Quantifying the Trade-Offs Between Stability versus Energy Use for Underactuated Biped Walking

Cenk Oguz Saglam and Katie Byl

Abstract— **In this paper, we address the problem of incorporating both energy consumption and stability into a cost function for bipedal walking. To solve the problem, we also propose a basic framework and demonstrate its effectiveness in simulation. This framework allows one to use a scalar coefficient to adjust the trade-off between stability and energy use. The optimal scalar value depends on the robot, terrain, task and priorities. In order to implement the methods in this paper, multiple low-level walking controllers and meshing of a ten-dimensional state space are needed. This latter requirement would normally be impractical for a 10D system; however, we exploit the observation that our low-level controllers cause the step-to-step dynamics to fill only a small, quasi-2D region, thus enabling meshing and, correspondingly, dynamic programming based on the resulting Markov Decision Process (MDP). Both the introduction of the energy/stability trade-off problem and our proposed framework for its solution have potential for significant utility in the future, as robot locomotion is developed to operate in increasingly less structured (stochastic) environments.**

I. INTRODUCTION

For legged robots to walk untethered and on real-world terrain, both energy use and stability are critical concerns. Although these are not necessarily opposing goals, achieving an effective balance between them is an open challenge, which we address in this work.

One approach to address energetic efficiency is to base a robot design on passive dynamic walking [1]. Powered walkers designed to exploit the stable limit cycles of passive walkers have in fact been very energy efficient. Arguably the best example is the Cornell Ranger [2], which successfully walked a record-breaking 65.2 kilometres on a flat course on battery power. However, such designs share the fragile stability problems that passive walkers have on rough terrain, and they are not as capable as we would like walkers to be.

Control approaches emphasizing stability often focus on planning of the Zero Moment Point (ZMP). ZMP-based algorithms aim to avoid underactuation at all times as a means of guaranteeing stability, but this occurs at the expense of high energy consumption. The most famous example would be Honda's Asimo [3].

In this paper, we will be measuring energetics with cost of transport (COT), which is the non-dimensionalized energy expenditure per unit weight and unit distance [4]. The lower COT is, the more energetically efficient a robot is during walking. For example while Cornell Ranger has COT around 0.2, COT of Asimo is estimated to be 3.2 [5]. However, it is obvious that Asimo is much more stable and capable compared to Cornell Ranger. To quantify stability in this paper, we will be using metastability analysis in which Mean First Passage Time (MFPT) quantifies the average number of steps before falling [6]. A higher MFPT therefore corresponds to more stable walking. We will also estimate the average COT using the mentioned metastability analysis, along with calculations of work-per-step for each particular state and action combinations possible.

A debate about whether energetics or stability comes first seems inevitable. We argue that while both are very important, the relative weighting of each will ultimately depend on the specific robot, terrain, task and personal risk aversion of the operator. Our goals are to provide a method for adjusting the relative importance of stability, energetics (and potentially other metrics) in a smooth and tunable way and to quantify resulting risk and energy. We expect future robots to be capable of walking in energy-saving and performance modes, just like our laptops. In this paper, we introduce a new framework towards achieving that.

To apply the methods of this paper, the robot does not need exact terrain information. For example, it may be walking blindly with only an estimate of terrain variability. However, since this paper is based on our previous work [7], where we assumed robot knows the slope ahead for one step, we will assume the same here also.

II. MATHEMATICAL MODEL

Figure 1 illustrates a five-link planar biped that is based on RABBIT [8] and used in this paper. The same figure defines what femur, tip, stance leg and swing leg stand for. The relative angles are given by $q := [q_1 \ q_2 \ q_3 \ q_4 \ q_5]^T$, and the 10 dimensional state is defined as $x = [q^T \ \dot{q}^T]^T$. Note that the biped has point feet, and actuations are only present at four internal joints while the ground contact at the point foot is a passive pivot. As a result, this biped is underactuated.

Walking consists of steps in sequence. Each step consists of a single and a double support phase. The single support phase is when only the stance leg is contact with the ground. This phase is expressed in the following form:

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

C.O. Saglam and K. Byl are with the Electrical and Computer Engineering Department, University of California, Santa Barbara, CA 93106 USA saglam at ece.ucsb.edu, katiebyl at ece.ucsb.edu

This work was supported by the Institute for Collaborative Biotechnologies through grant W911NF-09-0001 from the U.S. Army Research Office. The content of the information does not necessarily reflect the position or the policy of the Government, and no official endorsement should be inferred.

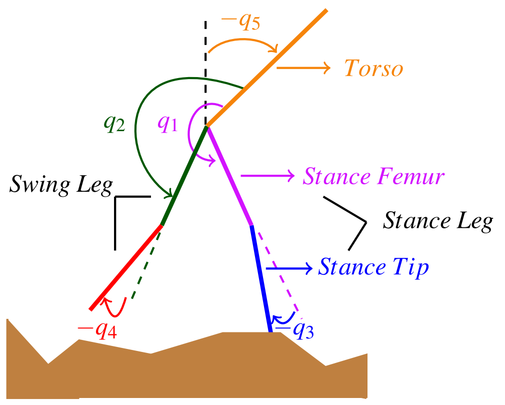

Fig. 1. Illustration of the five-link biped

where u is the input, and matrices D, C, G, B are obtained by the Lagrangian approach. (1) can be equivalently expressed by

$$\dot{x} = \begin{bmatrix} \dot{q} \\ D^{-1}(-C\dot{q} - G + Bu) \end{bmatrix} =: f(x,u). \tag{2}$$

We will model the double support phase as an instantaneous impact event, denoted by g.

$$x^+ = g(x^-) \tag{3}$$

The impact event is nothing but a mapping from the state just before the impact, x^-, to the state just after the impact, x^+. The exact equation is derived using conservation of energy and the principle of virtual work [8], [9].

