Mesh-Based Switching Control for Robust and Agile Dynamic Gaits

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

***Abstract*— In this paper, we describe and analyze mesh-based tools to control bounding motions of an 8 degree-of-freedom planar quadruped model with limited footholds on terrain. There are two complementary goals in our presentation. First, we aim to clarify potential advantages and disadvantages of our mesh-based approach in planning agile motions for a legged system. A key advantage is the ability to map the reachable states and their feasible transitions, given a relatively high-dimensional nonlinear dynamic system for which traditional meshing techniques would be impractical. A suspected disadvantage is that meshing has finite resolution, and robustness of mesh-based results should correspondingly be considered. Our second goal is to discuss appropriate frameworks for optimizing agility. Unlike typical locomotion optimization studies, in which control is designed for a limit cycle behavior that minimizes energy use or improves robustness to perturbations, here we focus on quantifying the performance of sets of controllers that together enhance reachability of the controlled system. In planning agile motions for our legged system model, we find that our mesh-based policies predict future dynamics robustly for plans up to about a 5-step horizon, and in quantifying controller sets, we emphasize that both the number of and parameterizations for such controllers should be considered in tandem during optimization.**

I. INTRODUCTION

One of the central aims in legged robotics is to be able to control foothold placement, for example to cope with intermittent terrain where wheels can not be used. However, it is generally much more challenging to design a control framework where step length can be adjusted, with the dynamics correspondingly varying at each step, than it is to design a single inter-step controller aimed at driving the system to a stable limit cycle. This problem gets particularly challenging as the desired step lengths increase, requiring the system to be more dynamic, with significant periods of ballistic flight between intermittent contacts at the ground.

A variety of successful approaches have been developed for control of bounding quadruped models [1]–[6], and even for real-world robots designed for bounding with a flexible spine [7]–[9]. By contrast, achieving the dual goals of planning for agile foothold placement and of quantifying and optimizing such planning goals for highly dynamic legged robots remains an open problem, to our knowledge.

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]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

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

The most relevant work we have found for quantifying agility is [8]. However, the authors focus on the ability of a robot to change its energy (e.g., accelerate and decelerate), rather that the ability to accurately control its end state and/or to exploit limited footholds.

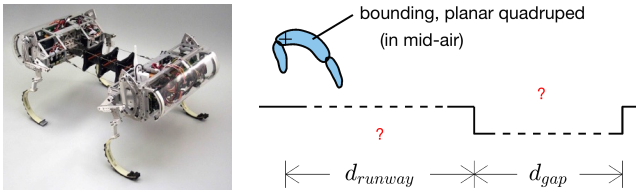

Fig. 1: A long jump problem, as one test of agility. The flexible-spine Canid robot (left) from the Kod*lab at UPenn inspires our planar quadruped model. The goal (right) is to jump across a range of gap sizes, d_{gap}, given a varying range of distances to the start of the gap, d_{runway}.

A promising recent approach for deriving control of a dynamic legged system with constraints on available foothold placement is presented in [10]. This work is part of a growing effort of research in trajectory optimization for legged systems, in a framework of Sequential Quadratic Programming (QP), yielding impressive results in recent years across implementations designed to optimize a variety of desired objective functions [10]–[14]. However, such methods are designed to solve for particular trajectories and corresponding actuator inputs for robot motions over time, without analyzing robustness of the results to noise and other sources of variability. Also, these methods are not appropriate for online planning, for example, to respond to new information as a robot is in locomotion and only senses upcoming terrain with a limited lookahead horizon. In a realistic legged robot scenario, both the upcoming terrain and current state of the robot are varying over time, and control methods must either be "fast enough to compute" or "compact enough to pre-compute".

In this work, we explore the later option, using a mesh-based approach to pre-compute reachable states and controller-dependent transitions among them. The approach studies one of the planar models presented in [15], but this paper focuses on several new and important topics, including analyses of meshing resolution, how different combinations of controllers affect performance in our example agility task, and whether there is a trade-off between jumping far and maintaining long-term dynamic stability. For this last aspect, visualize an olympic long-jumper. A crash-landing at the end allows for a longer jump than could be achieved if the athlete

were required to continue forward after finishing the jump.

Our approach and analysis is largely inspired by a growing body of work within our lab on mapping the reachable state space of a running or hopping system [16], [17], and on our previous work on meshing [18] as a method of minimizing fall rates (i.e., optimizing mean first-passage time), as originally proposed in [19]. The basic framework and theory are described in [20]. In its original formulation, our mesh-based approach aimed at minimizing fall rates for a walking model on rough terrain. In a general sense, it aimed at optimizing the robustness of the system to perturbations caused by variations in terrain height.

