2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)
October 25-29, 2020, Las Vegas, NV, USA (Virtual)

Fast Global Motion Planning for Dynamic Legged Robots

Joseph Norby and Aaron M. Johnson

Abstract— **This work presents a motion planning algorithm for legged robots capable of constructing long-horizon dynamic plans in real-time. Many existing methods use models that prohibit flight phases or even require static stability, while those that permit these dynamics often plan over short horizons or take minutes to compute. The algorithm presented here resolves these issues through a reduced-order dynamical model that handles motion primitives with stance and flight phases and supports an RRT-Connect framework for rapid exploration. Kinematic and dynamic constraint approximations are computed efficiently and validated with a whole-body trajectory optimization. The algorithm is tested over challenging terrain requiring long planning horizons and dynamic motions in seconds – an order of magnitude faster than existing methods. The speed and global nature of the planner offer a new level of autonomy for legged robot applications.**

I. Introduction

Legged robots must be able to autonomously execute dynamic motions to succeed in useful mobility applications. Figure 1 shows an example of terrain commonly found in outdoor mapping, inspection, or delivery tasks and that demands the ability to step and leap without falling. Automating these tasks requires the ability to plan ahead to ensure the robot has the appropriate positioning and velocity to execute the desired motion. This planning is challenging due to nonlinear dynamics, underactuation, intermittent contact including flight phases, and dependence on the terrain itself.

Model-based trajectory planning has proven an effective tool for planning motions while accounting for these challenges. The most straightforward form of trajectory planning is to give the planner full knowledge of the forces the robot can apply to the world and the constrained manner in which it can apply them, enabling the planner to predict how it can feasibly navigate the terrain and generate corresponding control inputs [1]. These methods have demonstrated impressive results in simulation, yet their numerical complexity renders them infeasible for real-time hardware deployment.

Rather than solve for the entire motion all at once with a high fidelity model, many successful robot implementations employ hierarchical control structures [2–4]. Such an architecture breaks the problem into multiple sub-problems that can be solved in parallel. Typically these layers include a long-horizon "global" planner to determine a rough path through the terrain to the destination, a short-horizon "local" planner that refines this motion and often selects footholds, and a low level controller that computes joint torques to send to the motors, Fig. 2. Distributing model complexity – with higher complexity for shorter horizons – ensures that each layer can be computed in real-time, yet presents the challenge of ensuring that any model simplifications can be resolved by lower layers.

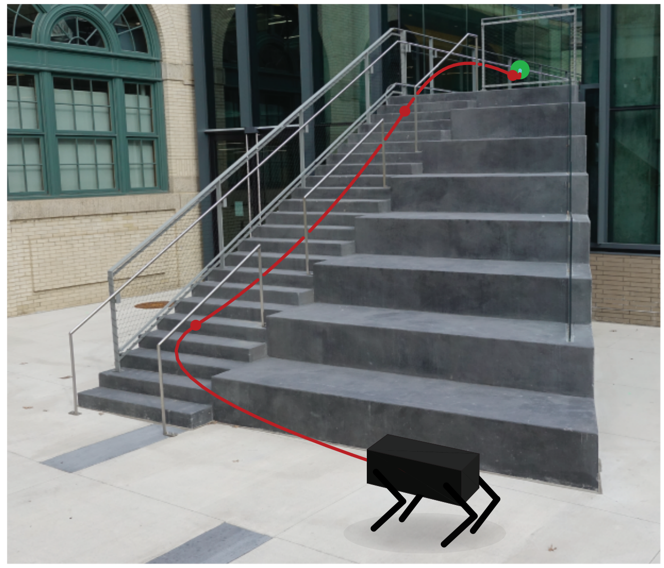

Fig. 1. Demonstration of the presented algorithm generating a long dynamical plan over terrain challenging for a legged robot in under three seconds. The trajectory from the start pose to the goal (green circle) is overlaid on an image of the environment used to generate the plan.

The global planner is critical to this pipeline due to its place at the top of the planning stack. A good global planner must have an accurate idea of what the robot can and cannot do to ensure that the other layers can resolve the motion, but also must have a long enough horizon to avoid local minima which is difficult to achieve in real time. Most global planners used in existing hierarchies achieve fast solution times by either ignoring dynamics to employ geometric planning methods like A* [5], restricting the horizon to only a few steps [6], or relying on traversability maps that use simple heuristics like maximum step height to avoid certain areas of the terrain [7].

This work introduces a fast global motion planning framework that can compute long-horizon feasible robot pose trajectories (body position and orientation) from which whole-body motions can be found, while accounting for intermittent

*This research was sponsored by the Army Research Office under Grant Number W911NF-19-1-0080. The views and conclusions contained in this document are those of the authors and should not be interpreted as representing the official policies, either expressed or implied, of the Army Research Office or the U.S. Government. The U.S. Government is authorized to reproduce and distribute reprints for Government purposes notwithstanding any copyright notation herein.

The authors are with the Department of Mechanical Engineering, Carnegie Mellon University, Pittsburgh, PA 15213, USA, {jnorby, amj1}@andrew.cmu.edu

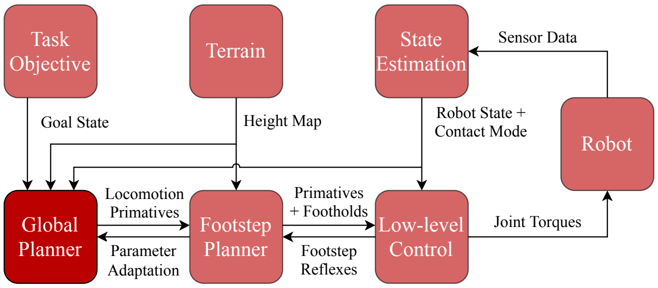

Fig. 2. The global planner presented in this work is at the top of a hierarchical planning and control stack. It is responsible for planning locomotion primitives to move the robot body from the current state to the goal state given terrain knowledge. It then passes those primitives down to footstep planning and low-level control for execution.

contact, underactuation, and constrained kinematics and dynamics. The framework plans with motion primitives that use a computationally efficient state and action parameterization in order to enable dynamic exploration of environments while also permitting the connection of two states for fast exploitation of trivial terrain. The framework also supports path length reduction methods to produce high quality paths, which is important for practical robot implementation. We present the results of four simulation experiments: the first demonstrates the validity of the constraint approximations with a whole-body trajectory optimization, the second analyzes the speed and horizon length of the planner, the third demonstrates path length reduction capabilities, and the fourth shows that this planner finds paths faster than state-of-the-art legged motion planning algorithms on a benchmark environment.

