Tools for the Design of Stable yet Nonsteady Bounding Control

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

***Abstract*— Most control development for running legged robots 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 or to recover and replan after perturbations. 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 and control methods have been proposed for bounding quadrupedal robots with passive compliant legs [4], but robots with flexible spines have only recently received significant interest. In pursuit of hypothesized gains in range of motion, thrust, and efficiency afforded by spine compliance [5], some models and control methods have been presented by [6]–[10], but little progress has yet been made toward achieving step-to-step agility.

Some quadrupedal robotic platforms which incorporate an articulated torso include Leeser's Planar Quadruped, Boston Dynamics' Cheetah and Wildcat platforms, and the MIT Cheetah [11]–[14]. These robots actuate the torso to gain additional force and range of motion, 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 parallel 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].

This work was supported in part by the U.S. Army's Robotics CTA through grant No. W911NF-08-R-0012, by an NSF CAREER award (CMMI 1255018), 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

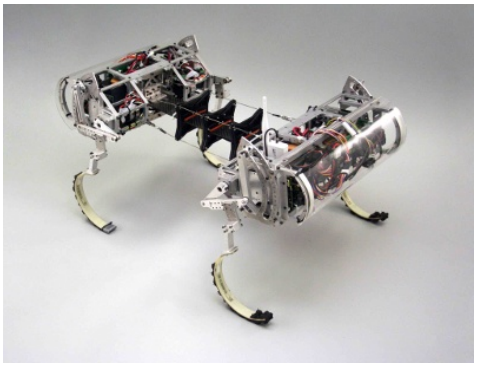

Fig. 1: Front view of the Canid robot.

In this paper, we describe a switching control framework for avoiding gap obstacles on flat terrain, applied to two planar simulations of Canid differing in their complexity and low-level control methods. In past work a finite set of controllers was used to generate a mesh of system states and control actions approximating the reachable space under the expected operating conditions [17]. We extend this by using task-specific controllers with an A^* search to find control sequences that will allow the robot to maintain dynamic stability while maximizing the distance between footholds and gaps in the terrain. Our first example employs step length control to robustly cross terrain gaps of varying width and lookahead, while the second achieves similar results using a single controller designed for steady-state running along with several short-term maneuvers.

The rest of the paper is organized as follows. Section II presents our high-level framework for mapping and searching the reachable state space associated to a given finite set of controllers. We then 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 six additional "maneuver" controllers, which do not demonstrate stable limit cycles. Finally, Section V gives conclusions and future work.

II. MESH-BASED OBSTACLE AVOIDANCE

Agility is a widely applied concept, but it is not yet clearly defined in legged robotics. Intuitively, it means being able to react to terrain variation by rapidly re-planning and changing the system trajectory while maintaining stability. Improving agility requires first defining some performance metric(s). For our task of bounding while avoiding terrain gaps, we

measure agility via the goals of maximizing the crossable gap width for a given lookahead and maximizing the distance between footholds and the gaps.

In this section, we present our framework for high-level switching control using a finite set of low-level gait controllers. Based on previous work [17]–[19], 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. This makes it 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

Originating from [17]–[19], our meshing algorithm Alg. 1 maps out a discrete approximation of all states the system can visit, along with all possible transitions between states. The mesh states are Poincaré snapshots of the robot, taken at a particular event in the gait cycle.

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

We begin the algorithm with at least two seed states: any fixed point(s) for limit cycle(s) of the controllers, and an absorbing failure state. 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:

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

where $\boldsymbol{x} \in \mathbb{R}^n$ is the system state, and σ_i is the standard deviation for state component i over all current points in the mesh. Any unexplored nodes enter a queue, and the algorithm keeps going until this queue is empty – indicating we have mapped out the reachable state space with the given resolution. The resulting mesh is a directed graph encompassing the reachable state space for the given terrain and set of controllers.