In this work, we instead focus on the goal of agility. Abstractly, a robust system aims "not to fail" when subject to various sources of variability. By contrast, being agile implies the ability to drive the system to any of a set of particular states or conditions, and to do so rapidly. There are many scenarios in which non-steady-state motion is useful, including locomotion on intermittent or rough terrain, recovery from perturbations, and adjustment of speed.

We argue that an intuitive notion of agility implies that a legged system (a) can move to any of a set of physical locations on terrain rapidly, and (b) can also reach a large range of state space. In other words, you wish not only to arrive at a particular physical location but also to do so with appropriate dynamics (e.g., capable of leaping a tall building, or of crossing a ditch, or simply of stopping in place, as required by higher-level planning objectives). In this work, we use a long jump task as a particular task requiring agility.

The rest of this paper is organized as follows. Section II describes our planar bounding model and gives an overview of the finite set of seven one-step, low-level controllers we use for high-level planning. Section III describes our algorithms to mesh the reachable state space and to derive approximate optimal control policies using this mesh. Section IV presents results for a long jump task, focusing on scalability of the mesh, accuracy of mesh-based predictions, and quantification of performance using various subsets of low-level controllers, and Section VI summarizes our conclusions and discusses directions for future work.

II. MODELING AND LOW-LEVEL CONTROL

We study an 8 degree of freedom (DOF) bounding model, shown in Fig. 2. All simulations for dynamics are done within Matlab, using equations of motion derived using the Lagrangian approach. The system includes masses at each foot as well as along the body, and has springs both within the legs and within a flexible spine. The model and low-level controls we use are outlined in more detail in [15]. We provide a general overview and highlight key aspects below.

We designed seven low-level controllers and allow for switching between controllers once per step for the planar, bounding model. Each low-level controller is characterized by three parameters: desired touchdown angle of the front leg, desired touchdown angle of the rear leg, and desired angle at which the ground reaction force acts during stance. We designed a particular (stable limit cycle) steady-state gait

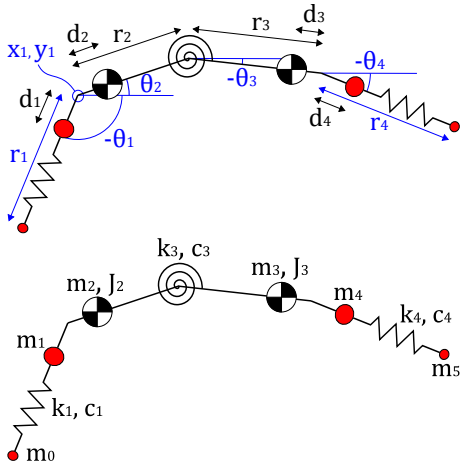

Fig. 2: (top) Geometry of the 8 DOF 2D model, with $d_1 = 0.1$ m, $d_2 = -0.048$ m, $d_3 = -0.049$ m, $d_4 = 0.09$ m, $r_2 = r_3 = 0.24$ m. (bottom) Inertial and passive elements with $m_0 = m_5 = 0.043$ kg, $m_1 = m_4 = 0.57$ kg, $m_2 = 4.4$ kg, $m_3 = 4.5$ kg, $J_2 = 0.0145$ kg-m^2, $J_3 = 0.015$ kg-m^2,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

controller and then created six additional controllers in which only one of the the three parameters was changed to be either smaller or larger than for the original controller, resulting in a total of seven controllers. Switching occurs when the rear hip of the model is at its apex.

More specifically, the 4-link model has three active torque inputs, so that the rotational internal joints are actuated while any pivot joint at either foot during ground contact is unactuated. Our control approach at each step drives each of the three actuated, internal joints to a desired angle. The control to do so uses Proportional plus Derivative (PD) control to set torque, based on a simplified feedback linearization approximation. Specifically, we use torque to cancel the spring and damper in the spine, and to account for figuration-dependent effects of inertia. For control of the rear and front leg joints, our feedback linearization accounts only for the pendulum dynamics of the leg. Our goal is to provide a method that does not depend on exact sensory knowledge of all states during control but still attempts to cancel out a large portion of the nonlinearities of the dynamics.

The control law for the spine is:

$$\tau_{sp} = k_3\theta_{sp} + c_3\dot{\theta}_{sp} + J_{eff}(-K_{p,sp}\theta_{sp} - K_{d,sp}\dot{\theta}_{sp}) \quad (1)$$

We select PD gains $K_{p,sp}$ and $K_{d,sp}$ that aim for the following desired second-order dynamics:

$$\ddot{\theta}_{des} = -\omega_n^2\theta - 2\zeta\omega_n\dot{\theta} = -K_p\theta - K_d\dot{\theta} \quad (2)$$

with natural frequency $\omega_n = 25$ rad/s and damping ratio $\zeta = 1$.

