




Regular Article

Quadrupedal Walking over Complex Terrain with a Quasi-Direct Drive Actuated Robot

Robert Griffin¹, Stephen McCrory¹, Sylvain Bertrand¹, Duncan Calvert¹, Inho Lee¹, Peter Neuhaus¹, Doug Stephen¹, Jay Jasper², Sisir Karumanchi², Ara Kourchians², Blair Emanuel², Emma Holmes³, Rachel Hegeman³, Jason Pusey⁴ and Jerry Pratt¹

¹IHMC Robotics, Pensacola, FL 32502

²NASA Jet Propulsion Lab, Pasadena, CA 91109

³Applied Physics Laboratory, Laurel, MD 20723

⁴Army Research Laboratory, Aberdeen, MD 21005

Abstract: In this paper, we present our approach to achieve autonomous walking over complex terrain on the quadrupedal robot, LLAMA. LLAMA is a prototype robot designed by NASA Jet Propulsion Lab as part of the Army Research Laboratory's Robotics Collaborative Technology Alliance. One of the major objectives of this robot is to be capable of traversing complex terrain autonomously to enable operating alongside a human squadron. This goal requires the robot to be capable of identifying practical footholds according to the environment, which may be sparse, and using these for walking. To accomplish this end, we first introduce two new contact planners. One is based on either desired body path plans; the other an A* graph-search based planner that find contacts over rough terrain given the environmental constraints. We then plan a dynamic trajectory using a custom Divergent Component of Motion planner, which is tracked using a whole-body inverse-dynamics control framework. We also introduce new methods for maintaining balance by adjusting step position and timing. We additionally discuss our approach for contact detection without the use of force sensors. We highlight the results in several experiments, both in our laboratory and at the RCTA Capstone Event at the Camp Lejeune Marine Corps base. We conclude with a discussion of these results, specific implementation problems, and lessons learned when developing such a control architecture on a quadruped.

Keywords: legged robots, control, motion planning

1. Introduction

Legged robots have the potential to assist humans in a wide variety of domains, from helping emergency first responders to accompanying and aiding warfighters. An inherent requirement to

Received: 29 September 2020; revised: 20 April 2021; accepted: 07 May 2021; published: 28 March 2022.

Correspondence: Robert Griffin, IHMC Robotics, Pensacola, FL 32502, Email: rgriffin@ihmc.org

This is an open-access article distributed under the terms of the Creative Commons Attribution License, which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited.

Copyright © 2022 Griffin, McCrory, Bertrand, Calvert, Lee, Neuhaus, Stephen, Jasper, Karumanchi, Kourchians, Emanuel, Holmes, Hegeman, Pusey and Pratt

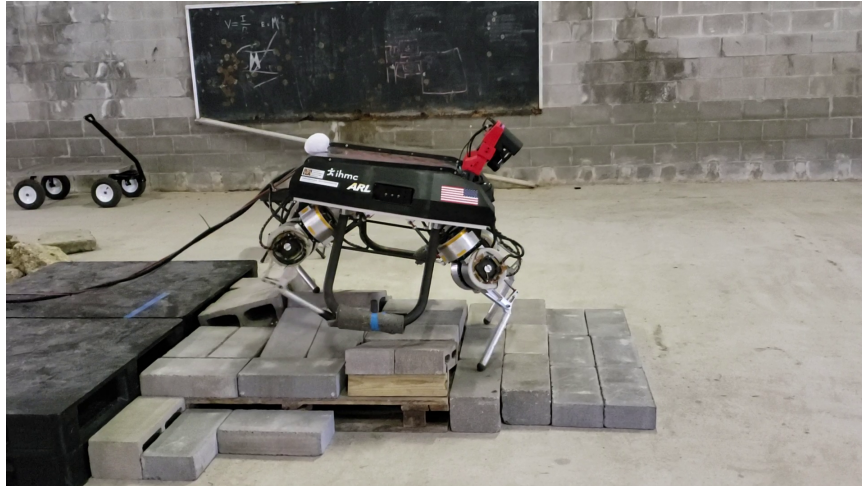


Figure 1. Image of the LLAMA v1 robot going down complex terrain inside a gymnasium at the Camp Lejeune Marine Corps base during the RCTA Capstone Event.

achieve this potential is the ability to navigate through complex, unstructured environments. To this end, the LLAMA quadrupedal robot shown in Figure 1 was designed and built as a large-scale research platform to explore algorithms for traversing these terrains.

The Army Research Laboratory’s Robotics Collaborative Technology Alliance (RCTA) was a 10 year program aimed at advancing robotics technology to perform at operational tempo with human squadrons in the field. Part of the last two years of this program focused on the development of the LLAMA quadrupedal robot. LLAMA was designed as a large-scale legged system to enable research into the mobility requirements for robots to perform at operational tempo in complex environments. Researchers have steadily improved the performance of quadrupedal robots, with many platforms capable of dynamic locomotion over compliant, uneven ground. As terrain increases in complexity, however, different mobility strategies are required to accommodate the sparsity of available contacts. Figure 2, for example, illustrates terrain offering limited footholds and few feasible paths for a legged robot.

In this work, we present our approach for enabling the LLAMA robot to negotiate complex environments and insights gained during development. The robot accomplishes these tasks using the system architecture shown in Figure 3. We leveraged existing software and control algorithms from our work on humanoid robots, while also developing a number of unique innovations specific to quadrupeds, including multiple methods for contact selection, a Divergent Component of Motion-based dynamic planning approach, and a unique methods for computing step position and timing adjustment. This system first models the environment using planar regions formed from LIDAR point clouds, which we discuss in Section 3, then determines appropriate contacts in the world to achieve a goal provided by an operator or some other higher level source, presented in Section 4, then executes these footholds using a custom momentum-based whole-body inverse dynamics controller (Section 5). We then demonstrate this system in Section 6 with a series of experiments conducted both in our laboratory and at the RCTA Capstone Event at Camp Lejeune. In addition to the overall novel approach and the individual enabling innovations, we believe our primary contribution in this work to be a variety of insights gained during the implementation, validation, and testing of our locomotion architecture. We developed the manuscript around discussion of these insights, which are found throughout the paper and in-depth in Section 7. We believe that these lessons learned are critical to the successful development of useful legged robotics that can achieve operational tempo in the field. We focus on this discussion while leaving a thorough comparison to other work on quadrupedal locomotion for future work.



Figure 2. Example challenging terrain environment at the Camp Lejeune Marine Corps base. This type of terrain requires a large-form robot to navigate and advanced planning algorithms to use the sparse available footholds.

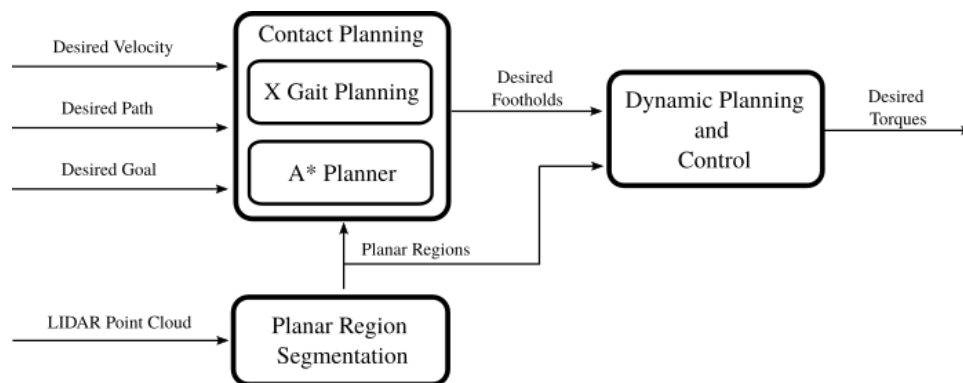


Figure 3. System architecture used in the presented framework. This system allows inputting multiple sources from an operator as a command to the robot, which then autonomously models the environment, plans footholds, and stably executes them, even in complicated terrains.

2. Background

The approach we use in this paper relies heavily on past work in legged robotic systems, both bipedal and quadrupedal. Here, we present a variety of research that inspired our methodology, as well as highlight significant accomplishments in the field. It is worth noting that significant progress has been made during the last few years in the field of quadruped robotics. We believe this research can be roughly separated into two categories on opposite ends of the spectrum: highly dynamic locomotion, which is typically blind, and gaits following a planned motion that accounts for terrain variations, which is typically slower.

One of the earliest and most famous examples of blind dynamic locomotion is the BigDog quadruped from Boston Dynamics (Raibert et al., 2008). More recently, this style of robust, blind walking has been shown very successfully on the Mini Cheetah robot from MIT, which relies on a convex model predictive controller to maintain balance (Katz et al., 2019; Kim et al., 2019). Although there has been good effort towards integrating knowledge about the robot’s surroundings, such as terrain variations, into these traditionally blind walking controllers, current solutions are still far from allowing the use of such a control framework for field operations, where the robot is expected to handle, in a smooth and predictable way, rough terrain with very sparse footholds.

