of the learned controller as the SDS search proceeds. Because we shape the problem by performing staged increases of the simulation durations, T_{sim}, there are step-wise discontinuities at points where a new optimization episode begins.

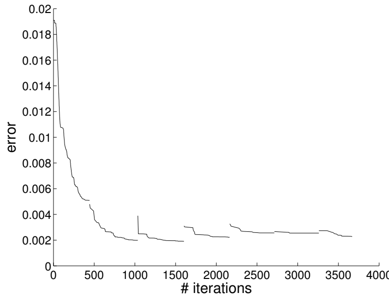

Fig. 9. Optimization evaluation as a function of the number of iterations. The optimization proceeds using multiple episodes of increasing simulation duration.

The search process requires approximately 450–23,000 cost function evaluations to produce useful controllers for our walking problems. At present, this requires from 1.5 minutes up to 2.5 hours on a 1 GHz Pentium 3 PC.

VII. Conclusions

A. Contributions

We have presented a method for learning to control planar bipedal walking gaits of a given style. The resulting controllers are tested for robustness with respect to a variety of terrain changes and to significant biped modeling errors. We also demonstrate our method on a variety of biped models. We address the potential problems of overly large search spaces and undesired local minima through several shaping techniques, including progressive changes of the cost function, suitable initialization, and the use of a mass-distance metric.

B. Future Work

Our method is not fully automated in that it does not succeed in producing stable walk controllers for all possible kinematic target motions. One issue is that the current cost function is based upon dynamic simulations from a single initial state extracted from the first keyframe of the target motion. The optimization has problems if this initial state is far away from any physically-realizable limit cycle. Instead of evaluating the cost function from one initial state, it could be more useful to evaluate it from multiple states that are distributed along the desired trajectory. Additionally, there are situations where the optimization makes sacrifices in terms of robustness in order to more precisely track the target motion. The addition of disturbances during the controller evaluation may encourage robustness by forcing the evaluated motions through areas of the state-space that are further removed from the achievable limit cycle.

Representations for motor control should ideally provide support for parameterization. It is natural to consider parameterizing a walking gait according to walking speed, stride length, and other such parameters. Synthesizing motions with a focus on both style and energy-efficiency would be desirable. Lastly, we wish to develop the equivalent capability to produce robust controllers for highly stylized 3D walks.

References

[1] Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, and K. Tanie, "Planning walking patterns for a biped robot," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 280–289, June 2001.

[2] R. Zhang and P. Vadakkepat, "An evolutionary algorithm for trajectory based gait generation of biped robot," in *Proceedings of the International Conference on Computational Intelligence, Robotics and Autonomous Systems*, 2003.

[3] 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," in *Proceedings of the IEEE International Conference on Robotics and Automation*, September 2003, pp. 1620–1626.

[4] K. Hirai, M. Hirose, Y. Haikawa, and T. Takenake, "The development of honda humanoid robot," in *IEEE International Conference on Robotics and Automation*, 1998, pp. 1321–1326.

[5] K. Yamane and Y. Nakamura, "Dynamics filter - concept and implementation of on-line motion generator for human figures," in *Proc. of the Intl. Conf. on Robotics and Automation*, 2000.

[6] J. E. Pratt, "Virtual model control of a biped walking robot, Tech. Rep. AITR-1581, 1995. [Online]. Available: citeseer.ist.psu.edu/article/pratt95virtual.html

[7] C.-M. Chew, J. Pratt, and G. Pratt, "Blind walking for a planar bipedal robot on sloped terrain," in *Proc. of the Intl. Conf. on Robotics and Automation*, 1999.

[8] L. Kaelbling, L. Littman, and A. Moore, "Reinforcement learning: a survey," *Artificial Intelligence Research*, vol. 24, pp. 237–285, 1996.

[9] J. Morimoto, G. Zeglin, and C. G. Atkeson, "Minimax differential dynamic programming: Application to a biped walking robot," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2003.

[10] M. Sato, Y. Nakamura, and S. Ishii, "Reinforcement learning for biped locomotion," in *International Conference on Artificial Neural Networks*, 2002, pp. 777–782.

[11] R. Tedrake, T. W. Zhang, and H. S. Seung, "Stochastic policy gradient reinforcement learning on a simple 3d biped," in *IEEE International Conference on Intelligent Robots and Systems*, 2004.

[12] J. Morimoto, G. Cheng, C. G. Atkeson, and G. Zeglin, "A simple reinforcement learning algorithm for biped walking," in *Proceedings of the IEEE International Conference on Robotics and Automation*, April 2004, pp. 3030 – 3035.

[13] M. van de Panne and E. Fiume, "Sensor-actuator networks," in *Proceedings of ACM SIGGRAPH*, 1993, pp. 335–342.

[14] J. Auslander, A. Fukunaga, H. Partovi, J. Christensen, L. Hsu, P. Reiss, A. Shuman, J. Marks, and J. T. Ngo, "Further experience with controller-based automatic motion synthesis for articulated figures," *ACM Transactions on Graphics*, vol. 14, no. 4, pp. 311–336, 1995.

[15] J. Nakanishi, J. Morimoto, G. Endo, G. Cheng, S. Schaal, and M. Kawato, "Learning from demonstration and adaptation of biped locomotion," *Robotics and Autonomous Systems*, vol. 47, pp. 79–91, 2004.

[16] S. Nakaoka, A. Nakazawa, and K. Yokoi, "Generating whole body motions for a biped humanoid robot from captured human dances," in *IEEE Intl. Conference on Robotics and Automation*, September 2003, pp. 3905–3910.

[17] V. Torczon, "On the convergence of pattern search algorithms," *SIAM Journal on Optimization*, vol. 7, no. 1, pp. 1–25, 1997.

[18] A. Y. Ng, D. Harada, and S. J. Russell, "Policy invariance under reward transformations: Theory and application to reward shaping," in *Proc. of the 16th Intl. Conf. on Machine Learning*, 1999, pp. 278–287.

[19] D. Sharon, "Synthesis of stylized walking controllers for planar bipeds," Master's thesis, University of British Columbia, Aug 2004.