Tools for the Design of Stable yet Nonsteady Bounding Control

Virgile Paris[1], Tom Strizic[2], Jason Pusey[3], and Katie Byl[2]

***Abstract*— Much work in developing control to achieve dynamic gaits for legged systems focuses on limit cycles and their stability. However, there are many practical situations where step-to-step variability is highly desirable, for example to achieve variable footholds, to recover and replan after perturbations, or to control forward speed. In this paper we present an effective, high-level switching control framework for overcoming terrain obstacles using the familiar A* algorithm to search a mesh over the reachable space for a given set of controllers. In support of this, we present new low-level control strategies for generating stable bounding with planar models of a spring-legged quadruped robot, and demonstrate their use crossing gaps in the terrain.**

I. INTRODUCTION

Animals that bound demonstrate distinct advantages in speed and agility, especially when the torso is long in relation to the legs [1], [2]. Additionally, mammals such as the cheetah and greyhound display pronounced articulation of the spine while executing high speed gaits [3]. Various models have been proposed to describe the dynamics of quadrupedal locomotion with passive compliant legs [4], [5], but the interaction between the legs and spine in propulsion is still not completely understood. Some initial insights have been reported by [6], and others have suggested that a quadruped with an actuated torso can leap higher and farther compared to a rigid-spined quadruped for about the same specific resistance [5], [7], [8]. In pursuit of these power gains, several groups have proposed control methods for limit-cycle bounding [7], [9], [10], but little progress has yet been made toward optimizing step-to-step agility.

Some quadrupedal robotic platforms which incorporate an articulated torso include Leeser's Planar Quadruped [11], and more recently Boston Dynamics' Cheetah and Wildcat platforms [12] and the MIT Cheetah [13], [14]. These robots actuate the torso to gain additional force and range of motion within the gait cycle, but none of these robots report intentionally designing compliance into their spine mechanisms. In order to explore the role of passive spine compliance in bounding a quadrupedal robot with a cable driven elastic spine named Canid (Fig.1) has been created through the Robotics Collaborative Technology Alliance (RCTA) between University of Pennsylvania and U.S. Army Research Laboratory (ARL) [15], [16].

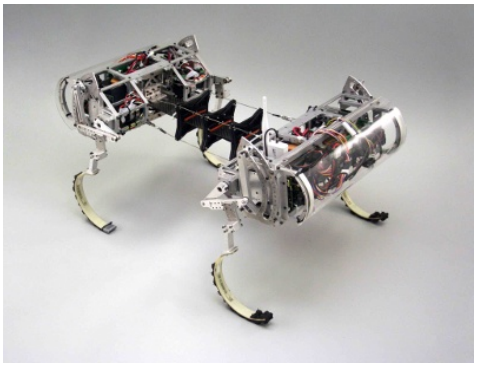

Fig. 1: Front view of the Canid robot.

In this paper, we describe a method for avoiding gap obstacles on terrain, applied to two planar simulations of Canid with varying complexity and different low-level control options. We build on past work meshing reachable robot states given some set of control actions [17], adding an A^* search to find control sequences that will avoid terrain gaps while maintaining stability and maximizing the distance between footholds and the gaps. Our first example employs step length control to robustly cross gaps with varying lookahead distances, while the second approach uses a controller designed for steady-state running along with several short-term maneuvers, for similar effect.

The rest of the paper is organized as follows. Section II presents our high-level framework, in which we map out the reachable state space, given arbitrary switching between a finite number of low-level controllers. We demonstrate this framework on both example systems. Section III presents a 4 degree-of-freedom (DOF) planar model, with switching among nine controllers, each of which has a stable limit cycle. Section IV presents an 8-DOF model and employs one limit cycle controller and five additional "maneuver" controllers, which do not demonstrate stable limit cycles. Finally, Section V presents conclusions and future work.

II. MESH-BASED OBSTACLE AVOIDANCE

Agility is an intuitive concept, but it is not yet clearly defined in legged robotics. Intuitively, it means being able to replan rapidly, to cope with known variability ahead. Improving agility requires first defining some performance

This work was supported in part by the U.S. Army's Robotics CTA through grant No. W911NF-08-R-0012, by the Institute for Collaborative Biotechnologies through grant No. W911NF-09-0001 from the U.S. Army Research Office, and by the Summer Student Research Participation Program at the U.S. Army Research Laboratory administered by the Oak Ridge Institute for Science and Education.

[1]Virgile Paris is with the Bordeaux Institute of Technology, ENSEIRB-MATMECA, 1 avenue du Dr Albert Schweitzer B.P. 99 33402 Talence, France paris.vgle@gmail.com

[2]Katie Byl and Tom Strizic are with the Department of Electrical and Computer Engineering, University of California Santa Barbara, Santa Barbara, CA 93106-9560, USA katiebyl@ece.ucsb.edu, tstrizic@ece.ucsb.edu

[3]Jason Pusey is with the U.S. Army Research Laboratory, Aberdeen Proving Ground, MD, USA jason.l.pusey.civ@mail.mil

metric(s). For our task of bounding while avoiding terrain gaps, we measure agility via the goals of minimizing required lookahead and maximizing achievable gap width.

In this section, we present our framework for high-level control. In our approach, we have a finite set of low-level controllers. Based on previous work [18]–[20], we hypothesize that switching among a finite set of low-level controllers constrains the dimensionality of step-to-step (Poincaré) snapshots of the system to a much lower dimensional manifold (e.g., roughly 2D) within its full state space, so that it is tractable to map out a relatively small (<10,000 elements), non-uniform mesh of the reachable state space, using a deterministic algorithm. Below, we present this meshing algorithm, which is used in Sections III and IV.

