Reactive Footstep Planning for a Planar Spring Mass Hopper

Ömür Arslan, Uluç Saranlı and Ömer Morgül

Abstract— **The main driving force behind research 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 against model uncertainty and environment noise. In this paper, we introduce a new method for dynamic, fully reactive footstep planning for a simplified planar spring-mass hopper, a frequently used model for running behaviors. Our approach is based on a careful characterization of the model dynamics and an associated deadbeat controller, used within a sequential composition framework. This yields a purely reactive controller with a very large, nearly global domain of attraction that requires no explicit replanning during execution. Finally, we use a simplified hopper in simulation to illustrate the performance of the planner under different rough terrain scenarios and show that it is extremely robust to both model uncertainty and measurement noise.**

I. INTRODUCTION

Legged morphologies have always been considered necessary to achieve dynamic, robust and autonomous traversal of complex, outdoor terrain. Despite effective behaviors and performance demonstrated by tracked vehicles [19] and flexible multi-wheeled platforms [16], the pallet of behaviors realizable with such morphologies inevitably remains limited due to restricted directions in which forces can be applied to the robot body. On the other hand, while legged designs do not suffer from such limitations [15], their robust and maneuverable control on complex terrain is still a largely unsolved problem. Traditional approaches which perform planning and control separately do not perform well in the presence of model uncertainty and measurement noise. In contrast, existing reactive control methods often make rigid assumptions about their environment (e.g. flat ground or single obstacle of known size) and do not offer the scalability necessary for deployment on real-life problems.

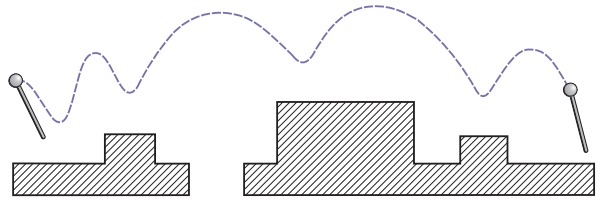

Fig. 1. A spring-mass hopper running over rough terrain.

Ö.Arslan is with the Dept. of Electrical & Electronics Eng., Bilkent University, 06800 Ankara, Turkey omur@ee.bilkent.edu.tr

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

Ö.Morgül is with the Dept. of Electrical & Electronics Eng., Bilkent University, 06800 Ankara, Turkey morgul@ee.bilkent.edu.tr

In this paper, we propose a novel algorithm to address these issues for the specific problem of purely reactive control and footstep planning for a simplified planar spring-mass hopper running on rough terrain, illustrated in Fig. 1. Our focus on the planar hopper is founded on the success of the well-known Spring-Loaded Inverted Pendulum (SLIP) model [17] both in accurately describing runners in nature [3] and in providing morphological inspiration and a high-level control interface to a family of robot runners [1, 11, 13]. Consequently, the contribution of a robust control and planning framework for this model promises to be applicable to a variety of robot sizes and morphologies ranging from monopedal and bipedal runners to hexapedal robots.

Motion planning for locomotion on rough terrain has been a topic of interest since the first days of legged robots. With controllers that regulate step-lengths, Raibert's bipeds [13] have been able to traverse both flat terrain with "holes" as well as terrain with significant height variations. Their method relied on preprocessing of the terrain structure to identify specific footholds in the planning step and used the execution controller to achieve the constructed plan, resulting in significant sensitivity to modeling uncertainty. A similar planning framework was also investigated in [20], particularly as it applies to the Bow-Leg platform. The proposed solutions still remained non-reactive with explicit replanning performed upon detection of plan failure. More recently, footstep planning for bipeds in complex environments received considerable attention with the availability of quasi-static but well actuated humanoid robots. Footstep planning for such platforms can rely on a kinematic characterization of their stepping patterns [12]. Since movements of such robots are usually rather slow, discrete abstractions of action sequences combined with search algorithms, possibly with replanning for dynamic or unpredictable environments, suffice to achieve reasonable performance [5, 6]. Unfortunately, for systems that must rely on their second order dynamics, either due to underactuation or to achieve high speeds, such kinematic methods quickly become inapplicable.

The presence of non-negligible second-order dynamics inevitably brings the need for reactivity since models for such systems are much less accurate. One of the most successful methods in integrating deliberate planning with reactivity for dynamically dexterous robots is the Sequential Composition, first introduced in the context of juggling [4] and later applied to other platforms such as planar mobile robots with different actuation modalities [7–9] and the Minifactory [14]. Sequential composition characterizes dynamic behaviors for a robotic system through their invariant domains and goal sets in the state space, ensuring proper activation order through a

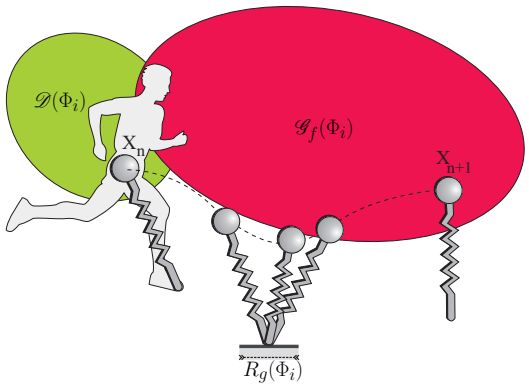

Fig. 2. An illustration of ground support $R_g(\Phi_i)$, policy domain $\mathscr{D}(\Phi_i)$, and feasible goal $\mathscr{G}_f(\Phi_1)$ regions for the spring mass hopper.

prioritization combined with reactive decision-making. Our approach is largely based on these ideas but deviates in our formulation of behavioral primitives and associated domain and 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 hoppers, as well as the application of resulting ideas to a specific, simplified hopper model supported by an analytical characterization of its apex states reachable from specific regions of allowable footholds.