II. Related Work

Legged robots have long shown remarkable ability to perform dynamic behaviors, demonstrating walking, running, and even front-flipping as early as the 1980s [8,9]. Though these robots exhibit surprisingly robust locomotion, they rely on hand-tuned motions or clever mechanical stability through compliant legs or tails. These simple methods (which form the basis for the presented algorithm) are adept at walking over relatively flat terrain but cannot easily handle obstacles that require leaping. More recent robots have demonstrated impressive obstacle-leaping behaviors [10,11], but without a global planner that includes flight phases they cannot autonomously determine when and how to perform these tasks in unstructured terrain.

Accounting for unstructured terrain requires a notion of the capabilities of the system, and a method to employ this knowledge to plan a path through the environment. The clearest way to achieve this is through trajectory optimization with a full-order representation of both the kinematics and dynamics of the robot, an accurate model of the terrain, and a nonlinear solver that can find a feasible trajectory from the start to the goal. This method has shown impressive results in walking, running, jumping, and object manipulation [1,12,13], yet fully representing the nonlinearities and intermittent contact of legged locomotion yields extremely large and highly constrained nonlinear programs that take hours to solve on state-of-the-art solvers. Researchers have shown that ignoring any limb dynamics and modeling just the centroidal momentum of the system is sufficient to perform challenging tasks like running, jumping, and brachiation [2,14,15]. This approach reduces computation times from hours to minutes, but the problem still cannot be solved fast enough to be used in real time.

These solution times can be improved by further reducing the model complexity while relying on the aforementioned hierarchical control structure to handle any small feasibility violations. One common method of reducing model complexity is approximating kinematic constraints with more efficiently computed heuristics such as bounding boxes [6], reachability sets [16], learned measures [17,18], or ignoring them entirely [19]. Conservative kinematic approximations reduce the space of trajectories, but overly optimistic approximations can lead to infeasible plans that cannot be resolved lower in the hierarchy. Other methods simplify the dynamics further by enforcing quasi-static, inverted pendulum, or Zero Moment Point dynamics [3–5,20]. These reductions allow for planning over a long horizon of many steps but result in slow motion and cannot leverage momentum to overcome obstacles.

Some planners augment the terrain map with a notion of traversability, such that regions of the environment are processed and labeled with regards to the capabilities of the robot [7,21]. Features such as terrain slope, surface roughness, and step height yield a terrain score that approximates the likelihood of successful navigation. Common sampling-based methods such as Rapidly-exploring Random Trees (RRT) and their optimal counterparts (RRT*) [22] can then be employed to maximize the traversability of the given path. This approach has some features of a dynamic global planner in that it can be computed in real-time and account for the dynamic capabilities of the robot, but the construction of the traversability map is highly heuristic, robot and controller dependent, and unable to synthesize new motions to navigate.

One of the most promising methods, [23], utilizes the RRT-Connect algorithm [24] to quickly synthesize kinodynamic motions. This algorithm connects states by solving linear programs (LPs) to determine feasible acceleration bounds, then interpolates between states with these bounds. The algorithm plans flight phases by identifying states that violate a reachability requirement and searching for ballistic trajectories to surpass them. The repeated solving of LPs and heuristic approach to handling flight phases restrict the horizon under which this method can plan. The algorithm in this work parameterizes the state and action spaces similarly for compatibility with RRT-Connect, but in a way that allows for the automatic generation of feasible actions by randomly sampling within bounds on force constraints. This parameterization also directly incorporates flight phases rather than identifying them with heuristics. In addition, the presented algorithm incorporates path length reduction methods such as RRT* and anytime planning with short-cutting to produce high quality paths more amenable to execution on a robot.

III. Planning Algorithm

The core of the presented algorithm is the reduction of the dynamics of legged locomotion to a state-action pair conducive to both RRT-Connect for rapid exploration of mostly flat terrain and kinodynamic RRT (KD-RRT) for exploration of challenging terrain [25]. This is achieved by reducing the system state to just the robot's body position, orientation, and their velocities, with double integrator dynamics to determine the overall path through the environment. So long as the resulting trajectories can be feasibly tracked with whole-body motions – as shown in Section IV-A – this reduction permits efficient long-horizon planning while allowing more complex tasks such as footstep planning to be solved with existing shorter-horizon methods [6,26].

Pose motion is expressed as a sequence of piece-wise polynomials due to their computational efficiency and closed form solutions. Expressing motion with polynomials also allows dynamic constraints such as ballistic motion during flight phases and non-adhesive forces, friction cones, and actuation limits in stance phases to be automatically satisfied through the selection of appropriate polynomial coefficients. Motions are then planned by stitching these piece-wise trajectories in a way that satisfies kinematic constraints, in particular collision avoidance and reachability.

A. State Parameterization

Sampling-based planning methods rely on a set of states and actions that map to other states. The algorithm presented here defines these discrete time states as the position and orientation of the body of the robot along with their time derivatives. This sort of center of mass (COM) trajectory planning is not itself a new concept in sampling-based planning [25,27], but here we explicitly reason about intermittent contact and formulate the actions to account for the dynamics and kinematics of the system.