For each leg, the PD gains $K_{p,l}$ and $K_{d,l}$ are selected as above with $\omega_n = 30\pi$ rad/s and $\zeta = 1$. As mentioned, these controllers aim to cancel the free pendulum dynamics of each leg. Torques for the rear leg, τ_{rl}, and for the front

leg, τ_{fl}, are:

$$\begin{aligned}\tau_{rl} &= (m_1 d_1^2 + m_0 r_1^2)(-K_{p,l}(\theta_1 - \theta_{1,des}) - K_{d,l}\dot{\theta}_1) \\ \tau_{fl} &= (m_4 d_4^2 + m_5 r_4^2)(-K_{p,l}(\theta_4 - \theta_{4,des}) - K_{d,l}\dot{\theta}_4)\end{aligned} \tag{3}$$

The set points for the seven controllers do not vary dramatically and are hand-tuned to provide a range of behaviors for multi-step trials on flat terrain. Further details are not presented here, due to space limitations; our emphasis throughout is on evaluation of any given set of controllers used in switching, toward creating a framework for tuning and optimization of such a set, in future work.

III. MESHING AND HIGH-LEVEL CONTROL

This section describes our algorithm for generating a non-uniform mesh of reachable states, and of using this mesh for the high-level goals of finding optimal policies to return to a goal state, derived using value iteration, and brute force searches over feasible low-level control sequences to jump over a particular upcoming gap in terrain.

A. Meshing the Reachable State Space

Our meshing algorithm Alg. 1 (see [18], [20] for details) generates a discrete approximation of the full reachable set of states the system can visit, along with all possible transitions between states. The states in the mesh are Poincaré snapshots of the continuous-time dynamics, taken at the apex of each bounding motion, which is also the moment at which switching between controllers occurs. The mesh is generated by deterministically simulating each of the seven possible one-step controllers from each state in the mesh, only adding new points if they exceed a given distance metric from all other existing points in the mesh.

Algorithm 1 Meshing Algorithm

1: **Input :** Initial set of states P, set of controllers U, threshold distance d_{thr}
2: **Output :** State transition map M, Final set of states P
3: $Q_{next} \leftarrow P$
4: **while** Q_{next} not empty **do**
5: $\quad Q_{current} \leftarrow Q_{next}$
6: $\quad$ empty Q_{next}
7: $\quad$ **for** each $\boldsymbol{p}_j \in Q_{current}$ **do**
8: $\quad\quad$ **for** each $\boldsymbol{u}_k \in U$ **do**
9: $\quad\quad\quad$ 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.
10: $\quad\quad$ **end for**
11: $\quad$ **end for**
12: **end while**
13: **return** M, P

The algorithm begins with just two seed states: the fixed point for the limit cycle associated Controller 1, and an absorbing failure state, and as the total number of points in the mesh, N, grows over time, the threshold distance d_{thr} from every other point $\boldsymbol{p} \in \mathbb{R}^n$ in the mesh P is:

$$d(\boldsymbol{x}, P) = \min_{\boldsymbol{p} \in P} \sqrt{\sum_{i=1}^{N} \left(\frac{x_i - p_i}{\sigma_i} \right)^2} \tag{4}$$

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.

B. High-level control

Toward achieving agile motions, we explore high-level control to drive the system from the limit cycle behavior of Controller 1 (C1) to reach a wide variety of end states on terrain and in state space. With only seven controllers, it is possible to exhaustively explore all n-step combinations of low-level control up to some computationally practical limit in n. In practice, this limit is effectively $n = 10$ steps, corresponsing to roughly 4 to 6 meters ahead on terrain (depending on control actions chosen). Most of these sequences result in failures, where the robot crashes into the ground. We then search among the successful plans to examine two goals: (1) varying the location of the end state and (2) crossing a gap of particular size and location, ahead on terrain. Note that compared to our previous work in [15], we no longer use a tree-search and instead simply compute all possible transitions.

To investigate robust performance, we also solve for control policies to drive the system from an arbitrary state in our discrete mesh back to the limit cycle, or to any other point of interest in the mesh. We use the well-known value iteration algorithm to do so. Our implementation is summarized briefly below.