For a step to be successful, certain “validity conditions” must be satisfied, which are as follows. After impact, the former stance leg must lift from ground with no further interaction with the ground until the next impact. Also, the swing foot must have progressed past the stance foot before the impact of the next step occurs. Only the feet should contact the ground. Furthermore, the force on stance tip during the swing phase, and the force on the swing tip at the impact should satisfy

$$F_{friction} = F_{normal}\ \mu_s > |F_{transversal}|. \tag{4}$$

If validity conditions are not met, the step is unsuccessful and the system is modeled as transitioning to an absorbing failure state. This is a conservative model because in reality violating these conditions does not necessarily mean failure. Since walking consists of continuous and discontinuous parts, it has hybrid dynamics as explained in Fig. 2.

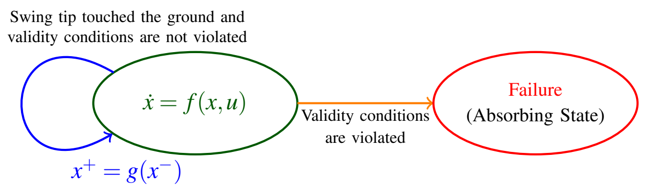
Swing tip touched the ground and validity conditions are not violated
$\dot{x} = f(x,u)$
$x^+ = g(x^-)$
Validity conditions are violated
Failure (Absorbing State)

Fig. 2. Hybrid model of a step and the failure state

III. CONTROL

Since the robot is underactuated, we cannot control all five angles at the same time. Instead, we need to chose four variables to be controlled. One choice, which is a result of our experience [7], is as follows.

$$q_c := [q_2 + q_5 \quad q_3 \quad q_4 \quad q_5]^T, \tag{5}$$

where q_c stands for angles to be controlled. Our methods in the following sections will be general and are designed for use with practically any preferred choice of low-level control structure to regulate q_c. In this particular paper, we adopt the Sliding Mode Control (SMC) scheme for finite time convergence [10] as summarized in the Appendix.

As mentioned in the introduction, our method requires multiple controllers to be available. Increasing their quantity would smooth the transitions, but would also cause higher computational cost. We start with six controllers, having in mind that we can increase the number if needed. We will use different piecewise constant and time-invariant references to get these six qualitatively different controllers. So, the only difference between the controllers will be their reference trajectories, namely q_c^{ref}. Let ζ_i represent the i^{th} controller. Then, the set of available controllers will be defined as

$$Z := \{\zeta_1,\ \zeta_2,\ \zeta_3,\ \zeta_4,\ \zeta_5,\ \zeta_6\}. \tag{6}$$

Figure 3 illustrates the steady-state walking gaits on flat terrain for ζ_1 and ζ_6. As can be verified from the figure, ζ_1 results with wider step width and larger ground clearance compared to ζ_6. Subfigures on the right column also show that it takes longer to complete one step using ζ_6. It will also become clear in later sections that ζ_1 is more stable, but it also has a larger COT, i.e., is less energy efficient. Trajectories of ζ_1 and ζ_6 were actually synthesized by design to be this way using the structure explained in [7] after a very short trial and error process. The reference trajectories of $\{\zeta_2,\ \zeta_3,\ \zeta_4,\ \zeta_5\}$ are linear (i.e., interpolated) combinations of q_c^{ref} for ζ_1 and ζ_6.

IV. MESH

A. Poincaré Map

Next, we build a mesh based on Poincaré sections at impacts. Specifically, for the rest of the paper, we will be interested in the states just before the impact, i.e., x^- states. We will refer to those simply as states, with notation x. So, x will be a 10 dimensional discrete variable capturing the state of the robot at each step. When x is seen in this way, the next state of the robot ($x[n+1]$) is a function of current state ($x[n]$), the terrain profile ahead ($\overline{\gamma}[n]$), and the controller ($\zeta[n]$).

$$x[n+1] = h(x[n], \overline{\gamma}[n], \zeta[n]) \tag{7}$$

Note that while angles need to satisfy $q \in [0, 2\pi)$, the velocities might be any real value, i.e., $\dot{q} \in \mathbb{R}$. However, the dynamics of (7) will not cover all this space. The controller set and terrain set force the robot to behave in a certain way, so there is a limitation to what $x[n+1]$ might be. On the other hand, mainly due to underactuation and torque limitations,

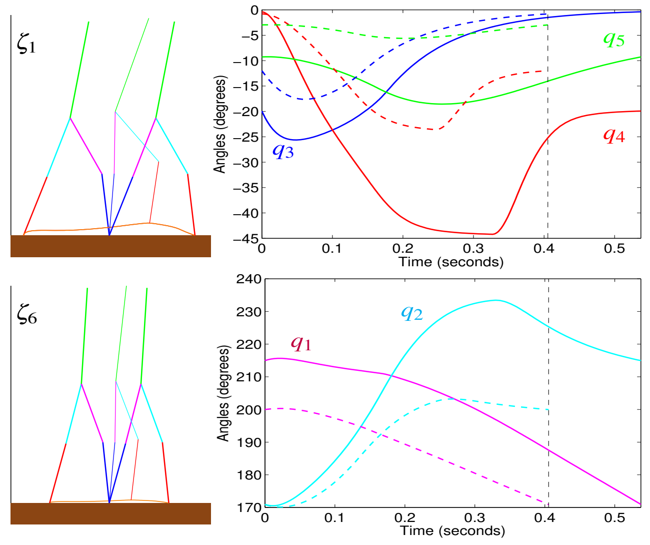

Fig. 3. Left column shows the steady-state walking gaits on flat terrain for ζ_1 and ζ_6. Two sub-figures on the right depict angles over time for those gaits. On the right column, ζ_1 is plotted with solid lines and ζ_6 is given with dashed lines. Dashed vertical line at $t = 0.405s$ indicates the end of step for ζ_6

