Reactive Planning and Control of Planar Spring-Mass Running on Rough Terrain

Ömür Arslan and Uluç Saranlı

***Abstract*—An important motivation for work on legged robots has always been their potential for high performance locomotion on rough terrain and the outdoors. Nevertheless, most existing control algorithms for such robots either make rigid assumptions about their environments (e.g flat ground), or rely on kinematic planning at low speeds. Moreover, the traditional separation of planning from control often has negative impact on the robustness of the system, particularly when dynamic behaviors are considered. In this paper, we introduce a new method for dynamic, fully reactive footstep planning for a planar spring-mass hopper. Our approach is based on a careful characterization of the model dynamics and the design of an associated deadbeat controller, used within a sequential composition framework. This yields a purely reactive controller with a large domain of attraction that requires no explicit replanning during execution. We show in simulation that plans constructed for a simplified dynamic model can successfully control locomotion of a more complete model across rough terrain. We also characterize the performance of the planner over rough terrain and show that it is robust against both model uncertainty and measurement noise, maintaining stability even under large disturbances such as misplaced footsteps without the need for replanning.**

***Index Terms*—reactive control, footstep planning, sequential composition, spring-mass running, robust control**

I. Introduction

Legged morphologies have always been considered necessary to achieve robust and autonomous traversal of complex, outdoor terrain. Despite effective behaviors and performance demonstrated by tracked vehicles [46] and flexible multi-wheeled platforms [43], behaviors realizable with such morphologies remain limited due to restricted directions in which forces can be applied to the robot body. Even leg/wheel hybrid designs and active suspension systems [22] suffer from the requirement of sustained contact with the ground, making traversal of broken terrain with holes or large obstacles infeasible. On the other hand, while legged designs, particularly those capable of dynamic dexterity, do not suffer from such limitations [32, 42, 45], their robust and maneuverable control on complex terrain is still a largely unsolved problem. Traditional approaches which separate planning and control perform well only when slow, quasi-static movement patterns are adopted [28, 29, 33, 39], with decreasing applicability in the presence of model uncertainty and measurement noise resulting from dynamic behaviors. In contrast, reactive control methods, relying on control policies with large domains rather than local stabilization of time trajectories, promise to address problems with model inaccuracies, but often lack any formal performance and stability guarantees, make rigid assumptions about their environment and do not offer the scalability necessary for deployment on more realistic settings [24].

In this paper, we propose a novel algorithm to address these issues for the specific but widely applicable problem of purely reactive footstep planning and control of a planar spring-mass hopper running on rough terrain with large height variations, such as the one illustrated in Figure 1. Our focus on planar hopping is founded on the success of the well-known Spring-Loaded Inverted Pendulum (SLIP) model [44] both in accurately describing runners in nature [5] and in providing morphological inspiration and a high-level control interface to many robot runners [2, 20, 26, 37]. Consequently, our robust control and planning framework for this model promises to be applicable to a variety of robot morphologies ranging from monopedal and bipedal runners to hexapedal robots. The present paper exclusively focuses on monopedal locomotion and its natural extension to bipedal running [37], acknowledging that the kinematics of foot placement for multilegged platforms, and the generalization of deadbeat control strategies are further challenges that need to be addressed for applicability to more complex morphologies.

Ö. Arslan is with the Dept. of Electrical & Systems Eng., University of Pennsylvania, Philadelphia, PA, USA `omur@seas.upenn.edu`

U. Saranlı is with the Dept. of Computer Engineering, Bilkent University, 06800 Ankara, Turkey `saranli@cs.bilkent.edu.tr`

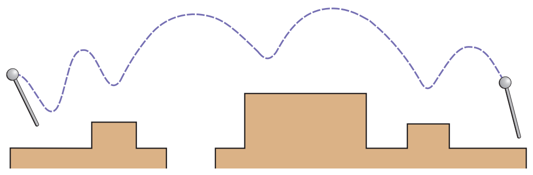

Fig. 1. Illustration of a spring-mass hopper running over rough terrain.

II. Related Work

A large body of work in the literature related to legged traversal of rough terrain focuses on kinematic trajectory planning and control of slow moving platforms [4, 30, 32]. Resulting simplifications in the control problem hence admit the investigation of both structural properties of trajectories themselves such as their static margins of stability [29, 39] as well as orthogonal issues such as energy efficiency, minimization of body undulations [48] or high level decision-making [13, 34]. Nevertheless, such quasi-static planning methods often necessitate relatively inefficient, fully-actuated and slow robot designs since they rely on the suppression of second-order dynamics through either velocity limits or explicit cancellation.

In contrast, exploiting second-order dynamics to achieve indirect controllability without explicit actuation was shown to enable much more efficient and capable robot morphologies [37], with behavioral capabilities far above those offered by quasi-static platforms [11, 36, 42]. However, their complex dynamics make it difficult to design locomotion controllers with any performance guarantees even on flat ground. Consequently, existing work on the traversal of rough terrain with such dynamically dexterous platforms relies mostly on scenario-specific heuristics or manual tuning, with only a few recent results on formal inquiries on stability and performance [31, 47].

In this context, an important line of research focuses on dynamic walking with the compass-gait model, introduced in [21] as the simplest model to capture the dynamics of walking. Initial intuitive controllers for complex terrain [15] were followed by more careful consideration of walking dynamics [38], leading to optimal control methods for rough terrain traversal [7, 27]. Recent results in this area recognize that trajectory stability on rough terrain is difficult to define [8], but exploit reduced dimension projections of walking dynamics to coordinate frames to which desired trajectories are transversal to achieve formally established stability properties [18, 31]. Similar ideas were also explored for more complex walking machines [14], with recent progress of extensions to rough terrain [47].

In contrast, running behaviors, generally modeled through the Spring-Loaded Inverted Pendulum (SLIP) model, pose additional challenges due to their more complex dynamics, as well as the practical necessity of only using intermittent, once-per-step control actions [41, 44]. Initial attempts at rough terrain traversal with this morphology were largely based on intuitive control and planning strategies [24, 37, 49] that were sensitive to modeling uncertainty due to their separation of planning and execution, relying on explicit replanning when necessary. However, unlike quasi-static legged platforms where such non-reactive planning strategies may succeed [10, 12, 19, 25, 28, 29, 35], reactivity, achieved through control policies with large domains of attraction, is necessary for systems that must rely on their second-order dynamics.

One of the most successful methods in integrating deliberate planning with reactivity for dynamically dexterous robots is Sequential Composition, first introduced in the context of juggling [6] and later applied to other platforms such as planar mobile robots with different actuation modalities [16, 17] and the Minifactory [40]. Sequential composition characterizes dynamic behaviors for a robotic system through their invariant domains and goal sets in their state space, ensuring proper activation order through a prioritization combined with reactive decision-making. The “backchaining” principle underlying this method has also been applied to planar dynamic walking [47] with recent work extending into three dimensions through the use of dynamic locomotion primitives [23].

An important necessity in using the sequential composition framework is the availability of behavioral controllers whose correctness and stability properties have been established. Even though experimental characterization of control laws is always possible [6], model-based controller design and analysis can lead to abstractions that are more generally applicable [17]. Fortunately, recent results provide us with simple but accurate analytic tools for the SLIP model [9, 41], supporting the design of effective controllers as well as their analytic characterization [1]. The reactive footstep controller we introduce in this paper for the SLIP model benefits from these results.

Our planning framework closely follows the sequential composition formalism but deviates in our representation of behavioral primitives and associated invariant domain and feasible goal sets. Among primary contributions of our paper are the formulation of a general framework for discrete, per-step application of sequential composition to a loosely constrained family of hopping robots, as well as the application of resulting ideas to both a simplified, analytically tractable running model and the much more relevant SLIP model. Some of the ideas in this paper were previously presented in [3], albeit without any analytic derivations for the Ball Hopper model, applications to the SLIP model or extensive simulations under noise.

III. Framework and Problem Statement

A. Running Behaviors on Rough Terrain

In this paper, we seek to construct a robust running controller for a planar, monopedal runner traversing rough terrain. In contrast to milder interpretations of roughness, we consider *rough terrain* to mean that the ground has substantial irregularities with magnitudes comparable to the leg length as exemplified by Figure 1. Consequently, finding suitable footholds during locomotion, together with sequencing of dynamic running strides for their realization are the two central problems addressed in this paper.

Generally, running trajectories for planar monopedal or bipedal runners exhibit a common structure: As shown in Figure 2, they alternatingly go through *flight* and *stance* phases, separated by *touchdown* and *liftoff* events as the foot comes into contact with, and leaves the ground, respectively. A minimal state vector for such systems can be defined in an inertial world frame $\mathcal{W}$ as

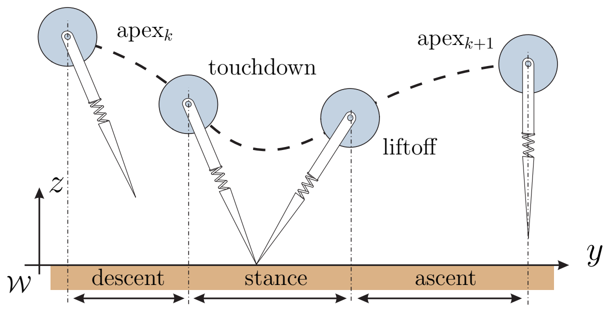

Fig. 2. General structure and definitions for monopedal running behaviors.

$$\mathbf{X} := [\ y,\quad z,\quad \dot{y},\quad \dot{z}\ ]^T\ . \tag{1}$$

An *apex* event associated with the highest point of the center of mass (COM) is also defined during flight with $\dot{z} = 0$. In this and the next few sections, we establish a definitional framework capturing common structural aspects of such gaits, making as few assumptions as possible about the underlying system beyond this structure in order to ensure general applicability of our reactive planning framework.

In modeling rough terrain, we assume that a planar legged platform is locomoting on a supporting surface described by a piecewise constant *elevation function* $h : \mathbb{R} \to \mathbb{R}$, possibly with a number of “holes” where no foot placement is possible. During flight, we assume that the robot COM follows a ballistic trajectory, whereas during stance, its dynamics are determined by its leg morphology and control, which we leave unspecified for the time being.

