Towards Reactive Control of Transitional Legged Robot Maneuvers

Jeffrey Duperret and Daniel E. Koditschek

Department of Electrical and Systems Engineering, University of Pennsylvania, 200 South 33rd Street, Philadelphia, PA 19104
jdup@seas.upenn.edu

Abstract. We propose the idea of a discrete navigation problem – consisting of controlling the state of a discrete-time control system to reach a goal set while in the interim avoiding a set of obstacle states – to approximate a simplified class of transitional legged robotic tasks such as leaping which have no well established mathematical description that lends itself to synthesis. The control relation given in Theorem 1 is (assuming a task solution exists) necessary and sufficient to solve a discrete navigation problem in a minimum number of steps, and is well suited to computation when a legged system's continuous-time within-stride controller anchors sufficiently simple stance mechanics. We demonstrate the efficacy of this control technique on a physical hopping robot affixed to a boom to reactively leap over an obstacle with a running start, controlling in continuous time during stance to exhibit a linear stance map.

1 Introduction

Roboticists are becoming increasingly successful at both implementing [1,2,3] and analyzing [4] dynamic steady-state legged robotic tasks such as running. Underlying this success is the mathematical understanding that many useful steady-state behaviors can be encoded as stable limit cycles of the controlled hybrid dynamics of the system. The dynamical systems literature [5] has given engineers a wealth of tools for encoding and controlling the asymptotic properties of such systems – for example, to recast the problem of generating stable locomotion as finding a control which brings the Jacobian of an appropriate Poincaré map of the system to have eigenvalues with magnitude less than unity at a designated fixed point.

In contrast, tasks such as leaping, turning, and dodging have no well-agreed upon mathematical description that lends itself to synthesis. These tasks – which we refer to as *transitional* and intuitively think of as exhibiting agility – are inherently not steady-state but represent, nevertheless, a canonical motivating setting for legged locomotion, for example in operation over unstructured or irregular terrain such as leaping from foothold to foothold in rubble during disaster response. The inherently transient nature of these maneuvers does not seem well-suited to straightforward encoding via asymptotic dynamical properties and seems to call for alternative formulations.

Several techniques exist in the literature for achieving transitional tasks in hybrid systems, albeit for limited classes of models. Our focus on hybrid systems is motivated by the intrinsically multifarious and abruptly changing leg-ground mechanics associated with making and breaking contact during stance and recirculation. Implicit model predictive control [6,7] has been used to great effect, for example, by recourse to

quadratic programming [8] over a finite horizon for leaping over obstacles. Such methods recompute a solution at every step even in a static environment and are generally sensitive to their cost functions, which often are constrained to take an artificial form due to the optimization methods available and may not readily express the designer's underlying intent or handle gracefully the inherent nonlinearity of a given problem. Exciting work in sum-of-squares verification [9,10] allows reactive local controllers via Lyapunov functions but thus far appear to be fragile in the face of model uncertainty as they can incur vanishingly small basins of attraction and it remains to be demonstrated that actuator saturation – perhaps the predominant constraint in legged locomotion – can be elegantly incorporated. Reactive techniques taking the form of sequential composition [11] such as [12,13] are often very robust and can incorporate actuator constraints, however known methods are conservative and require the by-hand deployment of local controllers in the environment instead of in an automated fashion. Classical sequential composition techniques also provide guarantees over an infinite time horizon, which can be overly restrictive for transitional tasks that may be as quick in duration as a single stride.

In this paper we explore application of the reachability ideas presented in [14] and the sizable literature it generated ([15,16,17] to name a few) that construct controllers for discrete-time systems to reach some target set in a minimum number of steps, a framework that intersects the similarly longstanding tradition of pre-image backchaining in the LMT literature [18,19,20] as well as explicit model predictive control [21,22]. Particularly, we directly employ the techniques of [23] which gives the complete class of minimum-time feedback laws to reach a target set, a modified formulation of which is presented in Theorem 1. To our knowledge, such techniques have not heretofore been used for reactive control on legged robots.

We connect this theory to legged locomotion by proposing a formal definition of a class of transitional tasks that we term a *discrete navigation problem*, which consists of controlling the state of a discrete-time control system to reach a goal set while in the interim avoiding a set of obstacle states. Here the passage from the continuous hybrid dynamics of a physical plant to the discrete-time system is achieved through the imposition of a suitable stance map via an appropriate continuous-time within-stance controller. Our main contribution is summarized in Figure 1. We empirically demonstrate the efficacy of the control relation given in Theorem 1 implemented on a physical hopping robot affixed to a boom (controlled in continuous time in stance to exhibit a linear stance map) for the encoding and execution of a reactive leap over an obstacle. We formulate this task as a discrete navigation problem such that - if a task solution exists - application of Theorem 1 is necessary and sufficient for its solution in a minimum number of steps. The real-time deployment of this reactive control relation is completely automated, however our understanding of how to effectively compute it is limited to affine control systems with polyhedral obstacles, goal sets, and control constraints.

This paper is organized as follows. Section 2 introduces the discrete navigation problem. Section 3 describes the reactive control relation and Section 4 approximates the task of a legged robot leaping over an obstacle as a discrete navigation problem in a manner suitable for computing the associated reactive control relation. Section 5

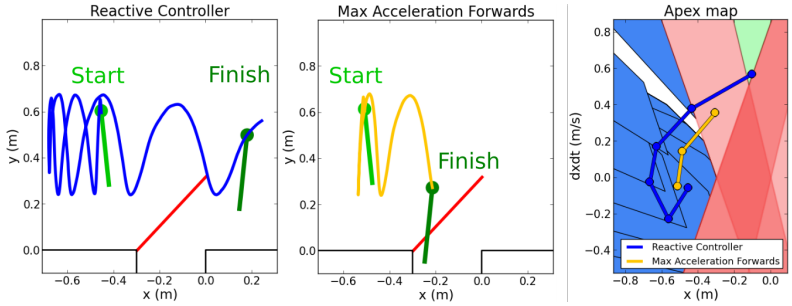

Fig. 1. Empirical data from two experiments using the hopper introduced in Figure 3 to perform a leap across a gap obstacle subject to a continuous-time within-stance controller that enforces the linear stance map described in Section 5. The first experiment – left – uses our new scheme to suggest a degree of behavioral autonomy by forcing the robot to back up so as to get a sufficient running start to clear the gap. The second experiment – middle – demonstrates that simply applying a maximum forwards acceleration is not sufficient to cross the gap. Our algorithm generates a hopping controller guaranteeing that the abstract representation of the mass center will leap over but not cross through the the red line segment - a coarse approximate obstacle set derived from the actual terrain (depicted in black in the two plots on the left). The apex states of both experiments – right – are plotted on top of the goal set of apices that will cross the gap (green), the obstacle set of apices will fall into the gap (red), and the set of apex states that can reach the goal in k steps for some integer $k > 0$ (light blue) as introduced in Section 3, while the remaining white set will enter the obstacle regardless of the applied control input and can be considered "as good as lost." These sets are explicitly constructed in Section 4 and [24]. The k-step goal-reachable blue sets "funnel" into the goal set and illustrate that reversing to get a longer running start to clear the gap is required from the robot's starting state.

empirically demonstrates the control relation deployment on a physical hopper to leap over an obstacle and is followed by concluding remarks in Section 6. This paper has a technical report companion [24] that provides proofs for the statements in Section 3 and calculates the control relation for a class of linear discrete-time control systems that describes the simplified hopper dynamics.

