# Robust Policies via Meshing for Metastable Rough Terrain Walking

Cenk Oguz Saglam and Katie Byl
University of California, Santa Barbara
{saglam, katiebyl}@ece.ucsb.edu

*Abstract*—In this paper, we present and verify methods for developing robust, high-level policies for metastable (i.e., rarely falling) rough-terrain robot walking. We focus on simultaneously addressing the important, real-world challenges of (1) use of a tractable mesh, to avoid the curse of dimensionality and (2) maintaining near-optimal performance that is robust to uncertainties. Toward our first goal, we present an improved meshing technique, which captures the step-to-step dynamics of robot walking as a discrete-time Markov chain with a small number of points. We keep our methods and analysis generic, and illustrate robustness by quantifying the stability of resulting control policies derived through our methods. To demonstrate our approach, we focus on the challenge of optimally switching among a finite set of low-level controllers for underactuated, rough-terrain walking. Via appropriate meshing techniques, we see that even terrain-blind switching between multiple controllers increases the stability of the robot, while lookahead (terrain information) makes this improvement dramatic. We deal with both noise on the lookahead information and on the state of the robot. These two robustness requirements are essential for our methods to be applicable to real high-DOF robots, which is the primary motivation of the authors.

## I. Introduction

Legged robots may potentially replace or assist humans in tasks involving difficult and/or dangerous environments not suitable for wheels, but where mobility is essential. As highly dynamic solutions to legged mobility, both hopping/running robots [16] and passive dynamic inspired bipeds [10] have been extensively researched and demonstrated. Stability is a major concern for such robots, and one aspect of performance that has not been widely studied is the optimal use of upcoming terrain information. Instead, work to date has typically focused either on remaining robust when blind to upcoming terrain [15, 13] or on achieving particular footstep lengths [7], without more generally addressing the issue of planning on partly-known terrain. A major reason such control problems are not yet adequately addressed in robotics, we argue, is the need for better methods to model legged systems and to quantify their stability. In this work, we present new methods to represent a high-dimensional, nonlinear dynamic system with a relatively small mesh and robustly employ the resulting meshes to optimize high-level policies for highly reliable (metastable) walking. We also apply these robust policies derived through meshing to the problem of walking with noisy terrain knowledge as a case example.

Demonstrations of stable limit cycles for passive walkers [10] have motivated the development of underactuated

dynamic powered walkers. One approach in control of such walkers is to minimize energy use [4, 2, 5]. However, optimizing for energetics often results in poor stability on rough terrain; such methods lack sufficient robustness. To plan on terrain with limited footholds, Zero Moment Point (ZMP) planning has become very popular. Here, walking motions are planned deliberately to avoid underactuation [23, 21, 14]. While very prominent in humanoid walking research, this approach often results in slow and inefficient walking, motivating many researchers to focus on improving robustness for dynamic walking solutions that more closely resemble human walking. One notable control method for dynamic walking is the Hybrid Zero Dynamics (HZD) approach [24, 25], which has been shown to be successful on a boom-mounted robot and is currently being tested on a 3D robot [6].

Recent work on underactuated bipeds has demonstrated tractable strategies for switching among multiple controllers to cope with rough terrain [19, 13]. A blind (to the terrain information) finite-state machine approach has been shown to increase robustness greatly [13], and switching control based on a one-step lookahead to the terrain slope is shown to be advantageous [19]. While robustness to unexpected (blind) perturbations is definitely useful, lookahead knowledge should also be used if it can improve performance. Our intuition and experience tells us that when we walk blindly, we are not as safe, comfortable, fast, or efficient as in sighted walking. Our work aims to quantify this perhaps obvious intuition more scientifically. In this work, we argue through modeling that sighted walking is dramatically better than optimal state-based switching with blind walking. However, the main focus will be on demonstrating robustness of our modeling, which is needed for our methods to be applicable to an experimental robot.

Although our goal is to keep our methods and results generic for all legged robots, the ideas will be illustrated on a planar, five-link, underactuated biped in this paper. We will assume there are qualitatively different low-level controllers, which might be designed using different control schemes. What is important is that they should behave differently and that we can make use of all of them effectively. We then mesh the state space to approximate hybrid dynamics in discontinuous time. Our algorithm differs from many others by being deterministic as opposed to being random as in [9]. To come up with high-level walking policies, we use dynamic programming. In this sense, our methods are aimed in particular at a machine-learning framework, which is shown to be useful for bipedal

locomotion research [11, 12, 22, 26].

In addition to our particular results on sighted walking presented in [19], key contributions of this work include (1) new methods for tractable meshing of high dimensional (10D) dynamics, (2) techniques for improving policy robustness, and (3) quantifying the validity of both our meshing and the resulting policy. The rest of the paper is organized as follows. Sec. II describes our problem statement and walking model, followed by a brief description of low-level controllers in III. Sections IV and V describe our meshing methods and value iteration algorithm, respectively, and VI investigates robustness. Finally, conclusions and comments on applicability and future work are presented in VII,and VIII.

## II. PROBLEM STATEMENT AND MODEL

Consider a legged robot that is walking, running, and/or hopping. Furthermore, assume there are multiple controllers available, each having advantages and disadvantages under different circumstances. Each of them might be obtained by different methods, and/or optimal for different cost functions. The problem we address is: How can the robot optimally switch among them?

We will consider biped walking, and our concern will be stability. There will be three controllers, designed for flat, downhill, and uphill grounds. Both state and noisy information about the terrain ahead will be available to the controller. The goal is to find a robust, near-optimal control policy.

### A. Model

The analysis in this paper will be carried out with a 5-link biped as shown in Figure 1. It has point feet and it is underactuated by 1 DOF. The angles shown in the figure form $q := [q_1\ q_2\ q_3\ q_4\ q_5]^T$. The ten dimensional state of the robot is defined as $x := [q^T\ \dot{q}^T]^T$.
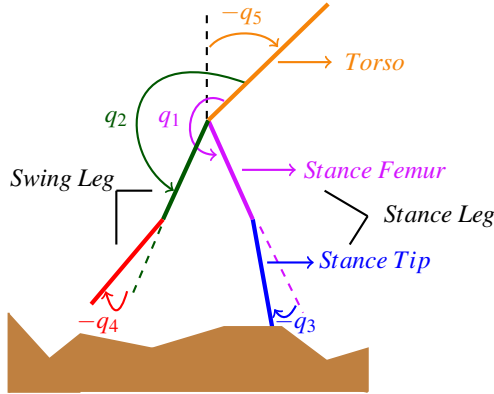


Fig. 1.    Illustration of the five-link biped

Depending on the number of legs in contact with the ground, the robot will be either in the single or double support phase. Walking consists of these two phases in sequence. The single support phase has continuous dynamics, which can be derived in the following canonical form using a Lagrangian approach.

$$D(q)\ddot{q} + C(q,\dot{q})\dot{q} + G(q) = Bu, \tag{1}$$

where $u$ is the input. As the robot has point feet, the double support phase can be well captured as an impact event.

$$x^+ = \Delta(x^-), \tag{2}$$

where $x^-$ and $x^+$ are the states just before and after the impact respectively. Conservation of energy and the principle of virtual work gives the mapping $\Delta$ [25], [8].