C. Value Iteration

We use a deterministic version of the Value Iteration algorithm to determine optimal control sequences between nodes in the mesh. Starting with a set of states $p \in P$, actions $a \in A$, and a state transition map $M(p, a)$, we select a goal state p_{goal}, and designate a failure state p_{fail}. The iteration begins by creating a vector U of utilities associated to each state, with $u_{goal} = 0$, $u_{failure} = 10^3$, and all others initialized to zero. On each iteration we update the utility for each state as $U(p) = C_{step} + \gamma \min_{a \in A} U'(M(p, a))$, where $C_{step} = 1$ is the step cost, $\gamma = 1$ is a discount factor, and U' is the utility vector from the previous iteration. By recording the control action that yielded the minimum utility for each state, we also update the optimal policy π^* on every iteration. This continues until the maximum change in utility for any state, δ, is less than a threshold $\epsilon = 10^{-4}$. For the mesh used in this study, δ converged to zero after 14 iterations, taking 0.7 s to compute.

IV. RESULTS AND DISCUSSION

This section presents our results for high-level motion planning, using our meshed approximation of the dynamic system. We focus on providing intuition for the feasibility of meshing and on providing a greater foundation for future directions within the legged robotics community, toward quantifying and improving agility.

Algorithm 2 Value Iteration

Input : Set of states P, actions A, state transition map $M(p, a)$, goal state p_{goal}, convergence threshold ϵ
Output : Vector of utilities for each state U, optimal policy from each state π^*
Local Variables : Utilities for last iteration U', max change in utilities from last iteration δ
$U(p_{goal}) = u_{goal}, U(p_{fail}) = u_{fail}$
while $\delta > \epsilon$ **do**
    $U' \leftarrow U$
    for all states p in P **do**
        $U(p) \leftarrow C_{step} + \gamma \min_{a \in A} U'(M(p, a))$
        $\pi^*(p) \leftarrow \arg\min_{a \in A} U'(M(p, a))$
    end for
    $U(p_{goal}) = u_{goal}, U(p_{fail}) = u_{fail}$
    $\delta \leftarrow \max_{p \in P} |U(p) - U'(p)|$
end while

A. Mesh Dimensionality

Our entire motivation for using a mesh-based approach is based on the observation that enforcing a low-level controller on the internal joints of a legged system significantly reduces the dimension of the reachable state space. For example, in our work with a 5-link point-footed walker, using a single sliding-mode, low-level controller during rough terrain walking resulted in an approximately 2D manifold of reachable Poincare' states [21]. Using a set of m low-level controllers, the reachable space is nearly (but somewhat larger than) a set of m 2D surfaces.

In this paper, we study a model of higher dimensionality, with 8 DOF instead of 5, i.e., with 16 total position and velocity states across. Meshing the full 16-dimensional space is not feasible, due to the curse of dimensionality, and so we have investigated similar deterministic meshing of the reachable state space. However, the planar bounding model includes springs in the legs, meaning the degree of underactuation is greater, and the PD control used here does not attempt to drive the internal states to their set points as aggressively as the sliding-mode control used for the walker.

Thus, the first aspect of interest in our results is to determine the effective dimensionality of the reachable state space, when switching control is implemented using our seven controllers. Depending on the structure of points in space, this can potentially be a fractional dimension [22]. Ultimately, what we care about is the rate at which the size of the mesh grows, as the desired resolution between nearby mesh points becomes more refined, which is what fractional dimensionality (e.g., fractal dimension) quantifies.

To estimate the dimensionality of our reachable state space, we first generated a "fine resolution" mesh, using a distance threshold of $d_{thr} = 0.25$ when using Equation 4 to test the distances to all other points in the mesh, during mesh generation. This resulted in a mesh of 52,826 points. Then, we randomized the order of points in the mesh and "repopulated" a new mesh, using a scaled version of the normalization set by the full mesh. (d_{thr} varies as the mesh is grown.)

Figure 3 shows our results, giving a log-log plot of mesh size (N) as a function of the allowed spacing between points (d). As expected, the data fall on a line as the mesh becomes coarse (to the right in the plot), since we have hypothesized $N \propto \left(\frac{1}{d}\right)^n$, where n is the "dimension" of the reachable state space. Toward the left, the mesh can never exceed the original size of 52,826 points, and the full data are bounded by these two extremes, depicting with the red and blue dashed lines.

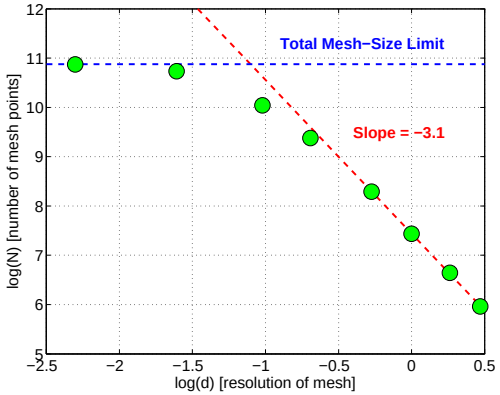

Fig. 3: Dimensionality of the mesh is approximately 3. Plot shows how the number of mesh points varies as the resolution of the mesh is reduced. Doubling the resolution increases the mesh size by about $2^{3.1}$, i.e., the mesh is approximately 3D.