$x[n]$s that are not destined to fall are also limited to a subset of $[0,2\pi)^5 \text{x} \mathbb{R}^5$. Starting from some initial set of states, meshing explores the reachable subset of state space for the robot. In other words, given an initial state (or a set of states), meshing captures the manifold which includes all the states the robot may visit over time, no matter how many steps it takes. Then the robot will never exit this subset if it is initialized in or ever entered the subset in question. The goal of the meshing is to obtain this subset and understand how the robot moves within it. The results of this paper verify our previous work which shows that our mesh points lie very close to a 2-dimensional manifold within the full 10D state space. [11].

B. Distance Metric

To scale the resolution of our mesh, we adopt a distance metric. Our experience shows that using the standardized (normalized) Euclidean distance is extremely helpful. Let a be a vector, and B be a set of vectors of the same dimension. In our case set B will grow in size while meshing. The distance of a from B is given by

$$d(a,B) := \min_{b \in B} \left\{ \sqrt{\sum_i \left(\frac{a_i - b_i}{r_i} \right)^2} \right\}, \tag{8}$$

where r_i is the standard deviation of b_i elements, which changes as new vectors are added to B. We also calculate the closest point in set B to point a as

$$c(a,B) := \operatorname*{argmin}_{b \in B} \left\{ \sum_i \left(\frac{a_i - b_i}{r_i} \right)^2 \right\}. \tag{9}$$

C. Terrain Profile

To carry out the calculations of the following sections, we need to express the possibilities of the terrain ahead for each step. Our choice was to assume that terrain consists of slopes, which can only change at impacts. In other words, we assume the terrain ahead is a constant slope for each step. With this assumption, (7) becomes

$$x[n+1] = h^t(x[n], \gamma[n], \zeta[n]), \tag{10}$$

where $\gamma[n]$ is the slope for step n, and superscript t on function h denotes the terrain assumption.

Unless stated otherwise, all the angles of the following sections will be in degrees.

D. Methodology

We aim to capture the dynamics of the robot written in (10). To do this, we mesh the state space robot walks in. As mentioned at the end of Section III, controller set Z has 6 controllers. We also adopt a slope set S, which is the integer values from -8 to 8.

$$S = \{k^\circ \ \mid \ k \in \mathbb{Z}, \ -8 \le k \le 8\} \tag{11}$$

Before starting the meshing process, we also need an initial mesh (set of states) Y_i. For this study, we used a Y_i with two states only. y_1 represents all the failure states. y_2 is the stable fixed point the robot state converged to when the ground was flat, and only ζ_1 was used. Different initial meshes are also possible, but they will result in similar final meshes. Algorithm 1 outlines the meshing process, which can output arbitrary accuracy depending on the inputs to the algorithm (e.g., smaller d_{thr} means higher accuracy). The meshing aims to cover the (reachable) part of the state space the robot visits. d_{thr} will determine how coarse or sparse sampling is in this region. The goal is to represent infinitely many points in the region with a finite number of states.

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, state-transition map, and information map

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

E. Deterministic State-Transition Matrix

The state transition map obtained with the Algorithm 1 gives us $h^t(y,s,\zeta)$ for all $y \in Y$, $s \in S$, and $\zeta \in Z$. We then define h^a function as

$$h^a(x[n],\gamma[n],\zeta[n]) := c(h^t(x[n],\gamma[n],\zeta[n]),Y). \tag{12}$$

We use the superscript a over h because the approximate dynamics of the walking is expressed by

$$y[n+1] = h^a(y[n],s[n],\zeta[n]). \tag{13}$$

In this case, the deterministic state-transition matrix is given by

$$T^d_{ij}(s,\zeta) = \begin{cases} 1, & \text{if } y_j = h^a(y_i,s,\zeta) \\ 0, & otherwise. \end{cases} \tag{14}$$

F. Stochastic State-Transition Matrix

Stochastic state-transition matrix is defined by

$$T^s_{ij} := Pr(y[n+1] = y_j \mid y[n] = y_i). \tag{15}$$

To calculate T^s we need a controller policy and a probability distribution for the slope ahead. Policy, noted by π, is what decides which controller to use at each step. We consider policies that are functions of the current state and slope ahead for one-step.

$$\zeta[n] = \pi(y[n],s[n]) \tag{16}$$

When such a policy is applied, (13) becomes

$$y[n+1] = h^a(y[n],s[n],\pi(y[n],s[n])). \tag{17}$$

The probability of having $s \in S$ as the slope ahead is defined as

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

In this paper we use a Gaussian distribution. Each slope will be normally distributed with mean μ_s and variance σ_s^2.

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

Then, the stochastic transition matrix defined in (15) can be calculated by the following summation over the discrete set of possible slopes.

$$T^s_{ij} = \sum_{s\in S} P_S(s)\; T^d_{ij}(s,\pi(y_i,s)) \tag{20}$$

G. Mean First Passage Time (MFPT)

In this subsection we will provide a brief and necessary summary of metastability analysis of walking. The details and proofs can be found in [6].