## III. CONTROL

In this paper, we propose a general framework that is independent of the structure of the controllers. We anticipate that the same ideas will apply when different control schemes are used. However, we will still provide a short explanation of the particular input-output feedback linearizing controller design used in this work, which is taken from [19].

One challenge for the control of bipedal locomotion is underactuation: $q$ is 5-dimensional, whereas $u$ is 4-dimensional. The latter suggests selecting 4 variables to be controlled, which we name as $q_c$.

$$q_c := [q_2 + q_5\ \ q_3\ \ q_4\ \ q_5]^T \tag{3}$$

Other definitions for $q_c$ are possible, but this choice is verified to work well. Then, adopting an input $u$ in the following form

$$u = (ED^{-1}B)^{-1}(v + ED^{-1}(C\dot{q} + G)), \tag{4}$$

$$where\ E = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix},$$

will lead to

$$\ddot{q}_c = v. \tag{5}$$

The way $v$ is controlled is a matter of choice. Here, we use the sliding mode controller (SMC) scheme as explained in [17] for finite time convergence. The error is defined by

$$e = q_c - q_c^{ref}. \tag{6}$$

And generalized error is given as

$$\sigma_i = \dot{e}_i + e_i/\tau_i, \quad i = \{1, 2, 3, 4\}. \tag{7}$$

Then $v$ in (4) can be chosen to be

$$v_i = -k_i|\sigma_i|^{2\alpha_i - 1}sign(\sigma_i), \quad i = \{1, 2, 3, 4\}. \tag{8}$$

Below is the list of controller parameters used, which were obtained by trial and error. For space considerations, we refer interested readers to the appendix of [20] for an explanation of these parameters.

TABLE I
CONTROLLER PARAMETERS

| $\alpha_1$=0.7, | $\alpha_2$=0.7, | $\alpha_3$=0.7, | $\alpha_4$=0.7 |
|---|---|---|---|
| $\tau_1$=1/10, | $\tau_2$=1/10, | $\tau_3$=1/20, | $\tau_4$=1/5 |
| $k_1$=50, | $k_2$=100, | $k_3$=75, | $k_4$=10 |

In this study, we will obtain different controllers using the same controller parameters, but different references. All references are piece-wise constant and time-invariant. Details of reference design are left out, due to space constraints, but qualitatively, each is designed to bias the walker toward suitability for uphill (leaning forward and smaller steps), flat, or downhill (more upright and larger steps) walking. We note that despite these general design goals, heuristic switching among controllers is far from optimal, and optimal policies are not trivial [19].

## IV. MESH

This section describes our meshing method, which uses a distance metric to adjust the total mesh size as desired within the 10D (5 angles and their velocities) state space. We also review theory for estimating the mean steps to failure for the system and provide numerical evidence that our estimation error converges as the mesh resolution improves. Our eventual meshing goals are to create one mesh with a small number of elements for value iteration and to use a much larger mesh as a means of verifying performance (of the sparse mesh policies), since the true dynamics are captured more accurately as meshing resolution increases.

For the rest of the paper, we are interested in states just before the impact. We define a Poincaré section at the transition from single support phase to double support phase, and refer to those 'states just before the impact' simply as states. So, for the rest of the paper, the term 'state' refers to a discrete-time variable for the walking system.

### A. Distance Metric

As the distance metric for meshing, we will be using standardized (normalized) Euclidean distance in this study: When $a$ is a vector, and $B$ is a set of vectors (growing in size, during meshing) each with the same dimension as $a$, the distance of $a$ from $B$ is calculated as

$$d(a,B) := \min_{b \in B} \left\{ \sqrt{\sum_i \left( \frac{a_i - b_i}{r_i} \right)^2} \right\}, \qquad (9)$$

where $r_i$ is the standard deviation of $b_i$ elements. In addition, the closest point in $B$ to $a$ is given by

$$c(a,B) := \operatorname*{argmin}_{b \in B} \left\{ \sum_i \left( \frac{a_i - b_i}{r_i} \right)^2 \right\}. \qquad (10)$$

### B. Terrain Profile

We will assume the terrain profile is angular, i.e., that it consists of slopes, noted by $\gamma$. The slope ahead only changes at impacts; i.e., it remains constant until the next step. This terrain assumption captures the fact that to calculate the pre-impact state, the terrain for each step can simply be interpreted as a ramp with the appropriate slope. Our general method is still applicable to more complicated terrain models, and we note briefly that the most important modeling detail for future work is to consider vertical (tripping) obstacles in between footholds.

For our piecewise-sloped terrain profile model, the next state of the robot, $x[n+1]$, is a function of the current state $x[n]$, the slope ahead $\gamma[n]$, and the controller used $\zeta[n]$, i.e.

$$x[n+1] = h^t(x[n], \gamma[n], \zeta[n]). \qquad (11)$$

### C. Methodology

Consider a slope set, $S$, and a controller set, $Z$. Two key goals in meshing are: First, we want to have a set of states, $Y$, which well covers the (reachable) part of the state space the robot visits. Secondly, we want to learn what $h^t(y,s,\zeta)$ is for all $y \in Y$, $s \in S$, and $\zeta \in Z$. To achieve these, we must first select what the controller set and slope set are. The controller set, $Z$, consists of the controllers designed and available to the robot. In this paper, we have a controller set with three controllers.

$$Z = \{\zeta_1, \ \zeta_2, \ \zeta_3\} \qquad (12)$$

As the slope set, $S$, we begin by considering all the integer values from -8 up to 8, in degrees.

$$S = \{k^\circ \ | \ k \in \mathbb{Z}, \ -8 \leq k \leq 8\} \qquad (13)$$

The difference between $\gamma$ and $s$ is that $s$ must be in the slope set $S$ for the learned policy, whereas the actual slope ahead, $\gamma$, might be any real value. The range and density of the slope set can be chosen depending on the controllers' performance, and on the robot. Also, the slope set does not have to be evenly spaced, it may be denser around slopes of particular interest. As we increase the density of the slope set, we are able to capture the dynamics more accurately at the expense of higher numbers of points in the final mesh, $Y$.