A very useful abstraction for the analysis and control of such systems is obtained through a Poincaré section at apex points, defining a reduced dimension discrete state vector as

$$\mathbf{Z} := [\ y,\quad z,\quad \dot{y}\ ]^T\ . \tag{2}$$

We assume that gait control is achieved with per-step control inputs $\mathbf{u}_k$ selected at each apex, allowing independent but possibly limited control of all three degrees of freedom for the next apex. Depending on the exact leg morphology, these controls may be realized either discretely or throughout the entirety of the following flight and stance phases. Nevertheless, they give rise to a controlled apex return map

$$\mathbf{Z}_{k+1} := f_a(\mathbf{Z}_k, \mathbf{u}_k)\ . \tag{3}$$

Note that these definitions are applicable to most planar monopedal or bipedal morphologies, including complex, multi-jointed leg designs.

B. Discrete Abstraction of Running Strides

In general, locomotory dynamics are symmetric with respect to positional variables. Consequently, we focus on a sufficiently expressive, discrete abstraction of a single running stride using a ground segment of length $2l$ as a foothold as shown in Figure 3.

To this end, we define a *stride policy template* Φ as a triple

$$\Phi = [\ R_E,\ R_V,\ \mathcal{U}\ ]\ , \tag{4}$$

where $R_E \subset \mathbb{R}$ and $R_V \subset \mathbb{R}$ respectively determine the initial apex energy and forward velocity ranges for which this policy may be invoked and $\mathcal{U}$ indicates the set of control inputs that can be used by this policy. We hence define the *domain* associated with a policy as

$$\begin{aligned}\mathcal{D}(\Phi) := \{\mathbf{Z} \mid \dot{y} \in R_V(\Phi);\ E \in R_E(\Phi);\\ \forall \mathbf{u} \in \mathcal{U}(\Phi).\ y_{f,\text{td}}(\mathbf{Z}, \mathbf{u}) \in [-l, l]\}\ ,\end{aligned} \tag{5}$$

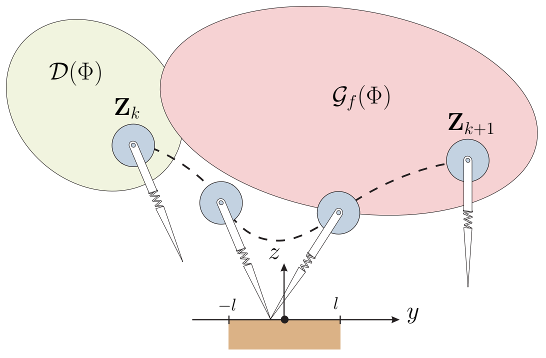

Fig. 3. Stride policy template for a single step using a ground segment of length $2l$ as a foothold. Also shown are the policy domain $\mathcal{D}(\Phi)$, and feasible goal $\mathcal{G}_f(\Phi)$ regions associated with the policy template Φ_i.

representing all apex states with admissible velocity and energy values from which the horizontal foot position at touchdown $y_{f,\text{td}}$ falls within the ground segment for all choices of allowable control inputs $\mathbf{u} \in \mathcal{U}(\Phi)$. This definition currently constrains our framework to systems with only a single point of ground contact.

Our abstraction of a controlled stride also incorporates a *feasible goal* region, $\mathcal{G}_f(\Phi)$, consisting of points reachable through admissible controls from any point within the domain. More formally, we define

$$\mathcal{G}_f(\Phi) := \left\{ \mathbf{Z}' \mid \forall \mathbf{Z} \in \mathcal{D}(\Phi).\ \exists \mathbf{u} \in \mathcal{U}(\Phi).\ \mathbf{Z}' = f_a(\mathbf{Z}, \mathbf{u}) \right\} . \qquad (6)$$

The primary motivation behind requiring accessibility from *all* domain points is to achieve runtime robustness against environmental and sensory noise, which might perturb system trajectories away from the predicted outcome of a step. In such cases, feasible control inputs should still exist to bring the system to the desired goal point as long as the previous state is in the domain of the policy. An illustration of both the domain and feasible goal regions is given in Figure 3.

These definitions can be used for any planar monopedal or bipedal runner. The number of stride policy templates that are appropriate for a specific system will depend on the shapes and sizes of the domain and feasible goal regions corresponding to different energy, velocity and control input ranges. We define the set of policy templates as

$$\mathcal{P} := \{ \Phi_i \mid i = 1,\ \dots,\ N \} . \qquad (7)$$

Even though this is left as a design choice at the current level of generality, we will give more specific guidelines and observations in the context of specific running models in subsequent sections.

C. Situated and Instantiated Stride Policies

Deterministic footstep planning must inevitably take into account the layout of the ground surface, which we assume to be known, or at least mapped sufficiently ahead of time. In order to make use of the stride policy templates defined in Section III-B, we discretize the elevation profile with a piecewise constant cover of ground segments of fixed length $2l$, each centered at a point $\mathbf{p}_j \in \mathbb{R}^2$. Assuming that there are M such segments, we then "situate" all stride policy templates Φ_i on each ground segment j such that their origin coincides with $\mathbf{p}_j$, resulting in a set of *situated ground policies*

$$\mathcal{P}_S = \left\{ \Phi_i^{\mathbf{P}_j} \mid i = 1,\ \dots,\ N;\ j = 1,\ \dots,\ M \right\} , \qquad (8)$$

with the domain and goal regions shifted accordingly.

Domain regions associated with policies in $\mathcal{P}_S$ determine which policies can be used for the stride following an initial apex state. However, the corresponding feasible goal regions $\mathcal{G}_f(\Phi_i^{\mathbf{P}_j})$ still leave a continuum of possibilities for which apex state to aim for. Planning for footsteps using a sequential composition approach, our framework will "instantiate" these situated policies with specific goal points from within the feasible set to yield the set of *instantiated stride policies*

$$\mathcal{P}_I := \left\{ \Phi_i^{\mathbf{P}_j}[\mathbf{Z}_g] \mid \mathbf{Z}_g \in \mathcal{G}_f(\Phi_i^{\mathbf{P}_j}) \right\} \quad \forall i, j \ . \qquad (9)$$

Note that our framework would also allow partitioning of the ground cover in segments of differing lengths, using corresponding policy template definitions. Our experiments show that a suitably chosen segment length $2l$ allowing at least two segments in each contiguous ground region provides enough flexibility to construct policies with sufficiently large domains of attraction. In light of these observations, and to keep the discussion focused, we only use a fixed ground segment length for the entire terrain in this paper.

In the following sections, we describe two models that are compatible with this formulation: First, the SLIP model as a realistic embodiment of running behaviors, then a "ball-hopper" model as a simplification of SLIP dynamics, admitting analytic derivations for effective computation of its domain and feasible goal sets.

IV. Dynamic Running Models

A. The Spring-Loaded Inverted Pendulum Model

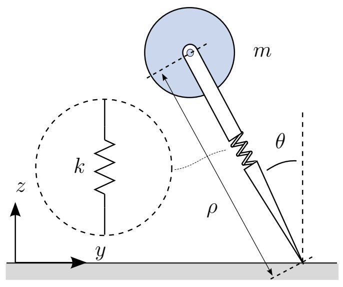

Fig. 4. The Spring-Loaded Inverted Pendulum model

1) System Dynamics: The Spring-Loaded Inverted Pendulum (SLIP) model, illustrated in Figure 4, consists of a point mass m, connected to a massless telescoping leg with compliance k. Running trajectories for the SLIP model have the same structure as the model shown in Figure 2. However, the stance dynamics of this model,

$$\begin{bmatrix} m\ddot{\rho} \\ m\rho^2\ddot{\theta} \end{bmatrix} = \begin{bmatrix} m\rho\dot{\theta}^2 - k(\rho - \rho_0) - mg\cos\theta \\ (-2m\rho\dot{\rho}\dot{\theta} + mg\rho\sin\theta) \end{bmatrix} , \qquad (10)$$

are non-integrable, but fortunately admit accurate approximate solutions previously presented in the literature [41]. Three discrete, once-per-step control inputs are available to this model: The leg angle at touchdown θ_{td} and two separate spring constants, k_c and k_d during the compression and decompression portions of the stance phase.

2) Position-Aware Deadbeat Control of SLIP Running: The execution phase of the reactive planning framework we propose in this paper relies on the presence of a reliable controller for individual steps during running. This controller should be capable of finding control inputs $\mathbf{u}$ to correctly realize any desired apex state in the feasible goal set for a policy $\mathbf{Z}^* \in \mathcal{G}_f(\Phi)$ when invoked from initial states within the associated domain $\mathbf{Z}^0 \in \mathcal{D}(\Phi)$ such that $\mathbf{Z}^* := f_a(\mathbf{Z}^0, \mathbf{u})$. However, all existing gait controllers for the SLIP model only focus on two of the three apex states: Forward velocity and hopping height. However, footstep planning also requires control over the horizontal position at apex. In this section, we present such a "position-aware" single-step deadbeat controller for the SLIP model based on the inversion of its apex return map.

Our controller design is based on the approximate analytic return map for the SLIP model proposed in [41] defined as

$$[y_a, z_a, \dot{y}_a]^T = \hat{f}_a([y_a^0, z_a^0, \dot{y}_a^0]^T, [\theta_{td}, k_c, k_d]^T) \ . \qquad (11)$$

Even though analytic inversion of these approximations is still not possible, the monotonicity of components in this return map admits the decomposition of the problem into two nested numerical optimization problems. Given θ_{td} and k_c, the decompression spring constant can be computed using the energy balance

$$k_d = k_c + \frac{m(\dot{y}^{*^2} - \dot{y}_a^2) + 2mg(z^* - z_a)}{(\rho_b - \rho_0)^2} \ , \qquad (12)$$

based on the energy input at the bottom instant through an approximate analytic computation of the maximal spring compression ρ_b [44]. Given this relation, the angular momentum of the SLIP system, and the associated liftoff leg angle θ_{lo} are monotonic functions of the touchdown angle as illustrated by Figure 5. Consequently, given k_c and k_d, we can choose the touchdown angle that minimizes the horizontal apex position through the one-dimensional minimization

$$\theta_{td}^* = \text{argmin}\,(C_1(\theta_{td})) \qquad (13)$$

$$C_1(\theta_{td}) := w_1 d_{lo}(\mathbf{Z}^0, [\theta_{td}, k_c, k_d])^2 + w_2(\dot{y}_a - \dot{y}^*)^2 \qquad (14)$$

of a cost function with the liftoff position error $d_{lo}(\mathbf{Z}^0, \mathbf{u})$, and the apex velocity error weighted by w_1 and w_2, respectively.