A. Meshing Algorithm

Our meshing algorithm maps out a discrete approximation of all states the system can visit, along with all the possible transitions among states. The mesh states are Poincarè snapshots of the robot, taken at a particular event in the gait cycle. We begin the algorithms with at least two seed states: any fixed point(s) for limit cycle(s) of the controllers, and an absorbing failure state. Any unexplored mesh points enter a queue, and the algorithm keeps going until this queue is empty – indicating we have mapped out the reachable state space with some given resolution.

Algorithm 1 Meshing Algorithm

Input : Initial set of states P, set of controllers U, threshold distance d_{thr}
Output : State transition map M, Final set of states P
$Q_{next} \leftarrow P$
while Q_{next} not empty **do**
  $Q_{current} \leftarrow Q_{next}$
  empty Q_{next}
  for each $\boldsymbol{p}_j \in Q_{current}$ **do**
    for each $\boldsymbol{u}_k \in U$ **do**
      Simulate one step and save the generated state vector $\boldsymbol{p}_j$ in P and Q_{next} if it is far enough (d_{thr}) from all $\boldsymbol{p} \in P$. The corresponding information about the step length are then stored in M.
    end for
  end for
end while
return M, P

The resulting mesh constitutes a directed graph, encompassing the reachable state space for a set of controllers or control actions selected once each step. The algorithm used to generate the mesh (see Alg. 1) originates from [18]–[20]. The mesh is grown (deterministically) from the seed points by simulating application of every available controller, and recording each end state as well as other important parameters of the step. This process is repeated from all of the resulting non-failure states that are farther than a threshold distance d_{thr} from every other point $\boldsymbol{p} \in \mathbb{R}^n$ in the mesh P as measured by:

$$d(\boldsymbol{p}, P) = \min_{\boldsymbol{p} \in P} \sqrt{\sum_{i=1}^{n} \left(\frac{x_i - p_i}{\sigma_i} \right)^2} \tag{1}$$

where $\boldsymbol{x} \in \mathbb{R}^n$ is the system state of interest, and σ_i is the standard deviation for state component i over all points in the mesh.

B. A^* Search

We explore agile behaviors by using the mesh as an approximate road-map for all potential actions within the normal operating range of the robot. By conducting a search over the mesh from the nearest neighbor to the robot's starting state, we can identify a stable sequence of control actions for a given task provided the mesh is stored along with relevant information such as the step-length or foothold locations. The A^* algorithm [21] (Alg. 2) is an ideal choice for this task as it can enable fast identification of optimal sequences. Here we expand a search tree from the given starting node in the mesh by exploring branches with the lowest estimated cost, f, to the goal, calculated as the actual cost of the path traversed so far, g, and a conservative heuristic of the remaining cost-to-go to the goal, h.

$$f = g + h \tag{2}$$

Algorithm 2 A^* Algorithm Pseudo Code

Input : Set of states X, state transition map M, starting state s, distance to gap x_{near}, gap width w_{gap}
Output : Control sequence $path$
$OPEN \leftarrow s$
while $OPEN$ is not empty **do**
  Remove node $current$ with lowest cost f from $OPEN$ and add to $CLOSED$
  if isgoal($current$) **then**
    return $path$ from s to $current$
  end if
  for all non-fail children of $current$ **do**
    Store total distance traveled and foothold locations
    $g(child) \leftarrow g(current) + step\ cost$
    $f(child) \leftarrow g(child) + h(child)$
    if $child$ is already in $OPEN$ or $CLOSED$ **then**
      If $f(child)$ is lower than existing node, replace and move to $OPEN$
    else
      add $child$ to $OPEN$ with pointer back to $current$
    end if
  end for
end while

Searches in the next two sections focus on finding step sequences to cross gaps of known width and distance ahead, while rewarding taking fewer steps and providing greater safety margins for footholds near gaps.

III. Step Length Control of a 4 Degree of Freedom Model

A. Model and Control Strategy

For our first application of the meshing and search algorithms, we model the robot as a planar three-link system

with massless legs and body masses coincident with the two hips (Fig. 2). We formulate the equations of motion using the Lagrangian approach with generalized coordinates:

$$\boldsymbol{q} := \begin{bmatrix} \theta_1 & \theta_2 & \theta_3 & l \end{bmatrix}^\top \tag{3}$$

These are defined graphically in Fig. 2 for the case when the robot is in rear stance. The dynamic state of the system is given by the vector $\boldsymbol{x} := \left[\boldsymbol{q}^\top, \dot{\boldsymbol{q}}^\top\right]^\top$.

The model has 4 DOFs (3 angles and 1 length) and is underactuated by 2 DOFs (as each hip has one actuator). For more information about this model see [22].

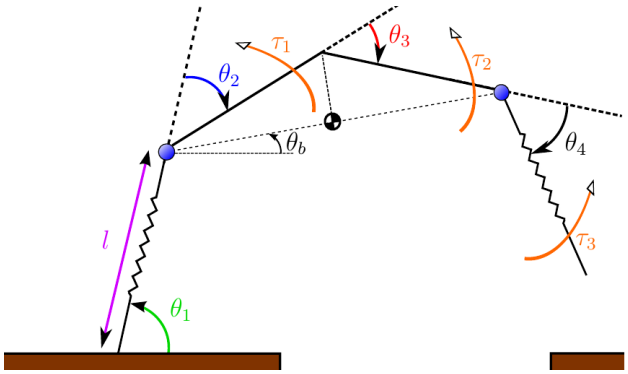

Fig. 2: 4-DOF Canid robot. Model parameters : hip masses $m_1, m_2 = 2.0kg$, joint inertia $J_1, J_2 = 0.0052kg.m^2$, neutral length of spring leg $l_0, l_0' = 0.254m$, section leg segment $L_2, L_3 = 0.254m$, leg stiffness $k_{leg} = 1550N/m$, leg damping $b_{leg} = 79N/(m/s)$. tau_2 spine stiffness (torsional) $K_{spine} = 60Nm/rad$, spine damping (torsional) $B_{spine} = 2Nm/(rad/s)$.