2 Discrete Navigation Problems

Consider a discrete-time control system $\mathbf{q}_{n+1} = f(\mathbf{q}_n, \mathbf{u}_n)$, where $\mathbf{q}_n \in \mathcal{D} \subset \mathbb{R}^m$, $\mathbf{u}_n \in \mathcal{U} \subset \mathbb{R}^p$, and the continuous map $f : \mathcal{D} \times \mathcal{U} \to \mathcal{D}$ is a homeomorphism of $\mathcal{D}$ for each fixed value of the second argument. We are interested in the task of controlling the state of such system to reach a nonempty goal set $\mathcal{G} \subset \mathcal{D}$ while in the interim avoiding a set of obstacle states $\mathcal{O} \subset \mathcal{D}$ where $\mathcal{O} \cap \mathcal{G} = \emptyset$. We call this task a *discrete navigation problem* since it is the discrete-time analogue of a continuous-time navigation problem where the state needs to reach (but not necessarily stay in) the goal set. [1]

[1] To give the reader some intuition for how this task differs from a more traditional task encoded with a limit-set goal – using an example well beyond the scope of this paper's experiments – consider a running leap to grab a vine or tree branch extending over a deep gorge so as to swing over it, where we separate the running leap to the vine from the brachiation task of grasping and

To provide a framework for deriving a controller for this task we mathematically represent the discrete navigation problem as executing (if it exists) a *K-step navigation plan* starting from $\mathbf{q} \in \mathcal{D}, \mathbf{q} \notin \mathcal{O} \cup \mathcal{G}$ for some unspecified $K \in \mathbb{N}^+$ [2]. We define a K-step navigation plan ($K \in \mathbb{N}^+$) from $\mathbf{q}_0 \in \mathcal{D}, \mathbf{q}_0 \notin \mathcal{O} \cup \mathcal{G}$ as the ordered pair $(\hat{\mathbf{q}}, \hat{\mathbf{u}})$ where $\hat{\mathbf{u}}$ is the length K control sequence $\hat{\mathbf{u}} = \{\mathbf{u}_0, \mathbf{u}_1, ..., \mathbf{u}_{K-1}\}, \mathbf{u}_i \in \mathcal{U}$ such that the length $(K+1)$ sequence of states $\hat{\mathbf{q}} = \{\mathbf{q}_0, \mathbf{q}_1, ..., \mathbf{q}_K\}$ given by the execution of the control sequence $\hat{\mathbf{u}}$ via $\mathbf{q}_{i+1} = f(f(...f(\mathbf{q}_0, \mathbf{u}_0), \mathbf{u}_1), ..., \mathbf{u}_i)$ has the properties (which we term *admissible* in the sequel) $\mathbf{q}_i \notin \mathcal{O} \cup \mathcal{G}$ for $i \in \{0, ..., K-1\}$ and $\mathbf{q}_K \in \mathcal{G}$.

We additionally introduce the notion of a *candidate K-step navigation plan* to assist in the later proofs, which we define as the ordered pair $(\hat{\mathbf{q}}, \hat{\mathbf{u}})$ that satisfies all the requirements of a K-step navigation plan except for possibly admissibility. Once admissibility of a candidate K-step navigation plan has been verified, it is a K-step navigation plan and we say that it is a *solution* to the discrete navigation problem, where by definition a candidate K-step navigation plan completes the discrete navigation task (is a solution) if and only if it is admissible.

3 Reactive Control for Discrete Navigation Tasks

This section derives a reactive control relation (given in Theorem 1) that – if a solution to the task exists – is necessary and sufficient to solve a discrete navigation problem, and concludes with remarks about its computability. Proofs of Proposition 1, Corollary 1, and Theorem 1 are provided in the technical report companion [24, Section 1].

We inductively define the set of all states $\mathcal{R}_k$ for which a control action exists that first completes the discrete navigation problem in k iterations as

$$\mathcal{R}_{k+1} := f^{-1}(\mathcal{R}_k, \mathcal{U}) \setminus (\mathcal{O} \cup \mathcal{G}), \quad \mathcal{R}_0 := \mathcal{G},$$

where we define $f^{-1}(\mathcal{S}, \mathcal{V}) := \{\mathbf{q} \in \mathcal{D} | f(\mathbf{q}, \bar{\mathbf{u}}) = \bar{\mathbf{q}}, \bar{\mathbf{q}} \in \mathcal{S}, \bar{\mathbf{u}} \in \mathcal{V}\}$ on sets $\mathcal{S} \subset \mathcal{D}, \mathcal{V} \subset \mathcal{U}$, noting that the set $\mathcal{R} = \bigcup_k \mathcal{R}_k$ gives the set of all states which are able to complete the discrete navigation problem.

Proposition 1 *A candidate K-step navigation plan* $(\hat{\mathbf{q}}, \hat{\mathbf{u}}) = (\{\mathbf{q}_0, ..., \mathbf{q}_K\}, \{\mathbf{u}_0, ..., \mathbf{u}_{K-1}\})$ *is admissible if and only if for every* $i \in \{0, ..., K-1\}$ *it holds that* $\mathbf{q}_i \in \mathcal{R}_{K-i}$ *and* $f(\mathbf{q}_i, \mathbf{u}_i) \in \mathcal{R}_{K-(i+1)}$ *(proof given in [24, Section 1]).*

Define the index set $\mathcal{I}_\mathbf{q}$ as $\mathcal{I}_\mathbf{q} := \{i \in \mathbb{N}^+ | \mathbf{q} \in \mathcal{R}_i\}$.

---

swinging from it. The goal state of the running leap might be designated as requiring an apex state of ballistic flight to be within reaching distance of the vine while the obstacle would be the gorge. This task differs from encoding limit sets in that there is no notion of continuing the task indefinitely to reduce error (as missing the vine goal set over the gorge would be catastrophic) or even remaining in the goal set for an arbitrary amount of time. Instead, the behavior of the modeled dynamics is irrelevant (in the encoding) after the goal is reached. In our example the forward flow would enter the gorge obstacle after task is finished, however phenomena outside the scope of the model take over. The vine is grasped and used to brachiate across the chasm. The brachiation itself could be considered a discrete navigation problem which could be composed with its predecessor.

[2] $\mathbb{N}^+$ denotes the positive integers.

Corollary 1. *There exists an admissible K-step navigation plan from* $\mathbf{q}$ *if and only if* $K \in \mathcal{I}_{\mathbf{q}}$. *If a solution to the discrete navigation problem exists, the minimum number of steps that it can be completed in from* $\mathbf{q}$ *is* $\min(\mathcal{I}_{\mathbf{q}})$ *(proof given in [24, Section 1]).*

We note that the sets $P_i = \{\mathbf{q} \in \mathcal{R} | i = \min(\mathcal{I}_{\mathbf{q}})\}$ for $i = 1, 2, ...$, together with $P_0 = \mathcal{R}_0$ form a partition of $\mathcal{R}$, where $\mathbf{q} \in P_i$ implies that an i-step solution from $\mathbf{q}$ exists and that i is the minimum number of steps that the navigation task can be solved in.