Next, an initial mesh, $Y_i$, should be chosen. In this study, we use an initial mesh consisting of only two points. One of these points ($y_1$) represents all (conservatively defined) failure states, no matter how the robot failed, e.g. a foot slipped, or the torso touched the ground. The other point is the stable fixed point of the robot model when the terrain is flat and only $\zeta_1$ was used. Then the process explained in Algorithm 1 is used. This algorithm is taken directly from [19], except that the mesh can be grown here in one pass, using only two initial mesh points, due to our distance metric.

### D. Deterministic State-Transition Matrix

Recall the step-to-step dynamics of walking are given by (11) and, thanks to the meshing (state-transition map), that we know $h^t(y,s,\zeta)$ for all $y \in Y$, $s \in S$, and $\zeta \in Z$. We define

$$h^a(x[n], \gamma[n], \zeta[n]) := c(h^t(x[n], \gamma[n], \zeta[n]), Y). \qquad (14)$$

where (10) is used, and the superscript $a$ stands for approximation. Then we write the approximate step-to-step dynamics:

$$y[n+1] = h^a(y[n], s[n], \zeta[n]) \qquad (15)$$

After that, the deterministic state transition matrix can be written as

$$T_{ij}^d(s, \zeta) = \begin{cases} 1, & \text{if } y_j = h^a(y_i, s, \zeta) \\ 0, & otherwise. \end{cases} \qquad (16)$$

**Algorithm 1** Meshing algorithm

**Input:** Initial set of states $Y_i$, Slope set $S$, Controller set $Z$ and threshold distance $d_{thr}$
**Output:** Final set of states $Y$, and state-transition map
1: $\bar{Y} \leftarrow Y_i$ (except $y_1$)
2: $Y \leftarrow Y_i$
3: **while** $\bar{Y}$ is non-empty **do**
4: $\quad \bar{Y}_2 \leftarrow \bar{Y}$
5: $\quad$ empty $\bar{Y}$
6: $\quad$ **for** each state $\bar{y} \in \bar{Y}_2$ **do**
7: $\quad\quad$ **for** each slope $s \in S$ **do**
8: $\quad\quad\quad$ **for** each controller $\zeta \in Z$ **do**
9: $\quad\quad\quad\quad$ Simulate a single step to get the final state $x$,
$\quad\quad\quad\quad\quad$ when initial state is $\bar{y}$, slope ahead is $s$,
$\quad\quad\quad\quad\quad$ and controller $\zeta$ is used. Store this information
$\quad\quad\quad\quad\quad$ in the state-transition map
10: $\quad\quad\quad\quad$ **if** robot did not fall and $d(x,Y) > d_{thr}$ **then**
11: $\quad\quad\quad\quad\quad$ add $x$ to $\bar{Y}$
12: $\quad\quad\quad\quad\quad$ add $x$ to $Y$
13: $\quad\quad\quad\quad$ **end if**
14: $\quad\quad\quad$ **end for**
15: $\quad\quad$ **end for**
16: $\quad$ **end for**
17: **end while**
18: **return** $Y$, state-transition map

---

Note that (16) is the result of our basic, nearest-neighbor approximation, which appears to work well. More sophisticated approximations are left as a topic for future work and would result with the matrix not just having one or zero elements, but also fractional values in between.

### E. Stochastic State-Transition Matrix

The stochastic state-transition matrix $T^s$ is defined by

$$T_{ij}^s := Pr(y[n+1] = y_j \mid y[n] = y_i). \tag{17}$$

To calculate this matrix, the first thing we need to do is assume a distribution over slope set, noted by $P_S$.

$$P_S(s) = Pr(s[n] = s) \tag{18}$$

In this paper, we will assume a normal distribution for $P_S$, with mean $\mu_s$, and standard deviation $\sigma_s$.

$$s[n] \sim \mathcal{N}(\mu_s, \sigma_s^2) \tag{19}$$

When only one of the controllers, say $\zeta_i$ is used (no switching), $T^s$ can be calculated as

$$T^s = \sum_{s \in S} P_S(s) \ T^d(s, \zeta_i) \tag{20}$$

The definition of $T^s$ will always remain the same, but its calculation will be updated when we consider switching between controllers.

### F. Mean First Passage Time

As mentioned, we will be illustrating our methods by optimizing stability of bipedal walking in this paper. To measure stability, we will be looking at the average number of steps before falling. The more steps a robot takes on average, the more stable we consider it to be. To describe the average

number of steps before falling, we borrow the somewhat misleading term Mean First Passage Time (MFPT) in [3], which we note actually characterizes average steps (not time) to failure. First, it is assumed that once the robot falls, it stays that way. The failure state, noted by $y_1$ in this work, is an absorbing state, and it is associated with the largest eigenvalue of $T^s$, $\lambda_1 = 1$. When the second-largest eigenvalue, $\lambda_2$, is very close to unity, its corresponding time constant, $\tau_2$, approximately equals the MFPT:

$$MFPT \approx \tau_2 = \frac{-1}{log(\lambda_2)} \approx \frac{1}{1 - \lambda_2}, \tag{21}$$

and the system is considered "metastable", with failures events being both inevitable but also very rare. Thus, the number of steps the robot is going to take on average can be approximated using the stochastic state transition matrix. For space considerations we are skipping further details which can be found in [3]. In this paper, we will use (21) as the basis for the graphs of the following sections.

### G. Convergence of the Mesh

Note that the number of points in the final mesh, and the accuracy obtained from (15) with this mesh are inversely related to parameter $d_{thr}$ and directly related to the density of the slope set. We argue that, for a dense enough slope set, as $d_{thr} \rightarrow 0$, the accuracy of the mesh, and as a result, the MFPT of the controllers, converges to a result capturing the true, hybrid dynamic system dynamics. For (13), we illustrate convergence numerically by first fixing $\sigma_s = 1.5$ (deg) and plotting six independently obtained meshes with $d_{thr} = \{0.1, 0.2, 0.3, 0.4, 0.5, 0.6\}$. In all plots that follow, each location on the x-axis assumes a different long-term mean in slope, resulting in a different stochastic transition matrix, from (17).

Figure 2 shows the difference in results for $d_{thr} = 0.1$ (more refined) versus $d_{thr} = 0.5$ (more coarse). Here, solid lines are associated with $d_{thr} = 0.1$. Table II shows the convergence. The second row gives the number of points in the mesh, while the third row shows the total area of the difference with $d_{thr} = 0.1$ plot.