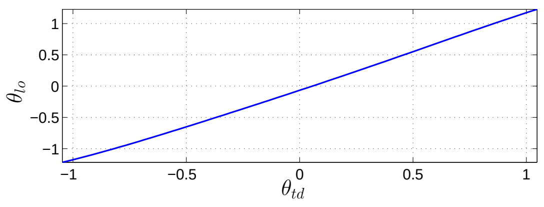

Fig. 5. Monotonic dependence of the liftoff angle θ_{lo} on the touchdown angle θ_{td} for the SLIP model.

This "inner" optimization yields the best touchdown angle solution for a given compression spring constant k_c. Another one-dimensional, "outer" optimization can now be used to solve for k_c as

$$k_c^* = \text{argmin}\,(C_2(k_c)) \qquad (15)$$

$$C_2(k_c) := w_3 \parallel [y_a, z_a] - [y_a^*, z_a^*] \parallel^2 + w_4(\dot{y}_a - \dot{y}^*)^2 \qquad (16)$$

where the cost function C_2 captures the apex position and horizontal velocity errors with gains w_3 and w_4, respectively. These nested numerical optimizations yield the desired single-step deadbeat controller that can simultaneously achieve all three components of the desired apex state while still being computationally feasible.

B. The Ball-Hopper (BHop) Model

1) System Dynamics: Despite the availability of simple analytic approximations for the apex return map of the SLIP model, they still do not admit analytic formulations of the domain and feasible goal regions defined in Section III-B. Consequently, we propose a new model that captures essential features of the SLIP model, including analogous control inputs, while being sufficiently simple to admit analytic representations of the domain and feasible goal regions.

Our "ball hopper" model summarizes the stance dynamics of the SLIP model with an instantaneous, controllable transition. As shown in Figure 6, the model consists of a point mass m, that comes into contact with a "virtual" ground positioned at $z = \rho_0$ during its descent phase. During flight, the system obeys simple ballistic flight equations

$$\begin{bmatrix} \ddot{y} \\ \ddot{z} \end{bmatrix} = \begin{bmatrix} 0 \\ -g \end{bmatrix} . \qquad (17)$$

In contrast, the stance phase is summarized as the transition function

$$\mathbf{X}_{lo} = F_s(\mathbf{X}_{td}) := A\mathbf{X}_{td} + B \ , \qquad (18)$$

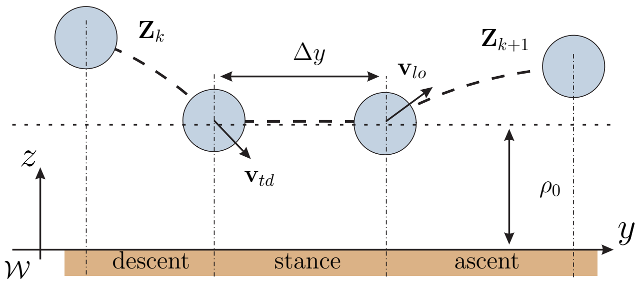

Fig. 6. The ball hopper model

with

$$A := \begin{bmatrix} \mathbf{I}_{2x2} & \mathbf{0}_{2x2} \\ \mathbf{0}_{2x2} & R(\theta)\begin{bmatrix} 1 & 0 \\ 0 & -k \end{bmatrix} R(-\theta) \end{bmatrix} \qquad (19)$$

$$= \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1-(1+k)\sin^2\theta & 0.5(1+k)\sin 2\theta \\ 0 & 0 & 0.5(1+k)\sin 2\theta & 1-(1+k)\cos^2\theta \end{bmatrix} \qquad (20)$$

$$B := [\ \Delta y, \quad 0, \quad 0, \quad 0\ ]^T \qquad (21)$$

This transition map incorporates three controllable parameters *designed* to closely match those available to the SLIP model:

- Touchdown angle θ : Corresponds to the SLIP touchdown angle and primarily controls the direction of the liftoff velocity,
- Velocity gain k : Corresponds to the ratio k_d/k_c of decompression and compression spring constants in the SLIP model and controls energy gain during stance by summarizing radial SLIP dynamics,
- Horizontal shift Δy : Corresponds to the average spring stiffness for the SLIP model and controls the horizontal displacement during stance.

Under these definitions, the apex return map for the ball hopper model can be formulated as the composition of the descent, stance and ascent maps to yield

$$F_a := F_u \circ F_s \circ F_d \ . \qquad (22)$$

where ballistic trajectories yield the descent and ascent maps as

$$F_d(\mathbf{X}_a) := [\ y_a + \dot{y}_a\sqrt{2z_a/g},\ 0,\ \dot{y}_a,\ -\sqrt{2gz_a}\ ]^T \qquad (23)$$

$$F_u(\mathbf{X}_{lo}) := [\ y_{lo} + \dot{y}_{lo}\dot{z}_{lo}/g,\ z_{lo} + \dot{z}_{lo}^2/(2g),\ \dot{y}_{lo},\ 0\ ]^T \ . \qquad (24)$$

2) Deadbeat Control of BHop Running: In this section we present a deadbeat controller for the BHop model similar to the controller presented in Section IV-A2 for the SLIP model.

The invertible ascent map, combined with the descent map, reduces the inversion problem to only the stance map as

$$\mathbf{u} = F_s^{-1}(F_d(\mathbf{X}_a^0), F_u^{-1}(\mathbf{X}_a^*)) \ . \qquad (25)$$

The horizontal shift control parameter is easily computed as

$$\Delta y = y_{lo}^* - y_{td}^0 \ . \qquad (26)$$

The remaining control parameters only effect the velocity states through the last two rows of (18). Inspection of the fourth row reveals

$$1 + k = \frac{\dot{z}_{lo}^* - \dot{z}_{td}^0}{0.5\dot{y}_{td}^0\sin(2\theta) - \dot{z}_{td}^0\cos^2\theta} \ . \qquad (27)$$

Subsequent substitution in (18) yields

$$-\frac{\dot{y}_{lo}^* - \dot{y}_{td}^0}{\dot{z}_{lo}^* - \dot{z}_{td}^0} = -\frac{\sin\theta}{\cos\theta}\left(\frac{-\dot{y}_{td}^0\sin\theta + \dot{z}_{td}^0\cos\theta}{\dot{y}_{td}^0\sin\theta - \dot{z}_{td}^0\cos\theta}\right) = \tan\theta \qquad (28)$$

whose solution for θ, in conjunction with (27) yields the solution for k through the identity $\tan^2\theta + 1 = 1/\cos^2\theta$ as

$$k = \frac{(\dot{z}^0_{td} - \dot{z}^*_{lo})^2 + (\dot{y}^0_{td} - \dot{y}^*_{lo})^2}{\dot{z}^0_{td}(\dot{z}^0_{td} - \dot{z}^*_{lo}) + \dot{y}^0_{td}(\dot{y}^0_{td} - \dot{y}^*_{lo})} - 1 \ . \tag{29}$$

These derivations yield a single-step deadbeat controller for the BHop model capable of reaching any point in the feasible goal set from any initial point within the domain.

V. Reactive Planning Framework

A. Sequential Composition of Stride Policies

The discrete abstraction for running steps defined in Section III-B, applied to either the BHop or the SLIP model, can be used as the basic building block for constructing a plan of footstep choices to reach a given apex goal state through backchaining. However, such a simplistic, offline sequence of footsteps can seldom be exactly realized in the presence of sensor or model noise as well as other large disturbances. Fortunately, the *sequential composition* framework [6, 16] provides a way in which backchaining can be combined with reactivity to eliminate the need for replanning.

Our application of sequential composition for footstep planning differs from its earlier uses in two important aspects. Firstly, behavioral policies in our domain are discrete in nature, summarizing the actions of a single step. Secondly, these single-step policies are *parametric* in their choice of goal states, requiring the planner to choose appropriate goal states from within the feasible sets.

Given the set of situated policies $\mathcal{P}_S$ as defined in (8), we capture the feasibility of backchaining two policies with the "can prepare" relation $\succeq_c \subseteq \mathcal{P}_S \times \mathcal{P}_S$, defined as follows.

Definition 1: A situated policy $\Phi_i^{\mathbf{P}_i}$ *can prepare* another policy $\Phi_j^{\mathbf{P}_j}$, denoted by $\Phi_i^{\mathbf{P}_i} \succeq_c \Phi_j^{\mathbf{P}_j}$, iff the following condition holds,

$$\mathcal{G}_f(\Phi_i) \cap \mathcal{D}(\Phi_j) \neq \emptyset \ . \tag{30}$$

This relation also forms the basis for instantiating stride policies with specific goal choices that lie within the intersection of the domain and feasible goal regions. More formally, we can obtain the set of instantiated stride policies through the construction

$$\mathcal{P}_I = \Big\{ \Phi^{\mathbf{P}}[\mathbf{Z}_g] \ \mid \ \exists\, \bar{\Phi}^{\bar{\mathbf{P}}} \in \mathcal{P}_S \ . \ \Phi^{\mathbf{P}} \ \succeq_c \ \bar{\Phi}^{\bar{\mathbf{P}}} \ , \\ \mathbf{Z}_g \in \mathcal{G}_f(\Phi^{\mathbf{P}}) \cap \mathcal{D}(\bar{\Phi}^{\bar{\mathbf{P}}}) \Big\} \tag{31}$$

where $\mathbf{Z}_g$, for which there are infinitely many choices, can be selected according to a number of different criteria considering, for example, how "safe" the choice of the goal state would be in the presence of noise. The selection of this particular goal point in this intersection has no effect on the prepares relation and does not change policy ordering in any way. Nevertheless, this choice impacts the robustness of the algorithm during runtime since inaccuracies in the deadbeat controller's ability to reach this goal point may result in apex states falling outside $\mathcal{D}(\bar{\Phi}^{\bar{\mathbf{P}}})$, leading to a different sequencing of policies during runtime. Our criteria for choosing these "intermediate" goal points is to maximize their distance to the boundary of the domain of the policy being prepared, computed through a sufficiently dense sampling of domain/goal intersections.