For ease of notation define $\mathcal{U}_{\mathbf{q},k} := \{\mathbf{u} \in \mathcal{U} | f(\mathbf{q}, \mathbf{u}) \in \mathcal{R}_{k-1}\}, k \in \mathbb{N}^+$.

Theorem 1. *If a solution to the discrete navigation problem exists, then the discrete navigation problem is solved in the minimum number of possible steps if and only if the following reactive control relation is observed at every step:*

$$\mathbf{u} \in \begin{cases} \mathcal{U}_{\mathbf{q},\min(\mathcal{I}_{\mathbf{q}})} & \mathcal{I}_{\mathbf{q}} \neq \emptyset, \\ \mathcal{U} & \textit{else}, \end{cases} \tag{1}$$

where $\mathbf{q}$ *is the state at any given iteration and* $\mathbf{u}$ *is the chosen control action at that iteration (proof given in [24, Section 1]).*

Solving the discrete navigation problem with this strategy has the practical utility that it is reactive. It reduces the problem of forming a full navigation plan – which might need to be re-planned in the case of state disturbance or uncertainty – to that of only choosing the next step from wherever the current state is.

In general there is no known method for computing $\mathcal{R}_k$, without which computing the control relation is infeasible and the user is relegated to techniques such as computing conservative approximations of these sets [20]. Even if computing a single $\mathcal{R}_k$ set is possible for a particular problem, the set of relevant $\mathcal{R}_k$'s might have infinite cardinality in which case the algorithm is not guaranteed to terminate.

However, specific cases can admit readily computable $\mathcal{R}_k$ sets and there are situations in which the cardinality of the set of all relevant $\mathcal{R}_k$ sets is finite. We show in [24, Section 2.2] a $\mathcal{R}_k$ computation method for a linear form of the system dynamics with polyhedral control constraints which is directly generalizable to affine system dynamics. Regarding the cardinality problem, if the region of operation $\mathcal{D}$ is compact (for example, if the state space is restricted to be a closed set sufficiently local to the robot) then there are problems where – for a sufficiently large K – the set $\bigcup_{i=0}^{K} \mathcal{R}_i$ has the property that for any $j \in \mathbb{N}$, $\mathcal{R}_j \subset \bigcup_{i=0}^{K} \mathcal{R}_i$, allowing the computation to terminate. Even if $\mathcal{D}$ is not compact, in some applications the user is only interested in solutions that complete the task within a maximum number of steps K (for example, if the task must be performed quickly) in which case specifying a maximum K guarantees termination. Since transitional maneuvers are typically local to a robot and must be performed quickly we could in theory adopt either of these methods but for simplicity we choose to specify a maximum K in our implementation described in Section 5.

4 Leaping Over an Obstacle With a Legged Hopper: Formulation as a Transitional Discrete Navigation Problem

This section presents our method for applying the control relation of Theorem 1 to the generation of autonomous leaping behavior in a simple sagittal-plane legged hopping model. It entails the interaction of three distinct abstracted representations of the

task, the environment, and the robot. The terrain height and robot template [25] dynamics representative of the physical machine are described in what we term the *sagittal-hopper model*. We make a local approximation of the sagittal-hopper model called the *local ballistic-approach (LBA) approximation* that gives a physically useful criterion for leaping over an obstacle in a way that is agnostic to the particular stance dynamics at the expense of making solutions slightly conservative and allowing a small but well-characterized (and rarely encountered) class of low-speed obstacle collisions as described more carefully in the third paragraph of this section's *Local Ballistic-Approach Approximation*. From this approximation we form the *task space* that forms the basis of a discrete navigation problem suitable for the application of the control relation in Theorem 1 and consists of the set of ballistic flight apex states – subsets of which form goal and obstacle sets – along with a control apex map. The control relation of Theorem 1 is explicitly calculated in [24, Equation 3] to complete the leap in the task space under the simplified dynamics presented in this section. We use these calculations as the basis for the physical experiments of Section 5.

Note that in this section we will write all set boundaries as closed to avoid the cumbersome notation of keeping track of which set boundaries are open and closed.

Sagittal-hopper model Consider a reduced-order point-mass sagittal-plane hopper tasked with leaping over without falling in or impacting a simple obstacle such as a gap, wall, or ledge from relatively flat ground. Denote the location of the hopper mass center by $\mathbf{x} = (x, y)$, where x is forward position and y is vertical height. The robot locomotes over ground represented by some terrain height function $h(x) : \mathbb{R} \to \mathbb{R}$. Suppose that the hopper's template dynamics together with the terrain height function h admit an apex map $\mathbf{q}_{n+1} = f(\mathbf{q}_n, \mathbf{u}_n)$, $f : \mathbb{R}^3 \times \mathcal{U} \to \mathbb{R}^3$ mapping the apex state $\mathbf{q}_n = (x_n, y_n, \dot{x}_n) \in \mathbb{R}^3$ at the n-th step to the $(n+1)$-th step according to the control vector $\mathbf{u}_n \in \mathcal{U} \subset \mathbb{R}^m$ chosen at step n, where $\mathbf{u}_n$ is representative of some member of a parametrized family of continuous stance-controllers.

We assume a single obstacle that is some user-specified or pre-sensed feature in the height function which is encoded with $p \in \mathbb{N}^+$ closed connected line-segments which the robot should leap over but not cross through. These line segments are individually denoted $\mathcal{L}(\mathbf{x}_{i,O1}, \mathbf{x}_{i,O2})$ in (x, y) space with endpoints $\mathbf{x}_{i,O1} = (x_{i,O1}, y_{i,O1})$ and $\mathbf{x}_{i,O2} = (x_{i,O2}, y_{i,O2})$, where $i \in \{1, ..., p\}$ and by convention we assume $\mathbf{x}_{i,O1} < \mathbf{x}_{i,O2}$ lexicographically. In future work we hope to incorporate multiple obstacles but expect interesting issues such as deadlock to arise where in certain states avoiding one obstacle would make another unavoidable. We specify which direction the hopper is to cross the obstacle by the variable σ, where $\sigma = 1$ if the hopper is tasked with traversing the obstacle forwards in x and $\sigma = -1$ if the hopper is tasked with traversing the obstacle backwards in x. Without loss of generality, we assume the hopper is initialized on the proper side of the obstacle so as to cross it in the direction determined by σ.

Local Ballistic-Approach (LBA) Approximation Passage from the sagittal-hopper model to the task space is achieved through an approximation called the *local ballistic-approach (LBA) approximation*. The LBA approximation uses only the ballistic apex map f and obstacle line segments $\mathcal{L}(\mathbf{x}_{i,O1}, \mathbf{x}_{i,O2})$, ignoring the terrain height and continuous hybrid dynamics except for their effect on the apex map. In place of the actual

continuous dynamics we associate with each ballistic apex state the parametrized set of configurations in the (x,y) plane over which the apex evolves under the influence of solely gravity's acceleration g as given by $\phi_\alpha(q) = \phi(q,\alpha) = (x+\dot{x}\alpha,\ y-\frac{g}{2}\alpha^2)^T$ where $\phi : \mathbb{R}^3 \times \mathbb{R} \to \mathbb{R}^2$. Such an evolution gives rise to an orbit $\mathcal{M}(q) := \{(x,y) \in \mathbb{R}^2 | (x,y)^T = \phi_\alpha(q), \alpha \in \mathbb{R}\}$. We write the orbit as a function of x via $y = f_q(x) = y_q - \frac{g}{2}\left(\frac{x-x_q}{\dot{x}_q}\right)^2$ or as a function of y for $y \leq y_q$ via the components $x = g_q^-(y) = x_q - \dot{x}_q\sqrt{\frac{2}{g}(y_q-y)}$ and $x = g_q^+(y) = x_q + \dot{x}_q\sqrt{\frac{2}{g}(y_q-y)}$, where g_q^- corresponds to the time leading up to apex and g_q^+ to the time subsequent to apex.