Since the slope in Figure 3 is approximately -3.1, the mesh grows as $N \propto (1d)^{3.1}$. With $n \approx 3.1$, we have an approximately "3D" set of points. With such a low-dimensional mesh, we still have hope of achieving sufficient resolution to predict step-to-step dynamics. However, if the dynamics are highly sensitive to initial conditions (e.g., chaotic), this hope is lost. Detemining sensitivity to initial conditions is an open topic of interest for this work, although simulations of true dynamics (presented in IV-D) are promising for time horizons up to about 5 steps.

B. Value Iteration Results

Figure 4 shows one particular 3D projection of the "non-doomed" states within our mesh points, with the limit cycle of Controller 1 shown as a magenta star. Here, color indicates the number of steps required to arrive at this limit cycle state, as calculated using our value iteration algorithm, which uses a nearest neighbor approximation in predicting state transitions. The plot shows some of the structure of dynamic system. There are several distinct shapes, each corresponding to a particular controller used on the previous step.

The lower portion of Figure 4 shows the full mesh, including states which are doomed to fail, regardless of future control actions. Such points account for approximately one third of the total mesh.

The value iteration results are useful for two purposes. First, they enable us to more efficiently evaluate the long-term stability of solutions found for our long jump task. It is quite possible that optimal solutions to cross a large gap would require extreme motions that essentially "crash" onto the other side, and in fact, we predicted this would likely be

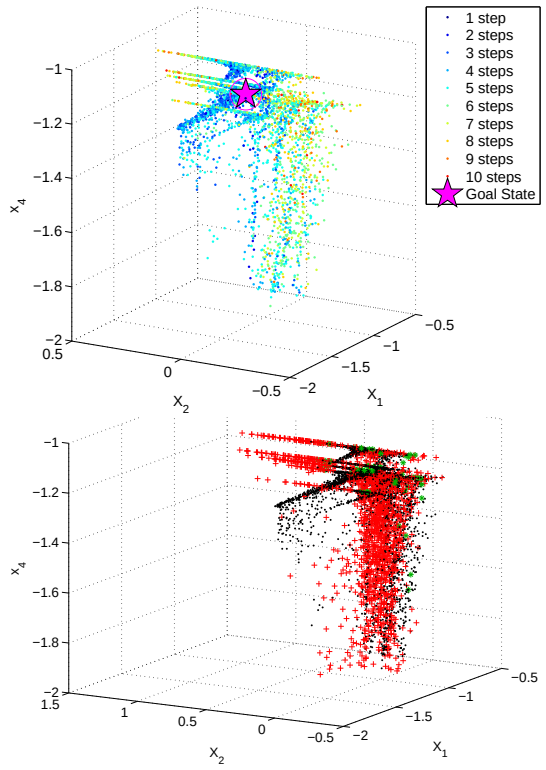

Fig. 4: 3D projection of mesh. In the upper plot, color indicates the number of steps required to return to the limit cycle motion. The lower plot highlights point in the mesh that are doomed to fail in either 1 step (red +) or two steps (green *). States (X_1, X_2, X_4) represent various joint angles here.

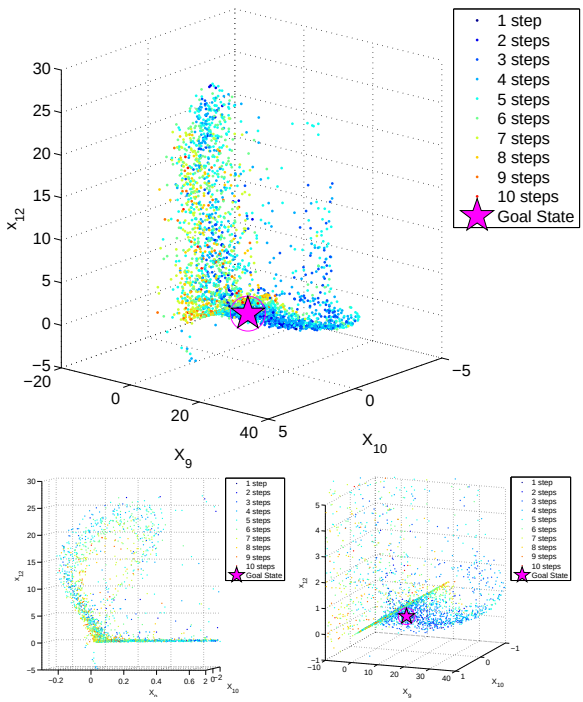

Fig. 5: Another 3D projection of the mesh states, using a different subset of states (X_9, X_{10}, X_{12}), which now represent various joint velocites.

the case. Interestingly, the optimal gap-crossing performance does not differ noticeably when we require that the final state has a feasible policy to return to the stable C1 limit cycle. Explaining this is a topic of interest for future work.