II. PLANNING FRAMEWORK

A. A Generic Hopper Model

Running trajectories for all one or two legged systems exhibit a common structure: They alternatingly go through *flight* and *stance* phases, separated by *touchdown* and *liftoff* events as the foot comes into contact and leaves the ground, respectively. It is also useful to define an *apex* event associated with the highest point of the center of mass (COM) during flight, whose height and forward velocity are often used as a representative state vector for the subsequent stride. In this section, we make as few assumptions as possible about the underlying system beyond this structure in order to ensure general applicability of our framework.

Throughout this paper, we assume that a planar one-legged hopper is running on a piecewise flat ground (e.g. Fig. 1), possibly with a number of "holes" on which 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. We also assume that gait control is achieved with per-step control inputs selected at each apex (but possibly realized throughout the entirety of the following flight and stance phases), allowing independent, possibly limited control of all three degrees of freedom for the next apex. This framework, for which a single stride is illustrated in Fig. 2, is consistent with most planar running robot morphologies in the literature ranging from the SLIP model to more complex, multi-jointed leg designs.

We associate with each ground segment, one or more single-step "local" families of control policies Φ_i that use that segment for their foothold during stance, while using control inputs from a constrained set $U(\Phi_i)$ to take the robot to an associated set of possible apex states. Our planning algorithm seeks to find a particular reactive sequencing of these policies to ensure that the robot is driven to a desired goal state from as large a set of initial conditions as possible.

In the spirit of sequential composition, we associate with each family of policies Φ_i, a *domain* $\mathscr{D}(\Phi_i)$, including those apex states from which the corresponding ground segment $R_g(\Phi_i)$ is reachable, as well as a *feasible goal set* $\mathscr{G}_f(\Phi_i)$ including only apex states that are achievable from every state within the domain within a single stride using allowable control inputs. In the sequel, we will use $X_n := \{y_n,\ z_n,\ \dot{y}_n,\ 0\}$ to denote the state of the hopper at the n^{th} apex event, and $F_a(X_n, u)$ to denote the apex return map that under a specific control input u. Formal definitions of the domain and feasible goal sets hence take the form

$$\begin{aligned}\mathscr{D}(\Phi_i) := \{X_n \mid\ & \dot{y}_n \in R_V(\Phi_i),\ E_n \in R_E(\Phi_i),\\ & \forall u \in U(\Phi_i).\ y_{f,td}(X_n, u) \in R_g(\Phi_i),\\ & T_D(X_n, \Phi_i) \subset \mathcal{FS}\ \}\ , \end{aligned} \tag{1}$$

$$\begin{aligned}\mathscr{G}_f(\Phi_i) := \{X_{n+1} \mid\ & \forall X_n \in \mathscr{D}(\Phi_i),\ \exists u \in U(\Phi_i).\\ & X_{n+1} = F_a(X_n, u),\\ & T_A(X_{n+1}, \Phi_i) \subset \mathcal{FS}\}\ , \end{aligned} \tag{2}$$

where the sets $R_V(\Phi_i)$ and $R_E(\Phi_i)$ are allowed ranges for the velocity and energy of the initial apex state, $U(\Phi_i)$ is the allowable set of control inputs, and $R_g(\Phi_i)$ denotes the ground segment associated with the policy on which the hopper will land as shown in Fig. 2. In order to prevent collisions with the ground we also require descent and ascent trajectories, denoted with $T_D(X_n, \Phi_i)$ and $T_A(X_{n+1}, \Phi_i)$, to be contained in free space $\mathcal{FS}$.

Intuitively, $\mathscr{D}(\Phi_i)$ captures apex states having energy and forward velocity values in the associated ranges $R_V(\Phi_i)$ and $R_E(\Phi_i)$ from where the hopper can reach the corresponding ground segment using available control inputs $u \in U(\Phi_i)$. In contrast, the feasible goal region $\mathscr{G}_f(\Phi_i)$, represents all apex states that are guaranteed to be reachable from the entire domain using whatever control input is necessary from within the allowable set $U(\Phi_i)$.

B. Reactive Footstep Planning

As described above, computation of the domain and feasible goal regions depends on the particulars of system dynamics. Nevertheless, once computed, they present a very convenient abstract interface between planning and control since the above construction ensures by design, the existence of a single-step controller that can take any apex state inside the domain to any subsequent apex state within the feasible goal region. In this section, we describe how to automatically construct a reactive, provably correct hybrid footstep controller that uses a given set of policies Φ_i.

