Reachability-based Control for the Active SLIP Model

Giulia Piovan and Katie Byl *

Abstract

One of the key topics in robotics is legged locomotion, taking inspiration from walking and bouncing gaits of bipeds and quadrupeds. A fundamental tool for analyzing running and hopping is the Spring Loaded Inverted Pendulum (SLIP), thanks to its simplicity but also effectiveness in modeling bouncing gaits. Being completely passive, the SLIP model does not have the ability to modify its net energy, which, for example, can be a prohibitive obstacle when traveling on a terrain with varying heights. The actuated version of the SLIP model, the active SLIP, considered in this paper, includes a series actuator that allows energy variations via compressing or extending the spring. In this work, we investigate how different actuator motions can affect the system's state, and we propose a control strategy, based on graphical and numerical studies of the reachability space, and updated throughout the stance phase, to drive the system to a desired state. We quantify its performance benefits, particularly in serving as an error recovering method. The objective of our control strategy is not to replace any leg-placement approach proposed by other works, but rather to be paired with any other leg-placement or path planning method. Its main advantage is the ability to reduce the effects of sensing errors and disturbances happening at landing as well as during the stance phase.

*G. Piovan and K. Byl are with the Robotics Laboratory, University of California at Santa Barbara, Santa Barbara, CA 93106, USA, giulia@engineering.ucsb.edu, katiebyl@ece.ucsb.edu

1 Introduction

Legged locomotion has long been a topic of study in robotics. While wheeled vehicles are extremely effective on continuous surfaces there are also terrains that are not accessible by them. However, these same terrains can easily be reached by most animals and humans: the continuous path of support needed by wheeled vehicles is substituted by a set of isolated footholds in the case of legged systems. Based on our daily experience, walking on legs seems to be a trivial task, and yet the capabilities of legged animals are far superior to those of any existing robot, due to mechanical and sensorial limitations (Full and Koditschek 1999). As a consequence, many robotics researchers have been focusing on studying and mimicking legged animals and humans (e.g., Collins et al. 2005).

The conventional model used for analyzing running and hopping is the so-called Spring Loaded Inverted Pendulum (SLIP). In combination with biological data fitting, this simple model can predict the center of mass dynamics of biped runners (Blickhan and Full 1993), but lacking an actuator it is incapable of increasing or decreasing its net energy. This is problematic, since it is clearly desirable to enable legged locomotion in situations with variability in both terrain height and forward velocity, requiring accompanying modification of net energy. The SLIP model can be improved by adding a linear piston-like actuator in series with the spring (e.g., Raibert 1986, Hodgins and Raibert 1991, Ahmadi and Buehler 1997). This actuated system, the so-called active SLIP, has the ability of supplementarily compressing or extending the spring, which will therefore store more or less energy. Changing the spring length allows us to alter the trajectory of the center of mass, thus compensat-

ing for sensing errors and energy losses. In this work, we study the effect that actuation has on the system's trajectory, and we propose a control strategy to drive the system to a desired location in state space.

The SLIP model has long been used to model running and hopping behaviors for both biomechanics and robotics (Blickhan 1989, Blickhan and Full 1993, and Farley and Ferris 1998). In the 80's Raibert (1986) presented the first and best-known monoped hopping robot, the Raibert hopper, opening the door to a long series of SLIP-based robots, from the bow-legged hopper (Zeglin 1999), RHex (Saranli et al. 2001) and the more recent BiMASC (Hurst et al. 2007), just to name a few. While the SLIP model is a very simple system, there is not an analytic solution during ground contact, limiting our ability to understand and control the SLIP dynamics. To overcome this issue, one early approximation to the closed-form solution has been given by Schwind and Koditschek (2000), followed by Geyer et al. (2005). These two solutions based upon conservation of momentum only work well for symmetric motions. Arslan et al. (2009) and Yu et al. (2012) have proposed approximations which include gravity corrections, improving the model performance. The classic SLIP model has been employed as a starting point in studies focusing on control strategies for path planning (Arslan and Saranlı 2012) or to stabilize the system in presence of disturbances due to uneven terrain or unexpected forces. Because of the completely passive nature of the system, its main tool in terms of balance is the placement of the swing leg. Seyfarth et al. (2003) propose a swing-leg retraction strategy, while Karssen et al. (2011) study its optimal retraction rate. On the contrary, in Ghigliazza et al. (2005) it is shown how choosing a fixed leg angle at touch-down results in asymptotically stable periodic gaits. Stability analysis of the SLIP model can also be found in Altendorfer et al. (2004), where a necessary condition for asymptotic stability is computed and used to quantify sensory costs for several feedback strategies previously proposed for the SLIP model. Studies on guinea fowls (Daley et al. 2007), and on insects (Dudek and Full 2006, and Sponberg and Full 2008), however, show that animals and insects use their legs' natural actuation to stabilize their body. This suggests that the passive SLIP is not fully able to model running and hopping gaits, and justifies the introduction of the actuated version of the SLIP. There are several methods to add actuation to the SLIP model: Seipel and Holmes (2007) added a clock-based torque at the hip of the system, while Raibert (1986), Schmitt and Clark (2009), Andrews et al. (2011), Rutschmann et al. (2012), Piovan and Byl (2012), Byl et al. (2012), and Piovan and Byl (2013) explore the possibility of adding and activating a series actuator during the stance phase to modify the net energy of the system. Vejdani and Hurst (2012) propose instead to modify the leg length during the flight phase, while Riese and Seyfarth (2012) consider the effect of modifying both leg length and spring stiffness during stance, limiting their analysis to the vertical hopping case. These added actuations have been used to reach different energy levels, meaning changing the velocity or height of the body, toward controlling step-size. While this prior work has relied on controlling both the touch-down angle and the actuation state during stance phase, in this paper we propose and evaluate an underactuated stance phase controller.

This work adds several contributions. First, we consider the active SLIP model and we determine the reachable state-space of the model using several different actuation strategies. Second, we investigate the variation of the reachable state with respect to the particular actuator motion, providing a relationship between direction of actuator displacement and location of the reached state. Third, we propose and evaluate an original numerical algorithm which aim at driving the system to a desired apex state. We conclude with a short discussion of possible applications.

The remainder of the paper is organized as follows. Section 2 reviews the characteristics of the active and passive SLIP models and their equations of motion, and underlines the differences between the two. Section 3 investigates and compares the shape of the reachability set for different actuator movements, including the effects of possible actuator's limitations. Section 4 considers in detail the evolution of the reachable state when activating the actuator during the stance phase, and Section 5 exploits those observations to introduce a control strategy to drive

the system to a desired apex state. In Section 6, performances of the given algorithm are discussed, and some examples provided. Finally, Section 7 presents a short discussion and conclusions.

2 SLIP model

2.1 Passive SLIP

The passive SLIP model consists of a point mass M, attached to a massless spring leg, L. Let us define k as the spring stiffness, L_k the length of the spring, and L_0 and $L_{k,0}$ the rest length of the leg and of the spring, respectively. The symbol θ is used to indicate the angle the leg forms with respect to the terrain, computed counterclockwise, and τ is the time.

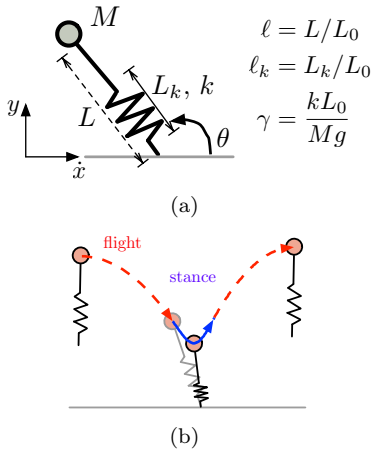


Figure 1: Subfig. 1(a) illustrates the passive SLIP model, and subfig. 1(b) shows the phases of motion.

For simplification purposes, we chose to reduce the system to its dimensionless counterpart, normalizing with respect to the gravity acceleration constant g, the leg length at rest L_0, and the mass M. Everything can then be written as a function of the leg length $\ell = L/L_0$, the spring length $\ell_k = L_k/L_0$, the dimensionless time $t = \sqrt{g/L_0}\tau$, and the relative spring stiffness $\gamma = \frac{kL_0}{Mg}$. Also, let us define x and y as the horizontal and vertical coordinates of the point mass, as shown in Fig. 1(a), such that $x = \ell\cos\theta$ and $y = \ell\sin\theta$. The function $\dot{(\cdot)}$ is the derivative with respect to the dimensionless time: $\dot{(\cdot)} = \frac{d}{dt}(\cdot)$.

The SLIP model can be considered a hybrid system with its motion divided in two parts: the *flight phase* and the *stance phase*. Flight phase is defined as the portion of the motion where there is no contact with the ground. The motion is governed by gravity only, where

$$\ddot{y} = -1, \quad \ddot{x} = 0.$$

The stance phase is defined by the contact between the terminal point of the leg (the foot) and the ground. The spring affects the motion only during ground contact, via a compression followed by an extension from its equilibrium length, $l_{k,0}$. The Lagrangian for the stance phase is:

$$Lag = \frac{1}{2}(\ell^2\dot{\theta}^2 + \dot{\ell}^2) - \ell\sin\theta - \frac{\gamma}{2}(\ell_k - \ell_{k,0})^2,$$

which results in the following equations of motion:

$$\ddot{\ell} = -\gamma(\ell_k - \ell_{k,0}) - \sin\theta + \ell\dot{\theta}^2. \tag{1}$$

$$\ddot{\theta} = \frac{-2\dot{\ell}\dot{\theta} - \cos\theta}{\ell}. \tag{2}$$

The *apex state* is defined as the highest point of the flight phase, characterized by $\dot{y} = 0$. Then, the apex state is a 3-dimensional vector $\{x, y, \dot{x}\}$.

The points that mark the transition between stance and flight phase are called *touch-down*, which is the instance between flight and stance characterized by first contact between the leg and the ground; and *take-off*, which separates stance from flight and corresponds to the first instance at which the leg leaves the ground. For the rest of this paper we will define *jump* the transition from one apex to the next, and we will use the subscripts TD and TO to refer to values at touch-down and take-off, respectively.