In this approximation we deem apex states whose ballistic evolutions pass over the obstacle line segments in the correct direction as determined by σ as completing the task and ballistic evolutions that pass through any of the obstacle line segments as immediately failing the task. More precisely, we define the set of apex states q completing the task as those with $sign(\dot{x}) = \sigma$ whose hypograph of f_q contains the line segment endpoints of $\mathcal{L}(\mathbf{x}_{i,O1}, \mathbf{x}_{i,O2})$ for all $i \in \{1,...,p\}$. We define the set of apex states q immediately failing the task by those having the property that, for some $i \in \{1,...,p\}$, the hypograph of g_q^- or g_q^+ contains one endpoint of $\mathcal{L}(\mathbf{x}_{i,O1}, \mathbf{x}_{i,O2})$ and the the epigraph of the same function contains the other endpoint. In this abstracted representation of the environment there is no accounting for stance interactions with the obstacle. Settings requiring perfect safety guarantees could only tolerate this simplification at the expense of dilating the obstacle diameter by the hopper's maximal shank length - a badly conservative excess. Instead, we adjoin to the set of "failing" apex states all those whose (if $\sigma = 1$) hypograph of g_q^- contain any of the obstacle endpoints or (if $\sigma = -1$) epigraph of g_q^- contain any of the obstacle endpoints so as to not let the state begin stance before the obstacle in x and end stance after the obstacle in x. These sets are graphically shown in Figure 2.

The reason for making the LBA approximation is because it provides a convenient method of approximating task success or failure from the ballistic apex state in a way that is agnostic to both the continuous stance dynamics and local characteristics of the terrain height function h. This generality comes at the price of (a) labeling some trajectories which complete the task in the sagittal-hopper model as failing in the LBA approximation, and (b) labeling some trajectories which fail the task in the sagittal-hopper model as not failing in the LBA approximation. Specifically, solutions in the sagittal-hopper model with ballistic apex q which complete the task but which transition from stance to flight directly above an obstacle line segment can be labeled in the LBA approximation as failing the task if $\mathcal{M}(q)$ intersects the line segment despite the state never doing so in the sagittal-hopper model. Additionally, trajectories in the sagittal-hopper model which pass through an obstacle line segment a nonzero even number of times in stance will be not counted as immediately failing the task in the LBA approximation. We expect such violations to be at low kinetic energies because they require a direction reversal[3] close to the obstacle and so we assume the robot structure can tol-

[3] We believe that one of the fundamental constraints of legged robot technology is limited actuation power, implying a limited affordance to change kinetic energy over the course of a stride.

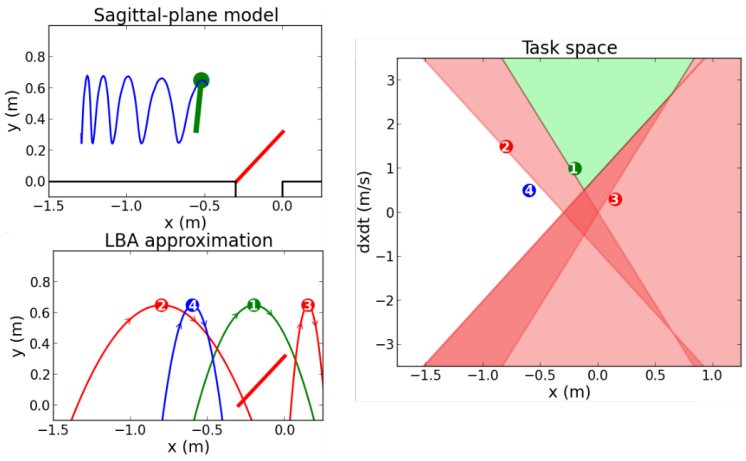

Fig. 2. Illustration of the problem setup given in Section 4. The sagittal-plane model in the upper left includes the terrain height (black) and the user-encoded or pre-sensed obstacle (red) that the mass center (green dot) should leap over in some pre-specified direction but not pass through. Here the line segment was user-specified to span the horizontal width of the physical obstacle and to vertically extend beyond the physical obstacle by the length of the hopper's leg so as to allow room for the hopper's leg to clear the physical obstacle in flight. Note that the obstacle can include multiple line segments to better "contour" the obstacle but a single line segment is used in this work for simplicity. Notional robot trajectories are given in blue. The local ballistic-approach (LBA) approximation of the sagittal-plane model in the lower left provides a convenient method of approximating task success or failure from only the ballistic apex state and obstacle line segments, associating with each ballistic apex state the parametrized set of configurations over which the apex evolves solely under the influence of gravity. The task space slice on the right shows the goal set (green) of apex states whose LBA evolution passes over the obstacle line segment and the obstacle set (red) of apex states whose LBA evolution parabola either passes through the obstacle line segment or is past the obstacle. For example, the apex state 1 passes over the obstacle in the LBA approximation so it is in the goal set. The apex state 2 will hit the obstacle in the LBA approximation so is in the obstacle set. The apex state 3 has already passed the obstacle so it is also in the obstacle set. The apex state 4 is before the obstacle and will neither pass over nor through the obstacle in the LBA approximation and therefore isn't in the goal or obstacle set.

erate such collisions. In simulation we have observed such violations when the robot is initialized very close to and slowly moving towards an obstacle. Thus this formulation is best suited for use in uncluttered environments where the obstacles are not extremely close to each other so that after navigating one obstacle the robot will not initialize itself adjacent to the next obstacle.

Task Space The task space is composed of the set of ballistic apex states – which evolve according to the dynamics of the control apex map f – together with a goal set and obstacle set. For simplicity in this work we restrict the set of apex heights under consideration to be those that are higher than the obstacle height, i.e. $y > max(y_{i,O1}, y_{i,O2}) \forall i \in \{1, ..., p\}$, so that g_q^- and g_q^+ are well-defined (if the robot starts below this threshold height we assume it is able to safely leap to above this threshold on the next leap). Let

the goal set $\mathcal{G}$ be the set of apex states which complete the leaping task according to the LBA approximation and similarly let the obstacle set $\mathcal{O}$ be the set of apex states which immediately fail the leaping task. Explicitly, they are given by[4]:

$$\mathcal{G} = \left\{ (x,\dot{x},y) \in \mathbb{R}^3 | \forall i \in \{1,...,p\} : \begin{bmatrix} 1 & \sigma\sqrt{\frac{2}{g}(y-y_{i,O1})} \\ -1 & \sigma\sqrt{\frac{2}{g}(y-y_{i,O1})} \\ 1 & \sigma\sqrt{\frac{2}{g}(y-y_{i,O2})} \\ -1 & \sigma\sqrt{\frac{2}{g}(y-y_{i,O2})} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} > \begin{bmatrix} x_{i,O1} \\ -x_{i,O1} \\ x_{i,O2} \\ -x_{i,O2} \end{bmatrix} \right\},$$