1) The Prepares Relation: The sequential composition formalism introduced in [4] defines a relation between local controller policies that captures whether their sequencing is feasible or not. Their policy definitions include only a single goal point, whose inclusion in the domain of another

policy is sufficient to ensure the validity of sequencing. Our formulation of the goal set is more general, in the sense that the feasible goal set $\mathscr{G}_f(\Phi_i)$ includes an entire range of possible goal points that can be chosen to yield a particular policy instance. Consequently, we define a more general relation, *can prepare*, indicating the availability of a goal choice that can guarantee proper sequencing.

Definition 1: A policy Φ_i *can prepare* another policy Φ_j, denoted by $\Phi_i \succeq_c \Phi_j$, iff the following condition holds,

$$\mathscr{G}_f(\Phi_i) \cap \mathscr{D}(\Phi_j) \neq \emptyset \ .$$

This freedom in choice for the goal point associated with a policy allows the planner to consider optimality or safety criteria to increase the efficiency and robustness of the final reactive controller. Much like the original sequential composition algorithm, this relation induces a directed, possibly cyclic *can prepare graph* $\mathcal{G} := \{\Phi_i, \succeq_c\}$ between all policies. Before we proceed with the description of our planning algorithm, we will find the following definition useful.

Definition 2: A *policy instance* $\hat{\Phi}_i(X_g)$ is a controller that will take the hopper from an apex state $X_n \in \mathscr{D}(\Phi_i)$ and bring it to a specific, feasible goal point $X_{n+1} = X_g \in \mathscr{G}_f(\Phi_i)$ using an allowable control input $u \in U(\Phi_i)$ and stepping once on the ground range $R_g(\Phi_i)$ for the policy.

2) Planning by Prioritizing Policies: The formulation of the "can prepare graph" above captures all relevant sequencing constraints between different control policies. However, a robot running across rough terrain must still decide at every step which one of these policies will be used to determine proper control inputs for the next stride. The original sequential composition method divides this problem into two stages. First, the prepares graph is converted into a total order whose top element is chosen as a policy that can take the robot to the desired global goal. The resulting explicit prioritization of policies is then used at runtime to determine which policy should be used from among those whose domains cover the measured robot state. Even though different alternatives such as sequence-based and automata-based planners are possible [8], we adopt the order-based method, adapted to deal with larger, non-point goal sets as well as the discrete nature of our system.

Suppose a global goal is supplied to the planner in the form of a desired apex state X_g. Our algorithm starts by choosing goal policies Φ_j such that $X_g \in \mathscr{G}_f(\Phi_j)$ and instantiates them for the specific desired goal as $\hat{\Phi}_j(X_g)$. The algorithm then proceeds by backchaining on $\mathcal{G}$, incrementally building a total order of instantiated policies until all policy nodes in the graph are traversed. The instantiation of each policy chooses a single goal point which is both in its feasible goal set as well as the domain of the policy that it prepares, using a heuristic cost function that takes into account appropriate safety and efficiency criteria to determine the best candidate from among available goal alternatives.

Table I gives the detailed planning algorithm that yields the final policy ordering to be used for reactive control. Note that for each instantiated policy, the algorithm computes a specific goal point $\mathscr{G}_s(\hat{\Phi}_j)$, and a priority $\mathscr{P}(\hat{\Phi}_j)$ according to which the final total order will be obtained. The procedure $CostFunction(\Phi_j, \hat{\Phi}_i)$ is expected to return both the best candidate goal point for policy Φ_j and a cost related to how effectively it prepares the instantiated policy $\hat{\Phi}_i$.

TABLE I

ORDER BASED POLICY PRIORITY PLANNER

Algorithm Order_Based_Policy_Planner(X_g, $\mathcal{G}$)

PolicyFIFO := $\emptyset$
$\mathcal{GP} := FindGoalPolicies(X_g,\ \mathcal{G})$
for all $\hat{\Phi}_j \in \mathcal{GP}$ do
  $\mathscr{G}_s(\hat{\Phi}_j) := X_g$
  $\mathscr{P}(\hat{\Phi}_j) := 0$
  $Push(PolicyFIFO, \hat{\Phi}_j)$
endfor
while not(isempty(*PolicyFIFO*))
  $\hat{\Phi}_i := Pop(PolicyFIFO)$
  $\mathcal{PP} := FindPreparingPolicies(\hat{\Phi}_i, \mathcal{G})$
  for all $\Phi_j \in \mathcal{PP}$ do
    $[C_T, X_{g_T}] = CostFunction(\Phi_j, \hat{\Phi}_i)$
    if $(C_T + \mathscr{P}(\hat{\Phi}_i)) < \mathscr{P}(\Phi_j)$ then
      $\mathscr{G}_s(\hat{\Phi}_j) := X_{g_T}$
      $\mathscr{P}(\hat{\Phi}_j) := C_T + \mathscr{P}(\hat{\Phi}_i)$
      $Push(PolicyFIFO, \hat{\Phi}_j)$
    endif
  endfor
endwhile

return $\{\hat{\Phi}_{1,N}\}$

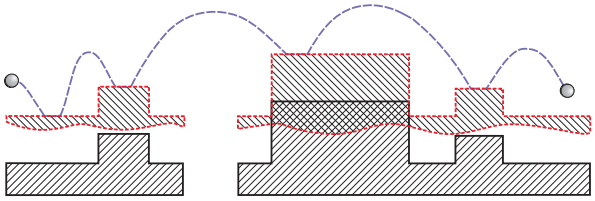

Fig. 3. An illustration of locomotion trajectories for the simplified "controllable ball" hopper model together with the "virtual ground" constructed from the scenario depicted in Fig. 1.