Subsequently, once the set of instantiated stride policies is defined, the *prepares* relation $\succeq \subseteq \mathcal{P}_I \times \mathcal{P}_I$, is defined as follows.

Definition 2: An instantiated policy $\Phi_i^{\mathbf{P}_i}[\mathbf{Z}_i] \in \mathcal{P}_I$ *prepares* another instantiated policy $\Phi_j^{\mathbf{P}_j}[\mathbf{Z}_j] \in \mathcal{P}_I$, denoted by $\Phi_i^{\mathbf{P}_i}[\mathbf{Z}_i] \succeq \Phi_j^{\mathbf{P}_j}[\mathbf{Z}_j]$, iff the following condition holds,

$$\mathbf{Z}_i \in \mathcal{D}(\Phi_j) \ . \tag{32}$$

The *prepares graph* $\mathcal{G}$ that results from Definition 2 captures all relevant sequencing constraints between instantiated stride policies. Its construction also provides a consistent criteria for choosing specific goal settings for each policy. At this point, the set of instantiated policies $\mathcal{P}_I$, together with the prepares relation $\succeq$ are sufficient to build a global, reactive control policy through sequential composition.

In practice, the complexity of computing the prepares graph is primarily associated with finding intersections between domain and goal sets and selecting specific goal instances to maximize the desired safety criteria. The former can be done rapidly when analytic representations of domain and feasible goal sets, such as those presented in Appendices A and B for the ball hopper model, are available. Even without such analytic representations (e.g. when body-ground collisions are also considered), the localized nature of policies in the horizontal direction (see Figure 9) limits the number of necessary pairwise comparisons, making algorithm complexity linear in the terrain length. For example, with 36758 policy instances for a ball hopper running on the rough terrain of Figure 1, a simple initial bounding box check reduces the number of domain/goal comparisons to less than 350 for each policy instance.

In contrast, the complexity of selecting goal instances strongly depends on the choice of safety criteria and whether it can be formulated analytically. In this paper, we maximize the distance of the selected goal to the boundary of the domain-goal intersection through dense sampling, incurring high computational cost. As such, the generation of the prepares graph for the SLIP model on the terrain in Figure 10 takes approximately 30 minutes to compute with our inefficient prototype implementation in Matlab on a modern desktop PC. This could be reduced through a more optimized implementation and a less strict but analytically feasible safety criterion. Note, also, that the prepares graph is fixed for a given terrain and can be reused for different goal choices. Consequently, it can be computed offline and efficiently stored in sparse matrix form as shown in Figure 12.

B. Policy Deployment and Execution

Given a global apex state goal $\mathbf{Z}_g$ to be reached, the principal idea behind our application of sequential composition is to convert the prepares relation into a total order for policy instances, starting from policies that can reach the global goal in a single step, extended with backchaining through the prepares relation. During execution, when the robot finds itself in a particular apex state, the controller goes through the policy in this total order, checking whether the apex state falls within the domain of any policy. If such an instantiated policy $\Phi^{\mathbf{P}}[\mathbf{Z}_i]$ is found, a single-step deadbeat controller is invoked to reach the corresponding intermediate goal $\mathbf{Z}_i$ and the process is repeated. Absent noise, this process is guaranteed to reach the goal state if the total order respects the sequencing constraints captured in the prepares graph [6].

Figure 7 illustrates the details of the deployment algorithm for our footstep planner. Functionalities of subroutines within this algorithm are as follows:

- findGoalPolicies() : Finds situated stride policies whose feasible goal sets include the desired global goal $\mathbf{Z}_g$.
- instantiatePolicies() : Instantiates all situated policies with specific goal points as described in Section III-C.
- buildPreparesGraph() : Computes prepares relations between pairs of instantiated policies and build the associated graph.
- pickbest() : Selects the best available policy from the queue based on a safety criteria (examples given in Section VI-B).
- findUnusedPrepares() : Processes the prepares graph to locate currently unused policies that prepare the current selection.

At the beginning, a priority queue, $PolicyQueue$ is initialized with goal policies and used to identify the next policy to add to

Fig. 7. Algorithm for the instantiation and deployment of stride policies towards a global goal $\mathbf{Z}_g$.

```
1:  procedure DEPLOY($\mathcal{P}_S$, $\mathbf{Z}_g$)
2:      $\mathcal{P}_G \leftarrow$ findGoalPolicies($\mathcal{P}_S, \mathbf{Z}_g$)
3:      $\mathcal{P}_I \leftarrow \mathcal{P}_G \cup$ instantiatePolicies($\mathcal{P}_S$)
4:      $\mathcal{G} \leftarrow$ buildPreparesGraph($\mathcal{P}_I$)
5:      $PolicyQueue \leftarrow \mathcal{P}_G$
6:      $PolicyList \leftarrow [\,]$
7:      while ! isempty($PolicyQueue$) do
8:          $\Phi^{\mathbf{P}}[\mathbf{Z}_i] \leftarrow$ pickbest($PolicyQueue$)
9:          $PolicyList \leftarrow$ append($\Phi^{\mathbf{P}}[\mathbf{Z}_i], PolicyList$)
10:         $\mathcal{P}_P \leftarrow$ findUnusedPrepares($\mathcal{G}, \Phi^{\mathbf{P}}[\mathbf{Z}_i], PolicyList$)
11:         $PolicyQueue \leftarrow$ insertall($\mathcal{P}_P, PolicyQueue$)
12:     end while
13:     return $PolicyList$
14: end procedure
```

the total order maintained in $PolicyList$. Backchaining is accomplished by extending the queue with preparing policies through the findUnusedPrepares().

Note that there is substantial freedom in how the pickbest() function chooses the next policy instance to be placed in the total order without compromising the correctness of the deployment. Different heuristics can be used to prioritize available policies including their safety with respect to unexpected collisions with the ground and the corresponding depth of the plan. We will explore some of these heuristics in subsequent sections.

Due to the positional locality of policies, the method findGoalPolicies() has negligible computational cost and returns only a small number of policy instances independent of the terrain length. For instance, only 321 situated policies were found to prepare the goal state in the example of Section VI. The cost associated with extending a precomputed prepares graph with such a small number of goal policy instances is negligible since domain/goal comparisons that must be performed for each are also limited by the positional locality of each policy as noted in Section V-A. Once the updated prepares graph is finalized, backchaining involves straightforward following of links in the prepares graph and can be done efficiently with the use of appropriate data structures.

Fig. 8. Reactive execution controller for the deployed policy ordering, invoked at each apex $\mathbf{Z}_a$ to compute control inputs for the next stride.

```
1: procedure STRIDECONTROL($\mathbf{Z}_a$, $PolicyList$)
2:     for all $\Phi^{\mathbf{P}}[\mathbf{Z}_i]$ in $PolicyList$ do
3:         if $\mathbf{Z}_a \in \mathcal{D}(\Phi^{\mathbf{P}})[\mathbf{Z}_i]$ then
4:             return $\mathbf{u} =$ deadbeatControl($\mathbf{Z}_a, \mathbf{Z}_i$)
5:         end if
6:     end for
7:     return $Error$
8: end procedure
```

Once the ordered list of policies is obtained through the deployment algorithm, execution proceeds by invoking the stride controller given in Figure 8. At every apex, the state of the system $\mathbf{Z}_a$ is measured, compared against the domains of all instantiated policies in the order they are deployed and the goal associated with the first match is targeted through a deadbeat controller. In contrast to offline footstep planners with separate planning and execution, this scheme integrates planning with control, resulting in robust reactive control while still ensuring proper sequencing of footstep choices. Note, also, that this scheme does not explicitly prescribe and patch together system trajectories and hence does not necessitate additional measures to ensure continuity. The computational complexity of the reactive execution controller is minimal since it only performs domain inclusion tests, which are supported by bounding box checks as well as the relative ease in which analytic boundaries for policy domains can be computed through ballistic flight equations.

VI. RUNNING WITH THE BALL-HOPPER MODEL

In this section, we apply our reactive planning framework to running with the BHop model across rough terrain. All our results presented below use the analytic region derivations for the BHop model which are detailed in Appendices A and B, respectively.

A. Simulation Environment and Policy Templates

All ball-hopper simulations in subsequent sections were obtained through numerical integration of the BHop dynamics detailed in Section IV-B1 using the ode45 function of Matlab with $m = 80kg$ and $\rho_0 = 1m$. 20 different policy templates were constructed, using combinations of different velocity and energy ranges

$$\begin{aligned} R_V &\in \{\, [-2.5,-1], [-1,0], [0,1], [1,2.5] \,\} \\ R_E &\in \{\, [120,160], [160,240], \\ &\qquad [240,400], [400,640], [640,1120] \,\} \end{aligned} \tag{33}$$

in MKS units. The ground segment length was chosen as $2l = 0.15m$ and the energy gain control input was constrained with $k \in [0.7, 1.4]$. For policy templates with $R_V < 0$, the remaining control inputs were constrained as $\theta \in [-\pi/2, \pi/3]$ and $\Delta y \in [-0.25, 0]$, whereas for policy templates with $R_V > 0$, they were chosen as $\theta \in [-\pi/3, \pi/2]$ and $\Delta y \in [0, 0.25]$. Domain and feasible goal regions for these templates are illustrated in Figure 9.