The discrete time states are defined at the beginning of a stance phase and denoted as,

$$\mathbf{s}_i = \begin{bmatrix} \mathbf{q}_i \\ \dot{\mathbf{q}}_i \end{bmatrix} = \begin{bmatrix} q_{x,i} & q_{y,i} & q_{z,i} & q_{p,i} & \dot{q}_{x,i} & \dot{q}_{y,i} & \dot{q}_{z,i} & \dot{q}_{p,i} \end{bmatrix}^T, \quad (1)$$

where $\mathbf{s}_i \in \mathbb{R}^8$ is the state at the beginning of the ith stance phase, $\mathbf{q}_i \in \mathbb{R}^4$ is the position and pitch of the system, x, y, z subscripts denote Cartesian directions, and the p subscript denotes the pitch. Roll is set to zero and yaw is defined as $q_{y,i} = \mathtt{atan2}(\dot{q}_{y,i}, \dot{q}_{x,i})$. These assumptions improve the speed of the algorithm at the cost of some model veracity, yet they are consistent with common locomotion trends of regulating roll and aligning heading with velocity. A stance phase is defined as a period of motion where a ground reaction force (GRF) is applied to the robot (and may correspond to multiple overlapping stance phases for each individual foot). Each stance phase is then followed by a flight phase consisting of ballistic motion, after which the next state $\mathbf{s}_{i+1}$ denotes the beginning of the following stance phase.

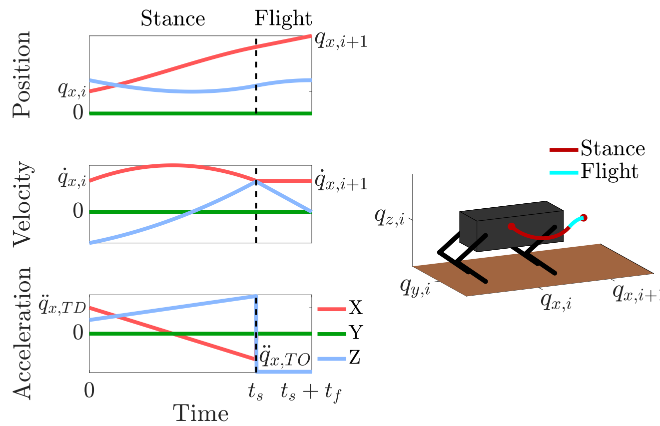

Fig. 3. Example state trajectory over one stance and one flight phase. Each planning state is defined as the body position, orientation, and velocity at the beginning of the stance phase. Each action is defined as a piece-wise linear acceleration trajectory during stance combined with the durations of stance and flight phases, yielding piece-wise cubic and quadratic polynomials for the position and velocity. During the flight phase the robot undergoes ballistic motion until it reaches touchdown. Robot legs are shown for context but not explicitly included in state or action definitions.

Figure 3 shows example states and the stance and flight phase trajectories connecting them. More complicated stance trajectories can be synthesized by setting the flight time t_f to zero to string multiple actions together.

Efficiently computed kinematic constraint approximations are applied at discrete intervals over the entire state trajectory. The ground clearance of the corners and center of the underside of the body are checked against a minimum height threshold to avoid collision with the ground. During the stance phase a maximum height threshold of the base of the leg linkage is also applied to ensure the ground is reachable for each leg. No such maximum is applied in flight, allowing the robot to reach terrain of different elevation by leaping up or falling down. These constraints can be written as,

$$h_{body}(\mathbf{q}(t)) \geq h_{min} \qquad \forall t \in [0, t_s + t_f], \quad (2)$$

$$h_{leg}(\mathbf{q}(t)) \leq h_{max} \qquad \forall t \in [0, t_s], \quad (3)$$

where $h_{body}(\mathbf{q}(t))$ and $h_{leg}(\mathbf{q}(t))$ are the clearance of the corners of the underside of the body and the base of the leg linkages, respectively, and h_{min} and h_{max} are the clearance thresholds.

B. Action Parameterization

These discrete time states are connected by actions that account for double integrator dynamics and synthesize piece-wise polynomial COM trajectories. Similar polynomial state trajectories have been employed in prior motion planners [23,28,29] but have not explicitly handled the intermittent contact of legged locomotion including flight phase dynamics. By specifying piece-wise linear ground reaction forces during stance, the resulting velocity and position trajectories are quadratic and cubic, respectively. Polynomials of this order are useful because cubic splines can efficiently connect states while maintaining dynamic feasibility. Each action is defined as,

$$\mathbf{a} = \begin{bmatrix} \ddot{\mathbf{q}}_{TD}^T & \ddot{\mathbf{q}}_{TO}^T & t_s & t_f \end{bmatrix}^T$$
$$= [\ \ddot{q}_{x,TD} \quad \ddot{q}_{y,TD} \quad \ddot{q}_{z,TD} \quad \ddot{q}_{p,TD} \quad \cdots$$
$$\ddot{q}_{x,TO} \quad \ddot{q}_{y,TO} \quad \ddot{q}_{z,TO} \quad \ddot{q}_{p,TO} \quad t_s \quad t_f\ ]^T \tag{4}$$

where $\mathbf{a} \in \mathbb{R}^{10}$ is the action taken, $\ddot{\mathbf{q}}$ denotes the acceleration of the system, t_s and t_f are the stance and flight time, and the subscripts TD and TO denote touchdown and take-off (start and end of stance times). Under this parameterization, the acceleration, velocity, and position of the system during stance can be written as,

$$\ddot{\mathbf{q}}(t) = (\ddot{\mathbf{q}}_{TO} - \ddot{\mathbf{q}}_{TD})\frac{t}{t_s} + \ddot{\mathbf{q}}_{TD}, \tag{5}$$

$$\dot{\mathbf{q}}(t) = (\ddot{\mathbf{q}}_{TO} - \ddot{\mathbf{q}}_{TD})\frac{t^2}{2t_s} + \ddot{\mathbf{q}}_{TD}t + \dot{\mathbf{q}}_{TD}, \tag{6}$$

$$\mathbf{q}(t) = (\ddot{\mathbf{q}}_{TO} - \ddot{\mathbf{q}}_{TD})\frac{t^3}{6t_s} + \ddot{\mathbf{q}}_{TD}\frac{t^2}{2} + \dot{\mathbf{q}}_{TD}t + \mathbf{q}_{TD}, \tag{7}$$

where $t \in [0, t_s]$ is the time since the beginning of the stance phase. Examples of these trajectories can be seen in Fig. 3. During flight the only acceleration is due to gravity, yielding the following trajectories,

$$\ddot{\mathbf{q}}(t) = \mathbf{g}, \tag{8}$$

$$\dot{\mathbf{q}}(t) = \mathbf{g}\,(t - t_s) + \dot{\mathbf{q}}_{TO}, \tag{9}$$

$$\mathbf{q}(t) = \frac{\mathbf{g}\,(t - t_s)^2}{2} + \dot{\mathbf{q}}_{TO}\,(t - t_s) + \mathbf{q}_{TO}, \tag{10}$$

where $t \in [t_s, t_s + t_f]$ and $\mathbf{g}$ is the gravity vector. These polynomials can also be reversed in time to enable planning tree growth from the goal towards the start.