III. SIMPLIFIED HOPPER MODEL

Efficient, preferably analytic characterization of the domain and particularly the feasible goal regions for even the relatively well-studied SLIP model is a challenging, currently unsolved problem. Despite recent availability of very effective analytical maps for the stance dynamics of this model [2, 10, 18], numerical solutions are still needed for the characterization of the feasible goal region. Since our primary emphasis in this paper is the reactive planning framework, we will investigate the efficacy of our proposed method in a hopper model whose simplified dynamics will structurally mimic SLIP behavior, while admitting analytical characterization of the feasible goal region $\mathscr{G}_f(\Phi_i)$ for individual policies. This simplified model can be best described as a *controllable ball*, which will summarize the stance dynamics of the SLIP model with a "bounce" from a virtual surface elevated by a height equal to the SLIP spring rest length, changing the liftoff velocity and position of the body center of mass in a controllable fashion. Fig. 3 illustrates this idea for the scenario previously depicted in Fig. 1.

A. System Dynamics

During flight, the simplified hopper follows an uncontrollable ballistic trajectory, whose dynamics are given by

$$\dot{X} = \begin{bmatrix} \dot{y} & \dot{z} & \ddot{y} & \ddot{z} \end{bmatrix} = \begin{bmatrix} \dot{y} & \dot{z} & 0 & -g \end{bmatrix}$$

and are particularly important in computing both the domain and feasible goal regions for a given ground segment.

The stance phase for the SLIP model is governed by the compression and decompression of the leg spring until the liftoff event. For the simplified hopper, we capture this behavior by using a direct, "instantaneous" touchdown to liftoff map, controlled by a horizontal shift Δy, and adjustments θ and k on the angle and normal magnitude of the liftoff velocity with respect to a symmetric gait. These parameters have very close correspondence to control inputs frequently used for the SLIP model and associated robot morphologies:

- The liftoff velocity gain, k, roughly corresponds to the energy control for the SLIP model through the decompression and compression stiffness ratio k_d/k_c.
- The liftoff velocity angle adjustment closely corresponds to the touchdown leg angle for the SLIP model with respect to the neutral angle, $q_{\theta_t} - q_{\theta_n}$.
- The position shifting control, Δy, corresponds to the average stiffness of the SLIP leg, which can increase or decrease the positional span of the stance phase. In the SLIP model, this displacement nonlinearly depends on other control parameters, but can be independently chosen by adjusting k_c.

For example, a symmetric step can be obtained for the SLIP by choosing $k_c = k_d$ and $q_{\theta_t} = q_{\theta_n}$ with the horizontal liftoff position independently adjustable through the common stiffness value. For the simplified hopper, this corresponds to choosing $\theta = 0$ and $k = 1$ with Δy independently adjusting the horizontal displacement during stance.

The resulting stance map for the simplified hopper is hence

$$X_{lo} = AX_{td} + B \ , \tag{3}$$

where we define

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

$$B := \begin{bmatrix} \Delta y \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

B. Gait Control for the Simplified Hopper

The gait controller associated with each instantiated policy $\hat{\Phi}_i$ is responsible from finding control inputs necessary to bring the robot from any state $X_n \in \mathscr{D}(\hat{\Phi}_i)$ to the selected goal point X_g of $\hat{\Phi}_i$. To this end, we use a simple deadbeat controller for the simplified hopper, similar to those frequently used for the SLIP model, yielding reactive control inputs computed as $u = F_a^{-1}(X_n, X_g)$[1].

[1] $X_{n+1} := F_a(X_n, u)$ is the apex return map for a specific control u.

C. Derivation of the Domain Region

Before we proceed with an analytical representation of the domain region associated with a ground segment $R_g(\Phi_i)$, we add an additional constraint on the minimum apex height h_{min} to ensure that the leg can always clear the ground for protraction. This leads to a redefinition of the domain as

$$\mathscr{D}_{SH}(\Phi_i) := \{X_n | X_n \in \mathscr{D}(\Phi_i), z_n \geq h_{min}\}$$

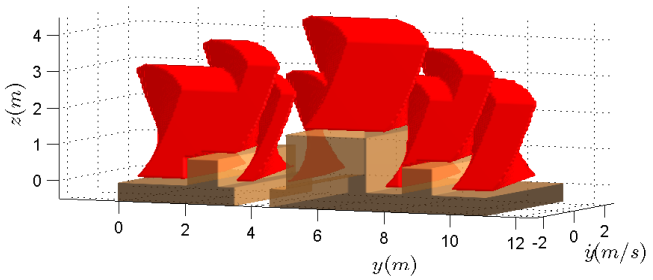

Fig. 4. Global domain coverage $\mathscr{D}_g := \bigcup_i \mathscr{D}(\Phi_i)$ for a planar rough surface, showing the union of all instantiated policy domains. Note that the depth axis represents the apex velocity.