and

$$\begin{aligned} \mathcal{O}_i = \Big\{ (x,\dot{x},y) \in \mathbb{R}^3 | \\ \left(\begin{bmatrix} 1 & \sqrt{\frac{2}{g}(y-y_{i,O1})} \\ -1 & -\sqrt{\frac{2}{g}(y-y_{i,O2})} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \geq \begin{bmatrix} x_{i,O1} \\ -x_{i,O2} \end{bmatrix} \right) \vee \left(\begin{bmatrix} -1 & -\sqrt{\frac{2}{g}(y-y_{i,O1})} \\ 1 & \sqrt{\frac{2}{g}(y-y_{i,O2})} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \geq \begin{bmatrix} -x_{i,O1} \\ x_{i,O2} \end{bmatrix} \right) \vee \\ \left(\begin{bmatrix} -1 & \sqrt{\frac{2}{g}(y-y_{i,O1})} \\ 1 & -\sqrt{\frac{2}{g}(y-y_{i,O2})} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \geq \begin{bmatrix} -x_{i,O1} \\ x_{i,O2} \end{bmatrix} \right) \vee \left(\begin{bmatrix} 1 & -\sqrt{\frac{2}{g}(y-y_{i,O1})} \\ -1 & \sqrt{\frac{2}{g}(y-y_{i,O2})} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \geq \begin{bmatrix} x_{i,O1} \\ -x_{i,O2} \end{bmatrix} \right) \vee \\ \left(\begin{bmatrix} \sigma & -\sigma\sqrt{\frac{2}{g}(y-y_{i,O1})} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \geq \sigma x_{i,O1} \right) \vee \left(\begin{bmatrix} \sigma & -\sigma\sqrt{\frac{2}{g}(y-y_{i,O2})} \end{bmatrix} \begin{bmatrix} x \\ \dot{x} \end{bmatrix} \geq \sigma x_{i,O2} \right) \Big\}, \end{aligned}$$

where $\mathcal{O} = \cup_{i=1}^{p} \mathcal{O}_i$, and where $\sigma = 1$ if the obstacle is to be traversed forwards in x and $\sigma = -1$ if the obstacle is to be traversed backwards in x. We note that for a constant y, $\mathcal{G}$ and $\mathcal{O}_i$ become polyhedra, a fact we use in the following section. When holding y constant, the goal set is described by four halfspace constraints for a given obstacle line segment. The first two halfspace constraints of the goal set require that $\mathcal{M}(q)$ pass over $\mathbf{x}_{i,O1}$ while the second two require that $\mathcal{M}(q)$ pass over $\mathbf{x}_{i,O2}$. The obstacle set is described by six polyhedra for a fixed y. The first four obstacle set polyhedra correspond to $\phi_\alpha(q)$ intersecting $\mathcal{L}(\mathbf{x}_{i,O1}, \mathbf{x}_{i,O2})$. There are four to account for the cases of intersections through either direction of $\mathcal{L}(\mathbf{x}_{i,O1}, \mathbf{x}_{i,O2})$ and for intersecting with positive or negative values of α in $\phi_\alpha(q)$. The point 2 in Figure 2 gives an example of such a case. The last two polyhedra encode states whose evolution $\phi_\alpha(q)$ with negative α pass under either of the obstacle endpoints to account for all apex states that have already passed the obstacle. The point 3 in Figure 2 gives an example of such a case.

Restriction to Linear Apex Map and Polyhedra Goal and Obstacles in Sagittal-Hopper Model We restrict our attention to a linear form of the apex map f predicated on the desire for iterated dynamics simple enough for analysis but complex enough to give physically relevant sagittal-plane behavior. To achieve such a map, we prescribe our point-mass hopper model a constant stance time u_T during which we apply an average horizontal stance acceleration $u_{\ddot{x}}$ on the mass center – where $u_{\ddot{x}}$ is constrained to

[4]These sets are calculated by solving the inequalities resulting from the hypograph/epigraph of f_q, g_q^-, or g_q^- containing obstacle line segment endpoints, as explained in this section's *Local Ballistic-Approach (LBA) Approximation*.

a closed interval containing zero – and some vertical force to achieve the (artificially imposed) constant ballistic flight apex height $y_n = y$. For simplicity we require that the beginning of stance occur when the point mass drops below a height of y_{TD} and that the hopper lift off from stance into flight at this height as well. While these assumptions (mainly the assumption of a constant apex height) severely restricts the behavior of the template, they still yield a reduced-order model capable of relevant sagittal plane behaviors such as hopping over gaps and onto ledges from a running start. These restrictions also endow the apex map with a linear form, making it suitable for the linear programming computations given in [24, Section 2]. We hope in future work to investigate restricted dynamics with more expressive behaviors that are also simple enough to be used with a similar algorithm.

Let $\mathbf{u} = u_{\ddot{x}} \in \mathcal{U} = [u_{\ddot{x},min}, u_{\ddot{x},max}] \subset \mathbb{R}$. We treat the constant apex height y and constant touchdown/liftoff height y_{TD} as parameters, along with the vertical acceleration due to gravity g. The linear controlled apex map $\mathbf{q}_{n+1} = f(\mathbf{q}_n, \mathbf{u}_n)$ of such a system is given by:

$$
\begin{aligned}
x_{n+1} &= x_n + u_T \dot{x}_n + (2\dot{x}_n + u_{\ddot{x},n} u_T) \sqrt{\frac{2}{g}(y - y_{TD})} + \left(\frac{u_T^2}{2}\right) u_{\ddot{x},n}, \qquad (2)\\
y_{n+1} &= y\\
\dot{x}_{n+1} &= \dot{x}_n + u_{\ddot{x},n} u_T,
\end{aligned}
$$

While many continuous stance dynamics could satisfy such an apex map, they can be as simple as applying a constant horizontal force and piecewise constant vertical force in stance for a time duration u_T to satisfy the desired forwards acceleration and vertical apex height.

5 Experiments

A series of running leap experiments using a robotic hopper on a boom were performed to demonstrate the utility of the control relation given in Theorem 1 under the linear dynamics assumption of Section 4.

The hopper, shown in Figure 3, consists of a two degree-of-freedom 3.2 kg leg fixed to a circular boom, constrained to allow translation but not rotation in the sagittal plane. The hopper leg is a parallel five-bar mechanism [26] actuated by two T-Motor U10-Plus 80KV motors[5] using Ghost Robotics motor controllers[6]. An STM-32 F303VC microcontroller[7] performs the control algorithm and sends commands to the motor controllers at 1kHz[8]. The only sensors used by the machine are encoders at the motor shafts to sense the leg kinematics and two encoders on the boom that sense the polar angle of the boom (used to calculate forward distance in the sagittal plane) as well as the azimuth angle of the boom (used to calculate the vertical height in the sagittal plane). No sensing of

[5] http://store-en.tmotor.com/goods.php?id=362

[6] https://www.ghostrobotics.io/

[7] http://www.st.com/en/microcontrollers/stm32-32-bit-arm-cortex-mcus.html?querycriteria=productId=SC1169

[8] The computations of [24] are not constrained to a 1kHz update rate and are only computed once per stride.

the obstacle is necessary as its location is preprogrammed into the algorithm, although future work will involve its sensing. An off-board 4-cell lithium polymer battery was used to power the hopper.