Metastable distribution is defined as the steady state distribution among states, given the robot did not fall yet, but has forgotten its initial conditions.

$$\phi_i := \lim_{n\to\infty} Pr(y[n] = y_i \mid y[n] \neq y_1) \tag{21}$$

So ϕ_i gives the probability of being at state y_i, given the robot took infinitely many steps, but did not fall yet. In practice, the systems we study rapidly forget initial conditions after just a few initial steps. Mean First Passage Time vector m is defined to satisfy the following equation

$$m_i = \begin{cases} 0, & \text{if } i = 1 \\ 1 + \sum_j T^s_{ij}\; m_j & \text{otherwise} \end{cases} \tag{22}$$

This equation says that given that the robot did not fall yet, the number of steps robot is going to take is one less after a step is taken. After calculating the Mean First Passage Time, the system-wide MFPT is simply

$$M = \sum_i m_i\; \phi_i. \tag{23}$$

The higher M is, the more stable a robot is said to be. The largest eigenvalue of the stochastic state-transition matrix T^s is equal to one, because we assumed the failure state to be absorbing. Let λ_2 be the second largest eigenvalue. To approximate the MFPT for easy calculation, we will note that the probability of taking a step successfully, i.e. not falling, given the robot haven't fallen yet is λ_2 [6]. Then, the probability of falling at a step is $1-\lambda_2$. As a result, the probability of taking n steps only is simply

$$Pr(y[n+1] = y_1,\; y[n] \neq y_1) = \lambda_2^{n-1}(1-\lambda_2). \tag{24}$$

The equation above can be verified considering falling at the first step (taking 1 step). When $n = 1$ is substituted we get $1-\lambda_2$ as expected. Then, the average number of steps can be then calculated as

$$M = \sum_{n=1}^{\infty} n\; Pr(y[n+1] = y_1,\; y[n] \neq y_1) = \frac{1}{1-\lambda_2}, \tag{25}$$

where we used the fact that $\lambda_2 < 1$.

H. Cost of Transport (COT)

Cost of Transport (COT) is defined as the total mechanical energy spent, divided by weight times distance.

$$COT = \frac{W}{mgd}, \tag{26}$$

where m is the mass, g is the gravitational constant, and d is the distance traveled. In this paper we will use the conservative definition of "energy spent" by regarding negative work is also done by the robot, i.e.

$$W = |W_{positive}| + |W_{negative}| = W_{positive} - W_{negative} \tag{27}$$

so that both acceleration and breaking require power, but there is no regenerative breaking. Remember that we stored the step width and energy information while meshing with Algorithm 1. We define $\delta(i,s,\zeta)$ and $w(i,s,\zeta)$ to be the step width and energy spent when a step is taken from state y_i using ζ and the slope was s. Then the COT associated with that step is simply

$$COT(i,s,\zeta) = \frac{w(i,s,\zeta)}{mg\delta(i,s,\zeta)} \tag{28}$$

V. VALUE ITERATION

With the stochastic transition matrix available, we cast the problem as a Markov Decision Process (MDP). To come up with policies, we use value iteration [12]. We slightly modify the common value iteration algorithm to come up with policies that are both functions of state and one-step lookahead (slope information for the upcoming one step), instead of just the first one. The value of state y_i is noted with $V(i)$ and defined as

$$V(i) := \sum_{s \in S} P_S(s) \max_{\zeta} \left\{ \sum_j P_{ij}(s,\zeta)\ (R_{ij}(s,\zeta) + \alpha\ V(j)) \right\}, \tag{29}$$

where s is the slope ahead and α is the discount factor, which needs to satisfy $0 \le \alpha < 1$. In this work, α is chosen to be 0.9. $P_{ij}(s,\zeta)$ is the probability of transitioning from y_i to y_j when ζ is used and slope ahead is s. Finally, $R_{ij}(s,\zeta)$ is the reward for using controller ζ when current state is y_i and s is the slope ahead.

Remember that y_1 represents the failure state. Its value will be initialized as zero, and it will always be zero due to the structure of the reward function.

$$V(1) = 0, \tag{30}$$

Values of the other states are initialized as 1 and (29) is iterated until values converge. Note that in this setting, the probability function is (essentially) nothing but the deterministic state-transition matrix,

$$P_{ij}(s,\zeta) = T^d_{ij}(s,\zeta) \tag{31}$$

for cases in which mesh state j is the exact state transitioned to and not simply the nearest (approximate) state in the mesh. For cases where (31) is not the case, [13] provides an algorithm to distribute probability across a set of nearest neighbors, for greater robustness.

In this paper, we propose a reward function that accounts for both stability and energetics with an adjustable trade-off.

$$R_{ij}(s,\zeta) = \begin{cases} 0, & j = 1 \\ 1 - \beta\ COT(i,s,\zeta), & \text{otherwise} \end{cases} \tag{32}$$

where $\beta \ge 0$ is the trade-off parameter. In this setting, the reward function we used in our previous work is just a special case with $\beta = 0$ [7]. When $\beta = 0$, the reward of taking a successful step is one, falling is zero. This tells the optimization structure not to account for energy usage at all, rewarding only stability. Since $\beta \ge 0$, a small β ensures the reward is always non-negative, i.e., $R_{ij}(s,\zeta) \ge 0$. Note that COT is always positive.