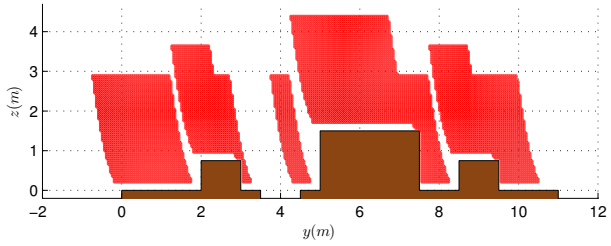

Fig. 5. A cross section of the global domain $\mathscr{D}_g$ at apex speed $\dot{y} = 1m/s$.

Apart from this adjustment, policy domains are simply constrained by the selected energy and velocity ranges together with the constraint of landing on the selected ground segment. For a given energy E and velocity $\dot{y}$, the ballistic flight trajectories yield simple expressions for the upper and lower limits of the horizontal position as

$$y_{min}(\dot{y}, E) = y_g - 0.5l_g - \dot{y}\sqrt{\frac{2(E - 0.5m\dot{y}^2 - mgz_g)}{mg}}$$

$$y_{max}(\dot{y}, E) = y_g + 0.5l_g - \dot{y}\sqrt{\frac{2(E - 0.5m\dot{y}^2 - mgz_g)}{mg}}$$

where (y_g, z_g) and l_g are the center and length of the ground segment associated with a policy.

The resulting simple analytical formulation yields a computationally efficient inclusion test for measured apex states. An illustration of the global domain of attraction $\mathscr{D}_g := \bigcup_i \mathscr{D}(\Phi_i)$ resulting from the deployment of such policies over the complex terrain shown in Fig. 3 is illustrated in Fig. 4. Similarly, Fig. 5 illustrates a cross section of the same domain at $\dot{y} = 1m/s$, showing positions from which the hopper can successfully recover from collision and find a foothold while travelling at $\dot{y} = 1m/s$.

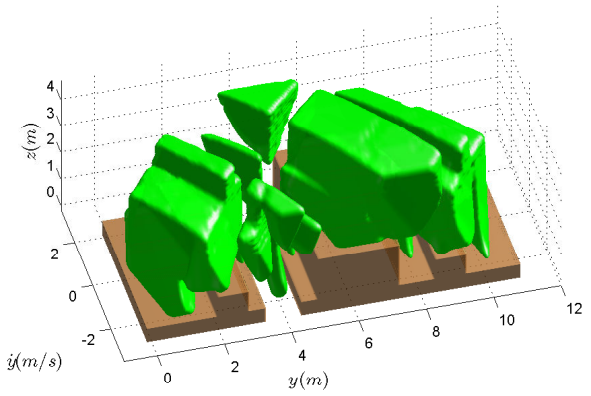

Fig. 6. Global goal coverage $\mathscr{G}_g := \mathscr{D}_g \bigcap (\bigcup_i \mathscr{G}_f(\Phi_i))$ over a planar rough surface, showing the union of feasible goal regions for all instantiated policies that are also inside the global domain. Note that the depth axis represents the apex velocity.

D. Derivation of the Feasible Goal Region

The feasible goal region for a policy includes all apex states reachable in a single step from every initial state in the domain, using only inputs from within the allowable set. In deriving a computational representation of this set, we proceed by analyzing each of the three dimensions in the apex state, namely the height z, the horizontal position y and the forward velocity $\dot{y}$. The following steps outline our method for finding the representation of $\mathscr{G}_f(\Phi_i)$ where we will omit analytical details for space considerations.

i) We first find the feasible z range, $[z_{min}, z_{max}]$, using analytical solutions to the equations

$$z_{min} = \underset{X_n \in \mathscr{D}(\Phi_i)}{\text{argmax}} \ (\underset{u \in U(\Phi_i)}{\text{argmin}} \ \pi_2 \circ F_a(X_n, u))$$
$$z_{max} = \underset{X_n \in \mathscr{D}(\Phi_i)}{\text{argmin}} \ (\underset{u \in U(\Phi_i)}{\text{argmax}} \ \pi_2 \circ F_a(X_n, u))$$

ii) Then, for any $z \in [z_{min}, z_{max}]$, we find the feasible y range, $[y_{min}, y_{max}]$, by analytically solving

$$y_{min}(z) = \underset{X_n \in \mathscr{D}(\Phi_i)}{\text{argmax}} \ (\underset{z_{n+1}=z, \ u \in U(\Phi_i)}{\text{argmin}} \ \pi_1 \circ F_a(X_n, u))$$
$$y_{max}(z) = \underset{X_n \in \mathscr{D}(\Phi_i)}{\text{argmin}} \ (\underset{z_{n+1}=z, \ u \in U(\Phi_i)}{\text{argmax}} \ \pi_1 \circ F_a(X_n, u))$$

iii) Finally, for any $z \in [z_{min}, z_{max}]$ and $y \in [y_{min}, y_{max}]$, we find the feasible velocity range, $[\dot{y}_{min}, \dot{y}_{max}]$, by

$$\dot{y}_{min}(z, y) = \underset{X_n \in \mathscr{D}(\Phi_i)}{\text{argmax}} \ (\underset{(z,y)_{n+1}=(z,y), \ u \in U(\Phi_i)}{\text{argmin}} \ \pi_3 \circ F_a(X_n, u))$$
$$\dot{y}_{max}(z, y) = \underset{X_n \in \mathscr{D}(\Phi_i)}{\text{argmin}} \ (\underset{(z,y)_{n+1}=(z,y), \ u \in U(\Phi_i)}{\text{argmax}} \ \pi_3 \circ F_a(X_n, u))$$