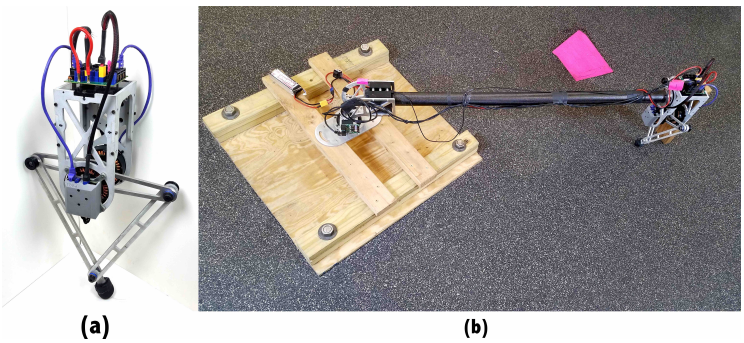

Fig. 3. The legged hopper (a) is used to empirically demonstrate the utility of the control relation given in Theorem 1 on a microcontroller to perform a leaping maneuver with a running start. The fully actuated two degree-of-freedom legs allow the command of arbitrary-direction sagittal-plane stance forces so as to anchor the linear apex map template model given in Equation 2 for use in the experiments. The experimental setup (b) shows the hopper constrained to the sagittal plane by a boom in front of the gap obstacle consisting of pink tape.

The template dynamics given in Equation 2 were achieved in stance by applying a constant feed-forward horizontal force $u_{\ddot{x}}$, a feed-forward piecewise constant vertical force to achieve the desired deadbeat apex height y, and limiting the stance duration time to u_T. Simple proportional derivative control was used as feedback to track the stance trajectory expected from the feed-forward signals. This anchoring [25] was achieved by mapping force commands to torque commands through the inverse transpose of the kinematic Jacobian, and saturating the infinity norm of the required torque according to the motor's torque limits. A singularity-avoidance controller that applies increasing amounts of torque as the leg approaches a singular configuration guarantees that the leg won't travel through a kinematic singularity during stance. We demonstrate the efficacy of this anchoring on the hopper at low speeds in Figure 4 and high speeds in Figure 5, although we save an in-depth description and experimental statistical analysis for future work on the more interesting case of a free-running untethered robot.

Figure 4 shows 30 instances of anchoring a low-speed stride on the hopper. The correspondence of the desired and actual trajectories at the apex events indicate that the linear apex-map template of Equation 2 is approximately achieved, however work remains to improve the in-stance anchoring as indicated by the the y-trace in which a variety of undesired phenomena are evident. We believe that the larger-than-expected vertical acceleration early in stance is due to the effect of a singularity avoidance controller activating as the leg approaches the edge of its workspace. We are also unclear exactly as to why the correct apex is achieved despite liftoff occurring at too great a height but suspect that some friction-like effect from the boom is responsible. For now we allow these effects to roughly cancel each other out but they deserve a more care-

ful treatment in future work when we implement this controller on an untethered robot unhindered by the dynamics of the boom.

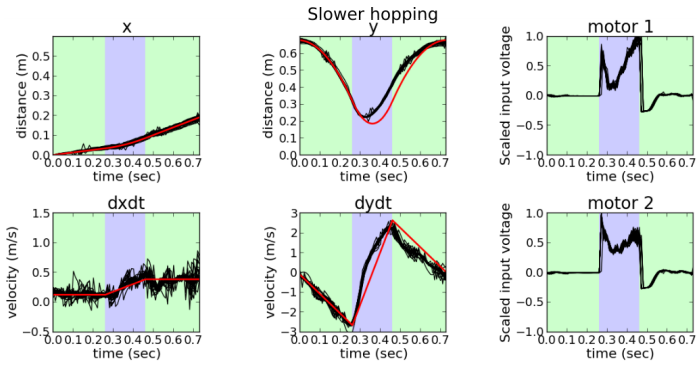

Fig. 4. Boom encoder readings and motor commands (black) from 30 runs of experiments of lower-speed hopper strides, superimposed with the desired trajectory of the robot (red), show that at slower speeds the linear apex-map template of Equation 2 is approximately achieved – as indicated by the correspondence of the desired and actual trajectories at the apex events occurring at times 0 and 0.7. However, work remains to improve the in-stance anchoring as indicated by the y-trace in which a variety of undesired phenomena are evident as discussed in Section 5. Note that the blue background indicates stance while the green background indicates ballistic flight. The data has been shifted in x so that every run starts from $x = 0$ as well as in time so that detected stance onset occurs simultaneously in every run.

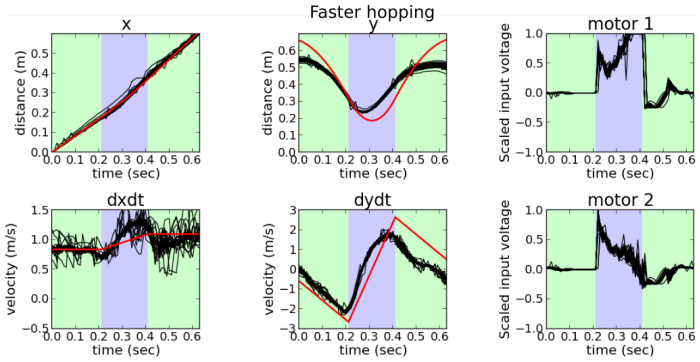

Fig. 5. Boom encoder readings and motor-commands (black) from 28 runs of experiments of higher-speed hopper strides, superimposed with the desired trajectory of the robot (red), show that at higher speeds the hopper is unable to anchor the linear apex-map template of Equation 2. This appears to be due to the fact that at higher speeds the kinematic configuration of the leg is such that, in the second half of stance, one of the two motors is saturated and the other one is doing almost no work as shown in the motor plots on the right as discussed in Section 5. Note that at these higher speeds the boom dynamics become more pronounced; ringing oscillations are evident in both the forward and vertical velocity traces later in the stride.

Figure 5 shows 28 runs of attempting to anchor a higher-speed stride on the hopper, demonstrating that at higher speeds the hopper is unable to anchor the linear apex-

map template of Equation 2. This appears to be due to the fact that at higher speeds the kinematic configuration of the leg is such that in the second half of stance one of the two motors is saturated and the other one is doing almost no work, as shown in the motor plots on the right. This manifests itself in a sharp drop-off in vertical velocity in the second half of stance so that the hopper is unable to generate enough liftoff velocity to attain the commanded apex height, indicating that the implementation of this algorithm on an untethered robot would benefit from a leg configuration that more evenly shares the burden of the motor affordance while running, bearing in mind that the requirement that the motor-leg kinematics be designed to support this class of templates will likely decrease the performance of some other task.