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 searching 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 sufficiently dense and state transitions are stored along with task-specific information such as the step-length or foothold locations. The A^* algorithm [20] (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 to the goal f, calculated as the actual cost of the path traversed so far, g, and a conservative heuristic estimate 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 lookahead, while penalizing low safety margins for footholds near the gaps. Because only one controller in the second example exhibits a stable limit cycle, its search includes the additional goal of returning to the limit cycle after crossing the gap.

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 length and angle of the non-stance leg are not included because we assume that the legs are massless and avoid having both feet on the ground at once. The model has 4 DOFs and is underactuated by 2 DOFs as each hip has one actuator. For more information about this model see [10].

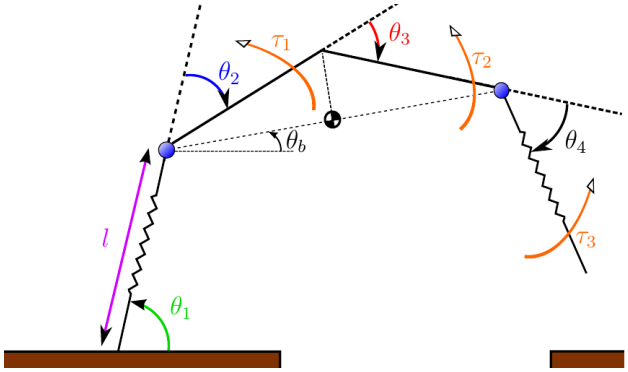

Fig. 2: 4-DOF Canid robot. Model parameters : hip masses $m_1, m_2 = 2.0kg$, joint inertia $J_1, J_2 = 0.0052kg.m^2$, neutral length of spring legs $l_0, l_0' = 0.254m$, spine segment lengths $L_2, L_3 = 0.254m$, leg stiffness $k_{leg} = 1550N/m$, leg damping $b_{leg} = 79N/(m/s)$, spine torque τ_2, spine stiffness (torsional) $K_{spine} = 60Nm/rad$, spine damping (torsional) $B_{spine} = 2Nm/(rad/s)$, and stance hip input torque τ_1.

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

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

where ξ is the input vector. Lastly, we form an augmented state vector $\dot{\boldsymbol{x}} = \left[\dot{\boldsymbol{q}}^\top, \ddot{\boldsymbol{q}}^\top\right]^\top$ so that the computation can be performed in MATLAB® with ode45().

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.

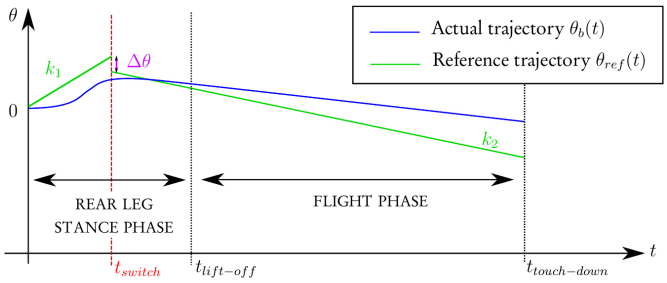

Fig. 3: Example reference and actual body angle trajectories.

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 $u_2 = -\theta_4$ are the desired leg touchdown angles set during the flight phase, 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. By tuning the reference trajectory parameters we can find actual trajectories with specific features such as step length, velocity, stability, or leap height.

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:

$$C(\boldsymbol{u}, \boldsymbol{x}^{(k)}) := \begin{cases} C_2, & \text{if the step fails} \\ C_1(\boldsymbol{u}, \boldsymbol{x}^{(k)}), & \text{otherwise} \end{cases} \tag{5}$$

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

$$C_1(\boldsymbol{u}, \boldsymbol{x}^{(k)}) = (\boldsymbol{l}(\boldsymbol{u}, \boldsymbol{x}^{(k)}) - \boldsymbol{l}_d^{(k)})^T W (\boldsymbol{l}(\boldsymbol{u}, \boldsymbol{x}^{(k)}) - \boldsymbol{l}_d^{(k)}). \tag{6}$$