Figure 5 shows another 3D perspective of our mesh points, again depicting the "value" (defined as steps-to-goal) through color. Subplots here attempt to show more detail for each of the two otherwise relatively flat-looking surfaces (horizontally and vertically).

C. Control of Distance Traveled on Terrain

Definitions of agility are difficult to decouple from particular high-level application goals. However, one general goal is the ability to reach a large, open set of locations on terrain, which should grow as a function of the number of steps taken. Here, we investigate how this growth occurs and also whether the full set of seven states is required for optimal (or near-optimal) performance.

Figure 6 shows the set of rear hip locations in x as a function of the number of steps taken, starting with the C1 limit cycle state. Recall that we have a total of 7 different *sets* of low-level control parameters, sets C1 through C7. While removing any one of these seven low-level controllers does reduce the reachable set somewhat, some subsets of one-step controllers are nearly as effective as using the entire set. Removing Controller 7 (C7) from the set of available has the most significant effect on long-jump performance, and this is depicted to give the reader some idea of its effect of removal. As the number of steps taken, n, gets larger, the set grows roughly linearly, as expected.

We were curious whether or not the ability to vary end location on terrain might correlate well with overall long jump performance, because it would be very useful to minimize the number of intermediate metrics required to gauge agility across a variety of somewhat different tasks, for instance, planning for a single large gap, or for multiple smaller gaps, or for vertical obstacles, or for catching a moving target, etc. To investigate this, we explored performance of various subsets of the full set of 7 low-level controllers, always including C1 as one of these.

The best performance achieveable by a set of only three controllers came by using C1, C3, and C7. Using only these options results in a set of achievable end points spanning approximately 3/4 the full range, as illustrated. At the other extreme, removing C6 seems to have almost no effect, and the worst set of four low-level controllers (C1, C2, C4 and C5) reduces performance dramatically. Of note, although C2 and C5 do not seem to add much (beyond using C1 alone) in this task, they are still essential for good performance across a range of potential gap locations for the long jump, as described in the next section.

D. Planning for a Long Jump

Our final results focus on the long jump task. Here, both the distance to the near edge of the gap and the length of

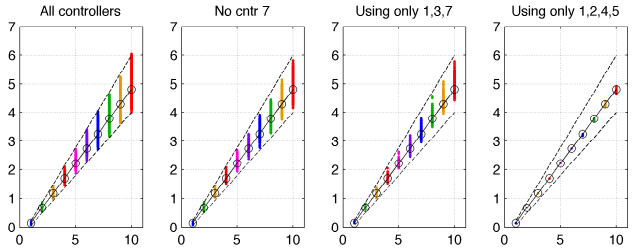

Fig. 6: Limiting policies affects n-step reachable distance on terrain. As the number of steps (x axis) increases, the set of reachable final footholds (y axis) increases almost linearly. The subplots illustrate how well various subsets of controllers perform, compared with the full set of seven; lines bounding the limits of the full set (leftmost subplot) are replotted on the other axes to aide in this comparison.

this gap are variables, and we determine the full range of conditions for which our mesh predicts a feasible control sequence exists. In simulations, we find that these predicted solutions are usually but not always successful.

More specifically, the median absolute value of the locations of footholds for predicted versus actual footholds across a 6-step planning horizon is less than one centimeter at each foothold. However, the distribution is bimodal, with approximately 10% of simulations showing large deviations from planned dynamics.

An important topic for future work is to determine if this is primarily due to limitations in meshing such dynamic systems, or simply indicates the low-level controllers can be better designed. We anticipate that using robustness of results as a benchmark during low-level control optimization may reduce sensitivity significantly, mirroring efforts in design of low level control for walkers which aimed directly at improving robustness of mesh-based policy performance [18].

Figure 7 shows performance for the long jump task. Here, there is a general trend in which the achievable gap length asymptotically reaches a maximum distance of 0.82 meters as the lookahead distance to prepare for the gap increases, along the x axis. The red dashed line is a first-order exponential decay, approximating this general trend. We also trace a lower bound on the worst possible performance bound, given the x axis represents a constant lookahead distance that a robot model has on terrain. In other words, with a lookahead of around 0.7 meters ahead, it is possible there will actually be a gap just short of one meter ahead, where there is a sharp, spikey drop in performance. The lower plot shows the same data on a semi-log plot, to emphasize the exponential approach to some maximal jump capability, as lookahead increases.