2.2 Active SLIP

The classic SLIP is a completely passive model, meaning that it is not possible to add or remove energy to/from the system. While this does not necessarily pose a disadvantage, there are situations when it would be beneficial to be able to modify the amount

of energy in the system. For example, when hopping up and down a stair step like terrain. This lack of actuation limits the reachable system states.

One approach to overcome these limitations is to add a piston-like linear actuator to the passive SLIP model, ℓ_{act}, placed in series with the spring, as shown in Fig. 2. We will then refer to this model as to the active SLIP. $\ell_{act,0}$ defines the initial/equilibrium value of the actuator. During flight, any movement of the actuator does not have an effect on the system's dynamics, while during the stance phase, the actuator can either compress ($\ell_{act} > \ell_{act,0}$) or extend ($\ell_{act} < \ell_{act,0}$) the spring, thus respectively adding or removing energy to/from the system. The dynamics of the leg length during stance phase, (1), become:

$$\ddot{\ell} = -\gamma(\ell_k - \ell_{k,0} - \ell_{act}) - \sin\theta + \ell\dot{\theta}^2.$$

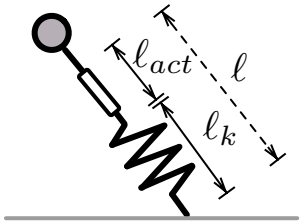


Figure 2: Schematics of the active SLIP model.

Displacing the actuator during the stance phase affects all three dimensions of the next apex state reached. However, any leg actuation strategy applied to its motion can only control two of the three dimensions simultaneously, as will be shown in Section 3. For this reason, in this work we choose to consider the apex height and forward velocity only, disregarding the forward position x, which is useful mostly in path planning scenarios. Hence, we will define the reduced apex state as $S = \{y,\, \dot{x}\}$. Let us then define the following function, $\mathcal{X}$, as the transition from one apex to the next:

$$\mathcal{X}(S_0,\, \theta_{TD}, \ell_{act}(t)) = S_1,$$

where S_1 is the apex state reached in one jump from the initial apex state S_0, with touch-down angle θ_{TD} and actuator value function during stance phase $\ell_{act}(t)$.

3 Reachability

In this section, we want to answer the following question: starting from an initial apex state, to which other apex states is it possible to drive the system, when moving the actuator during the stance phase?

3.1 Reachable space

Starting from an initial apex state $S_0 = \{y_a,\, \dot{x}_a\}$, we call the *reachable space* of S_0 the set of all apex states that can be reached from S_0 in one jump by positioning the leg and moving the actuator:

$$\mathcal{R}(S_0) := \{S_1 \mid \exists\, \theta_{TD},\, \ell_a(t) : \mathcal{X}(S_0, \theta_{TD}, \ell_a(t)) = S_1\},$$

where $\theta_{TD} \in [\pi/2,\, \pi]$, and $\ell_{act}(t)$ is a continuous function bounded by the physical limitations of the actuator, such as limited travel and velocity and torque limits.

In this work, we are interested in determining how the actuator's displacement affects the reachable space. Our study does not aim to propose a leg-placement strategy, but rather to understand the effect that the actuation has during the stance phase. Therefore, we limit our analysis to a reduced reachable space, restricted to the case where the touchdown angle, θ_{TD}, is fixed:

$$\mathcal{R}(S_0, \theta_{TD}) := \{S_1 \mid \exists\, \ell_a(t) : \mathcal{X}(S_0, \theta_{TD}, \ell_a(t)) = S_1\}.$$

Obviously, $\mathcal{R}(S_0, \theta_{TD}) \subseteq \mathcal{R}(S_0)$.

For the passive SLIP model, the reachable space is a point in the $\{y,\, \dot{x}\}-$space. Adding the series actuator has the effect of extending the reachable set to a 2-d surface, whose shape is determined by the "shape" of the function $\ell_{act}(t)$ during the stance phase. A few leg-actuation functions have been proposed and can be found in the literature. Among them, we chose two functions that we consider relevant because they provide an insightful representation of how different leg-actuation motions can affect the locus of all the points that can be reached in one step. Indeed, we want here to compare their reachability, to determine the best course of action when choosing an actuation strategy.

In Schmitt and Clark (2009), the authors propose a leg-length actuation inspired by studies on the net energy production of muscles groups in humans:

$$\ell_{act}(t) = \ell_{act,0} - \ell_{dev} \sin(\omega t), \tag{3}$$

where $\ell_{act,0}$ is the initial actuator length, ℓ_{dev} is the amplitude of the variation of the actuator length from its nominal length, and ω is the frequency of the actuator's motion. t is the time from the beginning of the stance phase, and $t = 0$ corresponds to the touch-down time. Let us use $\mathcal{R}_s(S_0, \theta_{TD})$ to indicate the reachable space computed with the actuator movement described in (3).

In Piovan and Byl (2013), the stance phase is divided in two parts marked by the point of maximal spring compression (which is equivalent to $\dot{\ell}(t) = 0$). The actuator is then moved from its initial position to a desired value $\ell_{c,1}$ during the first half of the stance phase, and a desired value $\ell_{c,2}$ during the second half:

$$\ell_{act}(t) = \begin{cases} \ell_{c,1}, & \forall t : \dot{\ell}(t) \leq 0 \\ \ell_{c,2}, & \forall t : \dot{\ell}(t) \geq 0. \end{cases} \tag{4}$$

According to the finite response of a physical actuator, the actuator does not move instantaneously, but it is assumed to move with its maximum allowable velocity and acceleration. Let $\mathcal{R}_p(S_0, \theta_{TD})$ be the reachable space computed with the actuator movement described in (4).

How do the reachable spaces $\mathcal{R}_s(S_0, \theta_{TD})$ and $\mathcal{R}_p(S_0, \theta_{TD})$ compare? And, how do they compare with respect to all the possible apex states reachable in one step for other actuator's motions? To answer to these questions, we compare the two reachable sets with the set of all reachable states obtained by moving the actuator at any time during the stance phase in any direction. Let us call this space $\mathcal{R}_{tot}(S_0, \theta_{TD})$. Fig. 3 shows the reachability space of the three methods considered, starting from the same initial conditions at touch-down, and with the same actuator's limitation. To take into account the limitations of a real-life actuator, we model the actuator dynamics as:

$$\ell_{act}(t) = \ell_{act}(t_i) + \dot{\ell}_{act}(t_i)(t - t_i) + k_{acc}(t - t_i)^2, \tag{5}$$

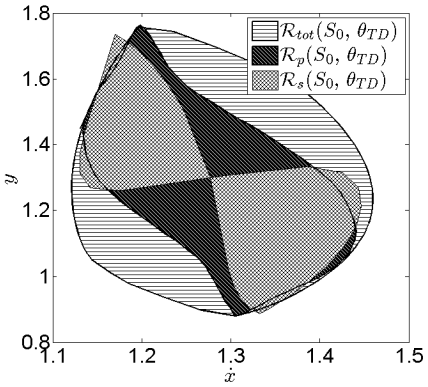


Figure 3: Reachable space of a passively symmetric jump $\mathcal{R}(S_0, \theta_{TD})$ computed for $S_0 = (1.3, 1.277)$, $\theta_{TD} = 57.855$ [deg], and $\gamma = 10$. Maximum actuator velocity allowed $v_{max} = 0.1596$ and maximum acceleration constant $k_{acc} = 1.0194$, which, for a leg length of $L_0 = 1$ [m], correspond to 0.5 [m/s] and 10 [m/s^2], respectively. The "corner" points are the points reached with maximum actuator movement in the positive (upper corner) or negative (lower corner) direction. Defining A_{tot}, A_p and A_s the areas of, respectively, $\mathcal{R}_{tot}$, $\mathcal{R}_p$, and $\mathcal{R}_s$, we have that $A_p \cong 0.619 A_{tot}$, and $A_s \cong 0.445 A_{tot}$. Note that, while $\mathcal{R}_p \subset \mathcal{R}_{tot}$, this does not necessarily hold true for $\mathcal{R}_s$, since its actuator dynamics (3) start with nonzero velocity.

where k_{acc} is the acceleration constant, and t_i is the time at which movement first occurs. When the actuator reaches its maximum velocity allowed, v_{max}, it stops accelerating and proceeds with a constant velocity motion. Additionally, the maximum actuator displacement is assumed to be 10% of the leg length at equilibrium, L_0, and the actuator is assumed to start its motion with zero velocity and displacement. In this particular example, initial conditions were chosen to achieve a passive symmetric jump, i.e., $\chi(S_0, \theta_{TD}, 0) = S_0$ with no actuator movement. Note that, due to the lack of closed-form solution for the stance dynamics of the SLIP model, the reachable spaces were numerically computed. This operation generally requires significant

computational effort; hence, it is not normally feasible to compute in real time the reachability set for any particular initial condition.

As we can see, the area of $\mathcal{R}_{tot}$ is bigger than the areas of $\mathcal{R}_s$ and $\mathcal{R}_p$. Furthermore, the convex ellipsoid-like shape of $\mathcal{R}_{tot}$ presents the advantage of reaching any point close to a desired state. This suggests that, in order to reach a wider range of apex states, it is beneficial to consider a strategy that involves updating the actuator movement throughout the entire stance phase, instead of choosing a priori some fixed parameters.

Consider also that the reachability space is a function of the actuator's characteristics, such as maximum and minimum compression and extension, and maximum velocity and acceleration allowed. Fig. 4 compares the complete reachability space for different values of the actuator's acceleration constant, k_{acc}. As one might expect, the higher the acceleration, the bigger the reachability space. Correspondingly, in the strategy proposed in this work, we will focus on moving the actuator with its maximum allowed parameters.