At the other end of the spectrum, there have been a number of efforts aimed at developing and assembling tools that enable identifying features in the environment, for instance, steppable regions and impassable obstacles, planning according to these features and the robot capabilities, and controlling the robot to follow a plan. The DARPA Learning Locomotion program laid much of the early foundation for quadrupedal motion planning. This program allowed researchers to ignore problems related to perception and state estimation, resulting in advances focused on planning and control for quadrupedal locomotion in rough terrain (Rebula et al., 2007; Pongas et al., 2007; Kolter et al., 2008; Kalakrishnan et al., 2011; Zucker et al., 2011; Shkolnik et al., 2011; Vernaza et al., 2009). What followed the Learning Locomotion program were improvements to planning and control algorithms to utilize data collected solely from on-board sensors (Kolter et al., 2009; Havoutis et al., 2013; Bazeille et al., 2014), thus removing the reliance on external sensing to traverse rough terrain.

In the wake of this DARPA program, work in the field has been investigating methods for overall improvements to robot capabilities. This effort has yielded a variety of whole-body control frameworks. One such method uses virtual model control with inverse dynamics for the trunk (Barasuol et al., 2013; Winkler et al., 2014), which provides physical compliance and is relatively lightweight from a computational perspective. Another common control framework accounts for the full rigid-body-dynamics of the robot (Winkler et al., 2015; Bellicoso et al., 2016), which tends to better capture the robot dynamics, but is more computationally expensive. Lastly, two-stage control frameworks have also been presented with low-rate Model Predictive Control and a high-rate whole-body controller (Villarreal et al., 2019), extending the robustness of a simple whole-body control framework by accounting for a longer time horizon.

A number of approaches for planning contacts have also been developed, which techniques can be largely separated into two categories: coupled planners (Aceituno-Cabezas et al., 2017b; Aceituno-Cabezas et al., 2017a; Mastalli et al., 2020), which formulate the body path and contact sequence as a single planning problem, and decoupled planners (Geisert et al., 2019; Kolter et al., 2008; Mastalli et al., 2015; Vernaza et al., 2009; Kalakrishnan et al., 2009), which formulate body path and contact sequence as two separate, consecutive problems. The main challenge faced by coupled planners is in maintaining problem tractability and computational efficiency such that they can be used to re-plan footholds continuously. While the decoupled method represents a smaller problem by limiting the search space, it also makes it more challenging to reliably explore the space of feasible motions in the first sub-problem, often leading to a more conservative formulation. Additionally, there have been works that only plan a single step ahead, such as Jenelten et al., 2020, or simply setting step height of the “flat ground” steps to match that of a height map (Kim et al., 2020).

3. Environmental Representation

The first major challenge to developing a framework that can navigate over rough terrain is identifying an approach for modeling the environment that facilitates contact planning and control. There are several data structures that can be used to represent the environment with the goal to enable the robot to plan adequately and traverse rough terrain. One popular data structure for quadrupeds is the height map. Height maps are representative of most terrains, i.e. terrain where there is no overlap of multiple levels, while being computationally efficient at integrating new sensor

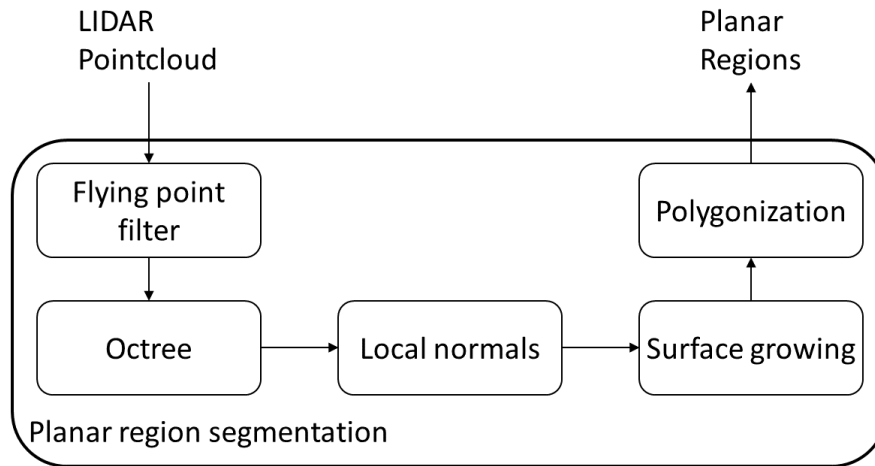


Figure 4. Point cloud processing steps overview: 1. a sensor specific filter is used to remove outliers in the data commonly called “flying points”, 2. the filtered point cloud is inserted in an octree used to store data over time, 3. the local normal for each occupied octree node is estimated, 4. octree nodes are segmented by growing planes of similar neighbors, 5. planar regions are obtained from the segmented nodes and the hull for each planar region is estimated.

measurements. A common approach is to extend a height map to contain a cost or reward per location that a footstep planner can use to choose appropriately desired footholds (Mastalli et al., 2020; Gutmann et al., 2008).

However, as the robot navigates through the world, the amount of data contained in the height map may increase rapidly as more of the environment is explored. One option to reduce the data structure size is to segment the perceived environment into features. To avoid reprocessing, it is also advantageous for these features to be a suitable representation for a footstep planner to use, while also being general enough to capture most of the environment and simpler than the original data structure.

To this end, a planar region, i.e. a 3D plane bounded by a concave hull, can be used. Unlike a height map, a list of planar regions tends to be much smaller, especially when the perceived area is large, and contains sufficient information for planning desired footholds. In this work, we use the same framework to process point cloud from depth sensors into planar regions as in Bertrand et al., 2020 that we previously used for planning footsteps on humanoid robots (Griffin et al., 2019), which we call the Robot Environmental Awareness (REA) module.

Figure 4 shows an overview of the environmental representation pipeline, which is described in depth in a separate work (Bertrand et al., 2020). The point cloud data is obtained from the LIDAR sensor mounted on LLAMA, which, in this case, is a Hokuyo UTM-30LX-EW attached to a Carnegie Robotics MultiSense. This point cloud is then stored in an octree (Hornung et al., 2013), after which we evaluate the local normal for each occupied node. The octree nodes are segmented by growing planes composed of neighbors with similar normals to a common 3D plane. Planar regions are obtained from the segmented nodes, on which we use the alpha-shape algorithm (Edelsbrunner and Mücke, 1994) to estimate the concave hull of each planar region. Finally, because concave hulls are difficult to perform geometric operations on, each concave hull is approximated by a set of convex polygons (Lien and Amato, 2006), which are used by the footstep planner. Figure 5 shows the output of each individual step of the process.

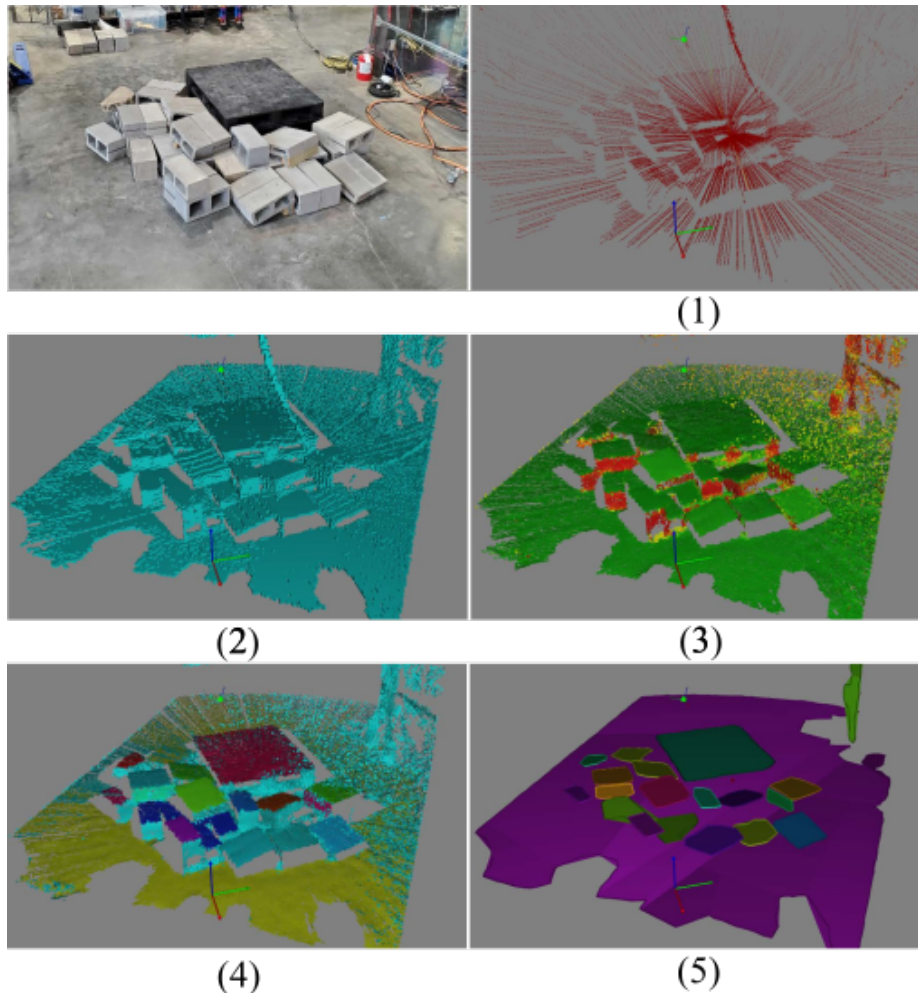


Figure 5. Example of segmentation into planar regions. The scene (top left, unlabeled) is scanned with LIDAR giving a point cloud (1). The point cloud is inserted into an octree (2). Then, the local normal for each octree node (3, coloring is based on the angle between the local normal and the vertical axis in world). The octree nodes are segmented by growing planes composed of neighbors with similar normals and close to a common 3D plane (4, coloring is used to highlight nodes that belong to the same plane). Finally, we polygonize the planes to obtain planar regions delimited by a concave hull which is approximated by a finite set of convex polygons (5, color code: the hue is used to better recognize planar regions, the brightness is used to visualize convex polygons approximating a planar region).