Using the Lagrangian method, the equations of motion can then be derived in the canonical form of

$$\ddot{\boldsymbol{q}} = M^{-1}(\boldsymbol{q}, \dot{\boldsymbol{q}})(C(\boldsymbol{q}, \dot{\boldsymbol{q}}) + \boldsymbol{\xi}) \tag{4}$$

$$\dot{\boldsymbol{x}} = \left[\dot{\boldsymbol{q}}^\top, \ddot{\boldsymbol{q}}^\top\right]^\top \tag{5}$$

Integrating the dynamics forward using equation (4), the entire dynamics of the system can be described as a function of an input vector $\boldsymbol{\xi}$. The torque τ_1 between the rear portion of the spine and the rear leg is applied to control the body angle θ_b using the simple PD control law $\tau_1 = -K_p(\theta_{ref} - \theta_b) - K_d\dot{\theta}_b$ ($K_p = 4000$ and $K_d = 120$), with one reference trajectory shown in Fig. 3.

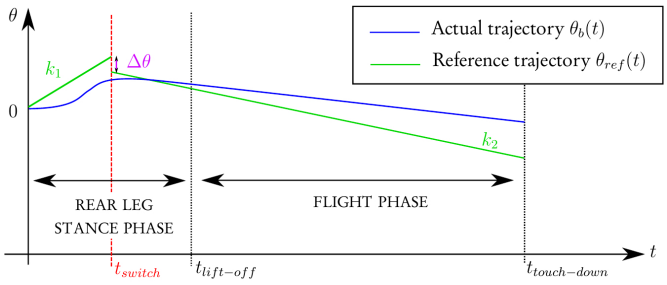

Fig. 3: General shape of the body angle trajectory and reference

The idea is to tune the parameters specific to the desired reference trajectory to find resulting actual trajectories with specific features (step length, velocity, stability, maximum apex height, touch-down body angle, ...). Our goal is to have a set of controllers associated with different bounding motions and to be able to switch between these to avoid obstacles while maintaining dynamic stability. The controllers are defined as vectors $\boldsymbol{u} := \begin{bmatrix} u_1 & \cdots & u_6 \end{bmatrix}^\top$, where $u_1 = -\theta_2$ and $u_2 = -\theta_4$ set the desired leg touchdown angles set during the flight phase,[1] and $u_3 = k_1$, $u_4 = k_2$, $u_5 = t_{switch}$, and $u_6 = \Delta\theta$ are parameters of the body angle trajectory shown in Fig. 3.

B. Search for Controllers

In order to generate a mesh of possible multi-step behaviors, we first designed a set of controllers with different step-lengths. We used the Newton-Raphson method to find parameters for body angle reference trajectories that minimize a cost function C, defined as:

$$C(\boldsymbol{u}, \boldsymbol{x}^{(k)}) := \begin{cases} C_2, & \text{if the step fails} \\ C_1(\boldsymbol{u}, \boldsymbol{x}^{(k)}), & \text{otherwise} \end{cases} \tag{6}$$

where $C_2 = 10^5$ ensures that the algorithm not output controllers which cause the robot to fall in one step, and

$$C_1(\boldsymbol{u}, \boldsymbol{x}^{(k)}) = (\boldsymbol{l}(\boldsymbol{u}, \boldsymbol{x}^{(k)}) - \boldsymbol{l}_d^{(k)})^T W (\boldsymbol{l}(\boldsymbol{u}, \boldsymbol{x}^{(k)}) - \boldsymbol{l}_d^{(k)}). \tag{7}$$

Here $\boldsymbol{l}(\boldsymbol{u}, \boldsymbol{x}^{(k)}) = \begin{bmatrix} \boldsymbol{\rho} & (\boldsymbol{x}^{(k+1)})^\top \end{bmatrix}^\top$ is a vector composed of n gait parameters $\boldsymbol{\rho}$ for the current step and the robot state at the end of the step $\boldsymbol{x}^{(k+1)}$. Similarly, $\boldsymbol{l}_d^{(k)} = \begin{bmatrix} \boldsymbol{\rho}_d & (\boldsymbol{x}^{(k)})^\top \end{bmatrix}^\top$ is a vector of the desired gait parameters $\boldsymbol{\rho}_d$ and end state $\boldsymbol{x}^{(k)}$. These parameters could be, for example the step length, flight apex height, body angle at lift-off, etc... In this case $\boldsymbol{\rho} = \rho = L$ and $\boldsymbol{\rho}_d = \rho_d = L_d$, where L and L_d are the step length and desired step length respectively, defined as the location of the rear leg foothold of the current step relative to the front leg foothold of the previous step.

The weighting matrix $W = diag(w_i) \in \mathbb{R}^{(8+n)\times(8+n)}$ is defined as:

$$W = \begin{pmatrix} W_p & 0_{n\times 8} \\ 0_{8\times n} & W_s \end{pmatrix} \qquad \begin{cases} W_s = w_s * \frac{1}{|L - L_d|} I_8 \\ W_p = w_p * |L - L_d| \end{cases} \tag{8}$$

where L and L_d are in centimeters and $w_s = 1$, $w_p = 5$ were chosen empirically. Intuitively, the weights were chosen so that the focus is on reducing the error $|L - L_d|$ first, and on stabilizing the system when the step length error is small.

The optimization was then computed using Alg. 3, where we chose the maximum number of allowed iterations $N_{max} = 50$, and convergence threshold $\varepsilon = 10^{-13}$. Recall that $H_C^k(\boldsymbol{u})$ and $\boldsymbol{\nabla} C$ are respectively the Hessian matrix and the nabla of C with respect to $\boldsymbol{u}$ around the estimate $\boldsymbol{u}^{(k)}$, where k is the number of iterations in Alg. 3.