As mentioned in Section 2.2, we consider here only two of the three dimensions of the apex state: apex height, y, and forward velocity, $\dot{x}$. As we claimed earlier, the reason behind this choice is the fact that any actuation movement applied during the stance phase can only control two of three dimensions. Indeed, we can see in Fig. 5(a) and 5(b), the apex states $\{x,\, y,\, \dot{x}\}$ reachable in one step from a fixed touch-down angle form a 2-dimensional manifold, suggesting that the three coordinates are not independent from each other. While an interesting topic, we do not provide a characterization of this manifold, since it is beyond the goal of this work. It is worth noting that changing the angle at touch-down inevitably changes the reachability space. It is than more precise to say that the reachability space for any initial condition consists of a set of 2-dimensional manifolds, one for every touchdown angle employed, forming a fan-shaped 3-dimensional volume, as shown in Fig. 6(a) and 6(a).

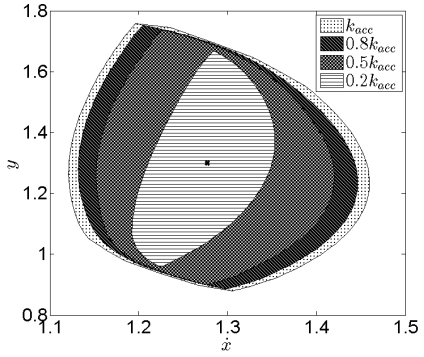


Figure 4: Reachability space for a passively symmetric jump with actuator moving at different acceleration values, given a maximum allowed velocity of $v_{max} = 0.1596$, and maximum acceleration constant of $k_{acc} = 1.0194$. The black cross represents the apex state passively reachable (i.e., without any actuation). From this graph we can see that the higher the maximum acceleration allowed, the bigger the reachability space is, in terms of area. Initial conditions chosen: $\gamma = 10$, initial apex state $S_0 = \{1.3,\, 1.277\}$, touch-down angle $\alpha_{TD} = 57.855$ [deg].

4 Actuator movement

In this section, we aim to understand and characterize the effects that actuation motion has on the next apex state reached. We will show the relationship between moving the actuator at different times during the stance phase and the variation of the apex state thereby reached.

Our objective is to find an actuator movement strategy to (i) move the system to a desired apex and (ii) counteract the effects of perturbations (on terrain height or on initial position at apex). As previously stated in Section 2.2, compressing or extending the actuator during the stance phase corresponds to an increase or decrease of the system's energy: the additional potential energy stored in the spring through actuation will then be transformed into additional ki-

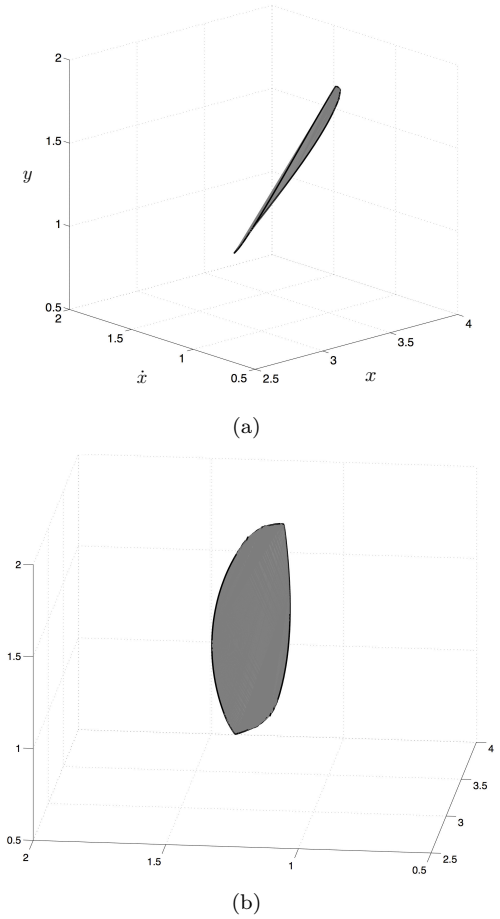


Figure 5: Subplots (a) and (b) show two different views of the 3-dimensional reachable space in one step for a fixed touch-down angle. As we can see, the set is a 2-dimensional manifold.

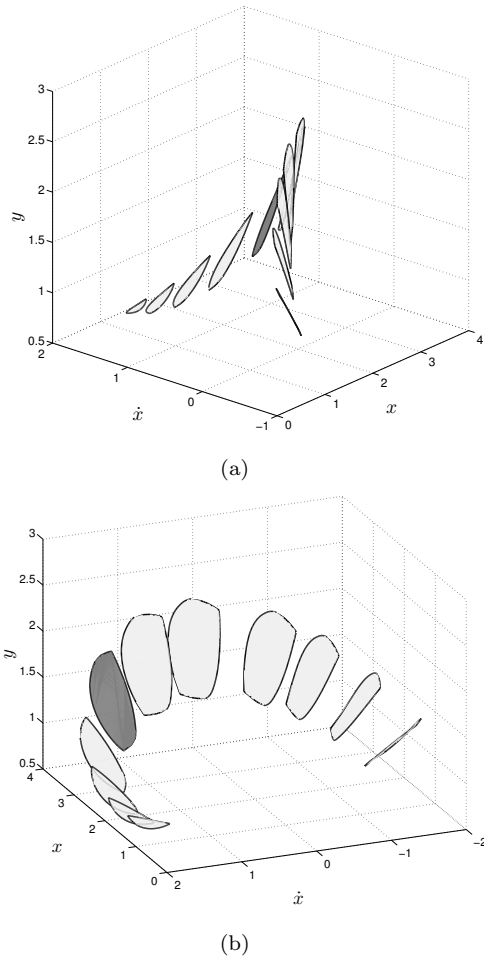


Figure 6: Subplots (a) and (b) show two different views of the 3-dimensional reachable space in one step for varying touch-down angles. The dark grey manifold corresponds to the reachability space in Fig. 5(a) and 5(b).

netic energy throughout the stance phase, until take-off, and vice versa. However, the system's energy variation is a function of the values of the states of the system when actuation is performed. Therefore, to drive the system from an initial state $S_0 = \{y_0, \dot{x}_0\}$ to a desired apex state, $S_{des} = \{y_{des}, \dot{x}_{des}\}$, it is crucial to understand the effects that the particular time during stance when actuation happens has on the evolution of the states, and, therefore, on the position of the reached apex.

The lack of closed-form solution for the system's dynamics constrains us to perform a numerical study. Toward generality in this particular analysis, we consider a step-type actuation, i.e., we assume that the actuator can be instantaneously moved to reach a

specified value, $\ell_{act,des}$:

$$\ell_{act}(t) = \begin{cases} 0, & \forall t < t_s \\ \ell_{act,des}, & \forall t \geq t_s \end{cases} \quad (6)$$

where t_s is the time at which the step actuation occurs. From an inital apex state and touchdown angle, we computed the passively-reached state, i.e., the state reached without actuation $S_{pass} = \{y_{pass}, \dot{x}_{pass}\}$. Given a positive and negative step actuation, we computed the apex reached as a function of the time at which the positive or negative actuation was applied: $S_r(\ell_{act,des}, t_s) = \{y(\ell_{act,des}, , t_s), \dot{x}(\ell_{act,des}, t_s)\}$. Fig. 7 shows an example of the variation of the two components of the reached apex with respect to the passively-reached apex, as a function of the time t_s at which the step actuation is applied:

$$\Delta y_r(\ell_{act,des}, t_s) = y_r(\ell_{act,des}, t_s) - y_{pass},$$
$$\Delta \dot{x}_r(\ell_{act,des}, t_s) = \dot{x}_r(\ell_{act,des}, t_s) - \dot{x}_{pass}.$$

As we can see, the variation of y_r follows the sign of the step input: any positive actuation will result in an increase of the next apex height, whereas any negative actuation will result in its decrease, regardless of the time at which the actuation is performed. Dissimilarly, the evolution of $\dot{x}_r$ initially increases or decreases according to the sign of the actuation, and then increases of decreases opposite of the sign of the actuation. By lacking analytical solution for the stance phase, it has not been possible to pinpoint the exact time at which the variation of apex state happens. However, we can see that the change in direction for $\Delta \dot{x}_r$ happens when the actuator moves close to the time corresponding to the point of half stance with respect to the spring movement, i.e., when the spring reaches its maximum compression. We can then summarize the above observations as:

- First half:
  - $\dot{\ell}_{act} > 0$: $y \uparrow$, and $\dot{x} \downarrow$,
  - $\dot{\ell}_{act} < 0$: $y \downarrow$, and $\dot{x} \uparrow$,
- Second half:
  - $\dot{\ell}_{act} > 0$: $y \uparrow$, and $\dot{x} \uparrow$,
  - $\dot{\ell}_{act} < 0$: $y \downarrow$, and $\dot{x} \downarrow$.

Another important insight we can gather from Fig. 7 concerns the different rates at which y and $\dot{x}$ vary throughout stance. While providing actuation during the first half of stance has the biggest effect on the variation of the $\dot{x}$ component of the next reached apex state, the second half has little effect on it. On the contrary, the y component is influenced throughout the entire stance phase, with maximum variation during the second part. This is an indicator and a confirmation of our idea that varying the actuator throughout stance gives the greatest control authority to reach a desired state, compared to choosing any fixed time during stance at which to activate the actuation. This, for example, also explains and is confirmed by the bowtie-like shape of the reachability set $\mathcal{R}_s$ in Fig. 3: in order to modify y without much variation of the otherwise passively reached value of $\dot{x}$, the actuation should mostly be applied during the first half of the stance phase. However, the choice

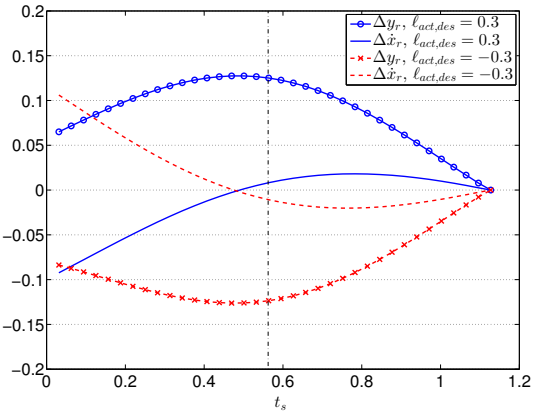


Figure 7: This figure shows the variation of Δy_r and $\Delta \dot{x}_r$ as a function of the time t_s at which the step actuation is applied to the system. The blue lines refer to a positive actuation $\ell_{act,des} = 0.3$, while the red lines refer to a negative actuation $\ell_{act,des} = -0.3$. The vertical dotted line signals the time at which the spring reaches its maximum compression.

of the sinusoidal function (3) with constant amplitude and frequency, distributes the actuation almost evenly throughout the entire stance phase, precluding the possibility of favoring one part of stance with respect to another.