4. Step Generation

The primary objective of our work is to achieve locomotion in rough terrain with potentially sparse footholds. This means that our control methodology is designed to try and achieve a specific set of desired foot positions. These positions are first planned offboard using one of two novel methods introduced by this work in this section. The planned steps are then sent to the robot for execution, as discussed in the next section.

To generate the planned foothold positions, we developed two distinct new architectures. The first attempts to achieve a body velocity commanded by the operator and streams footsteps to the robot. The second computes the necessary footholds to reach a specified robot end pose, requiring only this goal pose to be provided. Both approaches are built on the same method introduced here



Figure 6. Illustration of the different timings used to define the different gaits of the robot. The top gait is a trot, the middle is an amble, and the bottom is a pace.

to calculate a sequence of timed footsteps. This framework is characterized by a gait phase along with a number of timing parameters and is capable of producing amble, trot, and pace gaits, as well as a continuous family of gaits in between, the collection of which we call X-Gait.

4.1. Online Step Generation

We start by introducing the first of these two novel approaches, in which steps are continuously streamed to the robot based on an operator’s input. This input would typically take the form of commanded forward, lateral and angular velocities, but could also be a sequence of body-path waypoints.

The gait generator starts by scheduling step times based on a clock-cycle of period T , in which each quadrant steps once per cycle with swing duration t_{SW} . At each end of the robot, we enforce steps to be completely out of phase with respect to the opposite side and to have at least one foot in contact at all times, with $t_{DS} \geq 0$ being the duration of double support at each end. In practice, we encode the gait cycle in terms of swing and double support durations, giving a period of:

$$T = 2(t_{SW} + t_{DS}). \quad (1)$$

We then define a phase $\phi \in [0, 2\pi)$ to specify the relationship between the front and hind cycles. If a hind step on side s is started at time t_H , the subsequent front step on side s is started at

$$t_F = t_H + \frac{\phi}{2\pi}T. \quad (2)$$

The resulting timings are shown in Figure 6, which illustrates that a trot, walk and pace are generated at phases π , $\frac{\pi}{2}$ and 0, respectively. To plan foothold positions, we introduce a preferred stance configuration in the form of the X-Gait rectangle, as shown on the left in Figure 7. This rectangle is defined by a preferred stance length and width in body-centric frame. The robot’s initial pose is extrapolated along a commanded trajectory $\vec{r}(t) \in SE(2)$ to create a sequence of keyframes based on each step’s start and end times, as depicted on the right of Figure 7. For a step scheduled to start at t_i , the vertex of the box at $\vec{r}(t_i)$ is projected onto the environment to compute the touchdown position.

When blind walking, steps are projected onto a ground plane estimate based on each quadrant’s most recent position when in contact with the ground. Alternatively, steps could be projected on the planar regions perceived by the REA module presented in Section 3, as shown in Figure 9.

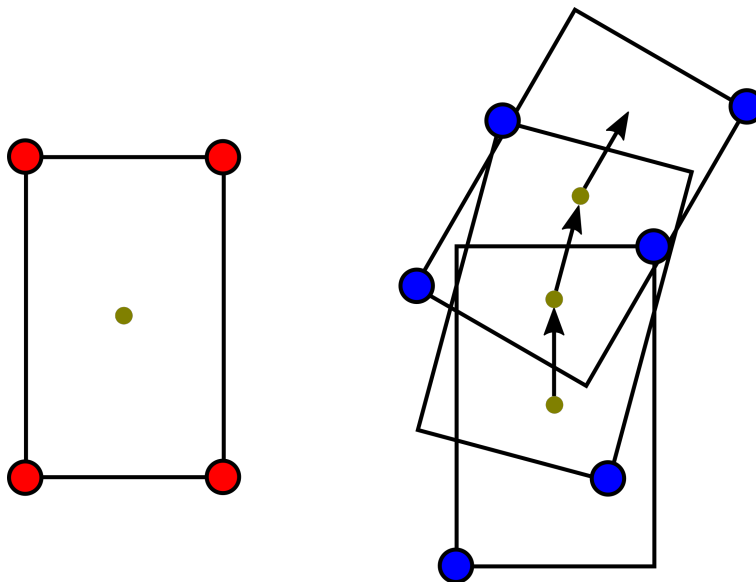


Figure 7. Left: Illustration of the X-Gait rectangle, which defines a preferred stance length and width. Right: Extruding the X-Gait rectangle according to the timing and desired velocities can be used to find the footsteps, shown in blue.

4.2. A* Planning Framework

The above framework in which steps are streamed to the robot works well for simple terrains but lacks the ability to plan over gaps, step ups, and other rough terrain. We adapted the proposed X-Gait to a novel A* search framework to address this. To perform a search, the operator provides a desired end pose and X-Gait phase ϕ and a timed step sequence is returned for review and execution.

To perform the graph search, we have a graph node represent a single xy footstep position and body orientation, as shown in Figure 8. Footsteps are restricted to be on an xy lattice with a spacing of 6cm, with a similar angular lattice used for body orientation. This affords the ability to cache node properties and prevent duplicated computation through the use of hash and equality functions (see Algorithm 1). The X-Gait rectangle is constructed relative to the footstep to describe the step’s nominal body pose. Although for the final sequence of steps the body pose differs depending on the placement of the other three quadrants, this allows for approximate checks of the step’s clearance from other quadrants and body collisions with the environment.

Algorithm 1 shows our method for performing the search. Nodes to be explored are placed in a priority queue which sorts in decreasing order of cost $c = g + h$, where g is the cost from start and h is the heuristic cost. On each iteration, the node at the top of the queue is expanded and, if accepted, added to a directed graph, which maintains optimal routes from the start for each node.

The `expand` function in Algorithm 1 starts by deciding which quadrant should follow the given node. The ordering of steps in X-Gait has only two modes, one when $\phi < \pi$ and another for $\phi > \pi$. We handle the simultaneous contact of $\phi = 0$ and $\phi = \pi$ by simply keeping the order of $\phi < \pi$ and $\phi > \pi$ (respectively) and scheduling zero delay in the final plan. When performing the search, the planner therefore determines the upcoming quadrant from the phase. To compute child step locations, a set of hand-tuned parameters defines a set of valid position and angular offsets from the X-Gait rectangle, shown in Figure 8 as orange circles. Additionally, a set of possible angular offsets is considered at each of these positions.

The `doProjection` function performs a simple vertical projection onto the planar regions detected from the REA module, as shown in Figure 9. If no projection is available, the node is marked as

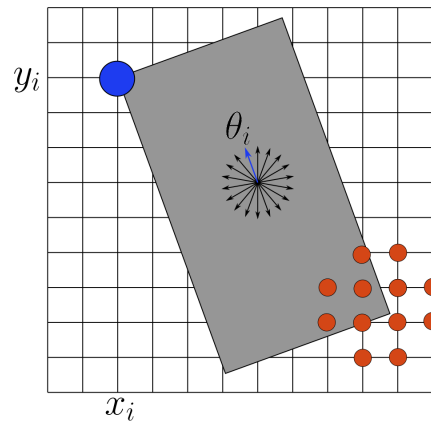


Figure 8. Graph node where the blue circle and blue arrow depict the node’s step position and body orientation, respectively. The orange circles show the candidate steps for the hind-right foot.

Algorithm 1. A* Search Algorithm

```

Function doAStarSearch(start, goal, planarRegions):
  queue.initialize(start)
  graph.initialize(start)
  while  $t < t_{max}$  and queue.size() > 0 do
    nodeToExplore = queue.pop()
    if nodeToExplore.equals(goal) then
      return nodeToExplore
    childNodes = expand(nodeToExplore)
    for (childNode : childNodes) do
      doProjection(childNode, planarRegions)
      if safeStep(nodeToExplore, childNode) then
        edgeCost = calculateEdgeCost(nodeToExplore, childNode)
        childNode.calculateHeuristics(goalNode)
        graph.addEdge(nodeToExplore, childNode, edgeCost)
        queue.push(childNode)
  return null

```

inaccessible. Steps within a threshold distance δ_{min} of a region’s edge are shifted inside to achieve this margin.