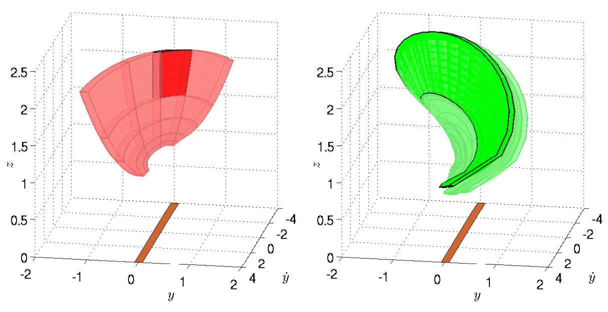

Fig. 9. Domain (left) and feasible goal (right) regions for all 20 ball-hopper policy templates in apex state coordinates. One of the stride policy templates with $R_V = [-1, 0]m/s$ and $R_E = [640, 1120]$ is highlighted for clarity. The patch on the bottom illustrates the ground segment with $2l = 0.15m$.

B. Rough Terrain Traversal

The rough terrain illustrated in Figure 1 features a rich collection of challenges, including substantial height irregularities and a "dangerous" gap whose size is comparable to the leg length. In this section, we apply our footstep planning framework to this terrain profile.

As described in earlier sections, the ground map is first discretized into segments of length $2l$, resulting in 240 segments for this terrain. Stride policy templates are then situated on these segments, yielding 1200 situated stride policies whose combined domain and feasible goal regions are illustrated in Figures 10 and 11, respectively. Note that both of these regions could be extended in the horizontal direction with additional situated policies but for the time being, we only focus

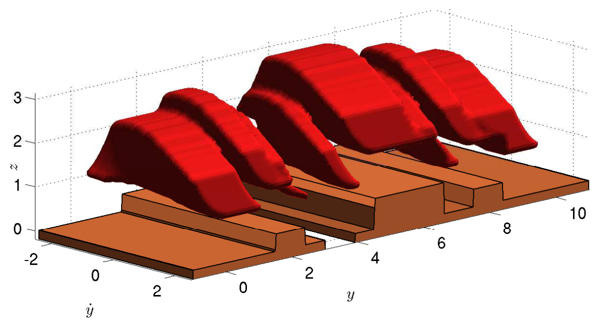

Fig. 10. Global domain coverage for the rough terrain example taking 1200 situated stride policies into account.

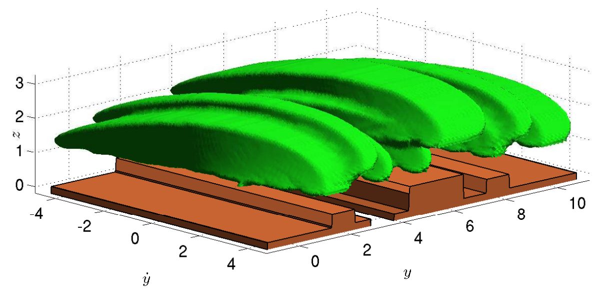

Fig. 11. Global feasible goal coverage for the rough terrain example taking 1200 situated stride policies into account.

on $y \in [0, 10]m$ for clarity. This yields 36758 instantiated policies, for which the resulting prepares relation is illustrated in Figure 12.

In preparation for handling sensor noise and map inaccuracies, we use a prioritization of policies within the deployment algorithm through the definition of the pickbest() function. In particular, we define a scalar "safety" measure for each policy instance, taking into account how much error can be tolerated in either the foot placement or the realization of the goal with the deadbeat controller. More formally, the safety of an instantiated policy $\Phi^{\mathbf{P}}[\mathbf{Z}_g]$ is defined as

$$h(\Phi^{\mathbf{P}}[\mathbf{Z}_g]) := w_e\ d_{\mathrm{e}}(\mathbf{p})^3 + w_g\ d_{\mathrm{g}}(\mathbf{Z}_g, \partial\mathcal{D}(\Phi_n^{\mathbf{P}_n}))\ , \qquad (34)$$

where d_{e} denotes the closest distance to the edge of the flat ground portion (*not* the small segment of length $2l$ but the contiguous ground region on which it resides) and $\partial\mathcal{D}(\Phi_n^{\mathbf{P}_n})$ denotes the boundary of the domain of the situated policy $\Phi_n^{\mathbf{P}_n}$ being prepared (i.e. $\Phi^{\mathbf{P}} \succeq \Phi_n^{\mathbf{P}_n}$). All simulations in subsequent sections use this safety criteria with manually tuned weights $w_e = 0.5$ and $w_g = 1.0$.

Figure 13 illustrates two example runs with the BHop model under our reactive controller, started from two different initial conditions. The execution algorithm of Figure 8 reactively selects the best policy corresponding to each measured apex state, leading the hopper to the global goal $\mathbf{Z}_g = [9.5, 1.3, 0]$. The reactive controller that results from the policy deployment is correct by construction and is guaranteed to take the hopper to the goal state from any state within the domain region illustrated in Figure 10. Note, also, that the foot safety criteria imposed by (34) ensures that footholds towards the middle of each contiguous ground region are preferred over other alternatives. In the next section, we will show the robustness of our algorithm in the presence of noise, owing both to its reactive nature as well as its prioritization of safe policies during deployment.

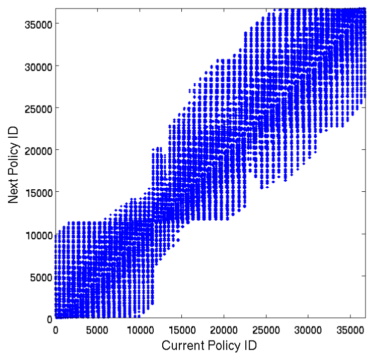

Fig. 12. A visualization of the prepares relation for the example terrain in Figure 1. Policy numbers increase with horizontal position along the terrain. Dots indicate when the corresponding current policy prepares the next policy.

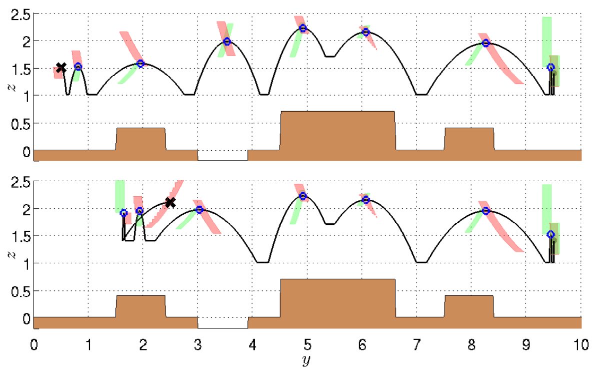

Fig. 13. Two example runs for reactive planning with the BHop model, started from different initial conditions but using the same reactive policy deployment towards the goal $\mathbf{Z}_g = [9.5, 1.3, 0]$ without any explicit replanning. Light green and dark red shaded regions illustrate cross sections of feasible goal and domain regions at each apex, respectively.

C. Robustness Against Sensor Noise and Map Inaccuracy

In this section we consider the performance of our reactive planner under three kinds of disturbances: Wind noise in the form of a constant, horizontal acceleration during flight, sensor noise in apex state measurements and map inaccuracy in the terrain height used by the planner.

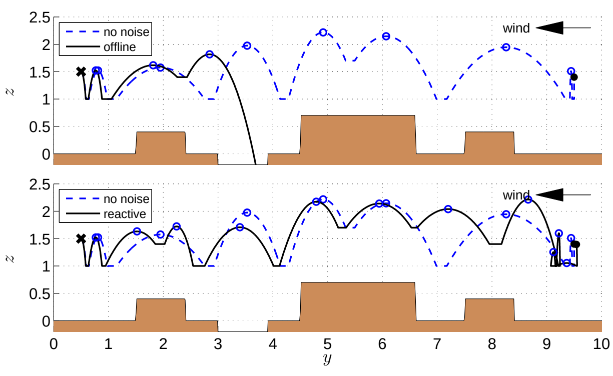

Fig. 14. Performance of our reactive controller under "wind noise". Top figure compares locomotion with and without noise using the same control inputs for each step. The bottom figure compares the no noise case with our reactive controller under noise. Wind magnitude is $\ddot{y} = -0.2m/s^2$ during flight.

Figure 14 illustrates an example run under wind noise. As shown in the top plot, an offline plan with precomputed control inputs at each step fails to react to the unexpected foot placement onto the high platform around $y = 2m$. In contrast, our reactive controller adapts to this unexpected large disturbance at the subsequent apex by choosing new control inputs without any explicit replanning.

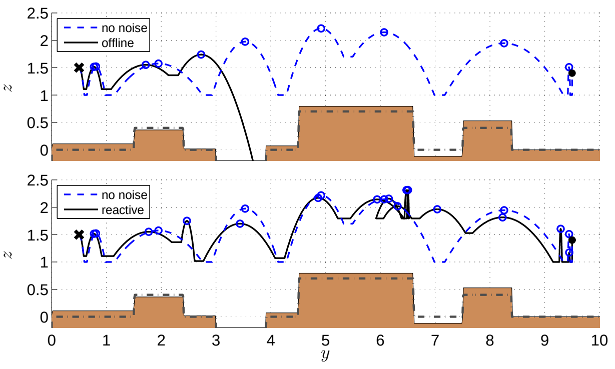

Fig. 15. Performance of our reactive controller under "ground noise". Top figure compares locomotion with and without noise using the same control inputs for each step. The bottom figure compares the no noise case with our reactive controller under noise. Ground heights perceived by the planner are shown as dashed lines.

Similarly, Figure 15 shows a scenario where the ground heights perceived by the planner are inaccurate with up to $0.15m$ errors in either direction. Once again, this causes unexpected foot collisions, leading to large disturbances and causing an offline planner to fail promptly. In contrast, the reactive planner chooses new policies (e.g. at $y = 2.2m$ and $y = 6.5m$), resulting in successful convergence to the goal $\mathbf{Z}_g = [9.5, 1.3, 0]$.