The use of a line search method such as the Armijo rule was crucial for the algorithm to converge. The Armijo rule consists in ensuring that the cost function decreases from one

[1]The exact time when those angle are set during the flight phase is not of concern since the inertia of the legs is not taken into account in this model.

Algorithm 3 Newton-Raphson Algorithm

Input : $\boldsymbol{u}^{(1)}$, $\boldsymbol{x}^*$
Output : $\boldsymbol{u}^{(k)}$, $\boldsymbol{x}$
for $k = 2 \rightarrow N_{max}$ **do**
    Determine α using the Armijo rule.
    Compute $\boldsymbol{u}^{(k)} = \boldsymbol{u}^{(k-1)} - \alpha H_C^{k-1}(\boldsymbol{u})^{-1}(\boldsymbol{\nabla} C)^T$ and store the new state vector $\boldsymbol{x}$ taken from the fixed point $\boldsymbol{x}^*$ with the controller $\boldsymbol{u}^{(k)}$.
    if $||\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}|| < \varepsilon$ **then**
        return $(\boldsymbol{u}^{(k)}, \boldsymbol{x})$
    end if
end for

step to another and that the decrease is significant enough. Initializing α to 1 gives us a first $C(\boldsymbol{u}^{(k)}, \boldsymbol{x}^*)$ that will be compared to the cost function of the previous iteration using the following test:

$$C(\boldsymbol{u}^{(k-1)}, \boldsymbol{x}^*) - C(\boldsymbol{u}^{(k)}, \boldsymbol{x}^*) < \sigma \beta^m s \nabla C d_{k-1} \quad (9)$$

where $d_k = H_C^k(\boldsymbol{u})^{-1}(\nabla C)^T$, $\beta = 1/3$, $\sigma = 10^{-3}$, $m = 0$, $s = 1$ and $\alpha = \beta^m s$ (see [23]). As long as (9) is not satisfied, m is incremented and $\boldsymbol{u}^{(k)}$ is computed again.

Recall that the objective is to find controllers associated with fixed points. It is therefore crucial to update the state from which each step is taken. In that prospect, we used the iterative process depicted in Figure 4.

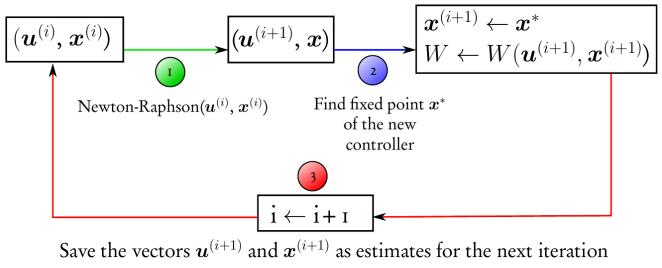

Fig. 4: Newton-Raphson iterative process

The counter i denotes the number of time Alg. 3 is used. The process is then reiterated with $\boldsymbol{x}^{(i)}$ and $\boldsymbol{u}^{(i)}$ until the desired precision is attained *i.e.* until $C(\boldsymbol{u}^{(i+1)}, \boldsymbol{x}^{(i+1)}) < \varepsilon$, where $\varepsilon = 10^{-2}$ was chosen empirically.

We were able to find controllers within 2 cm of the desired step length using this method. The range of the step lengths associated with the controllers found varies from -20 cm to $+20$ cm, with each controller having a step length 5 cm apart from the next. The 9 controllers found this way were each associated with a stable fixed point, which was unexpected as nothing in the algorithm guarantees multi-step stability.

C. Meshing and Path Finding Algorithm

The initial set of states X consists of the nine fixed points associated with the controllers found in addition to a failure state. The set of controllers U consists of the nine controllers found by the methods described above, and the distance threshold is chosen to be $d_{thr} = 5 \times 10^{-3}$. The mesh generated had 1720 nodes.

The obstacles used in this particular study were gaps of known width, separated by specific distances. The A^* cost functions were chosen to minimize the number of steps before leaping over a gap, while maximizing the distance between the footholds and the gap. Thus, let us define d_r as the distance of the rear foot from the start, d_f as the distance of the front foot from the start, and d_g as the distance of the gap. The distance from the rear foot to the gap d_{rg} and from the front foothold to the gap d_{fg} are:

$$d_{rg} := |d_r - d_g| \quad (10)$$
$$d_{fg} := |d_f - d_g| \quad (11)$$

The A* algorithm should maximize $min(d_{rg}, d_{fg})$ when the robot is close to a gap. The heuristic and movement cost functions of the A^* algorithm were chosen as:

$$h(node) := 1/d_f \quad (12)$$
$$g(node) := 1/min(d_{rg}, d_{fg}) \quad (13)$$

Mapping the terrain using the state transition map M in terms of nodes was also a critical part of the A^* algorithm, each node being associated to a distance of the rear and front leg from the start during the mapping. Mapping this way had the advantage of offering the choice not to build irrelevant nodes. In this study, only the reachable nodes were built, which were the nodes where no failure of the system were observed. If any switching from one controller to another led to the fall of the robot, then the corresponding node would be discarded. Likewise, the mapping discards the nodes in which footholds fall in the gap, i.e. $d_r \in$ [gap location, gap location+gap width] or $d_f \in$ [gap location, gap location+gap width]. In our case, the heuristic cost associated to each node could be precomputed, while the movement cost function was computed in the A^* algorithm (see Alg. 2).

D. Control Strategy Performance

Let us assume that the system becomes aware of the gap while in its default bouding gait. The performance of this control strategy can be observed in Fig. 5. This plot shows the ability of the system to leap over a gap when the gap is at a specific distance to the system and has a specific size.