Figure 8 shows the same data for the case in which only low-level controllers 1, 3, and 7 are used. A similar asymptotic improvement occurs, but the maximum jump length is now 0.55 meters, rather than 0.82 (m). Significant descreases in performance occur when even a single controller is removed from the available set, except for C4, which has no apparent effect on optimal performance for

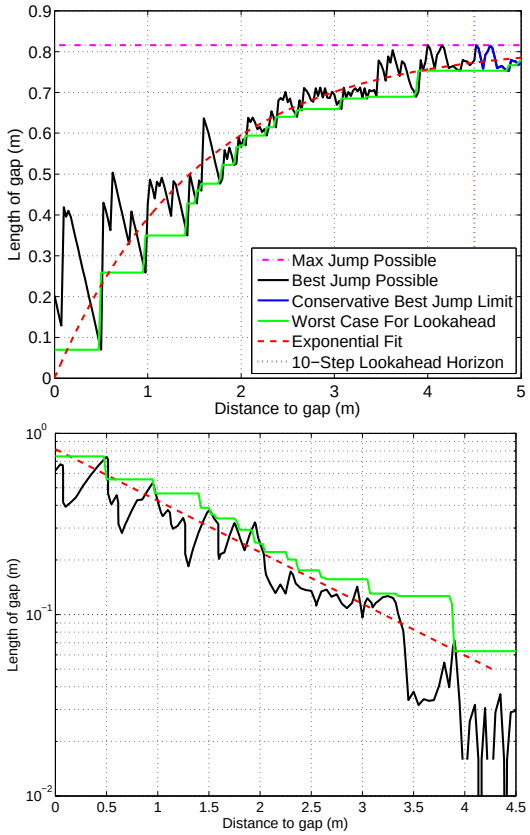

Fig. 7: Long jump performance: x axis show distance to the gap (i.e., lookahead), and y axis shows length of gap. Lower plot shows same data, with y now representing how much smaller the achievable gap width is, compared with the maximum gap length of 0.82 (m).

this task. Figure 9 shows footholds on terrain for particular optimal policies, for different gap widths and gap locations. Note that policies are rather non-intuitive: one does not simply use the limit cycle controller and then execute one, big jump. Corresponding, our mesh-based method is well-adapted to finding optimal policies and performance limits for such dynamic, nonlinear systems.

V. DISCUSSION

This approach, deterministically meshing the reachable state space for a dynamic system, is motivated by the need for better tools to optimize system parameters, low-level controllers, and high-level policies, although we focus only on the last of these topics within this work. Specifically, one can potentially optimize a vector of parameters describing both physical characteristics and low-level control parameters by repeatedly meshing the system, following an approach for low-level control optimization described in [23].

VI. CONCLUSIONS AND FUTURE WORK

This paper describes various aspects of a mesh-based approach for multi-step planning of dynamic legged system models, focusing on quantifying agility and investigating robustness. In describing agility, we focus on the ability to

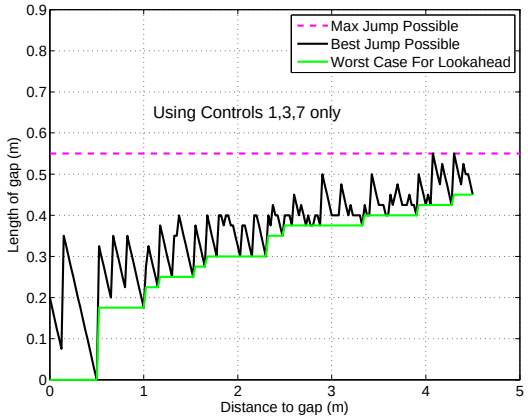

Fig. 8: Long jump performance, using only Controllers 1, 3, and 7.

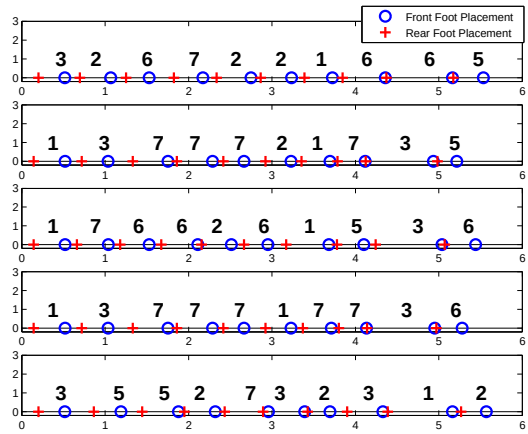

Fig. 9: Examples of optimal policies and their resulting foothold patterns.

reach a desired forward location in terrain in a particular number of steps, and of performing a long jump at variable distances ahead and for variable gap widths. Each task in turn requires access to a some region of state space, to change velocity, leg angles, and other variables. Motivated by the desired to improve agility in future work, this is a first step toward quantifying some important aspects of agility.