We implement the algorithm on-board the robot by computing the $\mathcal{R}_k$ sets up to a k of 20 when the robot initializes and then using the estimated apex state to calculate the control relation $\mathcal{U}_{\mathbf{q},\min(\mathcal{I}_\mathbf{q})}$ via [24, Equation 3] once per stride. The hopper then chooses the forward acceleration $u_{\ddot{x}}$ that is both consistent with the control relation and maximizes the distance of the next apex state to the boundary of the relevant $\mathcal{R}_k$ set containing it, achieving a degree of robustness in an informal sense that is elaborated on in [24, Section 3] . The obstacle location (represented by features in the black line-segment terrain in the right-hand-side column of Figure 6) is known a-priori to the control algorithm and encoded in the sagittal-hopper model using a single line segment (represented by the red line segments in the right-hand-side column of Figure 6), which is user-specified to span the horizontal width of the physical obstacle and to vertically extend beyond the physical obstacle by the length of the hopper's leg so as to allow room for the hopper's leg to clear the physical obstacle in flight. Future work will seek to automate this obstacle detection process from sensor data such as LiDAR readings.

Three experiments were performed to demonstrate the applicability of Theorem 1 to a physical machine using embedded hardware. In the first, the hopper is commanded to repeatedly leap over a ledge with a gap in front of it from various initial conditions to demonstrate repeatability. In the second and third, the hopper was commanded to cross other types of obstacles (a hurdle and a simple gap) to showcase a few of the different obstacle types that can be traversed and give a sense of their corresponding sets $\mathcal{G}$ and $\mathcal{O}$. In each case the hopping apex height y was set to 0.65m, the touchdown and liftoff height y_{TD} was set to 0.32m, the stance time u_T was set to 200ms, and the forward acceleration was limited to the modest $u_{\ddot{x},max} = -u_{\ddot{x},min} = 0.5$ m/s^2 due actuator saturation constraints.

The results of the experiments are shown in Figure 6. Figure 6(a) depicts 15 runs of the hopper leaping onto a ledge immediately preceded by a gap, and demonstrates that the task can be repeatedly completed from various initial conditions. Figure 6(b) depicts 5 runs of the hopper leaping over a simple gap obstacle, and Figure 6(c) depicts 6 runs of the hopper leaping over a hurdle obstacle. The variation in apex trajectories from similar initial conditions indicates our anchoring scheme of the linear apex dynamics given in Equation 2 remains to be improved (as depicted in Figures 4 and 5), however the task is still completed.

An additional experiment is shown in Figure 1 which demonstrates a degree of behavioral autonomy which this reactive control relation can provide. In this set of experiments the maximum acceleration was set to $u_{\ddot{x},max} = -u_{\ddot{x},min} = 1.0$ m/s^2. The

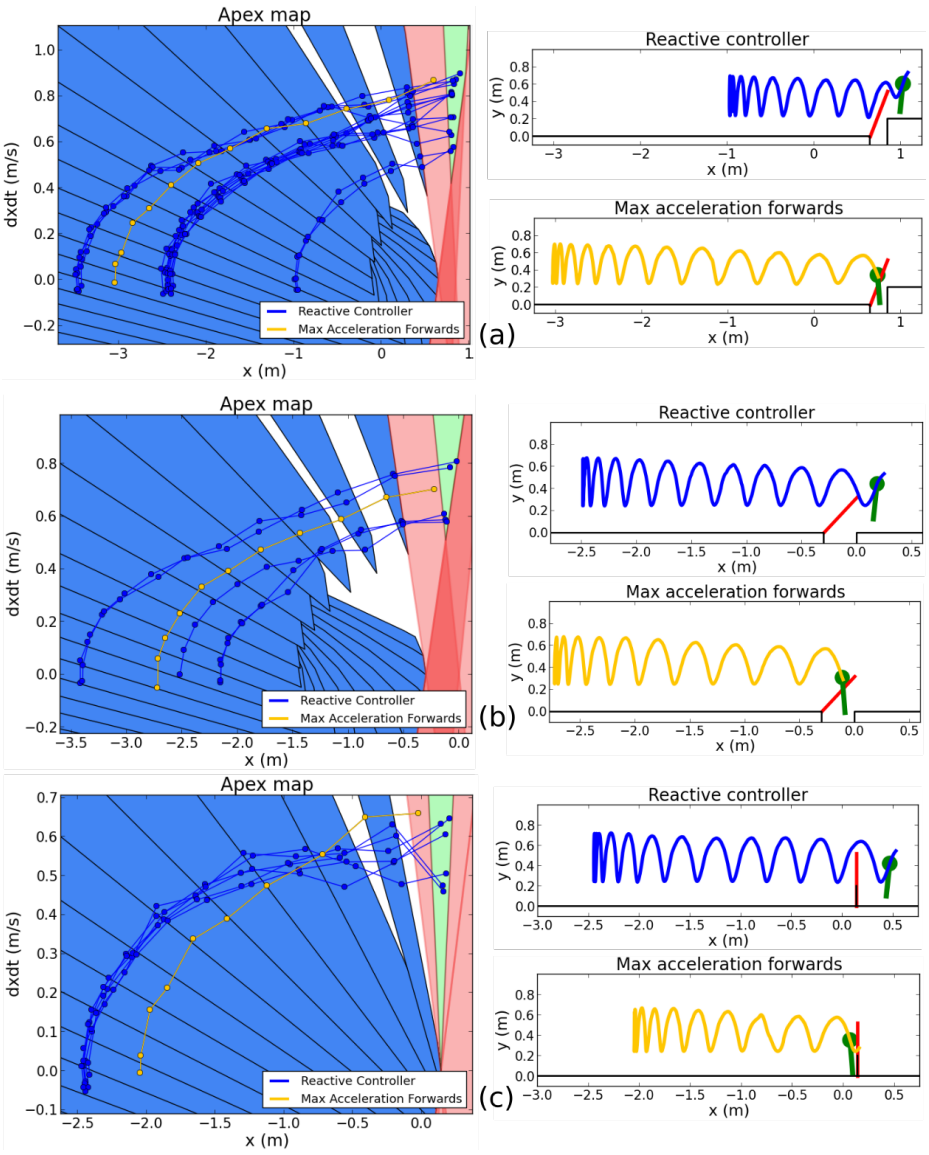

Fig. 6. Experimental results of the robotic hopper introduced in Figure 3 using the control relation of Theorem 1 to reactively leap over various obstacles from a running start. Row (a) depicts 15 instances of the hopper leaping onto a ledge immediately preceded by a gap from various initial conditions, (b) depicts 6 instances of leaping over a simple gap obstacle, and (c) depicts 5 instances of leaping over a hurdle obstacle. The right column of the figure shows the sagittal-plane representation of the environment with the terrain depicted in black and an example controlled robot trajectory from the experiments in blue, smoothed to filter out measurement noise. The yellow trajectory demonstrates failure when the robot simply accelerates at maximum. The red line segment is the encoded obstacle to be avoided by the mass-center. This line segment was user-specified to span the horizontal width of the physical obstacle and to vertically extend beyond the physical obstacle by the length of the hopper's leg so as to allow room for the hopper's leg to clear the physical obstacle in flight. Future work will seek to automate this obstacle detection process from sensor data such as LiDAR readings. The left column of the figure shows the apex-state representation of the leaping task where the robot's apex trajectory is depicted by dark blue dots sequentially connected by dark blue lines. The set of apex states which result in immediately passing through the obstacle as-per the approximations of Section 4 are shown in red and comprise the obstacle set $\mathcal{O}$, while the set of apex states which pass over the obstacle are shown in green and comprise the goal set $\mathcal{G}$. The lighter blue sets represent the set of states $\mathcal{R}_k$ which can reach the goal in k steps and which "funnel" into $\mathcal{G}$, while the remaining white area is the set of apex states which flow into $\mathcal{O}$ regardless of the applied control input.

robot is initialized too close to a gap to directly leap over it and – when using the proposed control scheme – reverses to gain enough of a running start to clear the gap. Without this direction reversal (when simply accelerating forwards at $u_{\ddot{x},max}$) the robot is observed falling into the gap.