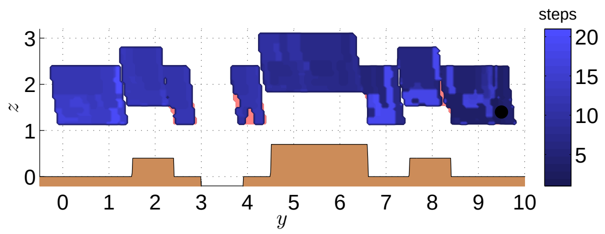

Fig. 16. Comparison of the domain of attraction with and without wind noise for $\dot{y}_0 = 0.5m/s$. Light red shaded patches (less than 1% of 4352 initial conditions in the global domain) show where the controller under wind noise fails to reach the global goal $\mathbf{Z}_g = [9.5, 1.3, 0]$ shown with a black dot. Darker shades illustrate the number of steps before reaching the goal.

To generalize, Figure 16 illustrates the effect of wind noise on the global domain of attraction. More than %99 of the 4352 initial conditions (with velocity $\dot{y}_0 = 0.5m/s$) in the domain successfully reach the goal, with failures primarily due to either a collision with the ground of falling into the hole, showing the robustness of our reactive planning framework despite substantial modeling inaccuracies.

Finally, Figure 17 illustrates the global domain of attraction under sensor noise in apex state measurements, uniformly distributed in $[-5, 5]cm$ for positional and $[-5, 5]cm/s$ for velocity variables. The magnitude of this noise is actually larger than safety margins associated with most intermediate goals and results in the hopper going through sequences of steps that are different than what the planner had anticipated. This is why even neighboring states go through different number of steps to reach the goal. Nevertheless, more than %99 of the 4352 initial conditions in the domain successfully reach the goal, showing that the reactive control strategy is robust to sensor noise.

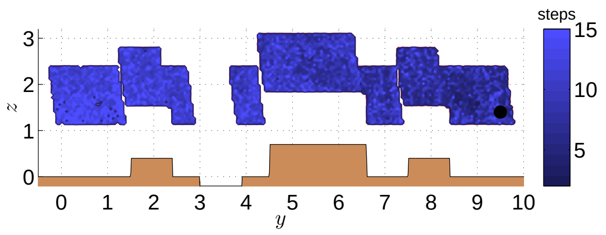

Fig. 17. Domain of attraction with and without sensor noise (uniformly distributed in $[-5, 5]cm$ for positional and $[-5, 5]cm/s$ for velocity variables) for $\dot{y}_0 = 0.5m/s$. Light red shaded patches (less than 1% of 4352 initial conditions in the global domain) show where the controller under sensor noise fails to reach the global goal $\mathbf{Z}_g = [9.5, 1.3, 0]$ shown with a black dot. Darker shades illustrate the number of steps before reaching the goal.

VII. Running with the Spring-Mass Hopper

In this section, we demonstrate how our reactive footstep planning framework can be used for the biologically and practically more relevant SLIP model. Surprisingly, we will be able to use policy deployments for the BHop model for the SLIP model, relying on the inherent reactivity and robustness of the framework to compensate for the discrepancies between the BHop and SLIP models.

A. Using Ball-Hopper Plans for the SLIP Model

By construction, control inputs available to the ball hopper closely correspond to those available to the SLIP model described in Section IV-A1. We need to, however, make sure that the allowable control inputs $\mathcal{U}$ used for BHop policy templates in (4) are consistent and feasible for the SLIP deadbeat controller described in Section IV-A2.

The touchdown angle control inputs θ and θ_{td} for the BHop and SLIP models are already in correspondence to each other. Similarly, the energy gain input k for BHop can be realized through the ratio of compression and decompression spring constants k_c and k_d for the SLIP model, with $k \in [0.7, 1.4]$ corresponding to the range $k_d/k_c \in [0.5, 2]$. On the other hand, the horizontal displacement during the stance phase of the SLIP model, despite being controllable through k_c, cannot be chosen as freely as the corresponding control input, Δy. In particular, the allowable range for this control input depends on the average angular momentum for the SLIP model during stance and hence has a much smaller range for low velocities.

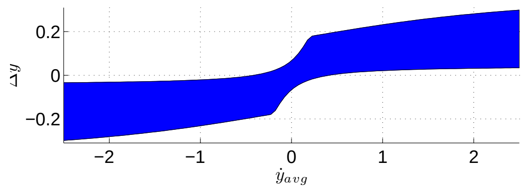

Fig. 18. Adjusted horizontal shift limits for BHop policy templates to be used for the SLIP model as a function of the average horizontal velocity $\dot{y}_{avg} := (\dot{y}_k + \dot{y}_{k+1})/2$.

We account for this discrepancy between control inputs by using velocity dependent limits on the Δy parameter for the BHop model that roughly capture the constraints of the SLIP system. In particular, we impose limits on the horizontal shift as

$$\Delta y \in [\, h(\dot{y}_{avg}),\ -h(-\dot{y}_{avg}) \,] \,, \tag{35}$$

where $\dot{y}_{avg} := (\dot{y}_k + \dot{y}_{k+1})/2$ is the average of the current and next apex velocities and $h(v)$ is a velocity dependent function defined as

$$h(v) := \begin{cases} C_m(\operatorname{atan}(C_a(v + C_o))/\pi - (1 - C_o)) & \text{if } v < C_o \\ C_m(\operatorname{atan}(C_b(v + C_o))/\pi - (1 - C_o)) & \text{if } v \geq C_o \end{cases} .$$

Manual tuning with $C_m = 1.11$, $C_a = 1$, $C_b = 10$ and $C_o = -0.1$ yields the limits shown in Figure 18, approximately capturing constraints on the range of horizontal displacement that can be accomplished by the SLIP model. Even though this is not, by any means, an exact representation of control input constraints for the SLIP model, it enables BHop policy instances to capture the capabilities of a SLIP stride reasonably accurately, with the reactive nature of our planner compensating for remaining errors.

In addition to this difference in the horizontal shift control input, the SLIP model also differs from the BHop model in its kinematics of touchdown. The horizontal position of the toe for the SLIP model depends on the touchdown angle control θ_{td} and is either in front of or behind the body. The policy domain derivations of Appendix A must be modified to take this discrepancy into account. We do this approximately through a velocity dependent average adjustment on the horizontal position range for the domain as

$$R_y^{\text{SLIP}}(\Phi, \dot{y}, z) := \left[\; -l - \dot{y}\sqrt{2z/g} - Y, \; l - \dot{y}\sqrt{2z/g} - Y \; \right] \quad (36)$$

$$Y := \dot{y}\sin(\theta_{\text{avg}})/|\dot{y}_{max}| \; . \quad (37)$$

where $\theta_{\text{avg}} = 0.2$ is a manually tuned, average touchdown angle for the SLIP model running with $\dot{y} = \dot{y}_{\text{max}}$. The derivations for the goal region adjusted accordingly.

B. Simulation Environment

As in Section VI-A, we compute SLIP trajectories through numerical integration of its dynamics. At each apex, we use the execution controller of Figure 8 on the policy ordering computed for the BHop model to determine the highest priority stride policy instance $\Phi^{\mathbf{P}}[\mathbf{Z}_i]$ that includes the current apex state. We then invoke the deadbeat controller for the SLIP model, described in Section IV-A2 to compute the best control input to reach the associated goal state $\mathbf{Z}_i$. This execution loop continues until either the goal state is reached (up to a certain error margin) or locomotion failure with a ground collision.

C. Planning Performance with the SLIP Model

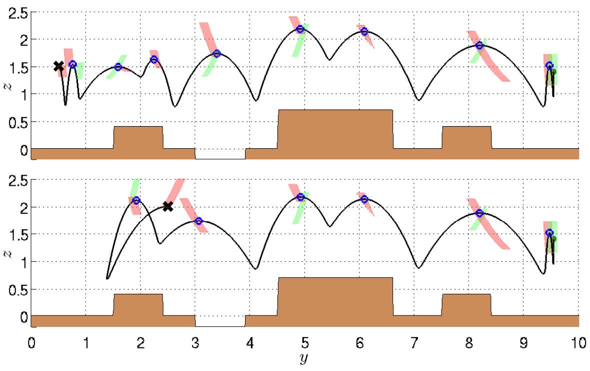

Fig. 19. Two example runs for BHop reactive plans used for SLIP running, started from different initial conditions but using the same reactive policy deployment towards the goal $\mathbf{Z}_g = [9.5, 1.3, 0]$ without any explicit replanning. Light green and dark red shaded regions illustrate cross sections of feasible goal and domain regions at each apex, respectively.

Figure 19 shows two example runs with the SLIP model using a reactive policy deployment based on the BHop model and the corrections described in Section VII-A. These examples have the same goals and initial conditions as those presented in Section VI-B for convenient comparison. In both cases, the SLIP model successfully reaches the global goal, reactively choosing new policies when the deadbeat controller is not capable of exactly realizing the intermediate policy goal settings deployed for the BHop model.

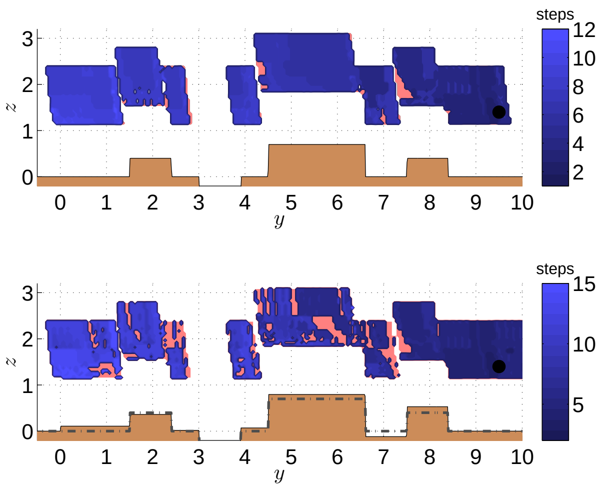

Fig. 20. Comparison of the domains of attraction for BHop and SLIP models without additional noise (top) and ground noise (bottom) for $\dot{y}_0 = 0.5m/s$. Light red shaded patches (top: 4%, bottom: %19 of 4352 initial conditions in the global domain) show where the SLIP model fails to reach the global goal $\mathbf{Z}_g = [9.5, 1.3, 0]$ shown with a black dot.