TABLE II
MESH CONVERGENCE

| $d_{thr}$ | 0.1 | 0.2 | 0.3 | 0.4 | 0.5 | 0.6 |
|---|---|---|---|---|---|---|
| Size | 394,420 | 65,066 | 21,726 | 10,531 | 6,126 | 4,154 |
| err | 0 | 3.3078 | 5.1959 | 7.9576 | 11.6452 | 14.4813 |

## V. VALUE ITERATION

For the following sections, the $d_{thr} = 0.5$ mesh, which has 6,126 points, will be used to solve for optimal policies.
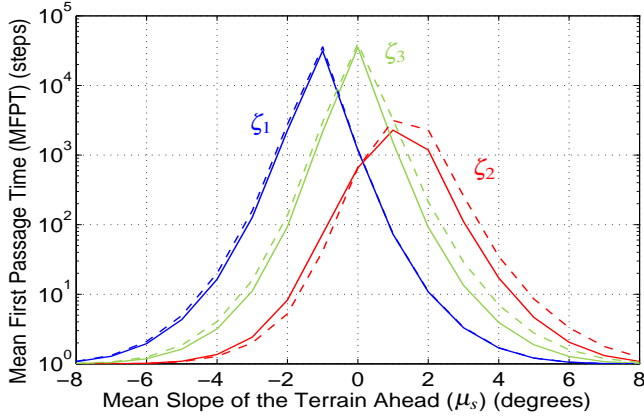
Fig. 2. Slopes ahead of the robot are assumed to be normally distributed as noted in (19) with $\sigma_s = 1.5$. Figure shows average number of steps before falling calculated using (21) versus $\mu_s$ for two independently obtained meshes. Solid line represents the mesh with 394,420 states (obtained using Algorithm 1 with $d_{thr} = 0.1$), whereas the dashed line is a result of choosing $d_{thr} = 0.5$, which results in 6,126 states.

### A. Blind Walking

We define policy, $\pi$, as the function determining the choice of controller, $\zeta[n]$. It is typical to assume a policy is a function of the state in Markov Decision Processes.

$$\zeta[n] = \pi(y[n]) \tag{22}$$

When this policy is applied, the approximate dynamics in (15) will become the following.

$$y[n+1] = h^a(y[n], s[n], \pi(y[n])) \tag{23}$$

In this case, (20) should be also updated as

$$T_{ij}^s = \sum_{s \in S} P_S(s) \; T_{ij}^d(s, \pi(y_i)). \tag{24}$$

We then use value iteration [1] to get the optimal policy.

$$V(i) := \max_\zeta \left\{ \sum_j P_{ij}(\zeta) \; (R(j) + \alpha \, V(j)) \right\} \tag{25}$$

where $V$ is the value, $P_{ij}(\zeta)$ is the probability of transitioning from $y_i$ to $y_j$ when $\zeta$ is used, $R(j)$ is the reward for transitioning to $y_j$, and $\alpha$ is the discount factor, which is chosen to be 0.9. This equation is iterated until convergence to get the optimal policy. Remember that the failure state is $y_1$. The value of the failure state will initially be zero, i.e.,

$$V(1) = 0, \tag{26}$$

and it will always stay as zero due to the following: the reward for taking a successful step is one, while falling has zero reward.

$$R(j) = \begin{cases} 0, & j = 1 \\ 1, & \text{otherwise} \end{cases} \tag{27}$$

Note that the reward function we use does not depend on the controller, slope ahead, or current state. Use of more sophisticated reward functions (e.g., considering energy, speed,

step width) is a topic of [20]. Substituting (26) and (27) into (25), we obtain

$$V(i) := \max_\zeta \left\{ \sum_{j \neq 1} P_{ij}(\zeta) \; (1 + \alpha \, V(j)) \right\}. \tag{28}$$

What is left is to write the probability term.

$$P_{ij}(\zeta) = \sum_{s \in S} P_S(s) \; T_{ij}^d(s, \zeta) \tag{29}$$

The optimization is done for each $\mu_s \in S$ and $\sigma_s = 1.5$. Figure 3 presents the results.
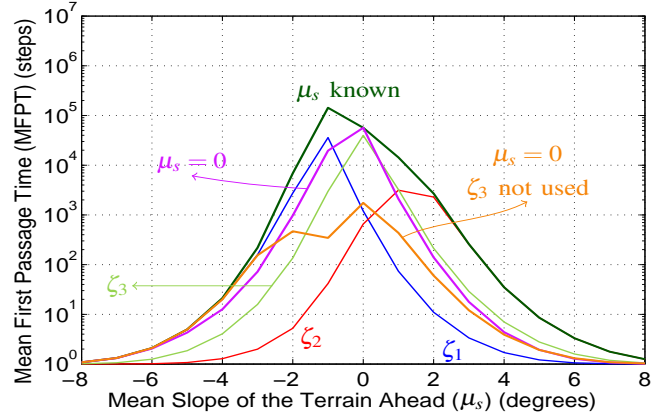


Fig. 3. Slopes ahead of the robot are assumed to be normally distributed as noted in (19) with $\sigma_s = 1.5$. Figure shows average number of steps before falling calculated using (21) versus $\mu_s$ for the mesh obtained via $d_{thr} = 0.5$. In addition to the three fixed controllers, there are three different policies, which walk "blindly", i.e., they have no information about the next slope ahead. The dark green (top) plot assumes mean is known to the controller, while the other two assume $\mu_s = 0$ when deciding the policy. Unlike the other two, the orange (bottom) plot uses only $\zeta_1$ and $\zeta_2$ for optimization. In the light of the performance degradation indicated by this plot, we conclude that use of a third controller, $\zeta_3$, is very helpful for blind walking.

### B. Sighted Walking with One-Step Lookahead

[19] investigates policies obtained using only sight information, without regard for the current state, and shows that this performs very poorly. Next, we consider policies that are functions of both the current state, and the slope ahead, i.e.

$$\zeta[n] = \pi(y[n], s[n]). \tag{30}$$

In this case, the approximate dynamics in (15) becomes

$$y[n+1] = h^a(y[n], s[n], \pi(y[n], s[n])), \tag{31}$$

and $T^s$ should also be updated as

$$T_{ij}^s = \sum_{s \in S} P_S(s) \; T_{ij}^d(s, \pi(y_i, s)). \tag{32}$$

To use the one-step lookahead in deriving policy, we will modify the value iteration.

$$V(i) := \sum_{s \in S} \max_\zeta \left\{ \sum_{j \neq 1} P_{ij}(\zeta, s) \; (1 + \alpha \, V(j)) \right\} \tag{33}$$