Here $\boldsymbol{l}(\boldsymbol{u}, \boldsymbol{x}^{(k)}) = \left[\boldsymbol{\rho} \quad (\boldsymbol{x}^{(k+1)})^\top\right]^\top$ 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, $\boldsymbol{l}_d^{(k)} = \left[\boldsymbol{\rho}_d \quad (\boldsymbol{x}^{(k)})^\top\right]^\top$ is a vector of the desired gait parameters $\boldsymbol{\rho}_d$ and end state $\boldsymbol{x}^{(k)}$. 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:

$$W = \begin{pmatrix} W_p & 0_{n\times 8} \\ 0_{8\times n} & W_s \end{pmatrix} \qquad \begin{cases} W_s = w_s * \frac{1}{|L - L_d|} I_8 \\ W_p = w_p * |L - L_d| \end{cases} \tag{7}$$

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 $\varepsilon_u = 10^{-13}$. Recall that $H_C^k(\boldsymbol{u})$ and $\boldsymbol{\nabla} C$ are respectively the Hessian matrix and the Jacobian 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 significantly from one step to another. 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

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 $||\boldsymbol{u}^{(k)} - \boldsymbol{u}^{(k-1)}|| < \varepsilon_u$ **then**
    return $(\boldsymbol{u}^{(k)}, \boldsymbol{x})$
  end if
end for

test:

$$C(\boldsymbol{u}^{(k-1)}, \boldsymbol{x}^*) - C(\boldsymbol{u}^{(k)}, \boldsymbol{x}^*) < \sigma \beta^m s \nabla C d_{k-1} \quad (8)$$

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 [21]). As long as (8) is not satisfied, m is incremented to decrease the step size, and $\boldsymbol{u}^{(k)}$ is computed again.

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

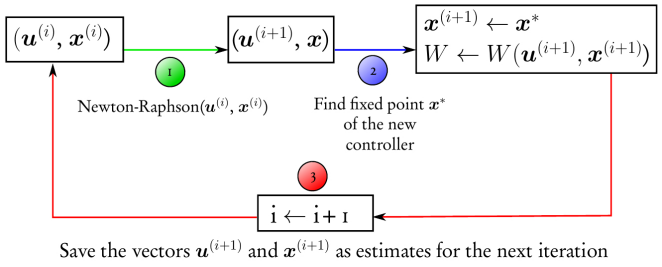

Fig. 4: Newton-Raphson iterative process

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

We used this method to find 9 controllers with step lengths in the range $[-20, 20]$ cm, each within 2 cm of the desired value and 5 cm apart from the next. It took about 5-10 hours to compute each controller, with the most time consuming operation being the computation of $H_C^k(\boldsymbol{u})$ that was done at each iteration of the Newton-Raphson aglorithm. All 9 were associated with a stable fixed point, which was unexpected as nothing in the algorithm guarantees multi-step stability.

C. Meshing and Path Finding Algorithm

We used these controllers to expand a mesh seeded from their 9 associated fixed points. Choosing a distance threshold $d_{thr} = 5 \times 10^{-3}$ resulted in a mesh with 1720 nodes. The obstacles used in this particular study were gaps of known width and lookahead, so the A^* cost functions were chosen to minimize the number of total steps, 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:

$$d_{rg} := |d_r - d_g| \quad (9)$$
$$d_{fg} := |d_f - d_g| \quad (10)$$

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:

$$h(node) := 1/d_f \quad (11)$$
$$g(node) := 1/min(d_{rg}, d_{fg}) \quad (12)$$

An important part of the implementation was to store the state transition map along with the distance of the front and rear footholds from the start during the mapping. This made it possible to precompute the heuristic cost, while the movement cost function was computed in the A^* algorithm (see Alg. 2).

D. Control Strategy Performance

In order to test the performance of our algorithm, we simulated A* motion plans for gaps of varying width w_{gap} and lookahead distance x_{near}. Each motion plan begins in the default bounding gait, and the lookahead x_{near} is calculated as the distance from the first front foot foothold to the near edge of the gap. Results are shown in Fig. 5.