As a more general measure of robustness, the top plot of Figure 20 shows convergence behavior of the SLIP model under our reactive planner for initial conditions within the global domain region with $\dot{y}_a = 0.5m/s$. %96 of all initial conditions converge to the goal even though the SLIP model dynamics are significantly different from the idealized BHop dynamics. This is a direct consequence of the reactive nature of the planner.

In contrast, the bottom plot of Figure 20 shows the performance of the reactive planner under the same ground sensing noise imposed on the BHop model in Section VI-C. Even under the large missed foothold disturbances under such ground sensing noise, %81 of all 4352 initial conditions in the ideal domain of attraction still converge to the goal, with gaps in the domain resulting from certain policy sequences leading to unexpected collisions with the ground.

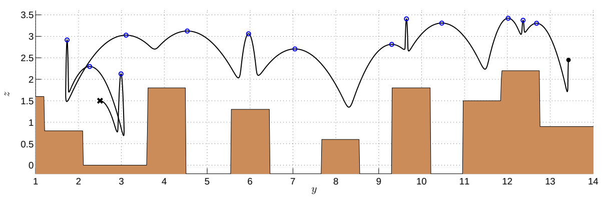

Fig. 21. SLIP running across a longer and more challenging terrain with the reactive planner. Initial and final apex states are marked with an x and a dot, respectively. We have included a supplementary MPEG file in our multimedia submission with an animation of the hopper across this surface.

Finally, Figure 21 illustrates reactive footstep planning with the SLIP model across a more challenging terrain profile. Starting from a deep well, the controller first realizes it has to step backwards in order to get on top of the hill at $y = 4$. At every apex, the policy deployment informs the controller what goal should be pursued next, with no explicit replanning done even though these goals are computed for the BHop model and the SLIP deadbeat controller can never exactly realize them. Despite this purely reactive control strategy, the SLIP model still undergoes seemingly deliberate stepping patterns such as the small hesitation steps around $y = 6$ and $y = 10$.

VIII. Conclusion and Future Work

In this paper, we presented a novel framework for reactive planning and control of planar monopedal running across rough terrain. Our approach is based on a uniform characterization of the preconditions (domain) and postconditions (feasible goal) of a single running stride, resulting in the definition of stride policies as sufficiently rich abstractions of running steps. Our planning framework then uses these policies within a sequential composition formalism to create a purely reactive controller that has a large domain of attraction from which the goal point is guaranteed to be reached.

We showed the validity and performance of this simultaneous planning and control framework through its application on a running model with simplified stance dynamics that capture relevant aspects of more accurate but complex models. The resulting controller was shown in simulation to be robust to different forms of noise and large disturbances. We also applied plans constructed for this simplified model to the much more complex but accurate SLIP model with only minor modifications on control input constraints and showed that convergence to the goal is still achieved with large domains of attraction even in the presence of additional noise.

A number of important challenges remain before an experimental realization of the proposed framework becomes possible. Most importantly, even though the three independent control inputs required by the ideal SLIP model are accurate descriptors of natural runners, they are difficult to instantiate in robot platforms. Most monopedal running robots built to date have been able to implement two of these three control inputs, with only Raibert's pneumatically actuated hoppers and the BiMASC platforms capable of independent control over all three. Consequently, such underactuated platforms would need to rely on a reformulation of the stride abstraction to focus on only two of these states (only positional variables, for example), rather than performing planning on all three, fully actuated dimensions of apex states. Unfortunately, once full controllability in apex states is sacrificed, the prepares relation becomes much more constrained, resulting in a much more sparse distribution of applicable policies. Nevertheless, we consider both the construction of a suitable platform and the adaptation of our framework to lower dimensional stride abstractions to be the natural continuations of this work.

One of the possible extensions of this planning framework is the incremental construction of the prepares graph in an online setting with limited-range sensing of the ground profile. In such cases, new policies could be situated as new data is received, with both the prepares graph and the deployment updated incrementally. This would require the deployment algorithm to rely not only on backchaining, but rather a combination of forward and backward chaining to incorporate new policy instances ahead of the existing deployment. We foresee realistic applications of this method to have such an incremental character.

We believe that our contributions in this paper, including the validity of the ball hopper model for capturing relevant aspects of SLIP locomotion, as well as the application of the sequential composition principles to achieve reactive footstep planning, show that footstep planning for rough terrain traversal and purely reactive control strategies need not be mutually exclusive. They can be combined to yield robust control strategies with large domains of attraction at least within mathematical models that were shown to be accurate with respect to both natural and artificial runners.

Appendix A
Domain Regions for the Ball-Hopper

We start the analytic derivation of the domain region for a stride policy template for the BHop model by assuming that the velocity and energy ranges are specified as closed, bounded and connected intervals of the real line with

$$R_V(\Phi) := [\ \dot{y}_l^{\mathcal{D}},\ \dot{y}_u^{\mathcal{D}}\ ] \quad \text{and} \quad R_E(\Phi) := [\ E_l,\ E_u\ ]\ . \tag{38}$$

Given a particular forward velocity $\dot{y} \in R_V(\Phi)$, the energy constraint yields boundaries for the apex height as

$$R_z(\Phi, \dot{y}) := \left[\ (E_l - 0.5m\dot{y}^2),\ (E_u - 0.5m\dot{y}^2)\ \right] / (mg)\ . \tag{39}$$

Every height value in this range gives rise to an associated range in the horizontal position coordinates as

$$R_y(\Phi, \dot{y}, z) := \left[\ (-l - \dot{y}\sqrt{2z/g}),\ (l - \dot{y}\sqrt{2z/g})\ \right]\ . \tag{40}$$

All together, these constraints lead to an analytic formulation of the policy domain region for the ball hopper system as

$$\mathcal{D}_{bh}(\Phi) := \Big\{ \mathbf{Z} = [\ y,\ z,\ \dot{y}\ ]^T \mid \dot{y} \in R_V(\Phi), z \in R_z(\Phi, \dot{y}), \\ y \in R_y(\Phi, \dot{y}, z)\ \}\ . \tag{41}$$

Appendix B
Feasible Goal Regions for the Ball-Hopper

In order to simplify analytic formulation of the feasible goal region, we first assume that the set of allowable control inputs has the form

$$\mathcal{U} := \Big\{ [\Delta y, \theta, k]^T \mid \theta \in [\theta_l, \theta_u], k \in [k_l, k_u], \Delta y \in [d_l, d_u] \Big\} \tag{42}$$

with simple interval constraints for each component. We first compute the boundaries of the feasible goal region $\mathcal{G}_f(\Phi)$ in the velocity dimension. Based on (6), analysis of the liftoff velocity in (18) and its dependence on both the control inputs and initial states yields

$$\dot{y}_l^{\mathcal{G}_f} = -0.5(1+k_u)\sqrt{2E_l/m} + 0.5(1-k_u)\dot{y}_l^{\mathcal{D}} \tag{43}$$
$$\dot{y}_u^{\mathcal{G}_f} = 0.5(1+k_u)\sqrt{2E_l/m} + 0.5(1-k_u)\dot{y}_u^{\mathcal{D}}\ . \tag{44}$$

Given $\dot{y}_a \in [\dot{y}_l^{\mathcal{G}_f}, \dot{y}_u^{\mathcal{G}_f}]$, we solve for the touchdown angle

$$\tan(\theta) = \frac{(1+k)\dot{z}_{td} + \sqrt{(1+k)^2\dot{z}_{td}^2 - 4(\dot{y}_a + k\dot{y}_{td})(\dot{y}_a - \dot{y}_{td})}}{2(\dot{y}_a + k\dot{y}_{td})} \tag{45}$$

and corresponding vertical liftoff velocity, $\dot{z}_{lo}$

$$\dot{z}_{lo} = 0.5(1+k)\left(\frac{2x^*}{1+x^{*2}}\dot{y}_{td} - \frac{1-x^{*2}}{1+x^{*2}}\dot{z}_{td}\right) + 0.5(1-k)\dot{z}_{td} \tag{46}$$

as functions of k and initial states. Inspection of this solution and the return map (with some effort) reveals that the lower and upper boundaries of the goal region coincide with the maximum and minimum energy levels, respectively and one of the two velocity boundaries of the domain region. Consequently, the height boundaries for the feasible goal region for a given velocity are given by

$$z_l^{\mathcal{G}_f}(\dot{y}_a) = \min(\dot{z}_{lo}(E_u, \dot{y}_l^{\mathcal{D}}), \dot{z}_{lo}(E_u, \dot{y}_u^{\mathcal{D}}))/(2g) \tag{47}$$
$$z_u^{\mathcal{G}_f}(\dot{y}_a) = \max(\dot{z}_{lo}(E_l, \dot{y}_l^{\mathcal{D}}), \dot{z}_{lo}(E_l, \dot{y}_u^{\mathcal{D}}))/(2g)\ . \tag{48}$$

Finally, for a given apex velocity and height within the intervals determined above, the horizontal position boundaries for the feasible goal region can be computed as

$$y_l^{\mathcal{G}_f}(\dot{y}_a, z_a) = l + \dot{y}_a\sqrt{2z_a/g} + d_l, \tag{49}$$
$$y_u^{\mathcal{G}_f}(\dot{y}_a, z_a) = -l + \dot{y}_a\sqrt{2z_a/g} + d_u. \tag{50}$$

Acknowledgment

We thank Prof. Ömer Morgül for his insights and support. This work was partially supported by the Scientific and Technological Research Council of Turkey (TUBITAK) project 109E032.

References

[1] M. M. Ankarali and U. Saranli. Stride-to-stride energy regulation for robust self-stability of a torque-actuated dissipative spring-mass hopper. *Chaos*, 20(033121), September 2010.