VI. THE OPTIMIZATION STRUCTURE

A. Average Statistics

Consider a policy π is decided. For non-fallen states, we calculate the average distance vector $\overline{\delta}$ by

$$\overline{\delta}_i = \sum_s \delta(i,s,\pi(y_i,s))\ Pr(s \mid h^a(y_i,s,\pi(y_i,s)) \ne y_1), \tag{33}$$

Note that $\overline{\delta}_i$ is the expected step width when a successful step is taken from y_i. Then, the system-wide average step width $\overline{\Delta}$ is

$$\overline{\Delta} = \sum_i \overline{\delta}_i \phi_i, \tag{34}$$

where we used metastable distribution ϕ_i defined in (21). Similarly, we define energy vector $\overline{w}$

$$\overline{w}_i = \sum_s w(i,s,\pi(y_i,s))\ Pr(s \mid h^a(y_i,s,\pi(y_i,s)) \ne y_1), \tag{35}$$

and system-wide average energy $\overline{W}$

$$\overline{W} = \sum_i \overline{w}_i \phi_i. \tag{36}$$

Once we have $\overline{\Delta}$ and $\overline{W}$, average cost of transport ($\overline{COT}$) can be easily calculated.

$$\overline{COT} = \frac{\overline{W}}{mg\overline{\Delta}} \tag{37}$$

As we mentioned in subsection IV-G, the average probability of taking a step successfully (i.e. not falling) is λ_2, the second largest eigenvalue of the stochastic state-transition matrix T^s.

B. Total Statistics

Let d_t and Δ_t denote the total distance to be traveled and expected total distance to be traveled respectively. Then, steps needed, noted by n_t, is simply

$$n_t = \left\lceil \frac{d_t}{\overline{\Delta}} \right\rceil, \tag{38}$$

where the ceiling function is used to guarantee n_t is an integer and $\Delta_t = n_t\overline{\Delta} \ge d_t$. As a result, expected total energy to be spent is

$$W_t = n_t \overline{W}. \tag{39}$$

The probability of traveling d_t (without falling) is

$$p_t := Pr(\Delta_t \ge d_t) = Pr(y[n_t] \ne y_1) = p^{n_t}, \tag{40}$$

where $p = \lambda_2$ is the average probability of taking a step successfully.

VII. RESULTS

The mesh used for this paper is obtained with $d_{thr} = 0.5$ in Algorithm 1, which resulted in 81,582 mesh points. For the slope probability distribution in (19), we assume $\mu_s = 0$, and $\sigma_s = 1.5$. Figure 4 shows a plot where the x-axis is the non-dimensional cost of transport and y-axis is the Mean First Passage Time. The figure includes both fixed controller performance and results for policies obtained by the value iteration given in (29) with different β values.

We immediately notice that energy efficiency rises as we move from ζ_1 to ζ_6. However, as we go from ζ_1 to ζ_6, the MFPT decreases (stability drops). So, for the fixed controllers we have, there is a trade-off between the stability and energy consumption. This is expected because, for example as shown in Figure 3, ζ_1 has a higher ground

TABLE I

CONTROLLER STATISTICS

				Average (for a step)		Total ($d_t = 10km$)		
	Controller Policy	$\overline{COT}$	MFPT	$\overline{\Delta}$: Distance (m)	$\overline{W}$: Energy (J)	Steps (n_t)	Probability (p_t)	W_t: Energy (J)
Fixed Controllers ($\pi = \zeta_i$)	ζ_1	0.6387	4.3e5	0.4703	94.29	2.12e4	95.24%	2e6
	ζ_2	0.5972	5.5e4	0.4438	83.20	2.25e4	66.86%	1.87e6
	ζ_3	0.5580	1.1e3	0.4162	72.90	2.40e4	7.9e-8%	1.75e6
	ζ_4	0.5206	96.7	0.3820	62.42	2.61e4	1e-115%	1.63e6
	ζ_5	0.4864	25.4	0.3378	51.57	2.96e4	0%	1.52e6
	ζ_6	0.4527	11.7	0.2916	41.44	3.42e4	0%	1.42e6
Policies obtained by value iteration given in (29) with different β values	$\beta = 0$	0.5834	1.1e7	0.4209	77.09	2.37e4	99.79%	1.83e6
	$\beta = 1e-6$	0.5625	1.1e7	0.4056	71.63	2.46e4	99.78%	1.76e6
	$\beta = 1e-5$	0.5242	8.3e6	0.3753	61.771	2.66e4	99.67%	1.64e6
	$\beta = 1e-4$	0.4870	3.2e6	0.3512	53.70	2.84e4	99.13%	1.52e6
	$\beta = 1e-3$	0.4696	8.4e5	0.3404	50.19	2.93e4	96.58%	1.47e6
	$\beta = 1e-2$	0.4594	2.2e5	0.3348	48.28	2.98e4	87.81%	1.44e6
	$\beta = 1e-1$	0.4544	5.1e4	0.3320	47.37	3.01e4	55.88%	1.42e6
	$\beta = 1$	0.4529	1e4	0.3309	47.04	3.02e4	5.39%	1.42e6

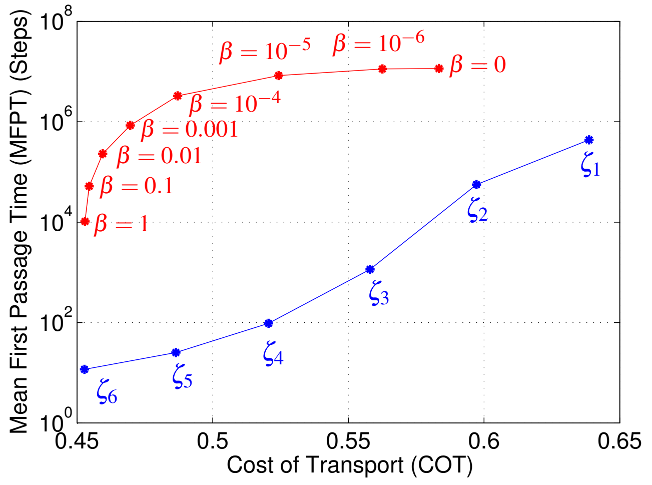

Fig. 4. Cost of Transport (COT) versus Mean First Passage Time (MFPT). Slopes ahead of the robot are assumed to be normally distributed as noted in (19) with $\mu_s = 0$, and $\sigma_s = 1.5$. The blue (bottom) line shows the performance of fixed controllers while the red (upper) line represents the results for various β values in (32).