Dynamic constraints are easily applied by computing bounds on the valid actions. Rather than bounding accelerations directly, or solving an LP to find acceleration boundaries as in [23], valid ground reaction forces at the beginning and end of stance are sampled directly, and then transformed into valid accelerations. This transformation is given by Newton's second law,

$$\begin{bmatrix} mI_3 & \mathbf{0} \\ \mathbf{0} & J_p \end{bmatrix} \ddot{\mathbf{q}} = \begin{bmatrix} \mathbf{f}_{GRF} + m\mathbf{g} \\ \tau_p \end{bmatrix}, \tag{11}$$

where $\mathbf{f}_{GRF}$ is the GRF, τ_p is the torque applied about the pitch axis, m and J_p are the mass and pitch inertia of the system, and I_3 is a 3×3 identity matrix. This transformation allows for actuation limits, non-penetration, and friction cone constraints to be respectively enforced by,

$$|\mathbf{f}_{GRF}| \leq f_{max}, \qquad |\tau_p| \leq \tau_{max}, \tag{12}$$

$$f_z \geq 0, \qquad \mu f_z \geq \sqrt{f_x^2 + f_y^2}, \tag{13}$$

where f_{max} is a fixed maximum ground reaction force, τ_{max} is a maximum torque threshold, f_z and $f_{\{x,y\}}$ are the normal and tangential components of the GRF respectively, and μ is the friction coefficient. Randomly sampling GRF vectors and torques at touchdown and takeoff within the bounds of these constraints, rotating into the world frame, and applying (11) allows for the automatic generation of feasible actions. The resulting piece-wise cubic position trajectories are then checked at regular intervals for kinematic feasibility to ensure collision avoidance and reachability. Upper and lower bounds are placed on the stance and flight times to aid the planner in producing feasible state-action pairs.

C. Planning Framework

These state-action pairs form the basis of the tree structure that RRT-Connect employs to explore the state space. While we refer the reader to prior literature for the detailed workings of RRT-Connect [24,27], the general strategy is to grow trees from the start and goal by alternately *extending* the trees towards randomly sampled states, then attempting to *connect* the trees together. These extend and connect functions can be thought of as two different locomotion primitives which together efficiently explore the environment. The planning framework presented here follows the RRT-Connect algorithm from [24], with particular instantiations of the extend and connect functions to account for the dynamics and kinematics of this problem.

The extend function in any RRT planner guides the planning tree towards new, unexplored areas of the state space by generating a random state, identifying the closest state in the tree, and leveraging a local planner to extend from that closest state towards the random one. The local planner implemented here samples desired states (1) uniformly but with zero pitch velocity to regulate orientation. As with KD-RRT [25], random actions described by (4) are sampled, checking for kinematic feasibility, and returning the closest new state to the desired under a weighted Euclidean distance metric. This extend function is effective at exploring challenging terrain by allowing frequent flight phases.

The connect function handles large portions of less challenging terrain more rapidly than the extend function. This is achieved by computing the unique closed form cubic spline that connects two states with zero flight time and a stance time that yields a nominal forward speed. Solving this boundary problem yields the following accelerations:

$$\ddot{\mathbf{q}}_{TD} = -\frac{6\,(\mathbf{q}_{TD} - \mathbf{q}_{TO}) - 2t_s(2\dot{\mathbf{q}}_{TD} + \dot{\mathbf{q}}_{TO})}{t_s^2}$$
$$\ddot{\mathbf{q}}_{TO} = \frac{6\,(\mathbf{q}_{TD} - \mathbf{q}_{TO}) - 2t_s(\dot{\mathbf{q}}_{TD} + 2\dot{\mathbf{q}}_{TO})}{t_s^2}. \tag{14}$$

The resulting forces are checked with (12)–(13) and discarded if dynamically infeasible, then checked for kinematic feasibility. If these constraints are satisfied the states are connected. If they are not satisfied, the algorithm considers the feasible portion of the trajectory and inserts a state into the tree corresponding to the midpoint of the feasible portion (to ensure the tree is not trapped against a constraint). No upper bound is placed on the distance of this connect operation to allow it to traverse large areas of terrain. The long "stance" phase produced by this operation can be thought of not as one single stance phase but as a trajectory induced by a collection of steps taken by the robot between flight phases, during which the lower level controller can easily find a valid whole-body motion.

Another benefit to such a connect function is the ability to exactly connect two states, which is helpful for reducing path length. We employ this ability in two separate ways and compare performance. One common way to reduce path length is to employ RRT*, which adds and removes connections through new states to reduce a cost function [27,30]. We implement the re-wiring algorithm described in [22], using the connect function to check for valid connections.

Though this method consistently reduces path length, re-wiring the connections whenever a new state is added slows the planner. A simpler short-cutting algorithm offers a faster but sub-optimal method of reducing path length after the completion of the planner [24]. This algorithm checks each state in the path for connections to other states. If a valid connection is found, those states are directly connected and all intermediate states are removed from the path. This process continues until the goal state is reached. Removing extraneous states in this fashion results in simpler, smoother paths without re-wiring throughout the planning process.

Path quality for this method is further improved by applying an anytime framework, wherein RRT-Connect is called repeatedly and only higher quality paths are accepted [31]. This anytime framework also improves the speed of the algorithm by restarting the planner if a solution is not quickly found, thus avoiding the increased complexity of adding more nodes to the planning trees.