Instead of modifying the value iteration algorithm, we could define a new 11-dimensional state, including the slope in

addition. However, (33) makes the analysis of the following parts easier, reduces computational cost, and requires less memory. The probability of 'having $s$ as the slope ahead' and 'transitioning from $y_i$ to $y_j$ when $\zeta$ is used' is simply the multiplication of these two probabilities.

$$P_{ij}(\zeta,s) = P_S(s) \ T_{ij}^d(s,\zeta) \qquad (34)$$

We optimize with $\mu_s = 0$ and $\sigma_s = 1.5$ to get Figure 4. Noting the logarithmic y-axis, it is clear that sighted walking is significantly better than blind walking, as one might expect. In the sighted walking case, we also found that knowing the mean helps very little, and we correspondingly only present a $\mu_s = 0$ plot. We note that *not* needing to know the long-term mean is a desirable result.
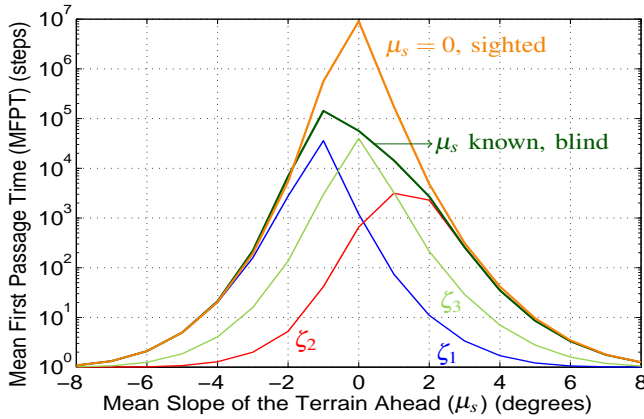


Fig. 4. Slopes ahead of the robot are assumed to be normally distributed as noted in (19) with $\sigma_s = 1.5$. Figure shows average number of steps before falling calculated using (21) versus $\mu_s$ for the mesh obtained via $d_{thr} = 0.5$. Three fixed controllers are repeated for reference. The best policy from Figure 3 (when $\mu_s$ is assumed to be known, but not the one-step lookahead) is also shown for reference. The top (orange) plot shows the optimal policy when only the state and the next slope are perfectly known.

## VI. ROBUSTNESS

Although the results of the previous section seem impressive, for this methodology to be applicable to real-life problems, the policies must also be robust to uncertainties. In this section, we describe several modifications that significantly improve several aspects of policy robustness.

### A. Noisy Slope Estimation

We start our study of robustness by considering the addition of noise to slope information. The slope ahead will still be defined by variable $s$, but the controller will think it is (closest to) $\tilde{s} \in S$, due to the noise $l \in S$. Their relationship will be given by

$$\tilde{s} = max(min(S), min(max(S), s+l)). \qquad (35)$$

The noise will be normally distributed with zero mean and standard deviation $\sigma_l$. $P_L(l)$ will be defined similarly to (18). In the presence of noise, the policy will be a function of $\tilde{s}$, not $s$.

$$\zeta[n] = \pi(y[n], \tilde{s}[n]) \qquad (36)$$

Then the approximate dynamics (15) will be

$$y[n+1] = h^a(y[n], s[n], \pi(y[n], \tilde{s}[n])) \qquad (37)$$

The stochastic state-transition matrix must also be updated to consider noise.

$$T_{ij}^s = \sum_{s \in S} \sum_{l \in S} P_S(s) \ P_L(l) \ T_{ij}^d(s, \pi(y_i, \tilde{s})) \qquad (38)$$

Figure 5 shows what happens to the policy we obtained in the previous section (orange-top plot), in the presence of $\sigma_l = 1$ noise (magenta-bottom plot). We lose almost all the improvements gained by switching! To account for noise in the slope information while optimizing, we first rewrite the modified value iteration algorithm.

$$V(i) := \sum_{\tilde{s} \in S} \max_{\zeta} \left\{ \sum_{j \neq 1} P_{ij}(\zeta, \tilde{s}) \ (1 + \alpha \ V(j)) \right\} \qquad (39)$$

The equation is essentially the same as (33), except we made clear $\tilde{s}$ is available to the controller instead of $s$. The probability of 'thinking $\tilde{s}$ is the slope ahead' and 'transitioning from $y_i$ to $y_j$ when $\zeta$ is used' is then

$$P_{ij}(\zeta, \tilde{s}) = \sum_{s \in S} \sum_{l \in S} P_S(s) P_L(l) \ f_S(s, l, \tilde{s}) \ T_{ij}^d(s, \zeta), \qquad (40)$$

where

$$f_S(s,l,\tilde{s}) = \begin{cases} 1, & \tilde{s} = max(min(S), min(max(S), s+l)) \\ 0, & \text{otherwise.} \end{cases} \qquad (41)$$

The green (middle) plot in Fig. 5 is the policy obtained and plotted by assuming $\sigma_l = 1$. Note that the new policy performs almost as well now with noisy slope information as the original policy did using noise-free data. Not surprisingly, as the noise goes down, the new policy performs better. More importantly, as the magnitude of the noise increases, we find performance does not suddenly drop. These data support our hypothesis that accounting for lookahead uncertainty is extremely important and can be done well without a precise noise model.

### B. Small Mesh Policy on Big (Refined) Mesh

Up until now, we optimized policies and plotted resulting performance using the same ($d_{thr} = 0.5$) mesh. In this section, we keep optimizing on the coarse ($d_{thr} = 0.5$) mesh, but we estimate performance using a bigger, more refined ($d_{thr} = 0.1$) mesh, intended to better approximate the true system dynamics. As presented in Table II, the small mesh has 6,126 points, while the big mesh has 394,420 points, meaning the small mesh requires significantly lower computational cost in meshing process and policy optimization. Thus, quantifying and improving robustness to mesh size are important issues.

As before, we will assume the small ($d_{thr} = 0.5$) mesh used to derive a policy is completely known. However, we will assume the larger, high-resolution mesh is not known during value iteration, thus it cannot be used while finding the policy. The larger mesh will be only used to estimate how well the small-mesh policy would work on the true system, i.e., it approximates how the policy behaves when substituted to (11).
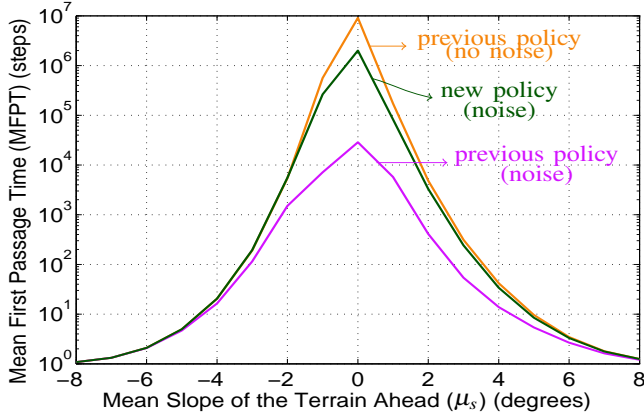
Fig. 5. Slopes ahead of the robot are assumed to be normally distributed as noted in (19) with $\sigma_s = 1.5$. Slope information experiences a noise with zero mean and standard deviation $\sigma_l = 1$. Figure shows average number of steps before falling calculated using (21) versus $\mu_s$ for the mesh obtained via $d_{thr} = 0.5$. The top plot of Figure 4 is repeated for reference. This policy performs poorly due to the noise (bottom plot). However, as the middle plot shows, we can recover the loss greatly by using (39).