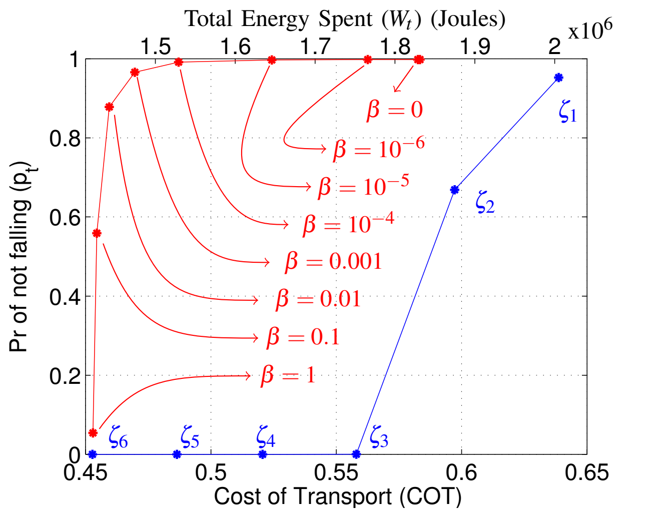

Fig. 5. Total Energy Consumption (W_t) (top x-axis) versus probability of completing d_t without falling (p_t). Slopes ahead of the robot are assumed to be normally distributed as noted in (19) with $\mu_s = 0$, and $\sigma_s = 1.5$. The blue (bottom) line shows the performance of fixed controllers while the red (upper) line represents the results for various β values in (32).

clearance which consumes more energy while potentially increasing the stability.

In Figure 4, we desire to move towards the upper left corner, because as we move up stability increases and as we move left energy consumption decreases. It is clear that switching between the controllers is better compared to using any one of the controllers alone, because we may gain increase in both stability and efficiency. However, it is not obvious which β to use in this figure. To discuss how to decide this, we will consider the case when $d_t = 10km$. Table I lists the statistics. To visualize, we present Figure 5, where top x-axis is the total energy spent and y-axis is the probability of walking 10 kilometers without falling.

Figure 5 reveals some facts: A tiny value of β (compared to a larger β) doesn't increase stability much, but causes inefficiency. When β is too big (compared to smaller βs), stability dramatically decreases while not providing any COT reduction. For $d_t = 10km$, $10^{-4} \leq \beta \leq 10^{-3}$ seems reasonable, which punishes "unnecessarily" energy consuming controller choices by reducing the reward. What we mean by "unnecessary" is that if the energy-saving controllers do as good as the most stable controller, then there is no need to go with the latter one.

Assuming 66% chance of achieving the goal distance is not enough, the only fixed controller that can be used is ζ_1. When $\beta = 10^{-3}$ is used, we are able to increase stability while saving more than 25% energy. If the distance and the battery capacity is known (e.g., 1.5 megajoules), we may pick β which maximizes the stability while having enough energy to finish the course using Figure 5.

Next, we present the cases when we change $\sigma_s = 1.5$ and $d_t = 10km$. Figure 6 depicts the cases when $d_t = 10km$ is fixed but $\sigma_s \in \{1.3, 1.5, 1.7\}$. Performance for both the fixed controllers and the policies drops as σ_s increases. This is expected since a higher σ_s means a rougher terrain, and thus a more difficult task. With a larger σ_s, policies seem to consume slightly more energy while the energy consumption of the fixed controllers almost stays constant. What is crucial is that this figure shows that the optimal choice of β depends on σ_s. When the terrain is not very rough, we would prefer a large β to decrease energy consumption, otherwise we would pick a small β to increase stability.

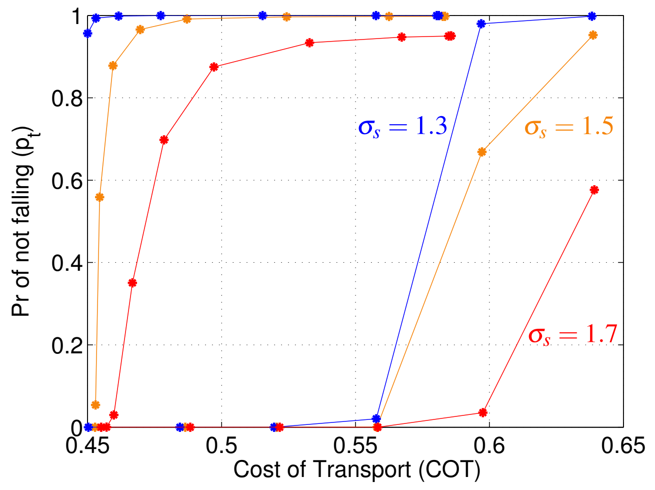

Fig. 6. Cost of Transport (COT) versus probability of completing $d_t = 10km$ without falling (p_t). Slopes ahead of the robot are assumed to be normally distributed as noted in (19) with $\mu_s = 0$, and $\sigma_s \in \{1.3, 1.5, 1.7\}$. Points are as labeled in Figure 5

Similarly, Figure 7 is obtained with $\sigma_s = 1.5$ and $d_t \in \{1, 10, 100\}km$. Obviously a larger d_t means a more difficult task, since the number of steps is proportional to the distance. As a result, the stability drops as the distance increases. In this sense note the qualitative similarity between the curves on Figures 6 and 7. Not surprisingly, the energy consumption per km stays the same for a fixed σ_s.