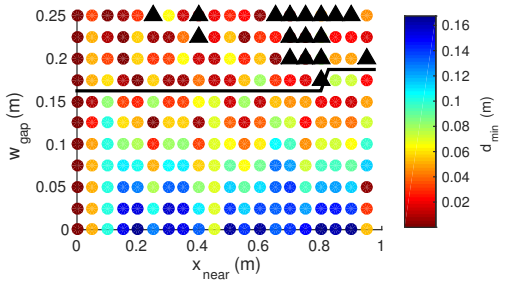

Fig. 5: Illustration of the control strategy performance

To create Fig. 5, the optimal control strategy was simulated, to verify accuracy of the A* results from meshing. The dots indicate that the system successfully leaped over the gap located at the distance x_{near} from its front foot and that

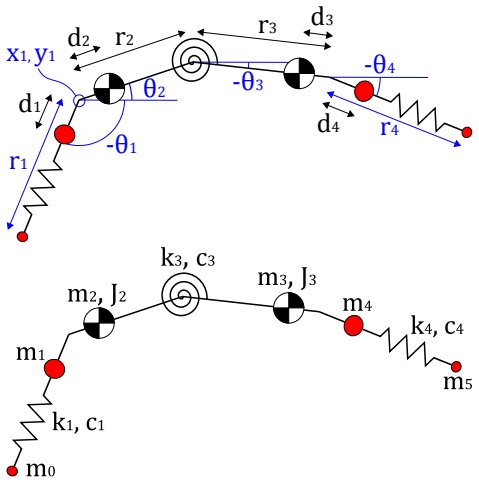

Fig. 6: (top) Geometry of the 8 DOF 2D model, with $d_1 = 0.1$ m, $d_2 = -0.048$ m, $d_3 = -0.049$ m, $d_4 = 0.09$ m, $r_2 = r_3 = 0.24$ m. (bottom) Inertial and passive elements with $m_0 = m_5 = 0.043$ kg, $m_1 = m_4 = 0.57$ kg, $m_2 = 4.4$ kg, $m_3 = 4.5$ kg, $J_2 = 0.0145$ kg-m^2, $J_3 = 0.015$ kg-m^2, $k_1 = 3300$ N/m, $k_3 = 11$ N-m/rad, $k_4 = 3460$ N/m, $c_1 = c_4 = 50$ N-s/m, $c_3 = 0.6$ N-m-s/rad

had a width of w_{gap} whereas the black triangles mean that the system failed to do so, even though an A* solution had been predicted to succeed. We expected occasional failures, and addressing meshing approximation errors is a key focus for future work. The colors represent the minimum distance the rear leg or front leg gets to either sides of the gap. The black line represents the planning horizon necessary to guarantee crossing gap with a width w_{gap}, given the gaps starts anywhere at or beyond the indicated x_{near}.

IV. Force Direction Control of an 8 Degree of Freedom Model

A. The Model

The geometry and parameter definitions for the 8 DOF model are shown in Fig. 6. This variation of the 2D Canid model is motivated by a need to simulate the effect of leg masses, dissipative elements in the robot, and energy losses due to ground contact. Increasing the dimensionality of the system might limit the tractability (or at least accuracy) of our meshing, which is a topic of on-going interest within our research group. Composite plate spine and body assemblies are represented as a two-link chain with centers of mass displaced from the hips. A linear torsion spring and damper at the central joint acts in parallel with the spine control torque τ_{sp} so that the total torque of the rear body segment on the front is:

$$\tau_{2,3} = -k_3(\theta_3 - \theta_2) - c_3(\dot{\theta}_3 - \dot{\theta}_2) + \tau_{sp} \tag{14}$$

where k_3 and c_3 are the spine spring rate and damping coefficient.

The upper half of Canid's legs feature 4-bar linkages that help to reduce toe stubbing by raising the legs during the swing phase of each step. This effect is mimicked in the 2D model by reducing the leg length used by the contact event handler until the leg is within 2° of the desired angle. The composite spring portion of the legs is represented as a 1 DOF linear spring-damper, with the radial force f_i between upper and lower portions computed as:

$$f_i = -k_i(r_i - r_{L,0}) - c_i\dot{r}_i \tag{15}$$

where i is 1 for the rear leg and 4 for the front, k_i and c_i are the spring rate and damping coefficient for each leg, and $r_{L,0} = 0.3$ m is the nominal leg length.

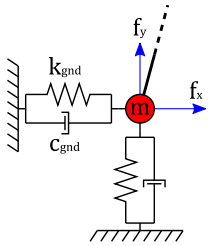

Fig. 7: Viscoelastic ground model.

We determine the ground reaction forces using a viscoelastic (Kelvin-Voigt) model [24], with decoupled vertical and horizontal components computed from the location and velocity of the toe mass relative to the touchdown location. The reaction forces are:

$$f_{i,x} = -k_{gnd} * (x_i - x_{i,g}) - c_{gnd} * \dot{x}_i \tag{16}$$

$$f_{i,y} = -k_{gnd} * (y_i - y_{i,g}) - c_{gnd} * \dot{y}_i \tag{17}$$

where i is the index of the mass in contact, $k_{gnd} = 7.5 * 10^4$ N/m is the spring rate, $c_{gnd} = 110$ N-s/m is the damping coefficient, (x_i, y_i) is the location of the toe, and $(x_{i,g}, y_{i,g})$ is the location of first contact.

The equations of motion were formulated using the Lagrangian approach for the robot in flight and subject to gravity [25]. The generalized coordinates are:

$$\boldsymbol{q} = [\theta_1, \theta_2, \theta_3, \theta_4, r_1, r_4, x_1, y_1]^\top \tag{18}$$

as defined in Fig. 6. The rear hip provides the reference point to the global Cartesian coordinate system while all angles are referenced to the global horizontal axis.