IV. Algorithm Analysis

To analyze the presented planner, the kinematic and dynamic model approximations are validated with a full-order model trajectory optimization framework. We then generate robot paths over several terrains, analyze the resulting performance, and compare to other multi-contact kinodynamic planning algorithms.

A. Trajectory Validation

Much of the efficiency of the presented algorithm stems from approximations made to the kinematics and dynamics of legged robot locomotion. These approximations must be sufficiently expressive to capture the dynamic behaviors of interest yet conservative enough for a short-horizon planner to resolve. To test this formulation of the kinematic and dynamic constraints we randomly generate one hundred state-action pairs that meet the feasibility criteria described above, employing kinematic and dynamic bounds shown in Table I that approximate an MIT Cheetah 3 quadruped [32]. The resulting trajectories are then passed to a whole-body hybrid trajectory optimization created in the FROST framework [33]. This optimization produces motions that track the planned trajectories over flat ground while satisfying full-order dynamics and kinematics constraints, as well as a DC motor model and friction cones. To avoid solving the contact-implicit problem to validate these trajectories [13], which is quite slow and prone to non-convergence, we specify the contact sequence as all four feet starting on the ground, lifting off simultaneously, then ending within reach of the ground. This is slightly conservative as an ideal footstep planner would select the footstep sequence best suited to a particular motion, but such a planner is outside the scope of this work.

TABLE I

Algorithm settings used in Section IV. Parenthetical values show where ANYmal parameters (IV-C) differed from Cheetah 3 parameters (IV-A and IV-B)

Parameter	Symbol	Value Cheetah 3 (ANYmal)
Maximum velocity magnitude	$\|\|\dot{\mathbf{q}}\|\|$	4.0 (2.5) m/s
Maximum pitch	q_p	± 1.0 rad
Maximum pitch acceleration	τ_p/J_p	± 10 rad/s^2
Maximum hip height	h_{max}	0.6 m
Minimum ground clearance	h_{min}	0.02 m
Robot mass	m	43 (30) kg
Maximum GRF magnitude	f_{max}	800 (500) N
Friction coefficient	μ	1.0 (0.5)
Maximum stance duration	t_s	0.3 s
Maximum flight duration	t_f	0.5 s
Actions attempted per “extend”	–	6
Nominal “connect” velocity	–	1.5 m/s
Kinematic constraint resolution	–	50 ms

The optimization framework was able to find solutions for 98% of the tested state-action pairs, indicating that this planner's heuristics closely approximate the full model. Those that it could not find a solution for nearly converged but could not overcome the conservative kinematic restrictions of requiring contact at all four feet during stance, and in each case it is likely that a full local footstep planner would still succeed.

B. Algorithm Performance

To test the performance of the presented planner, we created several environments that pose varying motion planning challenges. While their complexity does not span that of all possible environments, they qualitatively demonstrate performance in key terrains of interest. For each environment we specify a start and goal pose, provide the planner with a height map of the terrain, execute the anytime planner with short-cutting 100 times, and collect statistics. The planner finds a feasible path then executes the short-cutting algorithm but does not re-plan to prioritize algorithm speed. All trials are run in C++ on an Intel Core i7-8700K CPU at 3.7 GHz.

The “Rough Terrain” environment shown in Fig. 4 tests the planner's ability to overcome unstructured terrain including uneven ground, obstacles, and a ledge requiring a flight phase. The “Hallway” environment shown in Fig. 5 tests the ability to generate long-horizon paths in the presence of local minima. The “Slope” terrain shown in Fig. 6 tests the ability to handle steep slopes. Finally, the “Staircase” environment shown in Fig. 1 and 7 puts all these together, requiring a long-horizon, highly dynamic path that navigates slopes and local minima. The Staircase environment was tested with and without an obstacle blocking the middle of the small stairs, requiring the robot to navigate the larger stairs.

Table II shows the statistics averaged over each set of trials on the environments including the amount of time spent planning until a feasible path was found, the number

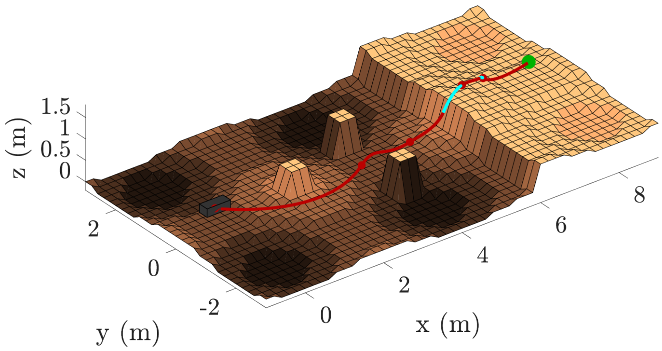

Fig. 4. The Rough Terrain environment includes a ledge requiring a flight phase, in addition to uneven terrain and obstacles. In this and the following figures, red lines indicate a stance phase, cyan indicates flight, the black box indicates the robot starting pose, and the green circle indicates the goal.

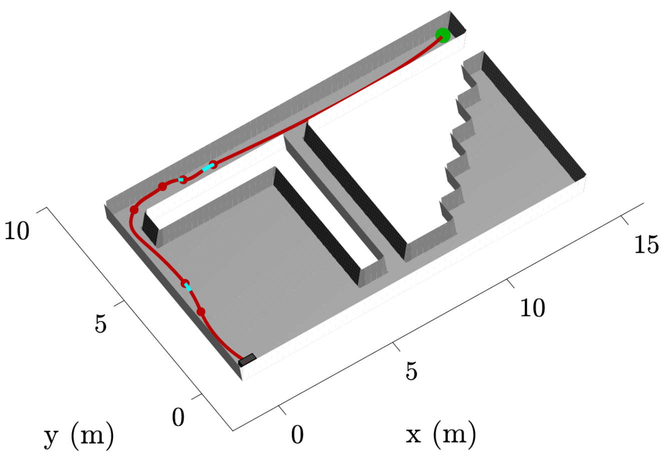

Fig. 5. The Hallway environment replicates an indoor areas with long stretches of flat terrain. The local minima presented by the large open portion or the area that approaches but does not reach the goal require a planner with a long enough horizon to avoid.