Of particular note, simulations of the dynamics for various initial conditions are generally but not always well-predicted by the mesh version of the dynamics when value iteration policies are used (to arrive at an arbitrary state in the mesh), which is a topic for further investigation. This mesh-based framework also provides a means of boot-strapping future design of sets of low-level controllers, with robustness between mesh-predicted and actual dynamics as a driving goal during optimization. That is, accuracy of the mesh is of greater importance than theoretical agility performance, based on mesh-based estimates: the estimates are only useful if accurate.

For future work, we also hope to combine multiple potentially conflicting factors in producing optimal policies. For example, [24] uses a metric to trade off robustness with energy use, and we envision eventually combining metrics for these two goals along with various metrics for agility.

REFERENCES

[1] 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, 2006. [Online]. Available: http://ijr.sagepub.com/content/25/7/669.abstract

[2] Q. Cao and I. Poulakakis, "Passive quadrupedal bounding with a segmented flexible torso," in *IROS*, 2012.

[3] U. Culha and U. Saranlı, "Quadrupedal bounding with an actuated spinal joint," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2011.

[4] U. Culha, "An actuated flexible spinal mechanism for a bounding quadrupedal robot," Master's thesis, Bilkent University, 2012.

[5] M. Khoramshahia, A. Sprowitzm, A. Tuleu, M. N. Ahmadabadi, and A. J. Ijspeert, "Benefits of an active spine supported bounding locomotion with a small compliant quadruped robot," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2013.

[6] 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, 94680C*, 2015.

[7] 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.

[8] J. M. Duperret, G. D. Kenneally, J. L. Pusey, and D. E. Koditschek, "Towards a comparative measure of legged agility," in *Proc. Int. Symp. on Experimental Robotics (ISER 2014)*, 2015.

[9] G. A. Folkertsma, S. Kim, and S. Stramigioli, "Parallel stiffness in a bounding quadruped with flexible spine," in *Proc. Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2012, pp. 2210–2215.

[10] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *International Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.

[11] L. Righetti, J. Buchli, M. Mistry, M. Kalakrishnan, and S. Schaal, "Optimal distribution of contact forces with inverse-dynamics control," *International Journal of Robotics Research*, vol. 32, no. 3, pp. 280–298, 2013.

[12] M. Posa and R. Tedrake, "Direct trajectory optimization of rigid body dynamical systems through contact," in *Algorithmic foundations of robotics X*. Springer, 2013, pp. 527–542.

[13] I. Mordatch, J. M. Wang, E. Todorov, and V. Koltun, "Animating human lower limbs using contact-invariant optimization," *ACM Transactions on Graphics (TOG)*, vol. 32, no. 6, p. 203, 2013.

[14] W. Xi, Y. Yesilevskiy, and C. D. Remy, "Selecting gaits for economical locomotion of legged robots," *Int Journal of Robotics Research [online first]*, Nov 2015, doi 10.1177/0278364915612572.

[15] V. Paris, T. Stizic, J. Pusey, and K. Byl, "Tools for the design of stable yet nonsteady bounding," in *Proc. American Control Conference (ACC) [in press]*, 2016.

[16] G. Piovan and K. Byl, "Reachability-based control for the active slip model," *Int Journal Robotics Research*, 2014.

[17] ——, "Approximation and control of the slip model dynamics via partial feedback linearization and two-element leg actuation strategy," *IEEE Trans. on Robotics [in press]*, 2016.

[18] C. O. Saglam and K. Byl, "Robust policies via meshing for metastable rough terrain walking," in *Proc. Robotics: Science and Systems X (RSS X)*. RSS, 2014, p. Paper 49.

[19] K. Byl and R. Tedrake, "Metastable walking machines," *Internation Journal of Robotics Research*, vol. 28, no. 8, pp. 1040–10 064, 2009.

[20] C. O. Saglam and K. Byl, "Metastable markov chains," in *Proc. IEEE Conference on Decision and Control (CDC)*, 2014.

[21] ——, "Stability and gait transition of the five-link biped on stochastically rough terrain using a discrete set of sliding mode controllers," in *Proc. IEEE Int. Conf. Robotics and Automation (ICRA)*, 2013.

[22] S. H. Strogatz, *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. Westview press, 2014.

[23] C. O. Saglam and K. Byl, "Quantifying and optimizing robustness of bipedal walking gaits on rough terrain," in *Proc. Int. Symp. Robotics Research (ISRR)*, 2015.

[24] ——, "Quantifying the trade-offs between stability versus energy use for underactuated biped walking," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS)*. IEEE, 2014, pp. 2550–2557.