Note that we have been able to derive fully analytical solutions to these equations, yielding a very simple inclusion test for membership in the feasible goal set. Fig. 6 illustrates the resulting goal regions $\mathscr{G}_g := \mathscr{D}_g \bigcap (\bigcup_i \mathscr{G}_f(\Phi_i))$ for the example of Fig. 3. Note that feasible goal regions that do not intersect any of the instantiated policy domains can be considered "as good as lost" since, by definition, states that are not in any the domain of any policies correspond to catastrophic situations from which none of the existing controllers are capable of recovering.

IV. SIMULATION RESULTS

A. Simulation Environment

In order to illustrate the effectiveness of our algorithm, we conducted a range of simulations of the simplified planar hopper on the rough terrain illustrated in Fig. 1, with different initial conditions and goal configurations under both ideal models as well as different noise conditions. All simulations were run in Matlab, numerically integrating the equations of motion described in Section III-A using control inputs selected by the reactive controller deployment.

B. Policy Generation and Deployment

Our planning algorithm assumes that a map of the environment is available in the form of the locations and heights of each flat ground segment. In practice, this information can be obtained through exteroceptive sensors such as cameras and range sensors as the robot locomotes over new terrain.

Before our planning algorithm in Table I can be applied, a collection of local control policies Φ_i must be generated. To this end, we start by coarsely discretizing the known map into a set of constant length ($l_g = 0.25m$) sections. We then consider for each such region, four "exiting" policies that can freely transition between forward ($R_V(\Phi_i) = [0, 2.5]m/s$) and backward ($R_V(\Phi_i) = [-2.5, 0]m/s$) locomotion (ff, bb, fb, bf) and two "goal" policies that can stop the robot from either slow forward ($R_V(\Phi_i) = [0, \epsilon]m/s$) or slow backward ($R_V(\Phi_i) = [-\epsilon, 0]m/s$) locomotion (fs, bs). Orthogonally, we also consider different energy levels by imposing a global constraint on the hopping height as $z \in [0.2, 3]m$ and dividing this range into as many energy levels as necessary to obtain policies whose domain and goal regions exhibit maximal overlap. This results in four different energy levels in our case: very low, low, medium and high, yielding a total of 6*4 = 24 policies associated with each ground segment.

For each policy, we also impose certain limits on control inputs. First, global limits on the velocity gain and angle adjustment are imposed with $k \in [0.5, 2]$, and $\theta \in [-\pi/2, \pi/2]rad$. Limits on the horizontal shift Δy are designed to ensure close correspondence to trajectories feasible in the SLIP model. For steps resulting in nonzero forward or backward speeds, we require $\Delta y \in [0, 0.5]m$, and $\Delta y \in [-0.5, 0]m$, respectively. In contrast, for stopping controllers, we require smaller displacements with $\Delta y \in [-0.25, 0.25]m$, realized through a two-step controller. As a result of this construction, each policy also gets assigned their corresponding ground segment $R_g(\Phi_i)$, apex velocity range $R_V(\Phi_i)$, apex energy range $R_E(\Phi_i)$, minimum apex height $F_{h_{min}}(\Phi_i)$, and allowable control set $U(\Phi_i)$.

Once all policies are generated (960 for 40 ground segments in the example of Fig. 1), we proceed with the generation of the prepares graph $\mathcal{G}$. Having the analytical representations and associated inclusion checks for the domain and goal regions described in Sections III-C and III-D, the construction of the prepares graph is straightforward and can be done offline. This graph, whose representation is very concise with only as many nodes as there are policies, can be reused every time a different apex goal state is supplied.

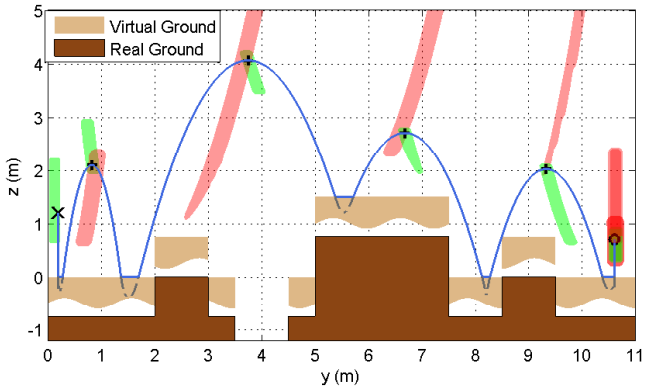

Fig. 7. An example hopper trajectory over rough terrain with reactive planning, starting from initial state $y = 0.2m, z = 1.2m, \dot{y} = 0$ and going to the goal $y = 10.6m, z = 0.7m, \dot{y} = 0$. Cross sections of domain (green) and feasible goal (red) regions are illustrated at every apex event.