These observations consider only one-step actuator movement during stance, but can be generalized to the case of subsequent actuation movements, with more realistic dynamics, as for example (5). We accomplish this by updating the actuator value every few time-steps during the stance phase. Let us define t as the time elapsed from the beginning of the stance phase (where $t_0 = 0$ is the touch-down time), and divide the stance phase in time intervals of length δt. Let $\tilde{S}_n$ be the apex state that would be reached if the actuator were to be moved as a certain function $f_n(t)$ from touch down to time $n\delta t$, and then stopped at the current value for the rest of the stance phase:

$$\tilde{S}_n \longrightarrow \ell_{act} = \begin{cases} f_n(t), & \text{if } t \in [t_0,\, n\delta t] \\ f_n(n\delta t), & \text{if } t > n\delta t \end{cases}$$

Then, $\tilde{S}_{n+1}$ will be the apex state reached if the actuator were to be moved as:

$$\tilde{S}_{n+1} \longrightarrow \ell_{act} = \begin{cases} f_{n+1}(t), & \text{if } t \in [t_0,\, (n+1)\delta t] \\ f_{n+1}((n+1)\delta t), & \text{if } t > (n+1)\delta t \end{cases}$$

where $f_{n+1}(t) = f_n(t)$, for $t \in [0,\, n\delta t]$.

Fig. 8(a) and 8(c) show the position of the next apex, $\tilde{S}_n$, when moving the actuator as respectively in Fig. 8(b) and 8(d) for values of $n = 1, 2, \ldots$ until take-off. A time interval of length $\delta t = 0.0626$ has been chosen (which, in the case of a leg length $L_0 = 1$ [m], corresponds to 0.02 [s]). As we can see, and according to earlier observations, at each time step n, compressing the spring results in an increase of y_n, while extending the spring results in a decrease of y_n with respect to its value at the $(n-1)-$th step. Conversely, any spring compression translates in a decrease or increase of $\dot{x}_n$ if this happens, respectively, during the first or second half of the stance phase. Vice versa, an extension of the spring applied during the first or second half of the stance phase results in an increase or decrease, respectively, of $\dot{x}_n$. This is summarized graphically in Fig. 9.

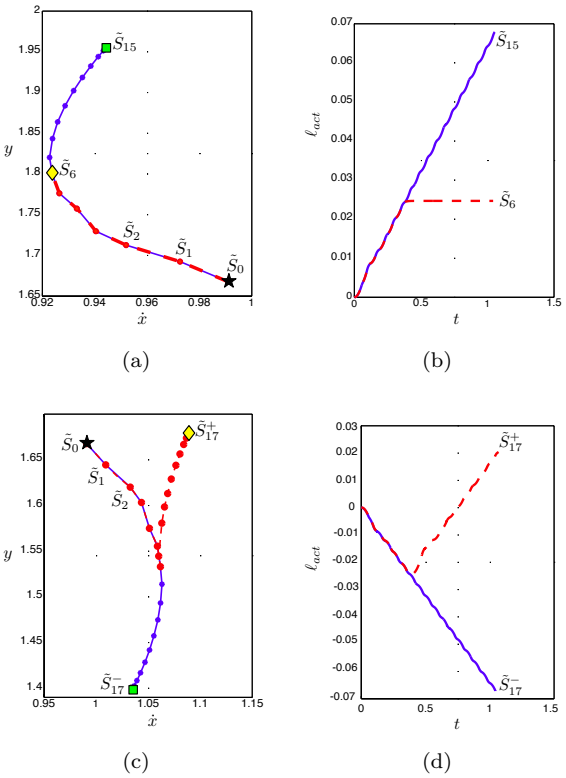


Figure 8: Subfig. 8(a) and Subfig. 8(c) show the next apex states $\tilde{S}_n$, $n = 1, 2, \ldots$ until take-off, for the different actuation motions shown in Subfig. 8(b) and 8(d). The blue line and the dotted red line in Subfig. 8(a) and 8(c) are the apex states reachable if the actuator motion follows the solid blue and dotted red trajectories in Subfig. 8(b) and 8(d), respectively. The black star symbol represents $\tilde{S}_0$, i.e., the passively reached state. The yellow diamonds represent the apex states reached following the dotted red actuator trajectory in Subfig. 8(b) and 8(d), while the green squares represent the apex states reached if the actuator follows the solid blue trajectory in Subfig. 8(b) and 8(d).

5 Control Strategy

We introduce here our control strategy to reduce the error between the reached and a desired apex state,

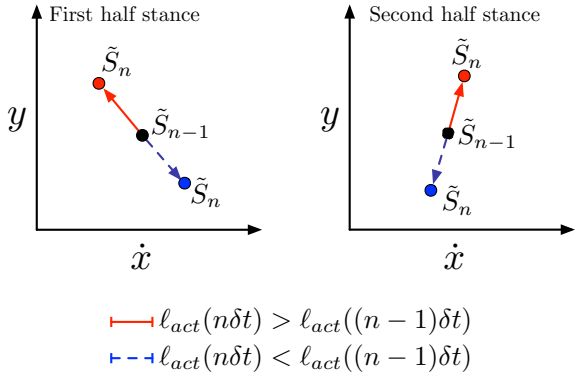


Figure 9: Position of the apex state that would be reached providing a positive or negative actuation during the following interval $[(n-1)\delta t,\, n\delta t]$. Note that the slope of the directional curve from $\tilde{S}_{n-1}$ to $\tilde{S}_n$ is a function of the particular actuation motion, the system's parameters and the initial condition $\tilde{S}_{n-1}$.

first giving an example, and then explaining in detail our proposed algorithm.

The objective of our control strategy can be summarized as follows.

Let $S_0 = \{y_0,\, \dot{x}_0\}$ be the dimensionless initial apex state, and θ_{TD} the angle of the leg at touch-down. Let us define $S_{des} = \{y_{des}, \dot{x}_{des}\}$ as the dimensionless desired apex state, and $S_r = \{y_r,\, \dot{x}_r\}$ the apex state reached with a certain actuation motion. Then, we want to find an actuation motion strategy to reduce the Euclidean distance between the desired apex and the actual apex reached:

$$Err = \parallel S_{des} - S_r \parallel_2 . \tag{7}$$

We explain here our proposed strategy to minimize error during stance.

During the stance phase, let us define each time interval of size δt as $\Delta t_n = [n\delta t,\, (n+1)\delta t]$, $n = 1,\, 2,\, \ldots$ until take-off. Our control strategy for actuator movement is to update the actuator's value every time-step δt of the stance phase, according to predictions of next apex state based on the current state. This means that at each time interval Δt_n, a new value for the actuator displacement at the next time interval, $\ell_{act}(\Delta t_{n+1})$, is computed.

At each interval Δt_n, we can compute the apex state, S_n, that the SLIP model would reach if the actuator remains at the current value (computed at Δt_{n-1}):

$$\ell_{act}(t) = \ell_{act}((n-1)\delta t), \quad \forall t \geq n\delta t,$$

i.e., the apex state reachable without further actuator movement. This is motivated by the observations highlighted in Section 4, i.e., that activating the actuator during the first part of the stance phase has the effect of increasing or decreasing the value of the forward height at the next apex, y, while at the same time decreasing or increasing, respectively, the value of the next apex's forward velocity, $\dot{x}$. By contrast, moving the actuator during the second half of the stance phase has the effect of increasing or decreasing the values of both y and $\dot{x}$ at the next apex state. Hence, to determine how to control the actuator's displacement at the next time step, it is imperative to determine the position of S_n with respect to the position of the desired apex state S_{des}. At the same time, the choice of the next value for the actuator, $\ell_{act}(\Delta t_{n+1})$, is computed predicting the position of S_{n+1} when moving the actuator for one step Δt to one direction or the other, i.e., compressing or extending the spring.

For example, let us assume that the relative position of S_{des} with respect to S_{pass} is given in Fig. 10 *I*. From what is graphically summarized in Fig. 7 and Fig. 9, we can already expect that the actuator motion necessary to reach the desired state will consist whether in an initial spring extension or compression, followed by a compression during the second half of the stance phase. What we aim to accomplish with our algorithm, is to control the amount of the desired compression/extension, by updating its value throughout the entire stance phase. At each time-step during the first half of the stance phase, we will compute the next apex reached if the actuator were to be compressed or extended during the next time-step, Fig. 10 *II*, kept at the reached value for the rest of the first half of the stance phase, and subsequently compressed during the entire second half of the stance

phase, as in Fig. 10 *III*. By computing and averaging the distance of the resulting apex states to the desired state, we can compute how much to move the actuator at the next step, Fig. 10 *IV*. Once the dynamics enter into the second half of the stance phase, Fig. 11 *I*, we can start compressing the actuator: we compute the states the system would reach if the actuator were to move at its maximum and at half its capabilities (e.g., actuator acceleration) at the next time step, and then kept at the current value for the rest of the stance phase, Fig. 11 *II*. Based on an average of the distance of these two predicted states with respect to the desired apex, the actuator's action to implement at the following time step can be computed, Fig. 11 *III*. Final result of our algorithm is shown in Fig. 11 *IV*.

Since at each step the algorithm predicts where the next apex state would be when displacing the actuator in the following time step and in the second half of the stance, the algorithm is essentially creating approximate "grid lines" (see Fig. 10 *III*) that are based on limited numerical simulations of the stance phase. Computing more intermediate states, i.e., higher order fits for the dynamics of the system subjected to actuation, would certainly result in a better estimate, but would require more computational time. For applicability, we want to ensure that the algorithm finishes well within the time δt between actuation updates. As a result, an error is introduced in the algorithm. Such error is the greatest during the first half of the stance phase, when we have more uncertainty about the direction the motion will go, and results in the bouncing motion of the running estimate of the next apex state, seen in Fig. 11 *IV*. Fortunately, the first half of the stance is also characterized by a higher time-to-apex, allowing for adequate time to significantly reduce the approximation error.