[2] M. M. Ankarali and U. Saranli. Control of underactuated planar pronking through anembedded spring-mass hopper template. *Autonomous Robots*, 30(2):217–231, 2011. 10.1007/s10514-010-9216-x.

[3] O. Arslan, U. Saranli, and O. Morgul. Reactive footstep planning for a planar spring mass hopper. In *Proc. of the Int. Conf. on Intelligent Robots and Systems*, pages 160–166, St. Louis, MI, October 2009.

[4] Y. Ayaz, T. Owa, T. Tsujita, A. Konno, K. Munawar, and M. Uchiyama. Footstep planning for humanoid robots among obstacles of various types. In *Proc. of the IEEE-RAS Conf. on Humanoid Robots*, pages 361 –366, Dec. 2009.

[5] R. Blickhan and R. J. Full. Similarity in multilegged locomotion: Bouncing like a monopode. *Journal of Comparative Physiology A: Neuroethology, Sensory, Neural, and Behavioral Physiology*, 173(5):509–517, November 1993.

[6] R. R. Burridge, A. A. Rizzi, and D. E. Koditschek. Sequential composition of dynamically dexterous robot behaviors. *The Int. J. of Robotics Research*, 18(6):534–555, 1999.

[7] K. Byl and R. Tedrake. Approximate optimal control of the compass gait on rough terrain. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 1258 –1263, May 2008.

[8] K. Byl and R. Tedrake. Metastable Walking Machines. *The Int. J. of Robotics Research*, 28(8):1040–1064, August 2009.

[9] S. Carver. *Control of a Spring-Mass Hopper*. Ph.d., Cornell University, January 2003.

[10] E. Celaya and J. Porta. A control structure for the locomotion of a legged robot on difficult terrain. *IEEE Robotics Automation Magazine*, 5(2):43 –51, June 1998.

[11] J. G. Cham, S. A. Bailey, J. E. Clark, R. J. Full, and M. R. Cutkosky. Fast and robust: Hexapedal robots via shape deposition manufacturing. *The Int. J. of Robotics Research*, 21(10):869–882, 2002.

[12] J. Chestnutt and J. Kuffner. A tiered planning strategy for biped navigation. In *Proc. of the IEEE-RAS Conf. on Humanoid Robots*, November 2004.

[13] J. Chestnutt, J. Kuffner, K. Nishiwaki, and S. Kagami. Planning biped navigation strategies in complex environments. In *Proc. of the Int. Conf. on Humanoid Robots*, October 2003.

[14] C. Chevallereau, J. W. Grizzle, and C.-L. Shih. Asymptotically stable walking of a five-link underactuated 3d bipedal robot. *IEEE Trans. on Robotics*, 25(1):37–50, February 2009.

[15] C.-M. Chew, J. Pratt, and G. Pratt. Blind walking of a planar bipedal robot on sloped terrain. In *Proc. of the Int. Conf. on Robotics and Automation*, volume 1, pages 381 –386 vol.1, 1999.

[16] D. Conner, H. Choset, and A. Rizzi. Flow-through policies for hybrid controller synthesis applied to fully actuated systems. *IEEE Trans. on Robotics*, 25(1):136 –146, feb. 2009.

[17] D. C. Conner. *Integrating Planning and Control for Constrained Dynamical Systems*. PhD thesis, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, January 2008.

[18] T. Erez and W. Smart. Bipedal walking on rough terrain using manifold control. In *Proc. of the Int. Conf. on Intelligent Robots and Systems*, pages 1539–1544, 29 2007-Nov. 2 2007.

[19] Y. Fukuoka, H. Kimura, and A. H. Cohen. Adaptive dynamic walking of a quadruped robot on irregular terrain based on biological concepts. *The Int. J. of Robotics Research*, 22(3-4):187–202, 2003.

[20] R. J. Full and D. E. Koditschek. Templates and anchors: Neuromechanical hypotheses of legged locomotion. *Journal of Experimental Biology*, 202:3325–3332, 1999.

[21] A. Goswami, B. Espiau, and A. Keramane. Limit cycles and their stability in a passive bipedal gait. In *Proc. of the Int. Conf. on Robotics and Automation*, volume 1, pages 246 –251 vol.1, Apr. 1996.

[22] C. Grand, F. Benamar, F. Plumet, and P. Bidaud. Stability and Traction Optimization of a Reconfigurable Wheel-Legged Robot. *The Int. J. of Robotics Research*, 23(10-11):1041–1058, 2004.

[23] R. Gregg, T. Bretl, and M. Spong. Asymptotically stable gait primitives for planning dynamic bipedal locomotion in three dimensions. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 1695 –1702, May 2010.

[24] J. K. Hodgins and M. N. Raibert. Adjusting step length for rough terrain locomotion. *IEEE Trans. on Robotics and Automation*, 7(3):289–298, June 1991.

[25] Q. Huang, K. Yokoi, S. Kajita, K. Kaneko, H. Arai, N. Koyachi, and K. Tanie. Planning walking patterns for a biped robot. *IEEE Trans. on Robotics and Automation*, 17(3):280 –289, June 2001.

[26] J. W. Hurst, J. E. Chestnutt, and A. A. Rizzi. Design and philosophy of the bimasc, a highly dynamic biped. In *Proceedings of the Int. Conf. on Robotics and Automation*, pages 1863–1868, April 10-14 2007.

[27] F. Iida and R. Tedrake. Minimalistic control of a compass gait robot in rough terrain. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 1985 –1990, May 2009.

[28] S. Kajita and K. Tani. Adaptive gait control of a biped robot based on realtime sensing of the ground profile. *Autonomous Robots*, 4:297–305, 1997. 10.1023/A:1008848227206.

[29] J. Kolter, M. Rodgers, and A. Ng. A control architecture for quadruped locomotion over rough terrain. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 811 –818, May 2008.

[30] J. Kuffner, J.J., K. Nishiwaki, S. Kagami, M. Inaba, and H. Inoue. Footstep planning among obstacles for biped robots. In *Proc. of the Int. Conf. on Intelligent Robots and Systems*, volume 1, pages 500–505, 2001.

[31] I. R. Manchester, U. Mettin, F. Iida, and R. Tedrake. Stable dynamic walking over uneven terrain. *The Int. J. of Robotics Research*, 2011.

[32] R. B. Mcghee and G. I. Iswandhi. Adaptive locomotion of a multilegged robot over rough terrain. *IEEE Trans. on Systems, Man and Cybernetics*, 9(4):176 –182, April 1979.

[33] D. Messuri and C. Klein. Automatic body regulation for maintaining stability of a legged vehicle during rough-terrain locomotion. *IEEE J. of Robotics and Automation*, 1(3):132 – 141, Sept. 1985.

[34] P. Michel, J. Chestnutt, J. Kuffner, and T. Kanade. Vision-guided humanoid footstep planning for dynamic environments. In *Proc. of the IEEE-RAS Conf. on Humanoid Robots*, pages 13 – 18, December 2005.

[35] D. Pai and L.-M. Reissell. Multiresolution rough terrain motion planning. *IEEE Trans. on Robotics and Automation*, 14(1):19 –33, Feb. 1998.

[36] I. Poulakakis, E. Papadopoulos, and M. Buehler. On the Stability of the Passive Dynamics of Quadrupedal Running with a Bounding Gait. *The Int. J. of Robotics Research*, 25(7):669–687, 2006.

[37] M. Raibert. *Legged robots that balance*. MIT Press series in artificial intelligence. MIT Press, Boston, 1986.

[38] S. Ramamoorthy and B. Kuipers. Qualitative hybrid control of dynamic bipedal walking. In G. S. Sukhatme, S. Schaal, W. Burgard, and D. Fox, editors, *Robotics: Science and Systems*. The MIT Press, 2006.

[39] J. Rebula, P. Neuhaus, B. Bonnlander, M. Johnson, and J. Pratt. A controller for the littledog quadruped walking on rough terrain. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 1467 –1473, April 2007.

[40] A. A. Rizzi, J. Gowdy, and R. L. Hollis. Distributed coordination in modular precision assembly systems. *The Int. J. of Robotics Research*, 20(10):819–838, 2001.

[41] U. Saranli, O. Arslan, M. M. Ankarali, and O. Morgul. Approximate analytic solutions to non-symmetric stance trajectories of the passive spring-loaded inverted pendulum with damping. *Nonlinear Dynamics*, 62(4):729–742, 2010.

[42] U. Saranli, M. Buehler, and D. E. Koditschek. RHex: A simple and highly mobile robot. *The Int. J. of Robotics Research*, 20(7):616–631, July 2001.

[43] P. S. Schenker, T. L. Huntsberger, P. Pirjanian, E. T. Baumgartner, and E. Tunstel. Planetary rover developments supporting mars exploration, sample return and future human-robotic colonization. *Autonomous Robots*, 14(2):103–126, Mar. 2003.

[44] W. J. Schwind. *Spring Loaded Inverted Pendulum Running: A Plant Model*. Phd, University of Michigan, 1998.

[45] D. Wooden, M. Malchano, K. Blankespoor, A. Howardy, A. A. Rizzi, and M. Raibert. Autonomous navigation for BigDog. In *Proc. of the Int. Conf. on Robotics and Automation*, May 3-8 2010.

[46] B. M. Yamauchi. Packbot: a versatile platform for military robotics. In *Proc. of SPIE: Unmanned Ground Vehicle Technology VI*, volume 5422, pages 228–237, September 2004.

[47] T. Yang, E. Westervelt, A. Serrani, and J. Schmiedeler. A framework for the control of stable aperiodic walking inunderactuated planar bipeds. *Autonomous Robots*, 27:277–290, 2009.

[48] K. Yoneda, H. Iiyama, and S. Hirose. Sky-hook suspension control of a quadruped walking vehicle. In *Proc. of the Int. Conf. on Robotics and Automation*, pages 999 –1004 vol.2, May 1994.

[49] G. Zeglin. *The Bow Leg Hopping Robot*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA, USA, October 1999.