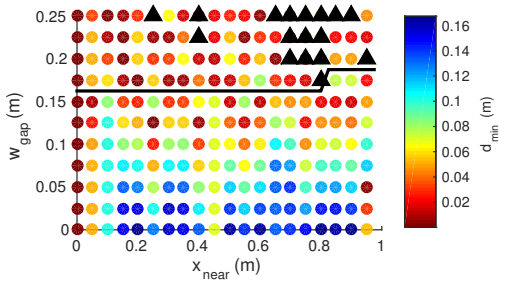

Fig. 5: Control strategy performance for gap widths w_{gap}, and lookahead x_{near}

In this plot, colored discs indicate the minimum distance d_{min} between the simulated footholds and the gap, while black triangles indicate cases in which the A* plan had been predicted to succeed, but failed when simulated. It is expected that the simulated trajectory should diverge from the predicted path due to the granularity of the mesh, and addressing approximation error is a key focus of future work. Lastly, the black line marks the largest gap that can be reliably crossed for a given lookahead.

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

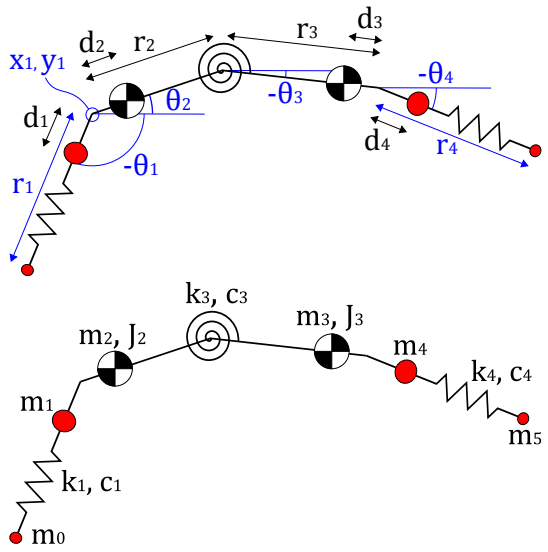

Fig. 6: (top) Geometry of the 8 DOF 2D model, with $d_1 = 0.1$ m, $d_2 = -0.048$ m, $d_3 = -0.049$ m, $d_4 = 0.09$ m, $r_2 = r_3 = 0.24$ m. (bottom) Inertial and passive elements with $m_0 = m_5 = 0.043$ kg, $m_1 = m_4 = 0.57$ kg, $m_2 = 4.4$ kg, $m_3 = 4.5$ kg, $J_2 = 0.0145$ kg-m^2, $J_3 = 0.015$ kg-m^2,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

model is motivated by a need to simulate the effect of leg masses, gaits including two-foot stance periods, and energy losses due to ground contact. Increasing the dimensionality of the system might limit the tractability of our meshing, which is a topic of on-going interest within our research group. The 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:

$$\tau_{2,3} = -k_3(\theta_3 - \theta_2) - c_3(\dot{\theta}_3 - \dot{\theta}_2) + \tau_{sp} \tag{13}$$

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:

$$f_i = -k_i(r_i - r_{L,0}) - c_i\dot{r}_i \tag{14}$$

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.

We determine the ground reaction forces using a viscoelastic (Kelvin-Voigt) model [22], with decoupled vertical and horizontal components computed from the location and

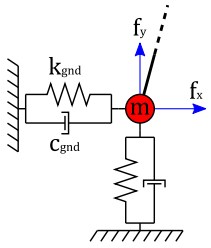

Fig. 7: Viscoelastic ground model.

velocity of the toe mass relative to the touchdown location (Fig. 7). The reaction forces are:

$$f_{i,x} = -k_{gnd} * (x_i - x_{i,g}) - c_{gnd} * \dot{x}_i \tag{15}$$

$$f_{i,y} = -k_{gnd} * (y_i - y_{i,g}) - c_{gnd} * \dot{y}_i \tag{16}$$

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 [23]. The generalized coordinates are:

$$\boldsymbol{q} = [\theta_1, \theta_2, \theta_3, \theta_4, r_1, r_4, x_1, y_1]^\top \tag{17}$$

as defined in Fig. 6. The rear hip provides the reference point to the global Cartesian coordinate system while all angles are measured counterclockwise from the positive x-axis.