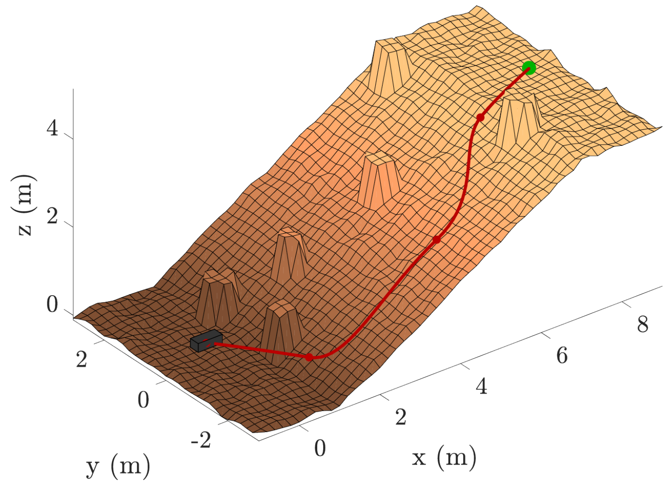

Fig. 6. The Slope environment tests the planner's ability to incorporate friction cones and usage of orientation to find more feasible regions of the state space.

TABLE II

ALGORITHM PERFORMANCE ON TEST ENVIRONMENTS. DATA REPORTED AS MEAN ± STANDARD DEVIATION OVER 100 TRIALS.

Environment	Plan Time (s)	States Generated	Plan Length (m)
Rough Terrain	2.3 ± 2.2	1100 ± 1000	14 ± 2.8
Hallway	1.8 ± 1.4	640 ± 420	25 ± 2.8
Slope	0.15 ± 0.2	150 ± 31	13 ± 2.1
Staircase	2.9 ± 2.5	720 ± 550	17 ± 2.6
Staircase w/obstacle	26 ± 20	4400 ± 2700	21 ± 3.8

of states generated during the planning call, and the length of the returned trajectory. Each environment figure shows an example trajectory found on that terrain, and animations of trajectories are shown in the video attachment.

The results shown in Table II demonstrate the effectiveness of the presented planner in handling a wide variety of challenging legged robot planning tasks, as every planning call was successfully completed. The length of each plan is on the order of tens of body lengths, showcasing a horizon long enough for global planning. Each plan – with the exception of the Staircase with the obstacle – is computed on the order of seconds and therefore fast enough for real-time deployment. The Hallway environment in Fig. 5 showcases the connect function's ability to rapidly traverse wide sections of trivial terrain and return a straightforward path with little erratic motion while avoiding local minima. Conversely, the performance on the Rough Terrain environment demonstrates the ability to navigate dynamically challenging terrain with flight phases which are unresolvable by geometric planners.

These challenges are combined in the Staircase environment. The algorithm automatically discovers that the regular stairs are more traversable than the large stairs and explores that area rapidly. This behavior is desirable as favoring less challenging motion improves the likelihood of resolution by a lower level controller. When those stairs are obstructed, the algorithm employs multiple flight phases to leap up the large stairs towards the goal, although it takes much longer to find such a path through the tightly constrained space.

The two methods of reducing path length – RRT*-Connect and anytime RRT-Connect with short-cutting – are tested on the Slope terrain environment in Fig. 6 and allowed to run for five seconds before termination. Each planner is called one hundred times and the results are averaged. The results are shown in Fig. 8, along with the average path length for regular RRT-Connect with short-cutting.

Figure 8 demonstrates a small advantage of anytime RRT-Connect with short cutting over RRT*-Connect in quickly reducing path length on the "Slope" environment, though both perform well compared to the non-optimal planners. The anytime framework is able to reduce cost by rapidly growing multiple trees in succession rather than re-wiring a single tree. Both of these methods outperform standard RRT-Connect in less than one second, demonstrating the utility of these cost-reducing strategies in yielding high quality paths.

C. Algorithm Benchmarking

We compare the presented algorithm directly against other state-of-the-art methods on the "Plinth" environment, based

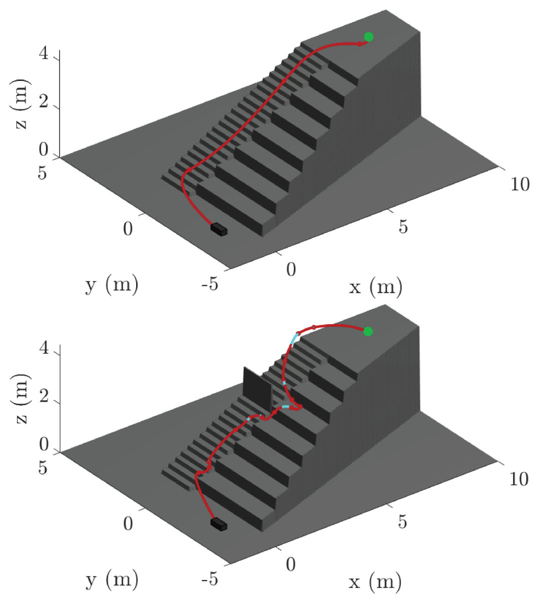

Fig. 7. The Staircase environment, top, based on the location from Fig. 1, presents a long-horizon task with discrete changes in ground height. When an obstacle is placed on the regular stairs, bottom, the planner must find a dynamic path over the larger steps.