First, we need to explain how the small-mesh policy can be applied when the slope ahead, $\gamma$, may not be in the slope set, and/or state $x$ may not be in the mesh. In these cases, we will apply the most basic idea: The controller will be picked assuming the slope ahead is $s \in S$ closest to $\gamma$ and the current state is $y \in Y$ closest to $x$. So the policy now is

$$\zeta[n] = \pi(c(x[n], Y), c(\tilde{\gamma}[n], S)), \quad (42)$$

where $\tilde{\gamma}[n]$ is the noisy slope ahead information, and function $c$ is as defined in (10). In its general form, let us denote the big mesh by $Y_b$, which is obtained from a slope set $S_b$. Using this mesh, we can approximate how (42) would behave in the higher fidelity mesh.

$$\zeta[n] = \pi(c(y_b[n], Y), c(\tilde{s}_b[n], S)), \quad (43)$$

The definition and calculation of $T^s$ remain the same, but this time $Y_b$ and $S_b$ will be used in obtaining it. Figure 6 shows the policy from the last section for reference (green-top plot). The magenta (bottom) plot shows what happens when this policy is applied on $Y_b$, when $Y_b$ is obtained with $S_b = S$, but $d_{thr} = 0.1$. (We consider $S_b \neq S$ later, in VI-D.) Comparing the purple and green curves in the figure we immediately note that there is a huge drop. We must once again refine our algorithm, this time to improve robustness to meshing discretization.

In our approach to fix this issue, we consider the following: While the actual state is $y_i$, the robot thinks it is $y_k$. To make this clear, we rewrite the value iteration algorithm.

$$V(k) := \sum_{\tilde{s} \in S} \max_{\zeta} \left\{ \sum_{j \neq 1} P_{kj}(\zeta, \tilde{s}) \, (1 + \alpha \, V(j)) \right\} \quad (44)$$

Note that we only exchanged $i$ with $k$, but this will make future notation easier to follow. The probability of 'thinking $\tilde{s}$ is the slope ahead', 'thinking the state is $y_k$', and 'transitioning to $y_j$ when $\zeta$ is used' is

$$P_{kj}(\zeta, \tilde{s}) = \sum_{s \in S} \sum_{l \in S} \sum_i P_S(s) P_L(l) \, f_S(s, l, \tilde{s}) \, P_{ik}^P \, T_{ij}^d(s, \zeta), \quad (45)$$

where $P_{ik}^P$ is the probability of being at state $y_i$ when robot thinks the state is $y_k$.

$$P_{ik}^P := Pr(y[n] = y_i \mid \tilde{y}[n] = y_k) \quad (46)$$

Finding the best calculation for $P_P$ is a question we want to answer in future work. However, it is intuitive that for a given state $k$ (the robot thinks the state is $y_k$), $P_{ik}^P$ should be smaller for $i$ for which $d(y_i, y_k)$ is larger. In this paper, we try the following

$$P_{ik}^P = P_C(c) / \sum_c P_C(c), \quad (47)$$

where $y_i$ is the $c^{th}$ closest state to $y_k$, and $P_C$ has the following form

$$P_C(c) = \begin{cases} \exp(-\lambda_c c), & \text{if } \exp(-\lambda_c c) > 10^{-5} \\ 0, & \text{otherwise.} \end{cases} \quad (48)$$

In this work, we use $\lambda_c = 1$ as the distribution parameter. These calculations result with the orange curve in Figure 6 (new policy).
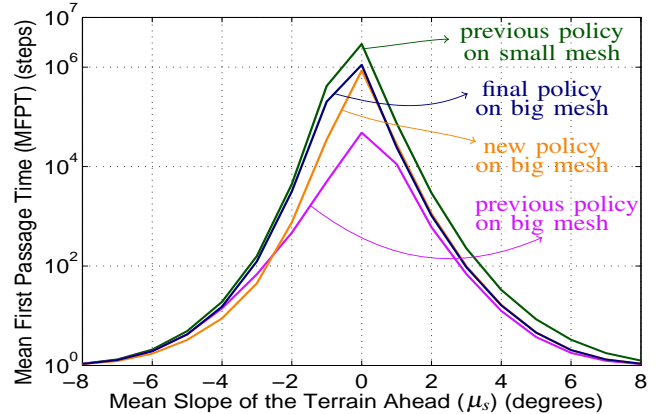


Fig. 6. Slopes ahead of the robot are assumed to be normally distributed as noted in (19) with $\sigma_s = 1.5$. Slope information experiences a noise with zero mean and standard deviation $\sigma_l = 1$. All polices are obtained using the mesh via $d_{thr} = 0.5$. However, Figure shows the results when these policies are evaluated on the $d_{thr} = 0.1$ mesh to estimate the actual performance. The green (top) plot is the middle plot of Figure 4 shown for reference. This policy performs poorly (bottom plot) on the denser mesh. However, as the middle two plot show, we can recover the loss greatly by using (44).

## C. A Final Adjustment

It is important to note that robustness to slope and state information is somewhat similar in the following sense: If the slope ahead or the current state is different from what the robot thinks, it may end up at a different point than it estimated beforehand. So in both cases, the optimization tries to account for this ending up in a different state. Since the approach in (47) is only ad hoc, we will also try increasing the robustness to slope information to see whether this actually helps when evaluating

MFPT on the big mesh. While optimizing policy, instead of a normal distribution for $P_S$ we assume a uniform distribution. Thus, we will optimize assuming the probability of having $-7°$ as the slope ahead is the same as the probability of having $2°$. This idea eliminates the need to choose both $\mu_s$ and $\sigma_s$ for optimization. We also assume sensing noise is $\sigma_l = 3$ (deg) and use $\lambda_c = 1/5$ in (48) while optimizing to increase robustness. These adjustments do improve performance, as shown by the dark blue curve in Figure 6 (final policy). We also tested the performance of this "final policy" using both smaller and larger magnitude terrain noise than the nominal $\sigma_s = 1.5$ assumption. Plots are not included, due to space limitations. However, when we test on terrain $\sigma_s = 1$ and $\sigma_s = 2$, we observe that the performance for the policy derived assuming $\sigma_s = 1.5$ still gives near-optimal stability results for the actual terrain noise present.