As in the first example, after finding the equations of motion we rearrange them to find the accelerations $\ddot{\boldsymbol{q}}$, then form an augmented state vector $\boldsymbol{x} := \left[\boldsymbol{q}^\top, \dot{\boldsymbol{q}}^\top\right]^\top$ so that the computation can be performed in MATLAB®.

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

$$J_{eff}\theta_{sp} = -k_3\theta_{sp} - c_3\dot{\theta}_{sp} + \tau_{sp} \tag{18}$$

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

$$J_{eff} = J_r J_f / (J_r + J_f) \tag{19}$$

$$J_f = J_3 + m_3(r_3 - d_3)^2 \tag{20}$$

$$J_r = J_2 + m_2(r_2 - d_2)^2 \tag{21}$$

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

$$\tau_{sp} = k_3\theta_{sp} + c_3\dot{\theta}_{sp} + J_{eff}(-K_{p,sp}\theta_{sp} - K_{d,sp}\dot{\theta}_{sp}) \tag{22}$$

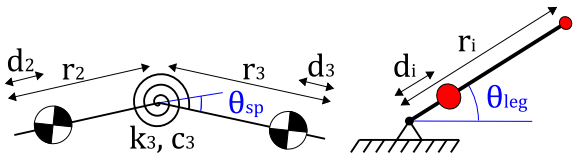

Fig. 8: Simplified models used for spine control (left), and leg control (right).

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

$$\ddot{\theta}_{des} = -\omega_n^2\theta - 2\zeta\omega_n\dot{\theta} = -K_p\theta - K_d\dot{\theta} \tag{23}$$

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:

$$\begin{aligned}\tau_{rl} &= (m_1 d_1^2 + m_0 r_1^2)(-K_{p,l}(\theta_1 - \theta_{1,des}) - K_{d,l}\dot{\theta}_1) \\ \tau_{fl} &= (m_4 d_4^2 + m_5 r_4^2)(-K_{p,l}(\theta_4 - \theta_{4,des}) - K_{d,l}\dot{\theta}_4)\end{aligned} \tag{24}$$

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 hip torque 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. We present a new feedforward control strategy in which the rear hip sets the angle of the total ground reaction force θ_f just forward of vertical. This has the effect of guiding the robot back into a ballistic trajectory while the slight forward bias overcomes impact losses to maintain speed. If leg angles and radial forces are known then the required torque is:

$$\tau_{rl} = r_1 \frac{f_1(cos\theta_1 tan\theta_f - sin\theta_1) + f_4(cos\theta_4 tan\theta_f - sin\theta_4)}{sin\theta_1 tan\theta_f + cos\theta_1} \tag{25}$$

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.

The control parameters for generating a particular gait are:

$$\boldsymbol{u} = [\theta_{rl,TD},\ \theta_{fl,TD},\ \theta_f] \tag{26}$$

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 [4]. 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 maneuver controllers. These were 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 distance x_{near} from the next front toe foothold to the near edge of the gap. Control begins at the apex of the flight stage for a limit-cycle bound just before this stance period so that the legs have time to move to the desired touchdown angles. The A^* cost functions favor control sequences that take a minimum number of steps to return to the limit cycle while placing the footholds as far as possible from the gap edges. The step cost is therefore:

$$g(node) = g(parent) + 1 + \alpha/\Delta x_{gap} \tag{27}$$

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 $h(node)$ includes an estimate of the number of steps to cross the gap, as well as an estimate of steps back to the limit cycle based on the divergence of the robot flight apex state:

$$h(node) = \begin{cases} \beta\sqrt{\sum_{i=1}^{n}\left(\frac{x_i - x_i^*}{\sigma_i}\right)^2} + \frac{x_{far} - x_1}{\mu_{\Delta x}}, & x_1 < x_{far} \\ \beta\sqrt{\sum_{i=1}^{n}\left(\frac{x_i - x_i^*}{\sigma_i}\right)^2}, & else \end{cases} \tag{28}$$