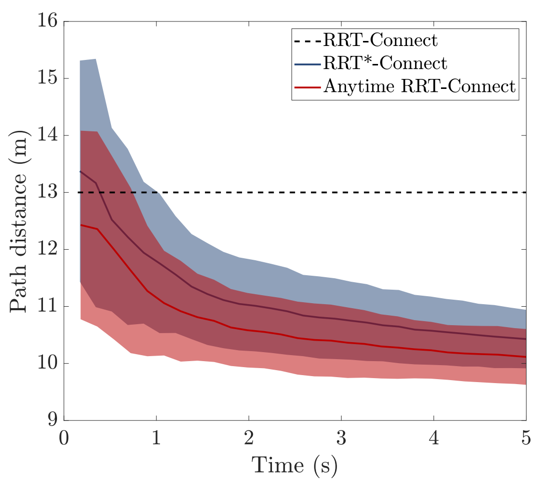

Fig. 8. Path length reduction for three variations of the presented planner. RRT-Connect returns after 0.15 seconds on average but does not further improve path quality. The solid lines indicate the mean and the shaded regions indicate one standard deviation over 100 trials.

on [34], shown in Fig. 9. In particular we benchmark against the root trajectory from the multi-contact RRT-Connect planner described in [23] and implemented in [34] as well as the contact-implicit trajectory optimization method presented in [6]. Each test is performed with model parameters that approximate the ANYmal quadruped [35], Table I. The presented algorithm and the trajectory optimization are executed on the CPU described above, and the statistic for the multi-contact RRT-Connect planner is obtained directly from [34] which utilized a similar CPU.

The results from the benchmarking are shown in Table III. The presented algorithm constructs plans six times faster

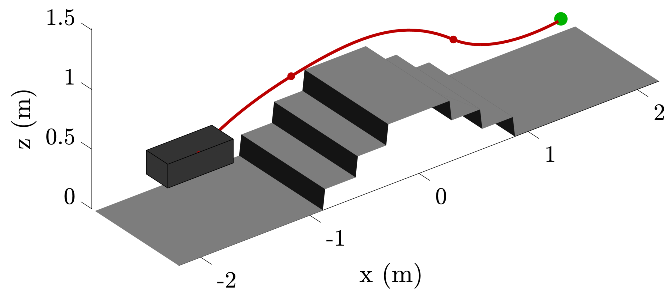

Fig. 9. The algorithm benchmarking is performed on the Plinth environment, which requires navigating height changes but does not explicitly require a flight phase.

TABLE III
ALGORITHM COMPARISON ON PLINTH ENVIRONMENT. DATA REPORTED AS MEAN ± STANDARD DEVIATION OVER 100 TRIALS.

Solver	Dimension	Plan Time (s)
Presented Algorithm	2D	0.20 ± 0.21
Presented Algorithm	3D	1.0 ± 0.91
Prior RRT-Connect [34]	2D	1.3 ± unreported
Traj Opt (w/contacts) [6]	3D	9.5 ± 0.082

than the multi-contact RRT-Connect method when similarly constrained to planar motion, and is still slightly faster when allowed to explore the environment in three dimensions. Plan lengths from each algorithm were comparable. This algorithm is also around an order of magnitude faster than the trajectory optimization method, although that method computes footstep locations in addition to a pose trajectory. This suggests that such a method could be an effective short-horizon planner to determine contact locations a few steps ahead once provided the global plan from the algorithm presented here.

V. Discussion and Conclusion

The diversity of the environments tested showcases the speed and ability to dynamically overcome obstacles of the presented algorithm. The trajectory validation results and optimality analysis indicate that the resulting paths are of high quality and likely to be resolved by a low level planner. These features together highlight the practicality of this algorithm for deployment as a global planner for legged robots.

By design, many of the assumptions and model approximations made in this planner are heuristic in nature and thus offer no model-based guarantees. The strength of this approach is rooted in the objective of the planner – finding a basic path through an unstructured environment that can serve as an initial seed for short horizon footstep and whole-body planners. Relaxing hard constraints on full-order feasibility allows for each constraint to be expressed in a more computationally efficient way. This enables longer planning horizons, ensuring that the footstep and whole-body planners can focus on refining motion rather than trying to escape local minima. In particular, reasoning about contact via the net ground reaction force rather than through each individual contact captures the hybrid nature of intermittent contact but does not enforce particular contact locations. This approach allows a shorter horizon footstep planner with a more expressive contact model to dedicate increased computation to selecting robust contact locations.

Nevertheless, some of these assumptions do require careful treatment for certain classes of systems. For example, the centroidal dynamics model is reasonably accurate for robots with negligible leg inertia, but may necessitate compensation for systems with high-inertia limbs such as those in many bipeds. This framework also assumes that the GRF vector always extends through the COM and that orientation can be independently actuated. Lower-level controllers therefore must carefully handle the coupling between leg and body control, particularly for bipedal robots or other highly underactuated systems.

It should be emphasized that the presented algorithm does not solve the entirety of legged locomotion planning, but rather provides the top level of a hierarchical framework through which the robot can autonomously reason about what path it should take though an unstructured environment. The other components of this hierarchical architecture – including a robust footstep planner, a high bandwidth whole-body controller, and reliable state estimation – are critical for the success of autonomous and dynamic legged robots, and constitute active areas of research.

ACKNOWLEDGMENT

The authors would like to thank Nikolai Flowers for coding assistance and Maxim Likhachev for helpful discussions.