After finding the equations of motion, we rearrange them to find the accelerations $\ddot{\boldsymbol{q}}$, then form an augmented state vector $\boldsymbol{x}$ so that the computation can be performed in MATLAB® with ode45():

$$\boldsymbol{x} := \left[\boldsymbol{q}^\top, \dot{\boldsymbol{q}}^\top\right]^\top \tag{19}$$

B. Low-Level Controls

Independent leg and spine angle controllers comprise the base level of the control hierarchy. The spine angle controller adds stiffness to the spine spring throughout all phases of motion to help couple leg stance torques to the body and damp out oscillations during flight. The control torque τ_{sp} includes a feedback linearization component based on a two-link model of the body in free space Fig. 8, and a

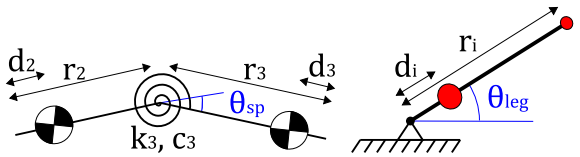

Fig. 8: Simplified models used for spine control (left), and leg control (right).

Proportional-Derivative (PD) portion which aims to attain the dynamics of a 2nd order reference model [26] [27]. Defining the spine angle as $\theta_{sp} = \theta_3 - \theta_2$, the equation of motion is:

$$J_{eff}\theta_{sp} = -k_3\theta_{sp} - c_3\dot{\theta}_{sp} + \tau_{sp} \tag{20}$$

where the combined effective moment of inertia J_{eff} for the front and rear assemblies is:

$$J_{eff} = J_r J_f/(J_r + J_f) \tag{21}$$
$$J_f = J_3 + m_3(r_3 - d_3)^2 \tag{22}$$
$$J_r = J_2 + m_2(r_2 - d_2)^2 \tag{23}$$

The following torque then cancels the natural dynamics and enforces a PD control law:

$$\tau_{sp} = k_3\theta_{sp} + c_3\dot{\theta}_{sp} + J_{eff}(-K_{p,sp}\theta_{sp} - K_{d,sp}\dot{\theta}_{sp}) \tag{24}$$

We select PD gains $K_{p,sp}$ and $K_{d,sp}$ with reference to the desired second-order dynamics:

$$\ddot{\theta}_{des} = -\omega_n^2\theta - 2\zeta\omega_n\dot{\theta} = -K_p\theta - K_d\dot{\theta} \tag{25}$$

where $\omega_n = 25$ rad/s and $\zeta = 1$ are the target natural frequency and damping factor.

The legs are controlled to the desired touchdown angles with PD feedback linearization controllers based on rigid pendulums in the absence of gravity, and PD gains $K_{p,l}$ and $K_{d,l}$ selected as above with $\omega_n = 30\pi$ rad/s and $\zeta = 1$. The control torques for the rear leg τ_{rl} and for the front leg τ_{fl} are:

$$\begin{aligned}\tau_{rl} &= (m_1 d_1^2 + m_0 r_1^2)(-K_{p,l}(\theta_1 - \theta_{1,des}) - K_{d,l}\dot{\theta}_1) \\ \tau_{fl} &= (m_4 d_4^2 + m_5 r_4^2)(-K_{p,l}(\theta_4 - \theta_{4,des}) - K_{d,l}\dot{\theta}_4)\end{aligned} \tag{26}$$

C. Bounding Control

It has been observed that bounding animals primarily use their rear legs for propulsion, and tend to favor gaits with front stance followed by rear and long flight periods in between [1] [2]. In order to control bounding, we therefore selected the rear leg motors as the source of thrust for ballistic flight, while the front legs passively catch each fall, and the spine provides stiffness for coupling the leg forces to the body. By controlling the rear leg torque τ_{rl} such that the direction of the total ground reaction force is constant and just forward of vertical, we favor the generation of ballistic flight with a slight forward bias to overcome step to step losses. With appropriate leg touchdown angles $\theta_{rl,TD}$ and $\theta_{fl,TD}$, the body naturally tilts up during front stance, and back down during the final moments of rear stance while the CG continues to rise. If leg angles and radial forces are known and the desired force direction is θ_f then the required torque is:

$$\tau_{rl} = r_1 \frac{f_1(cos\theta_1 tan\theta_f - sin\theta_1) + f_4(cos\theta_4 tan\theta_f - sin\theta_4)}{sin\theta_1 tan\theta_f + cos\theta_1} \tag{27}$$

The control parameters for generating a particular gait are:

$$\boldsymbol{u} = [\theta_{rl,TD},\ \theta_{fl,TD},\ \theta_f] \tag{28}$$

We verified gait stability by analyzing the Jacobian for the Poincaré map between consecutive flight apex states, with limit cycles identified using the Newton-Raphson method as in [28]. For the primary controller in Section IV-D the Jacobian's largest eigenvalue had magnitude $|\lambda_{max}| \approx 0.63 < 1$, indicating a stable fixed point.

D. Motion Planning for Pit Obstacles

In order to generate a mesh of possible robot states and control actions, we selected a stable primary controller with parameters $\boldsymbol{u}_1 = [-59.5°,\ -66.5°,\ 86°]^\top$, and six unstable controllers each the same as $\boldsymbol{u}_1$ in two parameters and with the third incremented by $\pm\Delta\theta$, where $\Delta\theta$ is $5°$ for the leg angles and $2°$ for the force angle. The primary controller produces a bounding gait with a flight velocity of 2.1 m/s, and a step length of 0.5 m. We then generated a mesh as described in Section II, with $d_{thr} = 0.5$, resulting in 5454 nodes.