Projected steps are then checked for feasibility in the `safeStep` function. These checks include:

- Surface incline
- Step height and length
- Clearance from other quadrants
- Deviation from nominal X-Gait position
- Environment collisions.

Note that the nominal step expansion assumes flat ground. The step length check restricts lateral movement based on the step’s height. Similarly, the `calculateEdgeCost` function computes the following costs for each step:

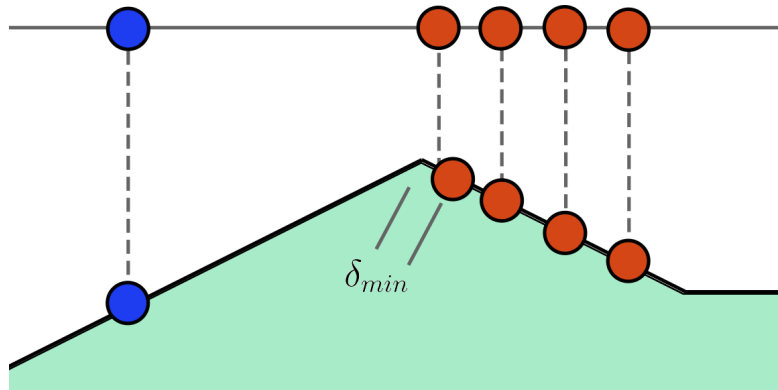


Figure 9. Illustration of the `doProjection` function in Algorithm 1. The candidate footsteps (orange) are projected vertically onto the planar regions coming from the REA module. Steps with a threshold distance δ_{min} of a region’s edge are shifted inward.

- Step height and length
- Constant cost per step
- Deviation from nominal X-Gait.

Finally, nodes are assigned a heuristic cost with the `calculateHeuristics` function. This was calculated in one of two ways. The first was a simple distance and yaw proximity to the goal, intended to create a turn-walk-turn behavior. Alternatively, the heuristic function could accept a list of waypoints, in which the cost would be the remaining distance along the path.

We found this search-based approach capable of navigating rough terrain in which the goal point is within a 5m radius of the robot. Above this threshold, performance was impaired by two factors. Firstly, the environment model at this distance was lower quality and occlusions became more prominent. Additionally, drift in the robot’s estimated state would eventually accumulate and result in significant tracking error. The details of the robot’s performance using both methods is discussed in Sections 6 and 7.

5. Control

Once the desired footholds have been calculated using the new approaches presented in Section 4, the robot must stably track the resulting motions to achieve these contact locations. We first use these footholds to plan dynamic trajectories using a novel Divergent Component of Motion (DCM) (Takenaka et al., 2009) dynamic planner presented here. The desired DCM is then tracked using a momentum-based controller, which computes a desired linear momentum rate of change. The necessary joint torques and accelerations to best achieve this net force are then computed using a whole-body inverse dynamics optimization. This control architecture is illustrated in Figure 10. Additional balance mechanisms including new step position and timing adjustment tools are also used to help the robot walk.

In the following section, we present each of these components in more detail, as well as provide observations we made during the development process.

5.1. Dynamic Planning and Control

In our framework, the center of mass of the robot is controlled by regulating the system’s linear momentum, which requires some reference value. For this, we compute a reference DCM trajectory from the provided contact sequence using a new approach presented here. Then, given the reference

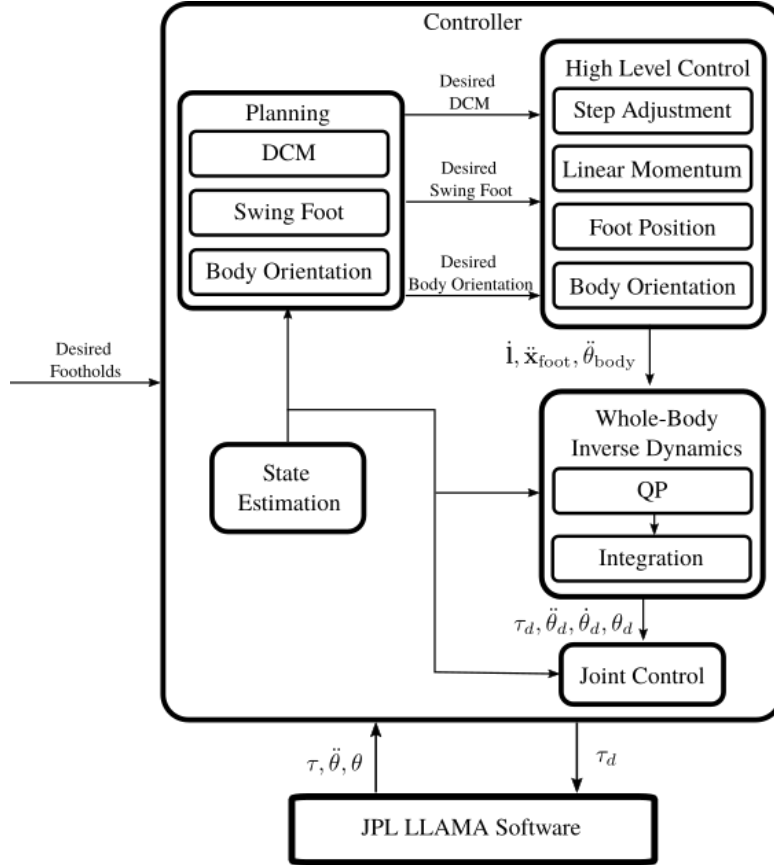


Figure 10. Proposed control architecture that computes desired torques to execute the provided footstep plan.

DCM position, we can compute the desired linear momentum rate of change and necessary step adjustment to maintain balance.

5.1.1. Divergent Component of Motion Planning

The first step in planning the DCM trajectory is to compute the reference center of pressure (CoP) sequence given the desired contacts. To do this, we present a novel approach for computing nominal centers of pressure for each of the individual contact phases given the contact configuration, as shown in Figure 11, and then planning the DCM trajectories in an approach similar to (Englsberger et al., 2014). Here, we compute a nominal loading of each foot using

$$w_f = \frac{0.5}{n_e}, \quad (3)$$

where w_f is the fraction of the total weight of the robot for the foot, and n_e is the number of feet in contact on that end. This results in the weight being evenly split between the front in the back, so that 50% of the weight is supported by the front and 50% is supported by the back, but with more desired weight on the side with more feet in contact. Once we have the weight distribution, we can compute the reference center of pressure by

$$\mathbf{r}_{\text{cop}} = \sum_i^N w_{f,i} \mathbf{r}_{f,i}, \quad (4)$$

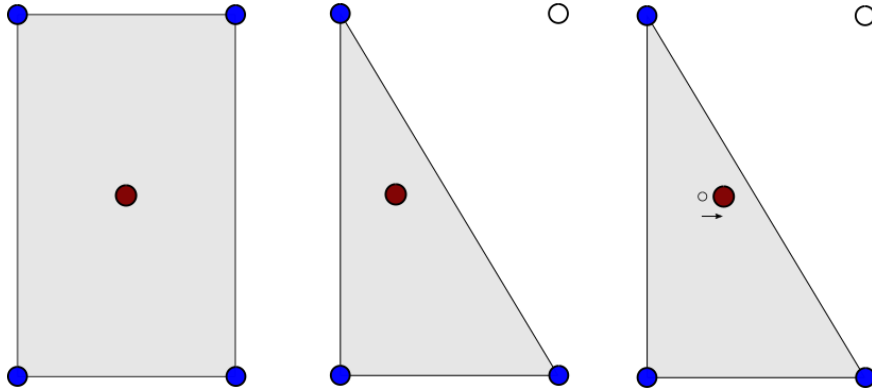


Figure 11. Illustration of the computation of the reference CoP, shown by the red circle. Each of the feet in contact (blue circles) are used to calculate the nominal CoP. This causes it to shift from side to side, as shown in the middle. We also introduce a shift factor that allows us to laterally shift the nominal CoP towards the center, based on the stance width, as shown on the right.

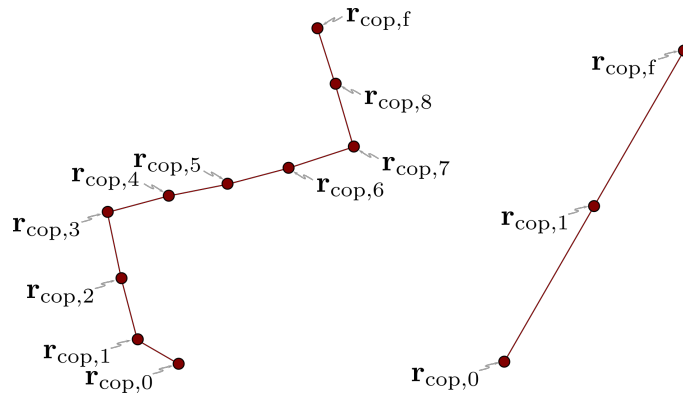


Figure 12. Illustration of the CoP waypoints computed using a dynamic amble (left) and trot (right) gait.

where \mathbf{r}_{cop} is the location of the center of pressure, and $\mathbf{r}_{f,i}$ is the location of the i^{th} foot, both expressed in world frame.