The last two figures motivates adjusting β on the fly to adopt to changing terrain roughness, remaining battery or updates in the task. We will illustrate the first one. Consider a terrain where μ_s and σ_s changes over distance as shown in Figure 8.

When the robot walks across such a terrain, the probability of completing the full course without falling versus the energy expected to be spent is as given in Figure 9. Apart from the previous plots, we also have a green star, which is obtained by adjusting β on the fly. The kth β value of the 9 cases shown in Figure 8 is selected by using

$$\beta_k = \underset{\beta \in B}{\operatorname{argmin}} \left\{ COT_k^2(\beta) + \alpha (1 - p_k(\beta))^2 \right\},$$
$$B = \{0, 10^{-6}, 10^{-5}, 10^{-4}, 10^{-3}, 10^{-2}, 10^{-1}, 1\}, \qquad (41)$$

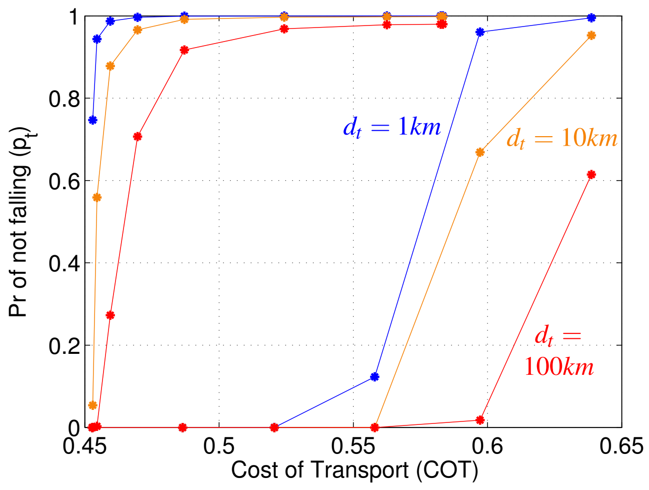

Fig. 7. Cost of Transport (COT) versus probability of completing $d_t \in \{1, 10, 100\}km$ without falling (p_t). Slopes ahead of the robot are assumed to be normally distributed as noted in (19) with $\mu_s = 0$, and $\sigma_s = 1.5$. Points are as labeled in Figure 5

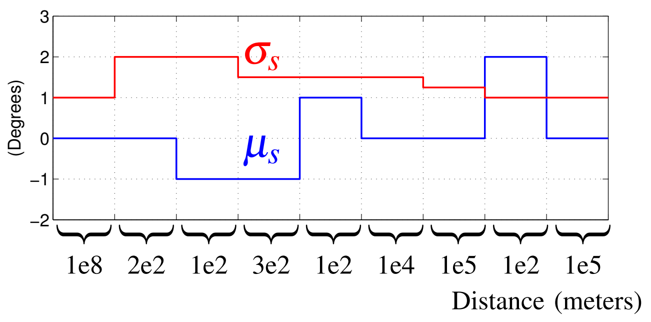

Fig. 8. Changing μ_s and σ_s over distance. Slopes ahead of the robot are assumed to be normally distributed as noted in (19) with μ_s, and σ_s. The x-axis is not in scale.

where $COT_k(\beta)$ and $p_k(\beta)$ are the COT and probability of achieving the kth case. They both depend on the policy which is tuned by the β parameter in (32). α is the trade-off parameter, which picks the appropriate β for each k. If α was zero, then we would have $\beta_k = 1$, and if α was infinity, then we would have $\beta_k = 0$. Equation (41) can be seen as minimizing a weighted distance from $(COT_k, p_k) = (0, 1)$, i.e., the unrealistically ideal (stable and passive) walking. $\alpha = 10^7$ is chosen by trial and error. Figure 9 shows the advantages of using such an adjustment mechanism. Compared to ζ_1, the green dot increases stability by 129% while decreasing energy consumption by 29%.

VIII. SCALABILITY

In this study, we accounted for energetics and stability only. However, considering other performance metrics is straightforward: simply update (32) with additional trade-off parameters.

Finally, [11] illustrates that one DOF underactuated planar biped dynamics reduce to a set of quasi-2D surfaces. We have been developing theory to explain this phenomenon more scientifically, which encourages us to believe that

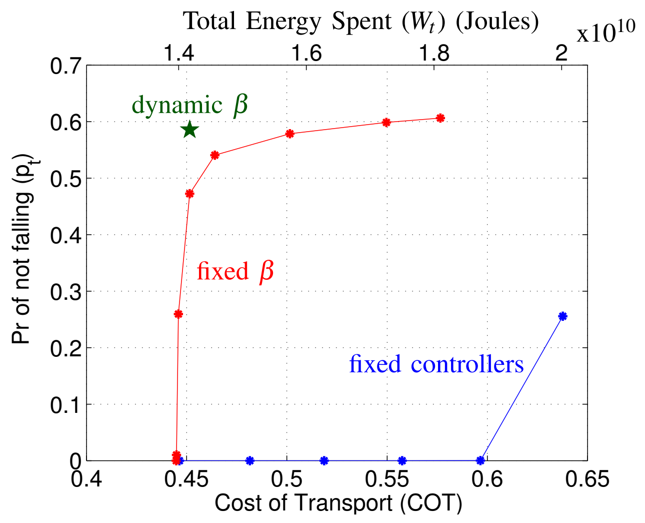

Fig. 9. Total Energy Consumption (W_t) (top x-axis) versus probability of completing d_t without falling (p_t). Slopes ahead of the robot are assumed to be normally distributed as noted in (19) with the terrain depicted in Figure 8. Green dot represents adjusting β on the fly using (41). Other points are as labeled in Figure 5.

the methods of this paper are already applicable to higher dimensional robots.