Because there is only one stable controller, in this case the goal of the A^* search is to find a control sequence that will allow a robot starting in the limit cycle associated with controller $\boldsymbol{u}_1$ to cross a gap in the terrain then return to the limit cycle. We assume that the gap width w_{gap} is known, as is the initial distance x_{near} from the front toe to the near edge of the gap. Control begins at the apex of the flight stage for a limit-cycle bound just after this stance period. The A^* cost functions favor control sequences that take a minimum number of steps to return to the limit cycle while placing the footfalls as far as possible from the gap edges. The step cost is therefore:

$$g(node) = g(parent) + 1 + \alpha/\Delta x_{gap} \tag{29}$$

where Δx_{gap} is the minimum distance between the footholds and the gap edge, and $\alpha = 1$ is selected to scale this cost appropriately relative to that for taking a step. The heuristic cost $h(node)$ is designed to estimate the number of steps to cross the gap, with an additional penalty for the normalized distance between state of the robot at the apex of flight for the limit cycle versus the end of the step.

$$h(node) = \begin{cases} \beta\sqrt{\sum_{i=1}^{n}\left(\frac{x_i - x_i^*}{\sigma_i}\right)^2} + \frac{x_{far} - x_1}{\mu_{\Delta x}}, & x_1 < x_{far} \\ \beta\sqrt{\sum_{i=1}^{n}\left(\frac{x_i - x_i^*}{\sigma_i}\right)^2}, & else \end{cases} \tag{30}$$

where $\beta = 4$ is a scaling factor, $\boldsymbol{x}^*$ is the limit cycle apex state vector, x_1 is the horizontal location of the rear hip, $x_{far} = x_{near} + w_{gap}$ is the location of the far end of the

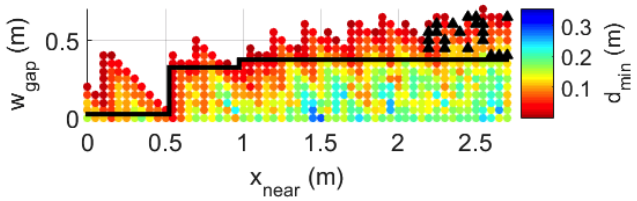

Fig. 9: Feasible combinations of gap width w_{gap} and distance to near edge of gap x_{near} from front foothold at start. Color indicates distance in meters from nearest foothold to gap when A^* found a working control sequence, while black triangles indicate sequences that caused the simulation to crash when applied. Black line gives largest gap w_{gap} that can be reliably crossed given a lookahead distance of x_{near}.

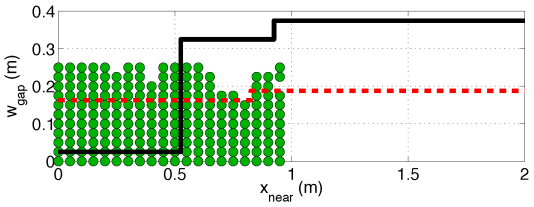

Fig. 10: Feasible cases for the controller developed with the 4-DOF model (green dots), along with max guaranteed gap width for a given lookahead for both 4-DOF scheme (red dashed line) and the 8-DOF case (solid black).

gap, σ_i is the standard deviation of the ith state across all points in the mesh, and $\mu_{\Delta x} \approx 0.5$ m is the average step length for all points in the mesh. Because this heuristic is can be larger than the total step cost for some paths, we trade optimal paths for short for run time. At each step of the search we then explore whichever state has the lowest estimated cost $f(node) = g(node) + h(node)$.

To verify this method, we generated A^* solutions for gap distances $x_{near} \in [0, 2.8]$ m and gap widths $w_{gap} \in [0, 0.7]$ m, sampling every 0.05 m for each. We then simulated application of each resulting control sequence, recording the minimum distance between the footholds and the gap as well as whether the robot crashed or the algorithm failed to find a solution. The results are presented in Fig. 9 as a scatter plot of gap width versus initial distance from the rear hip. A^* sequences that resulted in a successful simulation are indicated by green markers, sequences that failed upon simulation are shown in black, and empty spaces indicate no A^* solution.

For all combinations of gap width and distance tested, the A^* algorithm generally took less than 0.1 s to run, and successful motion plans required an average of 7 steps. While A^* solutions can be consistently found for gap widths below about 0.3 m when seen at least 0.5 m ahead, there are few solutions for gap widths greater than the limit cycle step length of 0.5 m. A sawtooth pattern appears in the results for gaps less than 0.5 m away, indicating that the footholds cannot be sufficiently altered in the first step to actively avoid a gap. For gap distances $x_{near} > 2.2$ m the results become unreliable as the actual robot behavior diverges from the mesh predictions.

V. Conclusion and Future Work

In this paper, we focus on two key goals. The first is the development of effective strategies for increasing the agility of a dynamic legged model, and the second is to propose preliminary tools and metrics for quantifying agility. While energy use is relatively easy to measure, robustness and agility are more challenging to quantify. Optimization algorithms require quantifiable metrics, and our long-term goals are to systematically improve agility for given task scenarios, via gradient-based methods, which in turn requires both good starting control strategies and metrics for "agility".

Focusing on the task of avoiding negative obstacles on flat terrain, we present two models for planar bounding, to explore scalability with increasing model complexity, and also present two general strategies for varying foothold placement, toward quantifying performance as gap width and/or lookahead knowledge of the terrain increase. On our 4-DOF model, we designed a set of controllers that each has a stable limit cycle tuned for a particular, unique step width. For the 8-DOF model, we instead design a single limit cycle controller, along with several more aggressive short-term "maneuver" controllers that do not demonstrate stable limit-cycles on their own. Figure 10 summarizes some key results. Data from Fig. 5 are replotted to show feasible combinations of lookahead and gap width that this system can handle. The dashed red line skims the tops of this feasible set to show the maximal region such that having a particular lookahead on terrain guarantees that the system can jump at least this w_{gap} distance. The solid black line gives the corresponding safe boundary for the 8-DOF system.

Summarized briefly, these data show that for a lookahead between zero and half a meter ahead, the 4-DOF control scheme (with multiple limit-cycle controllers) performs better, crossing any gap up to 0.18 m wide, compared with only about 0.04 m for the "aggressive maneuvers" of the 8-DOF system. However, with any longer lookahead, the 8-DOF system can cope with a gap of up to 0.375 (m), and if one is able to set the initial conditions of the system as desired, the 8-DOF system can handle a jump of over 0.7 meters.