In simulation, this CoP plan tended to work very well. However, when implementing on the LLAMA hardware, it was noticed that, particularly for slower steps, it results considerable lateral CoM motion. To address this, some additional tunable fields were introduced, allowing us to adjust the reference plan. These include a lateral CoP offset based on the width of the stance, and a forward CoP offset based on the upcoming foot position, as shown in Figure 11. We then ensure that this modified CoP is a certain distance within the support polygon, if the polygon has interior area (which may not always be the case given the contact configuration, e.g. during a trot). For trot and amble gaits, this results in the CoP plans shown in Figure 12, with the number of CoP waypoints in an amble plan being quite large due to the number of contact changes.

Once the CoP trajectory has been calculated, we can then use an approach similar to that outlined by Engelsberger (Engelsberger et al., 2014), which we will summarize here. This approach is based on a Linear Inverted Pendulum (LIP) model, where the DCM dynamics are a linear combination of the CoM position and velocity,

$$\xi = \mathbf{x} + \frac{1}{\omega} \dot{\mathbf{x}}, \quad (5)$$

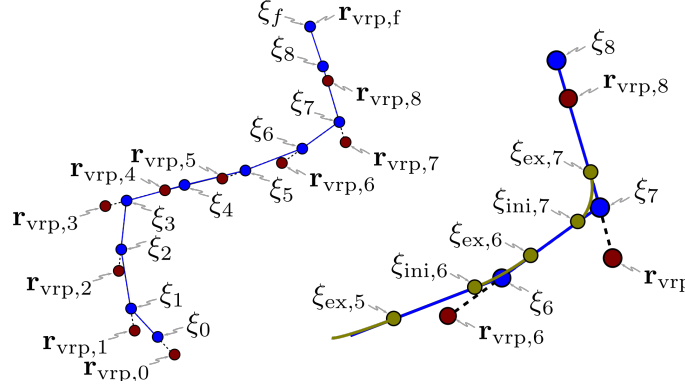


Figure 13. Left: Recursively computed DCM plan. This approach uses the continuous-time DCM dynamics assuming constant VRP positions. This results in a discontinuous VRP trajectory, where the VRP instantaneously shifts from one waypoint to the next. The VRP waypoints are the black circles, and the DCM positions at the beginning and end of each time segment are the blue circles. Right: Smoothed DCM plan. To achieve a continuous VRP trajectory, the DCM plan is smoothed at the contact shifting using splines in green.

where $\xi \in \mathbb{R}^3$ is the DCM position, $\mathbf{x} \in \mathbb{R}^3$ is the CoM position, and $\omega = \sqrt{\frac{g}{\Delta z}}$ is a time constant typically approximated by the natural frequency of the LIP, where the height is Δz . We can then define the DCM dynamics as

$$\dot{\xi} = \omega (\xi - \mathbf{r}_{\text{vrp}}), \quad (6)$$

where \mathbf{r}_{vrp} is the virtual repellent point (VRP). Since the DCM dynamics are first order and linear, we can compute a closed-form solution for the DCM position given a constant VRP location using

$$\xi(t) = e^{\omega t} (\xi_0 - \mathbf{r}_{\text{vrp}}) + \mathbf{r}_{\text{vrp}}, \quad (7)$$

where $\xi_0 = \xi(0)$. This implies that we can recursively compute the DCM trajectory by defining the final DCM location as fixed and computing the DCM trajectory for each of the m contact phases using the reverse-time DCM dynamics,

$$\xi(t_j) = e^{-\omega t_j} (\xi(T_j) - \mathbf{r}_{\text{vrp},j}) + \mathbf{r}_{\text{vrp},j}, \quad (8)$$

where t_j is the time in the j^{th} segment which has a duration T_j . This results in the DCM plan shown in the left of Figure 13.

One of the disadvantages of this constant VRP approach is that it results in a discontinuous VRP trajectory, as noted by Engelsberger (Engelsberger et al., 2014). When running the robot, it was found that this discontinuous change in the reference VRP position caused significant state estimator errors on contact switches. To address this, we introduce splines to smoothly transition between contact states, which run smooth around the DCM waypoint at ξ_i , splining between $\xi_{\text{ini},i}$ and $\xi_{\text{ex},i}$. This results in a more continuous DCM plan, shown in the right of Figure 13.

5.1.2. Momentum-Based Control

From the CoM and defined DCM dynamics, a desired linear momentum rate of change, $\dot{\mathbf{l}}$, can be computed given a VRP location as

$$\dot{\mathbf{l}} = m\omega^2 (\mathbf{x} - \mathbf{r}_{\text{vrp}}). \quad (9)$$

The next step, then, is to determine the desired VRP location to be used in Eq. 9. Given the reference VRP position from the DCM plan presented in Section 5.1.1, $\mathbf{r}_{\text{vrp},r}$, a reference linear momentum rate of change can be computed. However, even if this net reaction force were perfectly achieved (an unlikely event due to actuators not being perfect torque sources), things like model errors in the

CoM position would still result in tracking errors in open-loop execution. To compensate for this, we can define a simple proportional controller based on the DCM error for the x - y component,

$$\mathbf{r}_{\text{vrp},d,xy} = \mathbf{k}_p (\xi_{xy} - \xi_{r,xy}) + \mathbf{r}_{\text{vrp},r,xy}, \quad (10)$$

where ξ is the measured DCM position, ξ_r is the reference DCM position computed in Section 5.1.1, and $\mathbf{r}_{\text{vrp},d}$ is the desired VRP position with feedback. $\mathbf{r}_{\text{vrp},d,xy}$ can then be used with Eq. 9 to compute the desired linear momentum rate of change in x and y , $\dot{\mathbf{l}}_{d,xy}$. In this case, we are using the DCM setpoints and defined DCM model with the tuneable time constant as a way of shaping the closed-loop dynamics of the robot.

While the same approach can theoretically be used to compute the desired linear momentum rate of change in the z direction, $\dot{\mathbf{l}}_{d,z}$, we found that controlling the body height of the robot, rather than the CoM height, produced more favorable results. Controlling the body height then ignores the dynamic effects of the legs of the robot, which puts it more inline with the LIP model. We also believe these more favorable results are a product controlling the body height being more representative of the underlying objective, which is achieving a desired robot height. We specify the desired height in the support frame, so that it does not result in over-stretching one end of the robot, and control it using a simple PD controller, defined by

$$\ddot{x}_{z,d} = k_p (x_{z,d} - x_z) + k_d (\dot{x}_{z,d} - \dot{x}_z). \quad (11)$$

We can also introduce a tuning variable for the weight compensation of the vertical controller, α , which is used to compensate for an inexact weight and CoM height estimate. For example, if the robot model is heavier than the actual robot, the forces being exerted for “zero height acceleration” are too great, and would actually result in a upward acceleration. This tuning variable α allows the user to compensate for this type of uncertainty. Using this height acceleration, we can then compute the vertical momentum rate of change as

$$\dot{l}_z = m\omega^2 (x_z - r_{\text{vrp},d,z}) + m (\ddot{x}_{z,d} - (1 - \alpha)g). \quad (12)$$

However, because there is no closed-loop control being done on the DCM height, $r_{\text{vrp},d,z} \equiv x_z$, causing Eq. 12 to reduce to

$$\dot{l}_z = m (\ddot{x}_{z,d} - (1 - \alpha)g). \quad (13)$$

Combining Eqs. 9, 10, and 12, then, gives us the complete desired linear momentum rate of change to stabilize the robot,

$$\dot{\mathbf{l}} = m \begin{bmatrix} \omega^2 (\mathbf{x}_{xy} - \mathbf{k}_p (\xi_{xy} - \xi_{r,xy}) - \mathbf{r}_{\text{vrp},r,xy}) \\ k_p (x_{z,d} - x_z) + k_d (\dot{x}_{z,d} - \dot{x}_z) - (1 - \alpha)g \end{bmatrix}. \quad (14)$$

5.1.3. Step Adjustment

One of the main balance mechanisms for walking, particularly when walking more quickly, is step adjustment. There has been a tremendous amount of work performed for computing step adjustment, from classical Raibert based control laws to nonlinear system optimal controllers. In this work, we use a new strategy inspired by capture point techniques. The goal is to compute the amount of step adjustment needed to maintain balance from the current dynamic error. While one option is to simply adjust the foothold based on the dynamic error,

$$\Delta \mathbf{r}_{\text{foot}} = \xi - \xi_d, \quad (15)$$

this approach does not account for the dynamics continuing to diverge as the current step progresses. Instead, we can calculate an estimate of the error at the completion of the current step using the first-order DCM dynamics,

$$\xi_f = e^{\omega t_r} (\xi - \mathbf{r}_{\text{cmp}}) + \mathbf{r}_{\text{cmp}} \quad (16)$$

where t_r is the time remaining in the current state. Then, we can use this to compute the necessary step adjustment based on the predicted error,

$$\Delta \mathbf{r}_{\text{foot}} = e^{\omega t_r} (\xi - \mathbf{r}_{\text{cmp}}) + \mathbf{r}_{\text{cmp}} - e^{\omega t_r} (\xi_d - \mathbf{r}_{\text{cmp}}) + \mathbf{r}_{\text{cmp}} = e^{\omega t_r} (\xi - \xi_d). \quad (17)$$

In practice, we add a “safety factor” term $\alpha_s > 1$ so that we can manually increase the amount of step adjustment if desired,

$$\Delta \mathbf{r}_{\text{foot}} = \alpha_s e^{\omega t_r} (\xi - \xi_d). \quad (18)$$

In addition to step adjustment, we also introduced a swing speed up policy borrowed from our work with humanoids (Griffin et al., 2017). This approach predicts how much further along the dynamic plan in time the current robot state is by projecting the current capture point onto the capture point dynamics. It then uses this time difference to put the foot down more quickly to help stabilize the robot state when necessary.