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 gap, σ_i is the standard deviation of the i^{th} state across all points in the mesh, and $\mu_{\Delta x} \approx 0.5$ m is the average step length. Because this heuristic can overestimate the total step cost for some paths, we trade optimal plans for short run time.

We simulated application of A* motion plans for lookahead $x_{near} \in [0, 2.8]$ m and gap widths $w_{gap} \in [0, 0.7]$ m, sampling every 0.05 m for each. The results are presented in Fig. 9. As in the first example, colored discs indicate the minimum distance between footholds and the gap for motion plans that simulated successfully. Sequences that failed on simulation are indicated by black triangles, while the space is left blank if the A* search found no solution. The black line gives the largest gap the robot can reliably cross with a given lookahead.

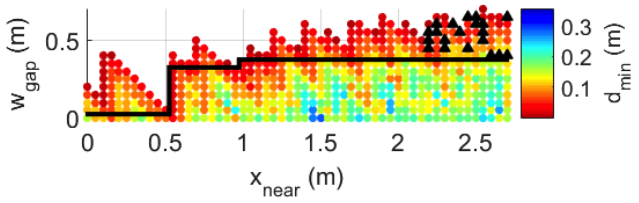

Fig. 9: Feasible combinations of gap width w_{gap} and lookahead x_{near}.

For all combinations of gap width and lookahead 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 the development of effective strategies for increasing the agility of dynamic legged robots, and propose preliminary metrics for quantifying performance in agile tasks. While energy use is relatively easy to measure and therefore optimize, it is more challenging to quantify agility because it is often seen as a multi-faceted task including the ability to e.g. quickly plan and efficiently execute a wide range of trajectories. Our long-term goal is to systematically optimize agility for specific task scenarios, which in turn requires both good starting control strategies and general 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. We also present two different gait control strategies toward quantifying performance as gap width and/or lookahead knowledge of the terrain increase. On our 4-DOF model we designed a set of 9 controllers, each tuned to a different step length, and capable of arbitrary switching without crashing. The use of a low-dimensional model with stable transitions between controllers helped to limit the size of the final mesh, and made it possible to cross terrain gaps up to 0.18 m with minimal lookahead. In contrast, the 8-DOF model used a single stable controller along with several ambitious maneuvers that do not permit arbitrary switching. This resulted in a much larger mesh, even when grown with a much higher distance threshold. The result was that the system was less capable of corrective action for nearby gaps, but was able to improvise "clever" trajectories for crossing large gaps when given sufficient lookahead.

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] I. Poulakakis, E. Papadopoulos, and M. Buehler, "On the stability of the passive dynamics of quadrupedal running with a bounding gait," *Int J of Robotics Res*, vol. 25, no. 7, pp. 669–687, July 2006.

[5] 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.

[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] 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.

[8] 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.

[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," *In Proc. SPIE 9468, Unmanned Systems Technology XVII*, 2015.

[11] K. F. Leeser, "Locomotion experiments on a planar quadruped robot with articulated spine," Thesis, Mass Inst of Tech, 1996.

[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 *IEEE Int Conf 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 *IEEE Int Conf 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.

[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.

[17] C. Saglam and K. Byl, "Switching policies for metastable walking," in *IEEE Int Conf on Dec 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 *Robotics: Science and Systems (RSS)*, 2014.

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

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

[22] J. S. Rossmann, C. L. Dym, and L. Bassman, *Introduction to Eng Mechanics: A Continuum Approach, 2nd ed*. CRC Press, 2015.

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