As mentioned, our primary goal in this work is to suggest new means of quantifying and improving agility for dynamic legged systems. Future work should focus on the development of more controllers for terrain with varying slope, steps, and hurdles, and strategies for predicting circumstances under which the search will fail to find a solution. We also plan to test models of higher complexity, to quantify the robustness of switching policies in the presence of realistic terrain sensing, and to adapt methods previously used to increase mesh robustness in quantifying failure rate on rough terrain [19] toward improving robustness of optimal switching control policies.

REFERENCES

[1] M. Hildebrand, "Motions of the Running Cheetah and Horse," *Journal of Mammalogy*, vol. 40, no. 4, pp. 481–495, Nov. 1959.

[2] ——, "Analysis of Asymmetrical Gaits," *Journal of Mammalogy*, vol. 58, no. 2, pp. 131–156, May 1977.

[3] P. E. Hudson, S. A. Corr, and A. M. Wilson, "High speed galloping in the cheetah (Acinonyx jubatus) and the racing greyhound (Canis familiaris): spatio-temporal and kinetic characteristics," *The Journal of Experimental Biology*, vol. 215, no. 14, pp. 2425–2434, July 2012.

[4] 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, Dec. 1999.

[5] Q. Cao and I. Poulakakis, "Passive quadrupedal bounding with a segmented flexible torso," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2012, pp. 2484–2489.

[6] Q. Deng, S. Wang, W. Xu, J. Mo, and Q. Liang, "Quasi passive bounding of a quadruped model with articulated spine," *Mechanism and Machine Theory*, vol. 52, pp. 232–242, June 2012.

[7] U. Culha and U. Saranli, "Quadrupedal bounding with an actuated spinal joint," in *2011 IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 1392–1397.

[8] T. Kamimura, Y. Ambe, S. Aoi, and F. Matsuno, "Body flexibility effects on foot loading based on quadruped bounding models," *Artificial Life and Robotics*, pp. 1–6, Sept. 2015.

[9] Q. Cao and I. Poulakakis, "Quadrupedal bounding with a segmented flexible torso: passive stability and feedback control," *Bioinspiration & Biomimetics*, vol. 8, no. 4, p. 046007, Dec. 2013. [Online]. Available: http://iopscience.iop.org/1748-3190/8/4/046007

[10] K. Byl, B. Satzinger, T. Strizic, P. Terry, and J. Pusey, "Toward agile control of a flexible-spine model for quadruped bounding," vol. 9468, 2015, pp. 94 680C–94 680C–11.

[11] K. F. Leeser, "Locomotion experiments on a planar quadruped robot with articulated spine," Thesis, Massachusetts Institute of Technology, 1996. [Online]. Available: http://dspace.mit.edu/handle/1721.1/11227

[12] "Boston Dynamics," http://www.bostondynamics.com.

[13] G. Folkertsma, S. Kim, and S. Stramigioli, "Parallel stiffness in a bounding quadruped with flexible spine," in *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2012, pp. 2210–2215.

[14] A. Valenzuela and S. Kim, "Optimally Scaled Hip-Force Planning: A control approach for quadrupedal running," in *2012 IEEE International Conference on Robotics and Automation (ICRA)*, May 2012, pp. 1901–1907.

[15] G. C. Haynes, J. Pusey, R. Knopf, A. M. Johnson, and D. E. Koditschek, "Laboratory on legs: an architecture for adjustable morphology with legged robots," vol. 8387, 2012, pp. 83 870W–83 870W–14.

[16] J. L. Pusey, J. M. Duperret, G. C. Haynes, R. Knopf, and D. E. Koditschek, "Free-standing leaping experiments with a power-autonomous elastic-spined quadruped," vol. 8741, 2013, pp. 87 410W–87 410W–15.

[17] C. Saglam and K. Byl, "Switching policies for metastable walking," in *2013 IEEE 52nd Annual Conference on Decision and Control (CDC)*, Dec. 2013, pp. 977–983.

[18] K. Byl, "Metastable legged-robot locomotion," Ph.D. dissertation, Massachusetts Institute of Technology.

[19] C. O. Saglam and K. Byl, "Robust policies via meshing for metastable rough terrain walking," *In Proc. Robotics: Science and Systems (RSS)*, 2014.

[20] ——, "Switching policies for metastable walking," *In Proc. IEEE Conference on Decision and Control (CDC)*, 2013.

[21] P. Hart, N. Nilsson, and B. Raphael, "A Formal Basis for the Heuristic Determination of Minimum Cost Paths," *IEEE Transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, July 1968.

[22] K. Byl, B. Satzinger, T. Strizic, P. Terry, and J. Pusey, "Toward agile control of a flexible-spine model for quadruped bounding," *In Proc. SPIE 9468, Unmanned Systems Technology XVII*, 2015.

[23] D. P. Bertsekas, *Constrained Optimization and Lagrange Multiplier Methods*. Athena Scientific, Belmont, Massachusetts, 1996.

[24] J. S. Rossmann, C. L. Dym, and L. Bassman, *Introduction to Engineering Mechanics: A Continuum Approach, Second Edition*. CRC Press, Apr. 2015.

[25] H. Goldstein, C. Poole, and J. Safko, *Classical Mechanics*, 3rd ed. Addison Wesley, 2002.

[26] G. F. Franklin, J. D. Powell, and A. Emami-Naeini, *Feedback Control of Dynamic Systems*, 5th ed. Upper Saddle River, N.J: Prentice Hall, Nov. 2005.

[27] J. P. Hespanha, *Linear Systems Theory*. Princeton University Press, Aug. 2009.

[28] I. Poulakakis, E. Papadopoulos, and M. Buehler, "On the Stability of the Passive Dynamics of Quadrupedal Running with a Bounding Gait," *The International Journal of Robotics Research*, vol. 25, no. 7, pp. 669–687, July 2006.