5.1 Algorithm

Let us call $\ell^m_{act}(\Delta t_i)$ and $-\ell^m_{act}(\Delta t_i)$ the maximum actuator movements possible in the time interval Δt_i with respect to the actuator's initial position and its specifications, respectively in the positive (spring compression) and negative (spring extension) direction. Let $\ell^{end}_{act}(\Delta t_i)$ be the value that the actuator assumes at the end of Δt_i.

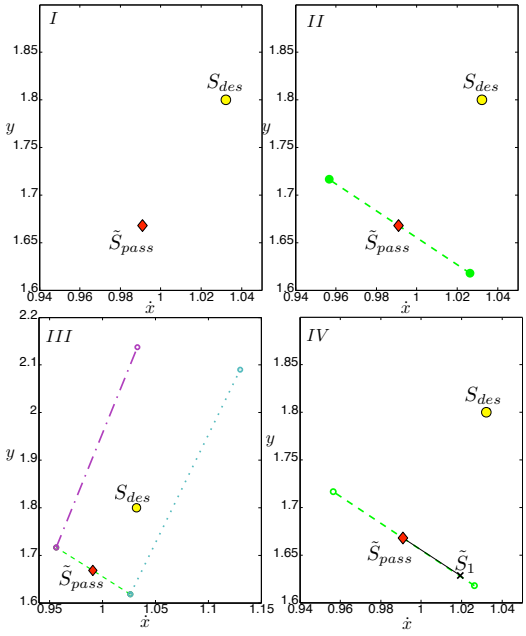


Figure 10: These plots show an example of the steps implemented during the first half of the stance phase in order to approach the desired apex state.

At each time interval Δt_n, with $n = 0, 1, \ldots,$ until the end of the stance phase:

(i) Compute the apex state, $\tilde{S}_{n+1}$, that the SLIP model would reach if keeping the actuator at the current value :

$$\ell_{act}(t) = \ell^{end}_{act}(\Delta t_n), \quad \forall t \geq (n+1)\delta t,$$

i.e., the apex state reachable without further actuator movements.

- If $\dot{\ell}(t) \leq 0$ (first half of stance phase):

  (iiA) Compute the apex state that the SLIP model would reach if we were to control the actuator's movement at its maximum negative or positive motion for a time interval Δt, with no further actuation movement occurring for the rest of the stance

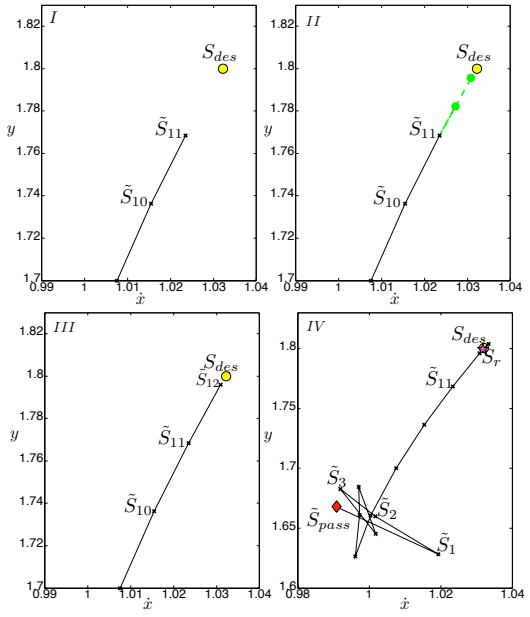


Figure 11: Plots I to III show an example of the steps implemented during the second half of the stance phase in order to approach the desired apex state. Plot IV provides a summary of the steps implemented during the entire stance phase.

phase.

$$\ell_{act}(t) = \begin{cases} \pm\ell^m_{act}(\Delta t_{n+1}), & \forall t \in \Delta t_{n+1} \\ \ell^{end}_{act}(\Delta t_{n+1}), & \text{otherwise.} \end{cases}$$

Let us call these points S^+_{n+1} and S^-_{n+1}.

(iiiA) Compute the apex state that the SLIP model would reach if we were to move the actuator with its maximum negative or positive motion for a time interval Δt and then stop until the beginning of the second half of the stance phase. Then, the actuator moves at its maximum negative or positive motion until the end of the stance phase.

$$\ell_{act}(t) = \begin{cases} \pm\ell^m_{act}(\Delta t_{n+1}), & t \in \Delta t_{n+1} \\ \ell^{end}_{act}(\Delta t_{n+1}), & \forall t : \dot{\ell}(t) \leq 0 \\ \xi\ell^m_{act}(\Delta t_{n+1}), & \forall t : \dot{\ell}(t) > 0, \end{cases}$$

The direction of movement during the second half ($\dot{\ell}(t) > 0$) is dictated by the position of S_{n+1} with respect of S_{des}, as shown in Fig. 12(a). In fact, during the second half of the stance phase, a positive actuation ($+\ell^m_{act}$) would increase both y and $\dot{x}$ values at the next apex, while a negative actuation ($-\ell^m_{act}$) will result in a decrease of y and $\dot{x}$. Therefore, if S_n is in zone I (see Fig. 12(a)), then $\xi = 1$; else, $\xi = -1$. Let us call these points P^+_{n+1} and P^-_{n+1}.

(ivA) Consider the line segments $s^+ = (S^+_{n+1}, P^+_{n+1})$ and $s^- = (S^-_{n+1}, P^-_{n+1})$, and compute the vector distance between the segments and the desired apex state S_{des}:

$$d^+ := \text{vector distance}(s^+, S_{des}),$$
$$d^- := \text{vector distance}(s^-, S_{des}).$$

(vA) Now, Compute the next actuator value desired, $\ell_{act}(\Delta t_{n+1})$, as follows.

$$\ell_{act}(\Delta t_{n+1}) = \begin{cases} \mu\ell^m_{act}(\Delta t_{n+1}), & \text{if } |d^+| \leq |d^-| \\ -\mu\ell^m_{act}(\Delta t_{n+1}), & \text{if } |d^-| \leq |d^+| \end{cases}$$

where the parameter μ is a weight $\in [0, 1]$. In particular, if the two vectors d^+ and d^- have a positive dot product, then $\mu = 1$, as shown in Fig. 12(c) and 12(d). Otherwise, the desired apex point is "between" the two line segments, and the required actuator movement will be a fraction of the previous predictions, as shown in Fig. 12(b):

$$\mu = \left| \frac{|d^-| - |d^+|}{|d^-| + |d^+|} \right|.$$

- If $\dot{\ell}(t) > 0$ (second half of the stance phase):

  During the second part of the stance phase, both $\dot{x}$ and y functions are increasing or decreasing, as seen in Fig. 7. Therefore, the choice of whether

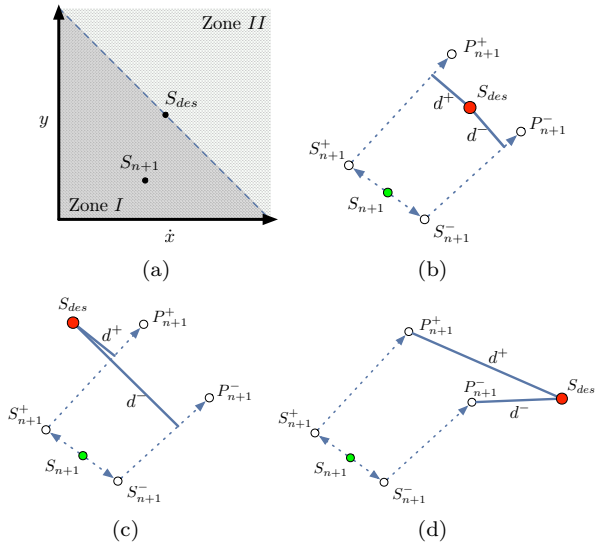


Figure 12: Subfig. 12(a) shows the two dividing zones where S_{n+1} can be with respect to the position of S_{des}. If S_{n+1} is in zone I, during the second half of stance the actuator displacement needs to increase; else, it needs to decrease. The algorithm creates approximate grid lines (dotted blue line segments) based on limited numerical simulations of the stance phase: only the states S^+_{n+1}, S^-_{n+1}, P^+_{n+1}, and P^-_{n+1} are computed. The actuator motion during the first half of stance is chosen based on the distance and location of the line segments with respect to the desired state: Subfig. 12(b), 12(c), and 12(d) demonstrate possible location of vectors d^+ and d^- with respect to S_{des}. Computing more states would result in a better fit for the direction along which the dynamics of the system move when subjected to actuation, which is in general not a straight line, and as a consequence it would result in a more accurate choice of actuator motion. However, this would be at the expense of the computational time, and could preclude the use of the algorithm in real time.

moving the actuator in the positive or negative direction is again done based on the position between S_{n+1} and S_{des}, and it is the same as the value of ξ computed in (iii) .

(iiB) Compute the apex state that the SLIP model would reach if we were to control the actuator's movement at its maximum velocity for the time interval Δt_{n+1}. Then, no further actuation movement occurs for the rest of the stance phase.

$$\ell_{act}(t) = \begin{cases} \xi \ell^m_{act}(\Delta t_{n+1}), & \forall t \in \Delta t_{n+1} \\ \ell^{end}_{act}(\Delta t_{n+1}), & \text{otherwise.} \end{cases}$$

Let us call this point P^+_{n+1}.

(iiiB) Compute the apex state that the SLIP model would reach if we were to control the actuator's movement at half of its maximum (positive or negative) displacement possible for a time interval Δt_{n+1}. Then, no further actuation movement for the rest of the stance phase.

$$\ell_{act}(t) = \begin{cases} \xi \frac{1}{2} \ell^m_{act}(\Delta t_{n+1}), & \forall t \in \Delta t_{n+1} \\ \ell^{end}_{act}(\Delta t_{n+1}), & \text{otherwise.} \end{cases}$$

Let us call this point P^-_{n+1}.