The deployment of a reactive controller for a specific apex goal involves the application of the algorithm shown in Table I to obtain a total order of instantiated policies $\hat{\Phi}_i$ from the prepares graph $\mathcal{G}$ constructed above. During execution, the reactive controller measures the system state at every apex, performs a prioritized inclusion test to determine which policy to activate, and then applies a single-step deadbeat controller to select control inputs that will bring the hopper to the goal state associated with the selected policy instance.

C. Results

Fig. 7 illustrates an example run with reactive control over rough terrain, starting from $y = 0.2m$ and going to the goal $y = 10.6m$. At every apex, the controller performs ordered inclusion tests on all policy domains and selects the first match as the policy to apply. The domain of the selected policy is illustrated with the red region in the figure while the feasible goal for the previously used policy is illustrated with the green region. As visible from the figure, the prepares relation is satisfied with nonempty intersections with the feasible goal and domain regions of successive policies. Furthermore, policies are instantiated with goals that are maximally safe, lying as far in the domain of the next policy as possible. Note that only activated policies are shown, but the union of all domains has substantially more coverage as shown in Fig. 4. This example illustrates that under ideal conditions with no model or measurement uncertainty, our planner and reactive controller performs as expected.

In contrast to the ideal environment with no model uncertainty, Fig. 8 illustrates simulation runs with a constant "wind" force, constantly pushing the hopper East. The top figure shows that if control inputs computed offline under an ideal model assumption are applied, hopper trajectories slowly deviate from the generated "plan" and eventually crash into the side of the wall around $y = 8.5m$. In contrast, the application of our reactive control method ensures that proper control policies are selected at each apex, safely taking the hopper across the terrain.

Finally, Fig. 9 illustrates a scenario wherein the "sensed" ground (i.e. the ground profile used by the planner, shown

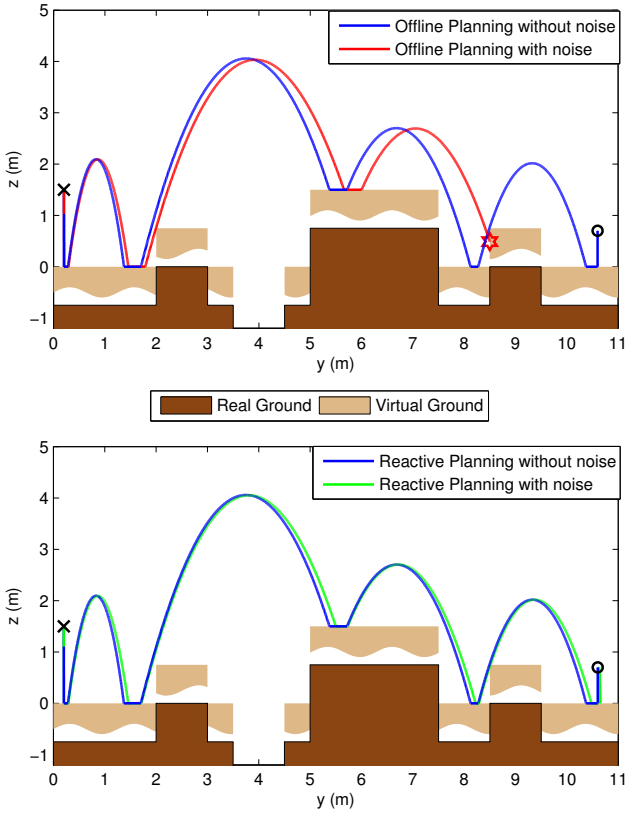

Fig. 8. Example hopper trajectories under a constant "wind" force of 0.05 N in the East direction. Top figure compares trajectories with no noise (green) to trajectories when control inputs computed offline are directly applied (red). The bottom figure compares trajectories with no noise (green) to trajectories resulting from our reactive control (blue).

with dashed lines) is different than the actual ground profile. This corresponds to a possibly more problematic situation since rather than the gradual noise introduced by the wind disturbance above, surface discrepancies may result in sudden, large disturbances that may quickly invalidate previously constructed plans. Indeed, as show in the figure, the large difference between the sensed ground and the actual ground profile in the range $y \in [0, 1]m$ causes the application of control inputs computed offline to fail catastrophically, causing a crash into the wall at $y = 5m$. However, our reactive controller, once it finds itself in a new, unexpected apex state, automatically selects the control policy that is guaranteed to eventually drive it to the overall goal, following a completely different plan than what was originally intended.

V. CONCLUSIONS AND FUTURE WORK

In this paper, we introduced a novel algorithm for the automated construction of a reactive footstep controller for a planar hopper. Our method is based on a careful characterization of the attracting domain and feasible goal sets of potential footholds on a piecewise flat surface map, combined through backchaining in a sequential composition framework to yield a full reactive control policy that guides the robot to a specified goal point, while providing an almost global region of attraction for the overall behavior.