5.2. Whole-body Control

The next step is to use some whole-body motion strategy to resolve the desired task-space, joint space, and momentum rate commands, some of which may be conflicting. There are several options for how to resolve these multiple motion tasks, including virtual model controllers (Pratt et al., 2001) and whole-body inverse dynamics controllers (Koolen et al., 2016). In this work, we use strategies similar to those employed by other robots that perform a whole-body inverse-dynamics based optimization (Mastalli et al., 2020).

The rigid body dynamics of the robot with n actuated degrees of freedom (DOF) can be defined as

$$\mathbf{H}(\mathbf{q}) \ddot{\mathbf{q}} + \mathbf{C}(\mathbf{q}, \dot{\mathbf{q}}) - \sum_c \mathbf{J}_c(\mathbf{q}) \mathbf{f}_c = \begin{bmatrix} \mathbf{0} \\ \boldsymbol{\tau} \end{bmatrix}, \quad (19)$$

where $\mathbf{q} = [\mathbf{q}_0^T \ \mathbf{q}_n^T]^T \in \mathbb{R}^{6+n}$ is the vector of joint positions, with $\mathbf{q}_0 \in \mathbb{R}^6$ representing the position of the 6 DOF floating base and $\mathbf{q}_n \in \mathbb{R}^n$ the current robot joint configuration, $\boldsymbol{\tau} \in \mathbb{R}^n$ the vector of actuated joint torques, $\mathbf{C}(\mathbf{q}, \dot{\mathbf{q}})$ the vector of centrifugal, Coriolis, and gravity torques, $\mathbf{H}(\mathbf{q})$ the joint-space inertia matrix, $\mathbf{J}_c(\mathbf{q})$ the Jacobian from the c^{th} contact point, and $\mathbf{f}_c \in \mathbb{R}^3$ the reaction force at the c^{th} contact point. From the rigid body dynamics, it can be seen that the actuated joint torques $\boldsymbol{\tau}$ are an affine function of the joint accelerations $\ddot{\mathbf{q}}$ and reaction forces \mathbf{f}_c , assuming known joint positions \mathbf{q} and velocities $\dot{\mathbf{q}}$. This means that we only need to solve for the necessary joint accelerations and reaction forces to determine the desired torques for the robot.

We can then define the centroidal dynamics of the robot. We know that the momentum rate of change, $\dot{\mathbf{h}}$, is equivalent to the sum of the forces acting on the system,

$$\dot{\mathbf{h}} = \begin{bmatrix} \dot{\mathbf{i}} \\ \dot{\mathbf{k}} \end{bmatrix} = \sum_c \mathbf{Q}_c \mathbf{f}_c + \mathbf{w}_g + \sum \mathbf{w}_e, \quad (20)$$

where \mathbf{Q}_c is the Jacobian that maps contact forces to wrenches acting about the CoM, \mathbf{w}_g is the gravity wrench, and \mathbf{w}_e is an external wrench acting on the system. \mathbf{h} is then related to the rigid body dynamics by the centroidal momentum matrix \mathbf{A} , which maps the joint velocities to the net momentum,

$$\mathbf{h} = \mathbf{A} \dot{\mathbf{q}} \implies \dot{\mathbf{h}} = \dot{\mathbf{A}} \dot{\mathbf{q}} + \mathbf{A} \ddot{\mathbf{q}}. \quad (21)$$

Given a desired spatial acceleration, $\dot{\mathbf{v}}_d$, its relationship to the joint tasks can be expressed using the task-space Jacobian, \mathbf{J}_t , which defines spatial velocities from joint velocities,

$$\mathbf{v} = \mathbf{J}_t \dot{\mathbf{q}} \implies \dot{\mathbf{v}}_d = \dot{\mathbf{J}}_t \dot{\mathbf{q}} + \mathbf{J}_t \ddot{\mathbf{q}}, \quad (22)$$

Then, to represent the friction constrained forces, we use a polyhedron inscription on the friction cone and define the force at each contact point as a function of the m basis vectors representing this polyhedron,

$$\mathbf{f}_c = \beta_c \rho_c, \quad (23)$$

where $\beta_c \in \mathbb{R}^{3 \times m}$ is the array of m unitary basis vectors that maps the m scalar $\rho_c \in \mathbb{R}^m$ basis vectors magnitudes to the c^{th} contact force. Note that the full set of scalar force values is noted as ρ .

We can then combine the set of desired spatial acceleration commands, momentum rate of change commands, and joint acceleration commands into a single objective vector of desired motion tasks, \mathbf{b} , a convective term Jacobian, $\dot{\mathbf{J}}$, and Jacobian, \mathbf{J} ,

$$\mathbf{b} = \begin{bmatrix} \dot{\mathbf{h}}_d \\ \ddot{\mathbf{q}}_d \\ \dot{\mathbf{v}}_{d,1} \\ \vdots \end{bmatrix}, \quad \dot{\mathbf{J}} = \begin{bmatrix} \dot{\mathbf{A}} \\ \mathbf{0} \\ \dot{\mathbf{J}}_{t,1} \\ \vdots \end{bmatrix}, \quad \mathbf{J} = \begin{bmatrix} \mathbf{A} \\ \mathbf{I} \\ \mathbf{J}_{t,1} \\ \vdots \end{bmatrix}. \quad (24)$$

However, based on the configuration of the robot, there may be a singularity in \mathbf{J} , such as when the the foot is in-line with the abduction-adduction joint. To handle this, we can define a set of PD controllers that attempt to drive the robot to some nominal configuration, \mathbf{q}_p , which provide desired accelerations by

$$\ddot{\mathbf{q}}_p = \mathbf{k}_{p,p} (\mathbf{q}_p - \mathbf{q}) + \mathbf{k}_{d,p} (\dot{\mathbf{q}}_p - \dot{\mathbf{q}}). \quad (25)$$

We can then project these desired accelerations into the nullspace of \mathbf{J} ,

$$\ddot{\mathbf{q}}_p^* = (\mathbf{I} - \mathbf{J}^\# \mathbf{J}) \ddot{\mathbf{q}}_p = \mathbf{J}_p \ddot{\mathbf{q}}_p, \quad (26)$$

where $(\cdot)^\#$ is the damped pseudo-inverse operator. This can then be appended as an additional task in \mathbf{b} and \mathbf{J} to yield $\hat{\mathbf{b}}$, $\hat{\mathbf{J}}$, and $\hat{\mathbf{J}}$. By projecting this joint-space task into the nullspace of \mathbf{J} , we can ensure that, if the nullspace of \mathbf{J} is empty, the matrix \mathbf{J}_p is empty, meaning $\ddot{\mathbf{q}}_p$ does not influence the solution of the task \mathbf{b} . However, if the nullspace is non-empty, we can bias the solution towards the direction defined by our ‘‘privileged configuration’’ \mathbf{q}_p .

By assembling these objectives, we can define the full quadratic program as

$$\begin{aligned} \min_{\hat{\mathbf{q}}, \rho} \quad & \left\| \hat{\mathbf{b}} - \hat{\mathbf{J}}\hat{\mathbf{q}} - \hat{\mathbf{J}}\hat{\mathbf{q}} \right\|_{\mathbf{Q}}^2 + \|\hat{\mathbf{q}}\|_{\lambda_{\hat{\mathbf{q}}}}^2 + \|\hat{\mathbf{q}} - \hat{\mathbf{q}}_{\text{prev}}\|_{\lambda_{\hat{\mathbf{q}}_{\text{prev}}}}^2 + \|\rho\|_{\lambda_{\rho}}^2 + \|\rho - \rho_{\text{prev}}\|_{\lambda_{\rho_{\text{prev}}}}^2 \\ \text{subject to} \quad & \mathbf{A}\dot{\mathbf{q}} + \mathbf{A}\ddot{\mathbf{q}} = \mathbf{Q}\beta\rho + \mathbf{w}_g + \sum \mathbf{w}_e, \\ & \rho_{\min} \leq \rho \leq \rho_{\max}, \\ & \ddot{\mathbf{q}}_{\min} \leq \ddot{\mathbf{q}} \leq \ddot{\mathbf{q}}_{\max}, \end{aligned} \quad (27)$$

which also includes terms for regularizing the resulting acceleration and reaction forces. Constraints to this equation include unilateral contacts, dynamics, and acceleration and force limits.