(ivB) Consider the line segments $s^+ = (S_{n+1}, P^+_{n+1})$ and $s^- = (S_{n+1}, P^-_{n+1})$. Compute the vector distance between the segments and the desired apex state S_{des}:

$$v^+ := \text{vector distance}(s^+, S_{des}),$$
$$v^- := \text{vector distance}(s^-, S_{des}).$$

(vB) Now, compute the next actuator value desired, $\ell_{act}(\Delta t_{n+1})$, as follows.

$$\ell_{act}(\Delta t_{n+1}) = \begin{cases} \mu \ell^m_{act}(\Delta t_{n+1}), & \text{if } |v^+| \leq |v^-| \\ \mu \frac{1}{2} \ell^m_{act}(\Delta t_{n+1}), & \text{if } |v^-| \leq |v^+| \end{cases}$$

where the parameter μ is a weight $\in [0, 1]$. In particular, if both or neither vectors are orthogonal to the respective line segment s^+ or s^-, then $\mu = 1$, as shown in Fig. 13(b) and 13(c). Else, the desired apex point is "between" the two

line segments, and the required actuator movement will be a fraction of the previous predictions, as in Fig. 13(a):

$$\mu = \left| \frac{|v^-| - |v^+|}{|v^-| + |v^+|} \right|.$$

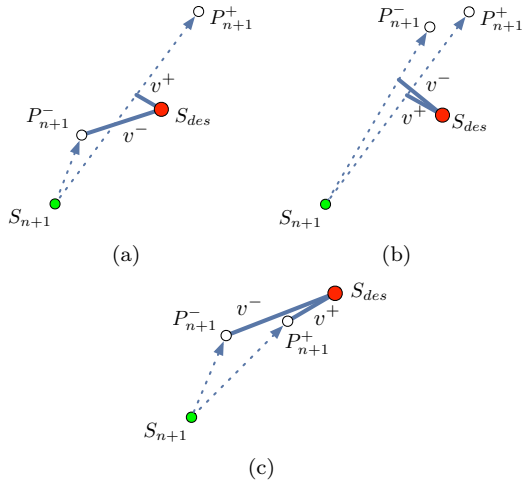


Figure 13: These figures demonstrate possible locations of the vectors v^+ and v^- with respect of S_{des}.

(vi) repeat (i) to (v) until the end of the stance phase.

Note that the choice of moving the actuator with its maximum velocity and acceleration allowed, derived from the fact that a higher velocity and acceleration correspond to a bigger reachability space (e.g., see Fig. 4 where reachability sets computed for different values of maximum velocities are compared). The algorithm can of course be implemented with any other velocity and acceleration values, provided they are in accordance with the limitations of the specific actuator used.

6 Performance

We show here the performances of our algorithm. In particular, we start with considerations on the required time delay for the algorithm implementation and the time interval used, and we continue with simulation results showing how effective the algorithm is in reducing the distance to the desire state.

6.1 Time interval and delay at first computational instance

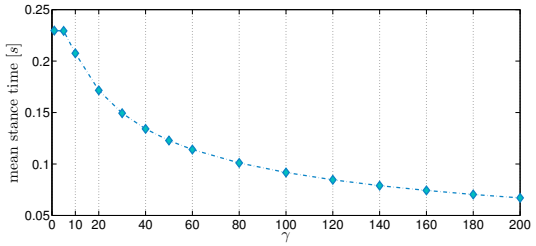
The choice of the time interval $\delta\tau$ (or, in its dimensionless version, δt), determines the frequency at which the algorithm is updated. Any control action applied in real-time during the stance phase needs to be executed in finite time, which corresponds to the duration of the stance phase itself. The execution of one update of the algorithm is only a function of the processor used, and therefore the number of iterations of the algorithm depends on the duration of the stance phase, which in turn is a function of the relative spring stiffness γ: higher values of γ correspond to a stiffer spring, and therefore a shorter stance phase; conversely, lower values of γ correspond to a longer stance phase caused by a softer spring. Let us choose initial conditions and parameters based on biological data for a typical human, as in Table 1. Then, the average stance time is shown in Fig. 14(a)

Parameters	
$g =$	9.81 m/s^2
$M =$	80 kg
$\ell_0 =$	1 m
$\ell_{k,0} =$	0.5 m
$\ell_{act} \in$	$[-0.1, 0.1]$ m
$y \in$	$[1, 2.5]$ m
$\dot{x} \in$	$[0.4, 5]$ m/s

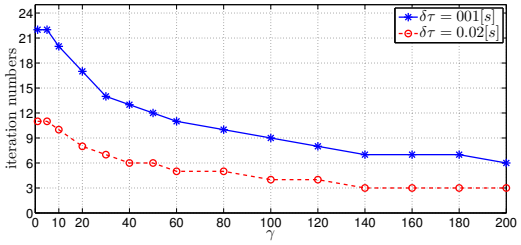
Table 1: Parameters based on biological data for a typical human.

as a function of the relative spring stiffness. Considering that on an average computer (Intel Core i7 eight core processor CPU, 2.8GHz) running Matlab R2012a, the average time to perform all the calculations required for one update of the algorithm is $\approx$ 0.01, 0.02 [s], one can ask what values of γ for a real robot are a good trade-off between number of algorithm iterations and considerations on biological inspiration. Fig. 14(b) shows how many algo-

rithm updates are possible on average during stance, for $\delta t = 0.01$ [s] and $\delta t = 0.02$ [s]. As expected, the smaller the value of γ, the more algorithm iterations can be performed. A study by Blickhan and Full (1993) shows that the relative spring stiffness assumes very similar values for various gaits in different animals (both quadrupeds and bipeds). For runners, values were found between $\gamma \in [7.1, 14.6]$, while $\gamma \in [7.7, 13.6]$ in hoppers. It is then reasonable to use values of $\gamma \in [8, 13]$, which correspond to a high number of algorithm iterations, validating the applicability of our control strategy.



(a)



(b)

Figure 14: Subfig. 14(a) shows the average stance time as a function of γ for a set of initial apex states and system's parameters as in Table 1. Subfig. 14(b) shows instead the corresponding number of algorithm iterations.

When implementing a control action, the time delay of such controller is usually a concern that requires considerations. In the case of the algorithm here proposed, each time interval Δt_n is dedicated to plan the desired actuation value at the next interval, $\ell_{act}(\Delta t_{n+1})$. This implies that at the initial time interval Δt_1 there will be no actuation; namely, the actuator will begin its motion from the second time-step. We want here to investigate how this will affect the performances of our algorithm. As one would expect, increasing time delay reduces the reachable space (Fig. 15) by limiting the maximum achievable actuator movement.

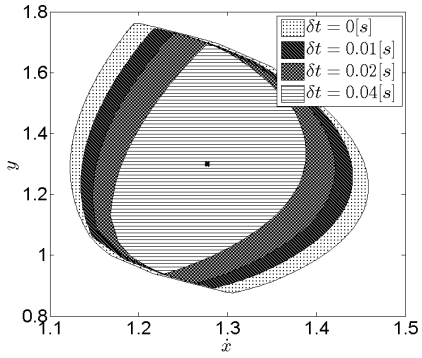


Figure 15: Reachability sets computed for different delay values. As expected, increasing δt reduces the reachable space.

The delay depends on the time interval chosen to perform our algorithm, δt, which in turn depends on the computational velocity of our computer. While the delay may seem a limitation, it is important to point out the following. On an average computer, the required (dimensional) time step is $\delta\tau \simeq 0.01$ [s]. In these circumstances, we can compare the reachable space with delay with the reachable spaces obtained with other control strategies (see Section 3), as shown in Fig. 16. As we can see, the reachability set with delay is still bigger (area-wise) than the case $\mathcal{R}_p$ and $\mathcal{R}_s$. Thus we conclude that even with a time delay, the proposed algorithm offers significant advantages to previous control strategies.

This performance study considers the ideal SLIP model, where the body is a point mass, and the leg is massless. Nonetheless, we expect to be able to utilize the same control strategy on a real platform with SLIP-like dynamics. Because of the nature of its dynamics, the real system will behave similarly to the ideal model when subjected to a piston-like ac-

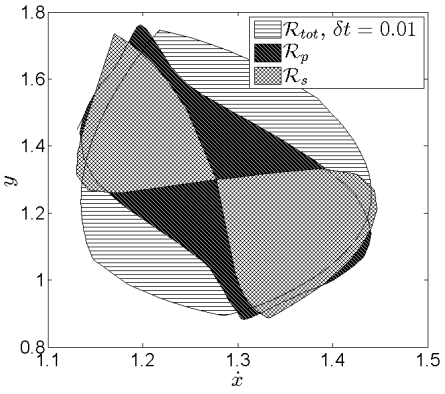


Figure 16: Reachability sets with $\delta\tau = 0.01$ delay, compared to $\mathcal{R}_p$ and $\mathcal{R}_s$.

tuation, e.g., it is reasonable to believe that a negative actuation will result in a decrease of the apex height. Additionally, at each time step throughout the stance phase, computations of the subsequent reachable state are performed numerically using equations of motion for a model of the system. Thus, the performance of the algorithm is strongly tied to how accurately the model that we construct of the physical system captures its actual dynamics.

6.2 Percentage change

We will evaluate our controller using equation (7). To quantify the reduction in Err, for an initial apex we introduce the percentage change, PC, as:

$$PC = 100\frac{\parallel S_{pass} - S_{des} \parallel_2 - \parallel S_r - S_{des} \parallel_2}{\parallel S_{pass} - S_{des} \parallel_2}, \quad (8)$$

where S_{des} is the desired apex state, S_{pass} is the state reached without actuation, and S_r is the apex state reached with our control strategy. Using percentage change as a performance index allows us to compare the system performance with and without control action over a broad range of desired states.

The performance of the controller has been tested as follows: an initial apex state S_0 and touch-down angle θ_{TD} are chosen, and a set of $10,000$ points are generated inside the corresponding reachability set without delay, $\mathcal{R}_{tot}(S_0, \theta_{TD})$. S_{pass} is defined as the passive apex state, i.e., the apex state that is reached without actuation movement. Then, for each point chosen inside the reachability set, the controller is tested by setting the desired apex state, S_{des}, to be one of those points, with a controller update time step of $\delta t = 0.01$ [s]. As we can see in Fig. 17 and Fig. 18, the effect of the controller is to dramatically reduce the error.