### D. Increasing Mesh Resolution for Slope Set, $S_b$

We finally show what happens if the slope set is different for optimization than for the system under evaluation, i.e., $S_b \neq S$. We now apply the final policy from Section VI-C on a new mesh obtained using $d_{thr} = 0.2$, and

$$S_b = \{(k/3)° \mid k \in \mathbb{Z}, -24 \leq k \leq 24\}. \tag{49}$$

This new big mesh has 226,489 points, and plotting is done with $\sigma_s = 1.5$, and $\sigma_l = 1$. Figure 7 presents the results, where final policy from Figure 6 is shown for reference. We see that the policy is quite similar when the slope set is denser, which encourages us to think that the final policy would also yield similar performance when simulating the full dynamics (Eq. 1). Monte Carlo simulations are not a computationally practical means of verifying this prediction when MFPT is very high, which has motivated our testing methodology throughout, using more refined meshing. However, a Monte Carlo study was presented in [18] for smaller number of steps and fixed controllers.

## VII. CONCLUSIONS

In this paper, we present methods for capturing the approximate dynamics of a walking model using a tractable mesh size, and we develop and test approaches for deriving robust, near-optimal switching control. To illustrate, we consider an underactuated five-link (10-state) walker with noisy one-step lookahead information. Our results quantify the intuition that even a one-step lookahead on terrain improves walking stability significantly. However, it is important to also use the current state information to set the policy. Using only slope information yields poor performance.

To estimate the number of steps before falling, we mesh the step-to-step dynamics of the system. With the meshing technique explained in this paper, we were able to do this accurately using only around six thousand points.

After getting the mesh, we use value iteration to make use of qualitatively different controllers and maximize number of steps before falling. We highlight that use of only three such controllers on rough terrain improves stability significantly,
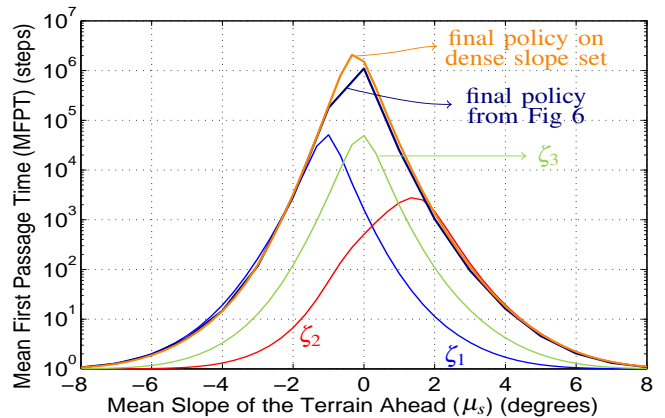


Fig. 7. Slopes ahead of the robot are assumed to be normally distributed as noted in (19) with $\sigma_s = 1.5$. Slope information experiences a noise with zero mean and standard deviation $\sigma_l = 1$. The policy was obtained using the mesh via $d_{thr} = 0.5$. However, Figure shows the results when this policy is evaluated on the mesh obtained with $d_{thr} = 0.2$ and denser slope set (as in (49)) to estimate the actual performance. The final policy from Figure 6 is shown for reference. This policy performs even better on an even denser mesh.

compared with using any one; in fact, eliminating any one of the three degrades stability significantly.

Our primary concern in this paper is to estimate how these policies would do in reality, on the true dynamic system. Thus, we focus on robustness. We were able to deal with both the effects of noise on slope information and of estimating MFPT on a big mesh using the policy obtained on a much smaller mesh. The drastic reduction in required mesh size in our work here (6,126 points, compared with over 139,000 in [19]) provides an important step toward eventually applying these methods to improve the number of steps before falling of an *actual* robot dramatically.

## VIII. APPLICABILITY AND FUTURE WORK

As illustrated in [18], the step-to-step dynamics of a planar biped with one degree of underactuation fall on a set of quasi-2D surfaces as a result of switching among a set of low-level controllers, no matter how many DOF the actual robot has. This is the key concept that allows for high accuracy with a small number of mesh points. A more detailed explanation is outside the scope of this paper but is planned for future publication. Based on this, we argue that the methods of this paper are already applicable to higher dimensional robot models. As for experimental applicability, we anticipate that a mesh obtained via simulation can be updated on the fly to improve performance, although initial performance would certainly depend on the accuracy of the hybrid model and terrain assumption.

REFERENCES

[1] Richard Bellman. A markovian decision process. *Indiana University Mathematics Journal*, 6(4):679–684, 1957. ISSN 0022-2518. URL http://www.iumj.indiana.edu/IUMJ/FULLTEXT/1957/6/56038.

[2] Pranav A. Bhounsule, Jason Cortell, and Andy Ruina. Design and control of ranger: an energy-efficient, dynamic walking robot. In *Proc. of the International Conference on Climbing and Walking Robots*, 2012. URL http://ruina.tam.cornell.edu/~pab47/CLAWAR12_Ranger_Design_Control.pdf.

[3] Katie Byl and Russ Tedrake. Metastable walking machines. *The International Journal of Robotics Research*, 28(8):1040–1064, August 2009. ISSN 0278-3649. doi: 10.1177/0278364909340446. URL http://ijr.sagepub.com/cgi/doi/10.1177/0278364909340446.

[4] Steve Collins, Andy Ruina, Russ Tedrake, and Martijn Wisse. Efficient bipedal robots based on passive-dynamic walkers. *Science*, 307(5712):1082–1085, February 2005. ISSN 0036-8075, 1095-9203. doi: 10.1126/science.1107799. URL http://www.sciencemag.org/content/307/5712/1082. PMID: 15718465.

[5] J. Maxwell Donelan, Rodger Kram, and Arthur D. Kuo. Mechanical work for step-to-step transitions is a major determinant of the metabolic cost of human walking. *Journal of Experimental Biology*, 205(23):3717–3727, December 2002. ISSN 0022-0949, 1477-9145. URL http://jeb.biologists.org/content/205/23/3717. PMID: 12409498.

[6] Kaveh Akbari Hamed and Jessy W. Grizzle. Event-based stabilization of periodic orbits for underactuated 3-d bipedal robots with left-right symmetry. *IEEE Transactions on Robotics*, 30(2):365–381, April 2014. ISSN 1552-3098. doi: 10.1109/TRO.2013.2287831. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6663683.