Given the solution to this optimization, we can then integrate up the desired accelerations $\ddot{\mathbf{q}}^*$ to compute desired velocities $\dot{\mathbf{q}}$ and positions \mathbf{q} . If we just directly integrate the acceleration, however, we can get diverging set points. This is intuitive - if there is poor tracking, there will always be a relatively large desired joint acceleration. However, the joints themselves will not achieve this desired acceleration, meaning that the integrated desired positions and velocities will diverge. To address this, we can introduce a leak rate α_{rate} and α_{pos} to the integration step,

$$\dot{\mathbf{q}}_{d,k+1} = \alpha_{\text{rate}} \dot{\mathbf{q}}_{d,k} + \Delta t \ddot{\mathbf{q}}_{k+1}^*, \quad (28)$$

$$\mathbf{q}_{d,k+1} = \alpha_{\text{pos}} \mathbf{q}_{d,k} + (1 - \alpha) \mathbf{q}_{k+1} + \Delta t \dot{\mathbf{q}}_{d,k+1}, \quad (29)$$

where the integrated desired velocity $\dot{\mathbf{q}}_{d,k+1}$ leaks towards zero velocity and the integrated desired position $\mathbf{q}_{d,k+1}$ leaks towards the current joint positions \mathbf{q}_{k+1} , with $\ddot{\mathbf{q}}_{k+1}^*$ is the current optimal joint acceleration solution of the optimization problem, Eq. 27.

When we initially implemented our whole-body controller on LLAMA, we began by using a Virtual Model Controller (VMC), a common approach utilized in the quadruped community (Ajallooeian et al., 2013; Winkler et al., 2014). For a quadruped, VMC is a fairly accurate approach due to the low link inertias of the legs, and is also relatively simple to implement. However, it does not typically allow for the ability to compute desired joint accelerations, velocities, and positions. We encountered difficulty with this approach due to the relatively high joint stiction found in the actuators on the robot, meaning we could not treat them as pure torque sources, resulting in very poor tracking and a break down of the assumptions of VMC. This required us to shift to a technique that easily allows computation of the desired joint positions and velocities. We can then augment the feedforward torques with closed loop position and velocity feedback. This informed our decision to switch to a whole-body inverse dynamics approach, as we already had significant experience and development in this area from our existing work.

5.3. Contact Detection

Because our control architecture attempts to decouple making and breaking contact from the gait clock, considerable effort was spent on developing new and incorporating existing contact detection methods. This is challenging, as there are no force sensors in the robot’s motors, and no contact sensors on the feet. To address this, we developed a set of virtual contact sensors in software. The first uses the canonical Jacobian transpose method, as presented in (Craig et al., 1989), to compute the virtual contact force at the foot from the motor currents in the leg. Then, if the virtual vertical force at the foot is above a certain threshold, the foot is said to be in contact.

The high joint speeds required in the leg, however, can result in high torques and correspondingly high virtual contact forces during normal swing. Additionally, because the swing trajectory occasionally results in the swing foot going through a singularity in the leg workspace on LLAMA, and the joints have a fairly high amount of friction, the virtual contact forces can be inaccurate. To address this, we also added an additional kinematics-based approach for calculating contact, which we have not seen used elsewhere. Here, we examine whether both the total spatial velocity magnitude and the vertical spatial velocity of the swing foot is below a certain threshold, at which point it is said to be in contact. By requiring both this velocity based foot switch and the force based foot switch to be true, it filters many of the false contacts where contact is reported early in the swing.

6. Experiments

We conducted an extensive number of experiments with the robot during the development of this control architecture. This includes a number of tests in our open-source software architecture, found at <https://github.com/ihmrobotics>, as well as experiments both in our laboratory and at the Camp Lejeune Marine Corps base. Here, we highlight a small subset of these hardware experiments, aimed at presenting the capabilities of this approach. Footage showing excerpts of tests performed at the IHMC Robotics lab can be found at <https://youtu.be/Pi4fH-ic1XQ>.

In our first experiment, we set up a short set of stepping stones in our lab. The gap between cinder blocks was uneven, but approximately 20cm between each of the pairs of blocks, with the difference in the height of the blocks and platforms being less than 1cm. The goal of this experiment was to validate the feasibility of extracting planar regions from an environment using LIDAR, planning footholds using these regions, and then executing these footholds with the robot. The results, shown in Figure 14, show the robot successfully traversing the stepping stones. In this experiment, no step adjustment was allowed in order to prevent the robot from accidentally stepping off the blocks. While step adjustment can be constrained to be within the planar region the step was planned over,

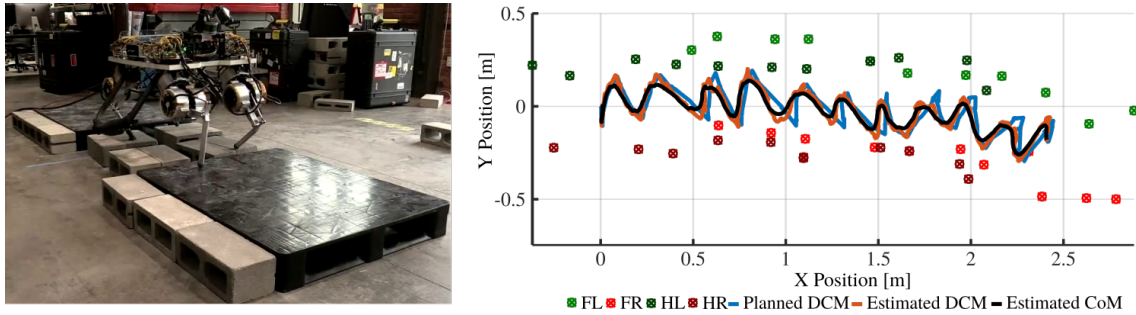


Figure 14. Planned (o) and adjusted (x) step locations of LLAMA when walking over stepping stones in our laboratory. The DCM plan, estimate, and CoM estimate are also shown. There is no step adjustment allowed during this experiment. The CoM closely follows the DCM, which stays well within the hull of all four feet throughout the experiment. Occasionally, the DCM does fall outside the support polygon when stepping, illustrating that this gait is, in fact, dynamic.

drift in the state estimate of the robot while planning results in the step area constraints improperly captured as the robot gets closer to the end of the plan. The average walking speed over these blocks was fairly slow at only 12.5 cm/s using an amble gait with a 0.5 s swing duration. The step length used by the robot was typically between 18 and 25 cm. It was noticed that the foot’s workspace is relatively limited, requiring the robot to take many shorter steps to position itself to take the longer step to cross blocks, ultimately limiting the forward velocity. This limited workspace is discussed more in Section 7.

The next three experiments were captured during the RCTA Capstone Event help at Camp Lejeune, a U.S. Marine Corps base in North Carolina, during June 2019. During this event, we aimed to further progress our planning and controls capabilities on LLAMA, and demonstrate their feasibility outdoors.

The first of these three experiments involved the robot blindly trotting up an outdoor incline at the base that was approximately 8 degrees. The incline angle was estimated by the robot from a short history of the foot positions, using a least squares fit to estimate a global ground plane. A desired velocity command is then provided to the X-Gait planner discussed in Section 4 from a joystick, which then plans in this estimated ground plane. In this experiment, the desired velocity was approximately 25 cm/s, using a swing duration of 0.33s and a double support duration of 0.1s, as defined in Section 4. The results of this experiment are shown in Figures 15 and 16.

During this experiment, step adjustment proved critical, which is to be expected when using a trotting gait as the center of pressure has very little ability to be modulated, and can be seen in Figure 15 as the difference between the (x) and (o) markers. Also worth noting is the slight lateral motions in the DCM plan. These were purposefully introduced, as we desired to push the CoM of the plan more towards the rear of the support polygon when stepping upwards to help avoid any potential shin collisions caused by the kinematic configurations of the legs. Lastly, the contact timings are shown in Figure 16, where it is apparent that the feet regularly were late on touching down. This is shown by the ‘x’ markers, where contact was expected but had not yet occurred. This late touchdown can be attributed a number of factors, including state estimation drift, poor ground plane estimation, and, most prevalent, poor foot tracking due to inaccurate force control.