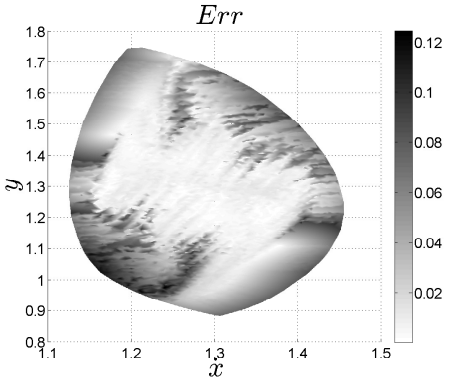


Figure 17: Error from desired point, $Err = \parallel S_r - S_{des} \parallel_2$, computed for $10,000$ points randomly taken inside the reachable set $\mathcal{R}(S_0, \theta_{TD}\}$, with $S_0 = \{1.3, 1.277\}$ and $\theta_{TD} = 58.85$ [deg]. The update time has been chosen to be $\delta\tau = 0.01$ [s].

Fig. 19 shows how many point inside the reachability set have a PC value between [0, 10], (10, 20],... (90, 100] [%], both with and without time delay ($\delta t = 0.01$ [s]), for $10,000$ random desired states. If we consider only the points inside the time-delay reachability set, we can see that $\sim 50\%$ of the points show a percentage change $PC \in (90, 100]$, meaning that the actuator motion has reduced the error by a factor between 90% and 100%. However, if we consider instead the set of all points reachable (i.e., without delay), the percentage of points with $PC \in (90, 100]$ is instead $\sim 42\%$, as expected smaller than its subset with delay. Indeed, due to the delay in the execution of the controller, points outside the subset $\mathcal{R} - \mathcal{R}_{gap}$ are not reachable anymore. In both cases, though, more than two thirds of the points have $PC \in (80, 100]$, highlighting the performance of our proposed controller,

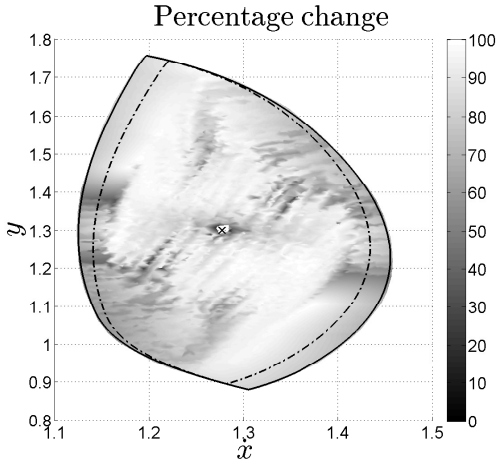


Figure 18: Percentage change graph computed for $10,000$ points randomly taken inside the reachable set $\mathcal{R}(S_0, \theta_{TD}\}$, with $S_0 = \{1.3, 1.277\}$ and $\theta_{TD} = 58.85$ [deg]. The update time is $\delta\tau = 0.01$ [s], and the inner dotted black line marks the border of the reachability space computed with a $\delta\tau$ gap. The black cross represents S_{pss}, the apex state reached without any actuator motion. As expected, the percent change of the subset $\mathcal{R} - \mathcal{R}_{gap}$ is on average smaller than the one on $\mathcal{R}$ alone, since points in $\mathcal{R} - \mathcal{R}_{gap}$ are not directly reachable with our proposed strategy. Note that the dark (lower improvement) and empty areas around S_{pass} are mainly due to the fact that since the desired apex points are so close to the passively reached apex, the margin for improvement is very slim.

even if it requires a delay.

Fig. 20, and Fig. 21(a) and 21(b) show an example of the performance of our controller. An initial apex state has been chosen to be $S_0 = \{1.3, 1.277\}$, and $\theta_{TD} = 58.85$ [deg], $\gamma = 10$. The desired apex state is $S_{des} = \{1.5, 1.213\}$. Maximum velocity and acceleration constant were chosen to be $v_{max} = 0.1596$ and $k_{acc} = 1.0194$. After running the controller with $\delta\tau = 0.01$ [s], the reached state is $S_r = \{1.505, 1.215\}$. Fig. 20 shows the values of S_n computed at each time interval Δt, while Fig. 21(a) and 21(b) show, respectively, the errors $\| S_{des} - S_n \|_2$

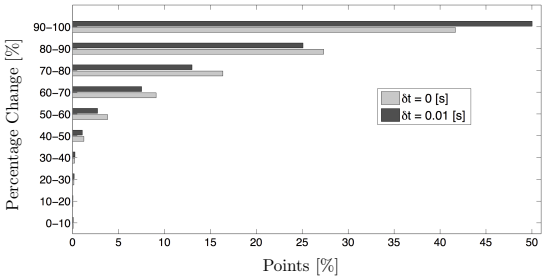


Figure 19: Percentage distribution of PC. The bars represent the percentage of points with a PC between [0, 10], (10, 20],... (90, 100] [%] in the reachability set without delay (light grey bars) and in the reachability set with delay $\delta\tau = 0.01[s]$ (dark bars).

and its $\dot{x}$ and y component at each time interval, and the actuator value $\ell_{act}(t)$ over time.

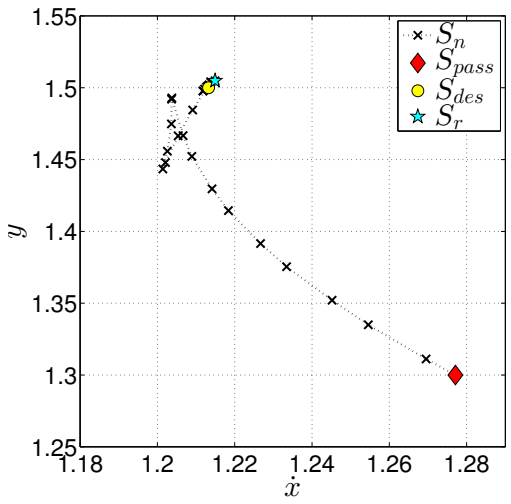
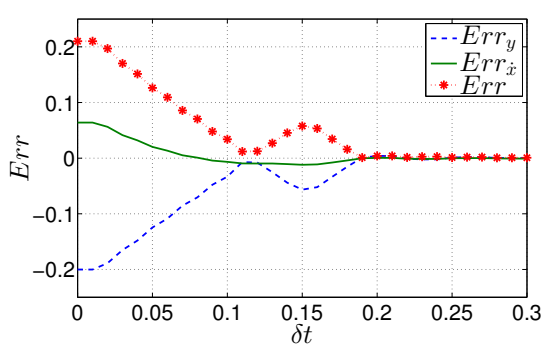
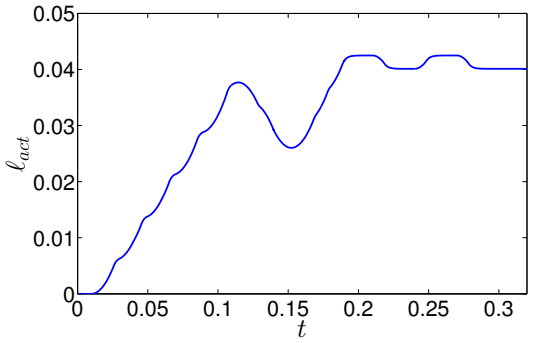


Figure 20: Example of the performance of the proposed control algorithm. The red diamond is the passive state, i.e., the state reached without any actuation. The crossed line represents the apex states S_n, $n = 1, 2, \ldots$ computed by the algorithm, while the yellow circle and the cyan star represent respectively the desired state, S_{des}, and the reached state, S_r.



(a)



(b)

Figure 21: Subfig. 21(a) and 21(b) show respectively the evolution of the error (7) and the evolution of its component in $\dot{x}$ and y, and the computed actuation for the example in Fig. 20.

6.3 Percentage change over a set of consecutive steps

Section 6.2 presents the performance of our control strategy for one specific set of initial conditions: starting from one apex state and the touch-down angle necessary to perform a symmetric hop, we showed the performance of the algorithm when trying to reach $10,000$ states inside its reachability set. To show that the presented results still hold for arbitrary initial conditions and touch-down angles, we consider a path of $1,000$ subsequent hops. At each hop, the touch-down angle is chosen randomly from the set of angles that allow a successful jump, not necessarily symmetric, where a successful jump is defined as any apex-to-apex hop characterized by a positive velocity and an apex height exceeding the leg length to avoid collision between the leg and the ground. Then, the desired apex state is set by adding to each dimension of the passively reached state an offset picked from a uniform distribution. Fig. 22 shows all the apex states reached by the system during the $1,000$ subsequent hops. As each one of these states is the initial condition of the following hop, our control algorithm has been tested for a wide variety of initial conditions which encompass the typical operating range for the SLIP model. Furthermore, generating desired apex states by adding uniformly distributed noise to the passive apex state ensures that our controller has been tested for a wide range of initial error. Fig. 23 shows the resulting percentage change for $1,000$ hops. As we can see, over two thirds of the states have a percentage change $PC \in [90, 100]$, meaning that the controller is able to reduce the distance to the desired state of a factor between 90% and 100%. Overall, over 80% of the states have a $PC \geq 70$. More in detail, Fig. 24 shows the output error $\| S_r - S_{des} \|_2$ as a function of the magnitude of the initial error $\| S_{pass} - S_{des} \|_2$. For each initial error interval of size 0.025, the mean value (grey circle) and standard deviation (black line) of the output error has been computed: as expected, while the output error is higher for higher initial error, there is on average an improvement of over 70%.