REFERENCES

[1] K. Mombaur, "Using optimization to create self-stable human-like running," *Robotica*, vol. 27, no. 3, pp. 321–330, 2009.
[2] S. Kuindersma, R. Deits, M. Fallon, *et al.*, "Optimization-based locomotion planning, estimation, and control design for the Atlas humanoid robot," *Autonomous Robots*, vol. 40, no. 3, pp. 429–455, 2016.
[3] M. Zucker, N. Ratliff, M. Stolle, *et al.*, "Optimization and learning for rough terrain legged locomotion," *The Intl. Journal of Robotics Research*, vol. 30, no. 2, pp. 175–191, 2011.
[4] P. Fankhauser, M. Bjelonic, C. D. Bellicoso, *et al.*, "Robust rough-terrain locomotion with a quadrupedal robot," in *IEEE Intl. Conference on Robotics and Automation*, 2018, pp. 1–8.
[5] A. Hornung, A. Dornbush, M. Likhachev, and M. Bennewitz, "Anytime search-based footstep planning with suboptimality bounds," in *IEEE Intl. Conference on Humanoid Robots*, 2012, pp. 674–679.
[6] A. W. Winkler, C. D. Bellicoso, M. Hutter, and J. Buchli, "Gait and trajectory optimization for legged systems through phase-based end-effector parameterization," *IEEE Robotics and Automation Letters*, vol. 3, no. 3, pp. 1560–1567, 2018.
[7] M. Wermelinger, P. Fankhauser, R. Diethelm, *et al.*, "Navigation planning for legged robots in challenging terrain," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2016, pp. 1184–1189.
[8] M. H. Raibert, *Legged robots that balance*. MIT press, 1986.
[9] J. Hodgins and M. H. Raibert, "Biped gymnastics," *Dynamically Stable Legged Locomotion*, vol. 79, 1988.
[10] A. M. Johnson and D. E. Koditschek, "Toward a vocabulary of legged leaping," in *IEEE Intl. Conference on Robotics and Automation*, 2013, pp. 2568–2575.
[11] H.-W. Park, P. M. Wensing, S. Kim, *et al.*, "Online planning for autonomous running jumps over obstacles in high-speed quadrupeds," in *Robotics: Science and Systems*, 2015.
[12] I. Mordatch, E. Todorov, and Z. Popović, "Discovery of complex behaviors through contact-invariant optimization," *ACM Transactions on Graphics*, vol. 31, no. 4, pp. 1–8, 2012.
[13] M. Posa, C. Cantu, and R. Tedrake, "A direct method for trajectory optimization of rigid bodies through contact," *The Intl. Journal of Robotics Research*, vol. 33, no. 1, pp. 69–81, 2014.
[14] H. Dai, A. Valenzuela, and R. Tedrake, "Whole-body motion planning with centroidal dynamics and full kinematics," in *IEEE Intl. Conference on Humanoid Robots*, 2014, pp. 295–302.
[15] D. E. Orin, A. Goswami, and S.-H. Lee, "Centroidal dynamics of a humanoid robot," *Autonomous Robots*, vol. 35, no. 2–3, pp. 161–176, 2013.
[16] A. Shapiro, E. Rimon, and S. Shoval, "A foothold selection algorithm for spider robot locomotion in planar tunnel environments," *Intl. Journal of Robotics Research*, vol. 24, no. 10, pp. 823–844, 2005.
[17] M. Kalakrishnan, J. Buchli, P. Pastor, and S. Schaal, "Learning locomotion over rough terrain using terrain templates," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2009, pp. 167–172.
[18] J. Carpentier and N. Mansard, "Multicontact locomotion of legged robots," *IEEE Transactions on Robotics*, vol. 34, no. 6, pp. 1441–1460, 2018.
[19] H. Audren, J. Vaillant, A. Kheddar, *et al.*, "Model preview control in multi-contact motion-application to a humanoid robot," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2014, pp. 4030–4035.
[20] K. Hauser, T. Bretl, J.-C. Latombe, *et al.*, "Motion planning for legged robots on varied terrain," *The Intl. Journal of Robotics Research*, vol. 27, no. 11–12, pp. 1325–1349, 2008.
[21] M. Brunner, B. Brüggemann, and D. Schulz, "Hierarchical rough terrain motion planning using an optimal sampling-based method," in *IEEE Intl. Conference on Robotics and Automation*, 2013, pp. 5539–5544.
[22] S. Karaman and E. Frazzoli, "Sampling-based algorithms for optimal motion planning," *The Intl. Journal of Robotics Research*, vol. 30, no. 7, pp. 846–894, 2011.
[23] P. Fernbach, S. Tonneau, A. Del Prete, and M. Taïx, "A kinodynamic steering-method for legged multi-contact locomotion," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2017, pp. 3701–3707.
[24] J. J. Kuffner and S. M. LaValle, "RRT-Connect: An efficient approach to single-query path planning," in *IEEE Intl. Conference on Robotics and Automation*, vol. 2, 2000, pp. 995–1001.
[25] S. M. LaValle and J. J. Kuffner Jr, "Randomized kinodynamic planning," *The Intl. Journal of Robotics Research*, vol. 20, no. 5, pp. 378–400, 2001.
[26] Y. Zhao and L. Sentis, "A three dimensional foot placement planner for locomotion in very rough terrains," in *IEEE Intl. Conference on Humanoid Robots*, 2012, pp. 726–733.
[27] S. Klemm, J. Oberländer, A. Hermann, *et al.*, "RRT*-Connect: Faster, asymptotically optimal motion planning," in *IEEE Intl. Conference on Robotics and Biomimetics*, 2015, pp. 1670–1677.
[28] K. Hauser and V. Ng-Thow-Hing, "Fast smoothing of manipulator trajectories using optimal bounded-acceleration shortcuts," in *IEEE Intl. Conference on Robotics and Automation*, 2010, pp. 2493–2498.
[29] T. Kunz and M. Stilman, "Probabilistically complete kinodynamic planning for robot manipulators with acceleration limits," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2014, pp. 3713–3719.
[30] S. Karaman and E. Frazzoli, "Optimal kinodynamic motion planning using incremental sampling-based methods," in *IEEE Conference on Decision and Control*, 2010, pp. 7681–7687.
[31] D. Ferguson and A. Stentz, "Anytime RRTs," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2006, pp. 5369–5375.
[32] G. Bledt, M. J. Powell, B. Katz, *et al.*, "MIT Cheetah 3: Design and control of a robust, dynamic quadruped robot," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2018, pp. 2245–2252.
[33] A. Hereid and A. D. Ames, "FROST: Fast robot optimization and simulation toolkit," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2017, pp. 719–726.
[34] M. Geisert, T. Yates, A. Orgen, *et al.*, "Contact planning for the ANYmal quadruped robot using an acyclic reachability-based planner," in *Annual Conference Towards Autonomous Robotic Systems*. Springer, 2019, pp. 275–287.
[35] M. Hutter, C. Gehring, D. Jud, *et al.*, "ANYmal-A highly mobile and dynamic quadrupedal robot," in *IEEE/RSJ Intl. Conference on Intelligent Robots and Systems*, 2016, pp. 38–44.