6 Conclusions and Future Work

We propose the notion of a discrete navigation problem – consisting of controlling the state of a discrete-time control system to reach a goal set while in the interim avoiding a set of obstacle states – to approximate a class of tasks useful for legged robotic applications such as leaping. We demonstrated the efficacy of the control relation given in Theorem 1, which is (assuming a task solution exists) necessary and sufficient to solve a discrete navigation problem in a minimum number of steps, on a physical hopping robot affixed to a boom to reactively leap over an obstacle with a running start, controlling the continuous stance dynamics to exhibit a linear stance map.

Future work will focus on implementing this algorithm on a pronking quadruped robot autonomously sensing obstacles with LiDAR. We also plan to apply this algorithm to multiple sequential obstacles in an "obstacle course"-like environment which we expect will raise interesting issues such as avoiding deadlock. Finally, we hope to be able to make formal claims on forming various control laws from the proposed control relation as it relates to criteria such as robustness and aggressiveness now that we have an exact representation of the solution space of discrete navigation problems as opposed to a conservative representation of it.

ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation Graduate Research Fellowship under Grant No. DGE-0822 held by the first author and in part by the Army Research Office under Grant No. W911NF-17-1-0229.

References

1. S. Seok, A. Wang, M. Y. Chuah, D. J. Hyun, J. Lee, D. M. Otten, J. H. Lang, and S. Kim, "Design principles for energy-efficient legged locomotion and implementation on the mit cheetah robot," *IEEE/ASME Transactions on Mechatronics*, vol. 20, no. 3, pp. 1117–1129, 2015.
2. D. J. Hyun, S. Seok, J. Lee, and S. Kim, "High speed trot-running: Implementation of a hierarchical controller using proprioceptive impedance control on the mit cheetah," *International Journal of Robotics Research*, vol. 33, no. 11, pp. 1417–1445, 2014.
3. "Boston dynamics," http://www.bostondynamics.com.
4. A. De and D. E. Koditschek, "Vertical hopper compositions for preflexive and feedback-stabilized quadrupedal bounding, pacing, pronking and trotting," *(under review)*, Dec. 2016.
5. J. Guckenheimer and P. J. Holmes, *Nonlinear oscillations, dynamical systems, and bifurcations of vector fields*. Springer Science & Business Media, 2013, vol. 42.
6. D. Q. Mayne, "Model predictive control: Recent developments and future promise," *Automatica*, vol. 50, no. 12, pp. 2967–2986, 2014.
7. J. H. Lee, "Model predictive control: Review of the three decades of development," *International Journal of Control, Automation and Systems*, vol. 9, no. 3, pp. 415–424, 2011.

8. H. . Park, P. M. Wensing, and S. Kim, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *Robotics: Science and Systems*, vol. 11, 2015.
9. R. Tedrake, I. R. Manchester, M. Tobenkin, and J. W. Roberts, "Lqr-trees: Feedback motion planning via sums-of-squares verification," *International Journal of Robotics Research*, vol. 29, no. 8, pp. 1038–1052, 2010.
10. A. Majumdar, A. A. Ahmadi, and R. Tedrake, "Control design along trajectories with sums of squares programming," in *Proceedings - IEEE International Conference on Robotics and Automation*, 2013, pp. 4054–4061.
11. R. Burridge, A. Rizzi, and D. Koditschek, "Sequential composition of dynamically dexterous robot behaviors," *The International Journal of Robotics Research*, vol. 18, pp. 534–555, 1999.
12. O. Arslan and U. Saranli, "Reactive planning and control of planar spring-mass running on rough terrain," *IEEE Transactions on Robotics*, vol. 28, no. 3, pp. 567–579, 2012.
13. D. C. Conner, H. Choset, and A. A. Rizzi, "Integrated planning and control for convex-bodied nonholonomic systems using local feedback control policies," in *Robotics: Science and Systems*, vol. 2, 2007, pp. 57–64.
14. D. P. Bertsekas and I. B. Rhodes, "On the minimax reachability of target sets and target tubes," *Automatica*, vol. 7, no. 2, pp. 233–247, 1971.
15. D. Q. Mayne and W. R. Schroeder, "Robust time-optimal control of constrained linear systems," *Automatica*, vol. 33, no. 12, pp. 2103–2118, 1997.
16. S. V. Rakovi and D. Q. Mayne, "Robust time optimal obstacle avoidance problem for constrained discrete time systems," in *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference, CDC-ECC '05*, vol. 2005, 2005, pp. 981–986.
17. S. V. Rakovi, F. Blanchini, E. Crck, and M. Morari, "Robust obstacle avoidance for constrained linear discrete time systems: A set-theoretic approach," in *Proceedings of the IEEE Conference on Decision and Control*, 2007, pp. 188–193.
18. T. Lozano-Prez, M. T. Mason, and R. H. Taylor, "Automatic synthesis of fine-motion strategies for robots," *The International Journal of Robotics Research*, vol. 3, no. 1, pp. 3–24, 1984.
19. M. Mason, "Automatic planning of fine motions: Correctness and completeness," in *Robotics and Automation. Proceedings. 1984 IEEE International Conference on*, vol. 1. IEEE, 1984, pp. 492–503.
20. M. Erdmann, "Using backprojections for fine motion planning with uncertainty," *International Journal of Robotics Research*, vol. 5, no. 1, pp. 19–45, 1986.
21. A. Alessio and A. Bemporad, *A survey on explicit model predictive control*, ser. Lecture Notes in Control and Information Sciences, 2009, vol. 384.
22. A. Bemporad, F. Borrelli, and M. Morari, "Model predictive control based on linear programming - the explicit solution," *IEEE Transactions on Automatic Control*, vol. 47, no. 12, pp. 1974–1985, 2002.
23. S. S. Keerthi and E. G. Gilbert, "Computation of minimum-time feedback control laws for discrete-time systems with state-control constraints," *IEEE Transactions on Automatic Control*, vol. 32, no. 5, pp. 432–435, 1987.
24. J. Duperret and D. E. Koditschek, "Technical report on: Towards reactive control of simplified legged robotics maneuvers," University of Pennsylvania, Tech. Rep., 2017.
25. R. J. Full and D. E. Koditschek, "Templates and anchors: Neuromechanical hypotheses of legged locomotion on land," *Journal of Experimental Biology*, vol. 202, no. 23, pp. 3325–3332, 1999.
26. D. A. Kenneally, Gavin and D. E. Koditschek, "Design principles for a family of direct-drive legged robots," in *Robotics: Science and Systems, Workshop*, 2015.