7 Conclusions

In this paper, we propose a control algorithm for an actuated SLIP model to reach a desired apex state. Its main application is to correct errors happening at touch-down and/or during the stance phase, independent of any foot placement strategy employed. Most of the control strategies developed in the past for the SLIP model (whether passive or active) are aimed at controlling the touch-down position of the leg, θ_{TD}, and/or the leg length at touch-down, in order to reach a stable gait even on unknown or partially known terrain, or in presence of system noise. For the actuated case, this choice was paired with

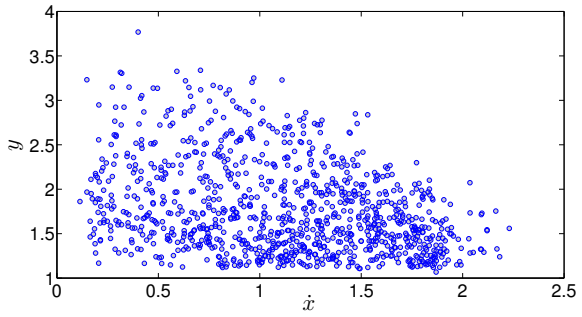


Figure 22: This plot shows the $1,000$ hops (initial conditions) used to test the controller. The initial apex state has been chosen as $S_0 = \{1.3, 1.277\}$. At each hop, the desired apex state has been chosen by adding to the passively reached state an offset drawn from a uniform distribution between $[-.3, , .3]$ for y and $[-0.15, 0.15]$ for $\dot{x}$. These bounds have been chosen based on the reachability set previously computed for the initial condition S_0. However, note that each hop drives the system to a new apex state, hence a new reachability set, whose computation is expensive and not feasible in real time. It is therefore possible at any given hop that the desired apex is not in the corresponding reachability set.

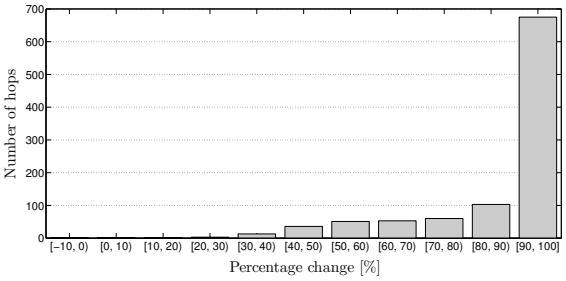


Figure 23: In this figure the resulting percentage change values are shown: as we can see, 675 points have a $PC \geq 90$, i.e., the initial distance to the desired apex is reduced of more than 90%. Overall, 832 states over $1,000$ have an improvement above 70%. Note that two states have negative percentage change, meaning that in these two cases our algorithm failed at improving the initial error, due to numerical approximation introduced by the algorithm.

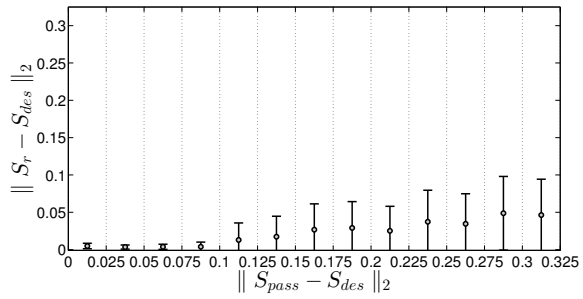


Figure 24: Plot of the initial error versus the final error. Each grey circle represents the mean value of the final error for all the initial errors in the corresponding interval, i.e., between $[0, 0.025)$, $[0.025, 0.05)$, etc. The black vertical lines are the corresponding standard deviations.

actuation displacement during the stance phase, and the desired actuation was precomputed during flight based on the knowledge on the terrain. Conversely, our strategy focuses on reaching a desired apex state without any insight on the choice of the touch-down angle: the actuation is computed online during the stance phase, based on the particular state of the system at each time interval. The main advantage of this procedure is not only the ability to counteract sensing errors at landing (e.g., ground sensing noise), which is a goal taken into consideration by other works as well, but especially the ability to reduce errors and disturbances that happen during stance. For example, a slipping of the foot, or an external force, such as a strong wind. Furthermore, we want to point out that our strategy does not aim to replace any leg-placement strategy: on the contrary, it is reasonable to think that our controller can easily be paired with any other leg-placement or path planning method.

Another contribution of this work is a qualitative characterization of the effects of the actuator motion on the reachable apex state. Throughout the stance phase, computing the spatial relationship between the current reachable state without further actuation and the desired state gives a strong indication of how

to provide additional actuation to the system.

Finally, by delineating the reachability space for a given apex state and touch-down angle, this work reveals the advantage of utilizing the actuation at its maximum capabilities. In fact, we show that updating the actuation values throughout the stance phase allows the system to reach a wider range of evenly distributed apex states with respect to the ones reached by pre-computing some fixed actuator motions.

Acknowledgments

This work is supported by DARPA (Grant No. W911NF-11-1-0077).

References

Ahmadi, M. and Buehler, M. (1997), 'Stable control of a suimulated one-legged running robots with a hip and leg compliance', *IEEE Transactions on Robotics* **13**(1).

Altendorfer, R., Koditschek, D. E. and Holmes, P. (2004), 'Stability analysis of legged locomotion models by symmetry-factored return maps', *International Journal of Robotics Research* **23**(10–11), 979–999.

Andrews, B., Miller, B., Schmitt, J. and Clark, J. E. (2011), 'Running over unknown rough terrain with a one-legged planar robot', *Bioinspiration & Biomimetics* **6**(2), 026009.

Arslan, Ö. and Saranlı, U. (2012), 'Reactive planning and control of planar spring-mass running on rough terrain', *IEEE Transactions on Robotics* **28**(3), 567–579.

Arslan, Ö., Saranlı, U. and Morgül, Ö. (2009), Approximate stance map of the spring mass hopper with gravity Correction for Nonsymmetric Locomotions, *in* 'IEEE Int. Conf. on Robotics and Automation', Kobe, Japan, pp. 2388–2393.

Blickhan, R. (1989), 'The spring-mass model for running and hopping', *Journal of Biomechanics* **22**(11/12), 1217–1227.

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

Byl, K., Byl, M., Rutschmann, M., Satzinger, B., van Blarigan, L., Piovan, G. and Cortell, J. (2012), Series-elastic actuation prototype for rough terrain hopping, *in* 'IEEE International Conference on Technologies for Practical Robot Applications', pp. 103–110.

Collins, S., Ruina, A., Tedrake, R. and Wisse, M. (2005), 'Efficient bipedal robots based on passive-dynamic walkers', *Science* **307**(5712).

Daley, M. A., Felix, G. and Biewener, A. A. (2007), 'Running stability is enhanced by a proximo-distal gradient in jointy neuromechanical control', *Journal of Experimental Biology* **210**, 383–394.

Dudek, D. and Full, R. J. (2006), 'Passive mechanical properties of legs from running insects', *Journal of Experimental Biology* **209**, 1502–1515.

Farley, C. T. and Ferris, D. P. (1998), 'Biomechanics of walking and running: Center of mass movements to muscle action', *Exercise and Sport Science Reviews* **26**, 253–283.

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

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

Ghigliazza, R. M., Altendorfer, R., Holmes, P. and Koditschek, D. (2005), 'A simply stabilized running model', *SIAM Rev.* **47**(3).

Hodgins, J. K. and Raibert, M. H. (1991), 'Adjusting step length for rough terrain locomotion', *IEEE Transactions on Robotics and Automation* **7**(3).

Hurst, J. W., Chestnutt, J. and Rizzi, A. (2007), Design and philosophy of the BiMASC, a highly dynamic biped, *in* 'IEEE Int. Conf. on Robotics and Automation', Roma, Italy.

Karssen, J. G. D., Haberland, M., Wisse, M. and Kim, S. (2011), The optimal swing-leg retraction rate for running, *in* 'IEEE Int. Conf. on Robotics and Automation', Shangai, China, pp. 4000–4006.

Piovan, G. and Byl, K. (2012), Enforced symmetry of the stance phase for the spring-loaded inverted pendulum, *in* 'IEEE Int. Conf. on Robotics and Automation', Saint Paul, MN, USA, pp. 1908–1914.

Piovan, G. and Byl, K. (2013), Two-element control for the active SLIP model, *in* 'IEEE Int. Conf. on Robotics and Automation', Karlsruhe, Germany, pp. 5636–5642.

Raibert, M. H. (1986), *Legged Robots that Balance*, MIT Press.

Riese, S. and Seyfarth, A. (2012), 'Stance leg control: variation of leg parameters supports stable hopping', *Bioinspiration & Biomimetics* **7**(1), 016006.

Rutschmann, M., Satzinger, B., Byl, M. and Byl, K. (2012), Nonlinear model predictive control for rough terrain hopping, *in* 'IEEE/RSJ Int. Conf. on Intelligent Robots & Systems', Villamoura, Algarve, Portugal, pp. 1859–1864.

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

Schmitt, J. and Clark, J. (2009), 'Modeling posture-dependent leg actuation in sagittal plane locomotion', *Bioinspiration and Biomimetics* **4**, 1–17.

Schwind, W. J. and Koditschek, D. E. (2000), 'Approximating the stance map of a 2 DOF monoped runner', *Journal of Nonlinear Science* **10**(5), 533–588.

Seipel, J. and Holmes, P. (2007), 'A simple model for clock-actuated legged locomotion', *Regular and Chaotic Dynamics* **12**(5), 502–520.

Seyfarth, A., Geyer, H. and Herr, H. (2003), 'Swing-leg retraction: a simple control model for stable running', *Journal of Experimental Biology* **206**, 2547–2555.

Sponberg, S. and Full, R. J. (2008), 'Neuromechanical response of musculo-skeletal structures in cockroaches during rapid running on rough terrain', *Journal of Experimental Biology* **211**, 433–446.

Vejdani, H. R. and Hurst, J. W. (2012), Swing leg control for actuated spring-mass robots, *in* 'Int. Conf. on Climbing and Walking Robots and the Support Technologies for Mobile Machines', Baltimore, MD, USA, pp. 536–542.

Yu, H., Li, M. and Cai, H. (2012), Approximating the stance map of the SLIP runner based on perturbation approach, *in* 'IEEE Int. Conf. on Robotics and Automation', Saint Paul, MN, USA, pp. 4197–4203.

Zeglin, G. (1999), The Bow Leg Hopping Robot, PhD thesis, Carnegie Mellon University, Pittburgh, PA, USA.