APPENDIX

In this study, we used the following control scheme. Once q_c is chosen as in (5), we pick u as

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

$$where\ E = \begin{bmatrix} 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix}. \qquad (42)$$

When substituted into (1), this control gives

$$\ddot{q}_c = v. \qquad (43)$$

What is left to do is choosing v such that q_c acts as desired. This can be achieved using a PID controller for example. However, for finite time convergence, we adopted Sliding Mode Control (SMC) [10]. We define error as

$$e = q_c - q_c^{ref}, \qquad (44)$$

and define the generalized error as

$$\sigma_i = \dot{e}_i + e_i/\tau_i, \quad i = \{1,2,3,4\}, \qquad (45)$$

where τ_is are time constants for each dimension of q_c. Note that when the generalized error is driven to zero, i.e. $\sigma_i = 0$, we have

$$0 = \dot{e}_i + e_i/\tau_i. \qquad (46)$$

The solution to this equation is given by

$$e_i(t) = e_i(t_0)\ exp(-(t - t_0)/\tau_i), \qquad (47)$$

which drives q_c to q_c^{ref} exponentially fast and justifies the name time constant. It can be also seen as the ratio of proportional and derivative gains of a PD controller.

Finally, v in (43) is given by

$$v_i = -k_i|\sigma_i|^{2\alpha_i - 1} sign(\sigma_i), \quad i = \{1,2,3,4\}, \qquad (48)$$

where $k_i > 0$ and $0.5 < \alpha_i < 1$ are called convergence coefficient and convergence exponent respectively. k_i can be seen as the gain common to both e_i and $\dot{e}_i$. $0.5 < \alpha_i < 1$ ensures finite time convergence. Note that if we had $\alpha_i = 1$, this would be a PD controller.

The controller parameters obtained after a short trial and error process are given in Table II.

TABLE II
CONTROLLER PARAMETERS

α_1=0.7,	α_2=0.7,	α_3=0.7,	α_4=0.7
τ_1=1/10,	τ_2=1/10,	τ_3=1/20,	τ_4=1/5
k_1=50,	k_2=100,	k_3=75,	k_4=10

Remember that the same controller parameters were used for all controllers in this paper, but q_c^{ref} was different for each controller.

REFERENCES

[1] T. McGeer, "Passive dynamic walking," *Internation Journal of Robotics Research*, vol. 9, no. 2, pp. 62–82, 1990.
[2] P. Bhounsule, J. Cortell, and A. Ruina, "Design and control of Ranger: an energy-efficient, dynamic walking robot," in *Proc. of the International Conference on Climbing and Walking*, pp. 441–448, 2012.
[3] Y. Sakagami, R. Watanabe, C. Aoyama, S. Matsunaga, N. Higaki, and K. Fujimura, "The intelligent ASIMO: system overview and integration," in *IEEE/RSJ International Conference on Intelligent Robots and Systems*, vol. 3, pp. 2478–2483, 2002.
[4] V. A. Tucker, "The energetic cost of moving about: Walking and running are extremely inefficient forms of locomotion. much greater efficiency is achieved by birds, fish-and bicyclists," *American Scientist*, vol. 63, no. 4, pp. 413–419, 1975.
[5] S. Collins, A. Ruina, R. Tedrake, and M. Wisse, "Efficient bipedal robots based on passive-dynamic walkers," *Science*, vol. 307, no. 5712, pp. 1082–1085, 2005.
[6] K. Byl and R. Tedrake, "Metastable walking machines," *Int. J. of Robotics Research*, vol. 28, pp. 1040–1064, Aug 2009.
[7] C. O. Saglam and K. Byl, "Switching policies for metastable walking," in *Proc. of IEEE Int. Conf. on Decision and Control (CDC)*, 2013.
[8] E. R. Westervelt, J. W. Grizzle, C. Chevallereau, J. H. Choi, and B. Morris, *Feedback Control of Dynamic Bipedal Robot Locomotion*. CRC Press, 2007.
[9] Y. Hurmuzlu and D. B. Marghitu, "Rigid body collisions of planar kinematic chains with multiple contact points," *Int. J. of Robotics Research*, pp. 82–92, Feb 1994.
[10] A. Sabanovic and K. Ohnishi, *Motion Control Systems*. John Wiley & Sons, 2011.
[11] C. O. Saglam and K. Byl, "Stability and gait transition of the five-link biped on stochastically rough terrain using a discrete set of sliding mode controllers," in *Proc. of IEEE Int. Conf. on Robotics and Automation (ICRA)*, 2013.
[12] R. Bellman, "A Markovian decision process," *Indiana University Mathematics Journal*, vol. 6, pp. 679–684, 1957.
[13] C. O. Saglam and K. Byl, "Robust policies via meshing for metastable rough terrain walking," in *Proc. Robotics: Science and Systems (RSS)*, 2014. accepted for publication.