[7] Jessica Hodgins and Marc Raibert. Adjusting step length for rough terrain locomotion. *IEEE Transactions on Robotics and Automation*, 7(3):289–298, June 1991. ISSN 1042-296X. doi: 10.1109/70.88138. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=88138.

[8] Yildirim Hurmuzlu and Dan Marghitu. Rigid body collisions of planar kinematic chains with multiple contact points. *The International Journal of Robotics Research*, 13(1):82–92, February 1994. ISSN 0278-3649. doi: 10.1177/027836499401300106. URL http://ijr.sagepub.com/cgi/doi/10.1177/027836499401300106.

[9] Steven M. LaValle and James J. Kuffner. Randomized kinodynamic planning. *The International Journal of Robotics Research*, 20(5):378–400, May 2001. ISSN 0278-3649, 1741-3176. doi: 10.1177/02783640122067453. URL http://ijr.sagepub.com/content/20/5/378.

[10] Tad McGeer. Passive dynamic walking. *The International Journal of Robotics Research*, 9(2):62–82, April 1990. ISSN 0278-3649. doi: 10.1177/027836499000900206. URL http://ijr.sagepub.com/cgi/doi/10.1177/027836499000900206.

[11] Jun Morimoto and Christopher G. Atkeson. Learning biped locomotion. *Robotics & Automation Magazine, IEEE*, 14(2):4151, 2007. URL http://ieeexplore.ieee.org/iel5/100/4263083/04264366.pdf?arnumber=4264366.

[12] Jun Nakanishi, Jun Morimoto, Gen Endo, Gordon Cheng, Stefan Schaal, and Mitsuo Kawato. Learning from demonstration and adaptation of biped locomotion. *Robotics and Autonomous Systems*, 47:7991, 2004. URL http://www.cs.cmu.edu/~cga/papers/nakanishi-ras04.pdf.

[13] Hae-Won Park, Alireza Ramezani, and Jessy W. Grizzle. A finite-state machine for accommodating unexpected large ground-height variations in bipedal robot walking. *Robotics, IEEE Transactions on*, 29(2):331345, 2013. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6399609.

[14] Ill-Woo Park, Jung-Yup Kim, Jungho Lee, and Jun-Ho Oh. Mechanical design of humanoid robot platform KHR-3 (KAIST humanoid robot 3: HUBO). In *Proc. of International Conference on Humanoid Robots*, pages 321–326, December 2005. doi: 10.1109/ICHR.2005.1573587. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1573587.

[15] Marc Raibert, Kevin Blankespoor, Gabriel Nelson, Rob Playter, and the BigDog Team. Bigdog, the rough-terrain quadruped robot. In *Proceedings of the World Congress of the International Federation of Automatic Control*, page 1082310825, 2008. URL http://web.unair.ac.id/admin/file/f_7773_bigdog.pdf.

[16] Marc H. Raibert. *Legged Robots that Balance*. Artificial Intelligence. MIT Press, 1986. ISBN 9780262181174. URL https://mitpress.mit.edu/books/legged-robots-balance.

[17] Asif Sabanovic and Kouhei Ohnishi. Motion control systems. John Wiley & Sons, 2011. ISBN 9780470825754. URL http://onlinelibrary.wiley.com/doi/10.1002/9780470825754.fmatter/summary.

[18] Cenk Oguz Saglam and Katie Byl. Stability and gait transition of the five-link biped on stochastically rough terrain using a discrete set of sliding mode controllers. In *IEEE International Conference on Robotics and Automation (ICRA)*, pages 5675–5682, May 2013. doi: 10.1109/ICRA.2013.6631393. URL http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=6631393.

[19] Cenk Oguz Saglam and Katie Byl. Switching policies for metastable walking. In *Proc. of IEEE Conference on Decision and Control (CDC)*, pages 977–983, December 2013. doi: 10.1109/CDC.2013.6760009. URL http://ieeexplore.ieee.org/xpls/abs_all.jsp?arnumber=6760009.

[20] Cenk Oguz Saglam and Katie Byl. Quantifying the trade-offs between stability versus energy use for underactuated biped walking. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, 2014.

URL http://www.ece.ucsb.edu/~katiebyl/papers/Saglam_IROS2014_preprint.pdf. accepted for publication.

[21] Yoshiaki Sakagami, Ryujin Watanabe, Chiaki Aoyama, Shinichi Matsunaga, Nobuo Higaki, and Kikuo Fujimura. The intelligent ASIMO: system overview and integration. In *IEEE/RSJ International Conference on Intelligent Robots and Systems, 2002*, volume 3, pages 2478–2483 vol.3, 2002. doi: 10.1109/IRDS.2002. 1041641. URL http://ieeexplore.ieee.org/xpls/abs_all. jsp?arnumber=1041641.

[22] Russ Tedrake, Teresa Weirui Zhang, and Sebastian Seung. Stochastic policy gradient reinforcement learning on a simple 3D biped. In *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS).*, volume 3, pages 2849–2854 vol.3, September 2004. doi: 10.1109/IROS.2004. 1389841. URL http://ieeexplore.ieee.org/stamp/stamp. jsp?tp=&arnumber=1389841.

[23] Miomir Vukobratovic and Branislav Borovac. Zero-moment pointthirty five years of its life. *International Journal of Humanoid Robotics*, 1(01):157173, 2004. URL http://www.worldscientific.com/doi/abs/10. 1142/S0219843604000083.

[24] Eric Westervelt, Jessy W. Grizzle, and Daniel Koditschek. Hybrid zero dynamics of planar biped walkers. *IEEE Transactions on Automatic Control*, 48(1):42–56, January 2003. ISSN 0018-9286. doi: 10.1109/TAC.2002.806653. URL http://ieeexplore.ieee. org/stamp/stamp.jsp?tp=&arnumber=1166523.

[25] Eric Westervelt, Christine Chevallereau, Benjamin Morris, Jessy Grizzle, and Jun Ho Choi. *Feedback Control of Dynamic Bipedal Robot Locomotion*, volume 26 of *Automation and Control Engineering*. CRC Press, June 2007. ISBN 978-1-4200-5372-2, 978-1-4200-5373-9. URL http://www.crcnetbase.com/isbn/9781420053739.

[26] Mingyuan Zhong and Emanuel Todorov. Moving least-squares approximations for linearly-solvable stochastic optimal control problems. *Journal of Control Theory and Applications*, 9(3):451–463, August 2011. ISSN 1672-6340, 1993-0623. doi: 10. 1007/s11768-011-0275-0. URL http://link.springer.com/ article/10.1007/s11768-011-0275-0.