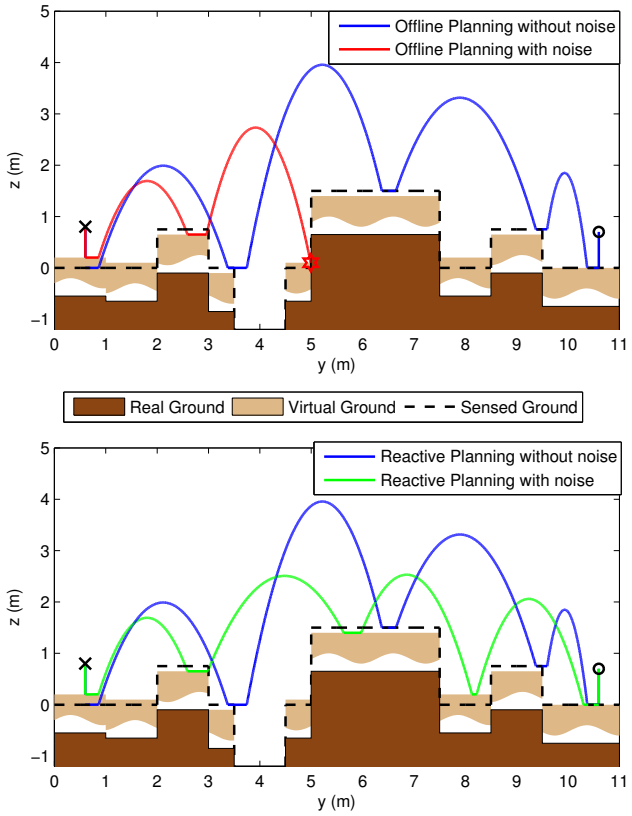

Fig. 9. Example hopper trajectories under a mismatch between sensed and actual ground heights. Top figure compares trajectories with no noise (green) to trajectories when control inputs computed offline are directly applied (red). The bottom figure compares trajectories with no noise (green) to trajectories resulting from our reactive control (blue).

We demonstrate the performance of our algorithm with a series of simulations of a simplified planar hopper, capturing essential properties of the popular Spring-Loaded Inverted Pendulum model while preserving analytical derivability of domain and goal regions for individual control policies and associated deadbeat controllers. We show that even in the presence of significant model and measurement noise, the global controller deployed by our algorithm is capable of successfully reaching the desired goal point, automatically taking alternative paths when the original, ideal plan fails due to unexpected disturbances. Compared to existing, mostly quasi-static footstep planning algorithms, both the ability of our algorithm to handle fully dynamic legged locomotion as well as its robustness against external, large sources of noise represents a significant step towards autonomous deployment of legged robots on realistic, rough terrain.

In the near future, we will extend our results and analytical domain and goal representations to the more realistic SLIP model. This will ensure that our results are immediately applicable to a large class of legged robots whose morphology and controls closely parallel those of the SLIP model. We will also demonstrate the experimental applicability of our method through a SLIP-like robot.

In the long term, automated deployment with variable length ground segment selection and the incorporation of inclined surfaces are the among possible future extensions to our proposed method. The consideration of ceiling constraints may also be interesting for indoor or otherwise covered settings. Finally, simultaneous mapping and policy deployment for fully autonomous utilization of our algorithm is among interesting future directions for this research.

References

[1] R. Altendorfer, U. Saranli, H. Komsuoglu, D. Koditschek, H. B. Brown, M. Buehler, N. Moore, D. McMordie, and R. Full. Evidence for Spring Loaded Inverted Pendulum Running in a Hexapod Robot. In *Proc. of the Int. Symp. on Exp. Robotics*, Honolulu, HI, 2001.

[2] O. Arslan, U. Saranlı, and O. Morgül. An aproximate stance map of the spring mass hopper with gravity correction for nonsymmetric locomotions. In *Proc. of the Int. Conf. on Robotics and Automation*, Kobe, Japan, 2009.

[3] 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.

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

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

[6] J. Chestnutt, M. Lau, K. M. Cheung, J. Kuffner, J. K. Hodgins, and T. Kanade. Footstep planning for the honda asimo humanoid. In *Proc. of the IEEE Int. Conf. on Robotics and Automation*, April 2005.

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

[8] D. C. Conner, H. Choset, and A. A. Rizzi. Integrated planning and control for convex-bodied nonholonomic systems using local feedback. In *Proc. of Robotics: Science and Systems II*, pages 57–64, Philadelphia, PA, August 2006. MIT Press.

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

[10] H. Geyer, A. Seyfarth, and R. Blickhan. Spring-mass running: simple approximate solution and application to gait stability. *Journal of Theoretical Biology*, 232:315–328, February 2005.

[11] J. W. Hurst, J. Chestnutt, and A. Rizzi. Design and Philosophy of the BiMASC, a Highly Dynamic Biped. In *Proc. of the Int. Conf. on Robotics and Automation*, April 2007.

[12] J. J. Kuffner, J. Koichi, N. S. Kagami, M. Inaba, and H. Inoue. Footstep planning among obstacles for biped robots. In *Proc. of IEEE Intl. Conf. on Intelligent Robots and Systems*, pages 500–505, 2001.

[13] M. H. Raibert. *Legged robots that balance*. MIT Press, Cambridge, MA, USA, 1986.

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

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

[16] 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.

[17] W. J. Schwind. *Spring loaded inverted pendulum running: a plant model*. PhD thesis, University of Michigan, Ann Arbor, MI, USA, 1998.

[18] W. J. Schwind and D. E. Koditschek. Approximating the Stance Map of a 2 DOF Monoped Runner. *Journal of Nonlinear Science*, 10(5):533–588, 2000.

[19] 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.

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