For the next experiment, to evaluate the ability for the robot to navigate terrain with sparse footholds and changing heights, we set up a course at Camp Lejeune inside a gymnasium using cinder blocks, pallets, and pieces of broken concrete. In this case, the cinder blocks leading up to the pallets were 8 cm high, the pallets were 14 cms themselves, and the tilted blocks were 8 degrees and a minimum of 9 cm high. This course can be seen on the left of Figure 17, as well as in Figure 1. Then, using this environment, we collected planar regions, and directed the robot to walk to the top of the platform. In this experiment, we used the A* contact planner presented in Section 4. The

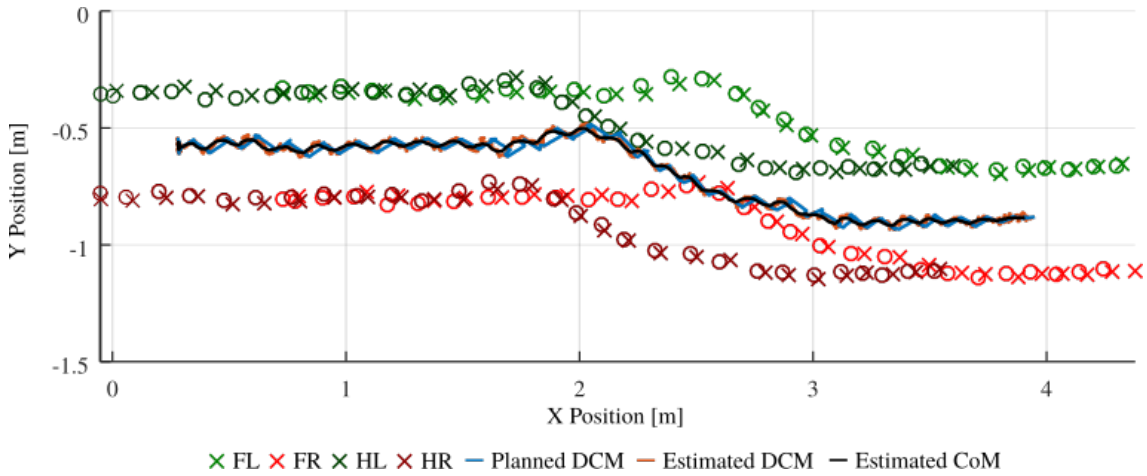


Figure 15. Planned (o) and adjusted (x) step locations of LLAMA when walking up a slope at Camp Lejeune. The DCM plan, estimate, and CoM estimate are also shown. As can be seen, the DCM plan has a slight lateral motion, which results from the robot shifting the CoP plan slightly towards the rear because of the incline. Some step adjustment occurs during execution.

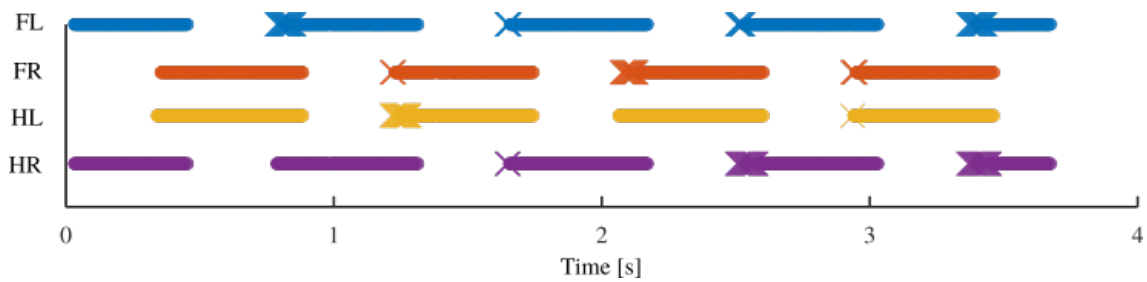


Figure 16. Sample of the gait timing used in the experiment shown in Figure 15. The 'x' marker indicates when touchdown was expected due to the gait schedule, but has not yet occurred. As can be seen, during certain portions of the gait, touchdown occurs fairly late.

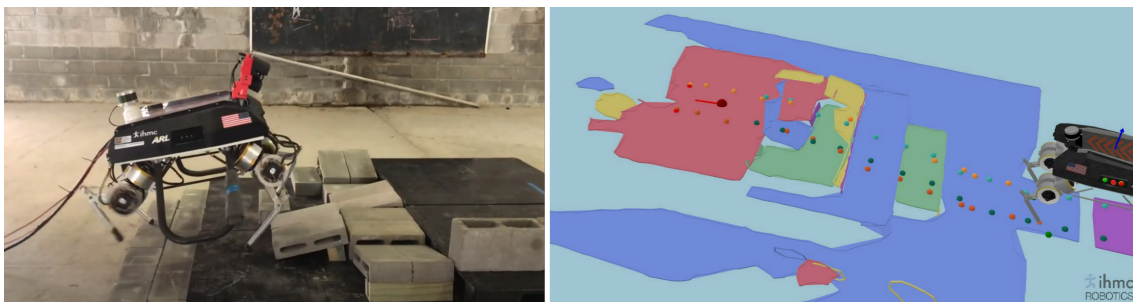


Figure 17. Left: Image of LLAMA walking up a set of cinder blocks and platforms inside a gymnasium at Camp Lejeune, North Carolina. Right: Planar region segmentation of the same environment, with planned footholds from the A* planner also shown.

resulting planar regions and foothold plan are shown on the right of Figure 17. This results in the step timing parameters shown in Figure 18.

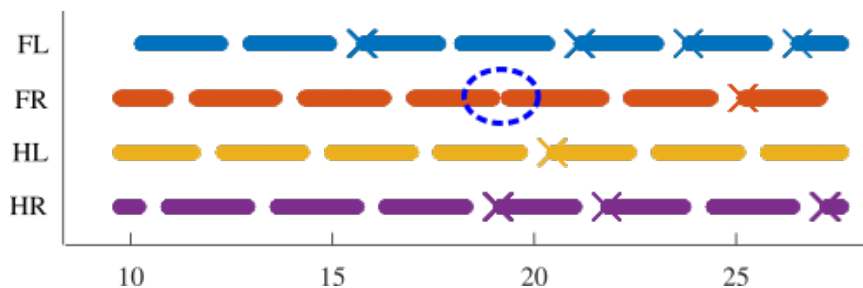


Figure 18. Gait timing used for the plan shown in Figure 17. Late touchdowns are shown by ‘x’, where the nominal gait phase would be in contact, but the foot had not yet hit the ground. The circled area highlights where the foot was set down more quickly to help regain balance, which is highlighted in Figure 21.

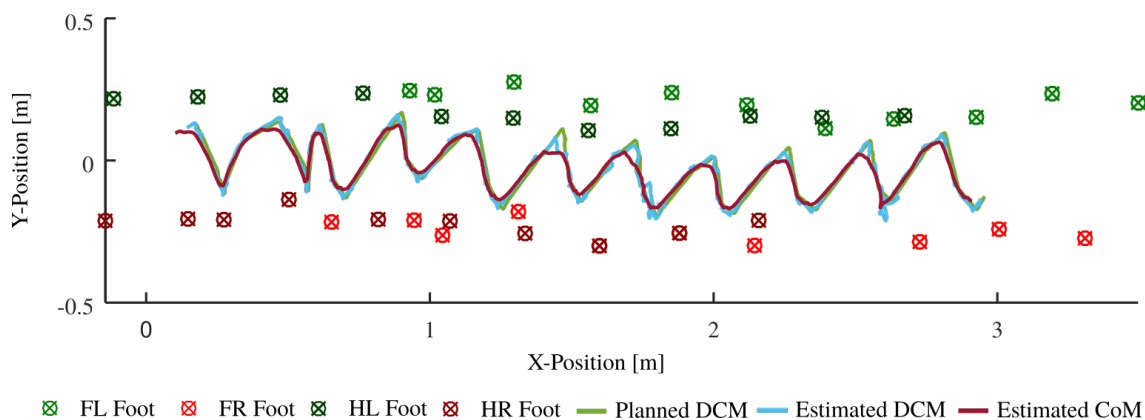


Figure 19. Step locations and DCM dynamics for the experiment shown in Figure 17. As is shown, the robot is not allowed to perform step adjustment during this experiment.

The robot was able to successfully navigate 2.91m to the top of the obstacle in 31.2s, as well as back down on the other side, as in Figure 1. This resulted in an overall average velocity of 9.3 cm/s, with an average step length of 26.5cm. Because we are not dynamically replanning the footstep positions online, to compensate for state estimation drift, we offset the planned footholds by any foothold height error measured at touchdown. This resulted in a net vertical drift of approximately 14 cm over the execution of the plan, which is fairly significant. Additionally, because of environmental modeling errors, the robot often encountered foothold height errors, resulting in very late touchdown, shown in Figure 18. The dynamic effects of these late touchdowns are also apparent in the DCM tracking, as shown in Figures 19 and 20. In both these figures, the reference DCM has several spikes of significant overshoot in the planned value, with an accompanying breakdown in DCM tracking. This is caused by the desired DCM location continuing to diverge from the base of support because of late touchdown.

These tracking errors do, however, allow highlighting the efficacy of the step timing adjustment strategy presented in (Griffin et al., 2017) and used here. Shown as the circled contact in Figure 18, one step touched down significantly early, as it was sped up. The dynamic configuration of the robot at this time is shown in Figure 21. Here, we can see that the measured DCM is significantly past the desired DCM, and towards the direction of the next step. By setting this foot down earlier, the CoM state is much more easily “captured”, allowing the robot to prevent falling.

In the last experiment conducted at Camp Lejeune, we aimed to validate this approach in an outdoor setting by requiring the robot to plan and traverse a sandy area covered in pieces of broken concrete, shown on the left in Figure 22. These pieces of concrete varied in height from 13cm to

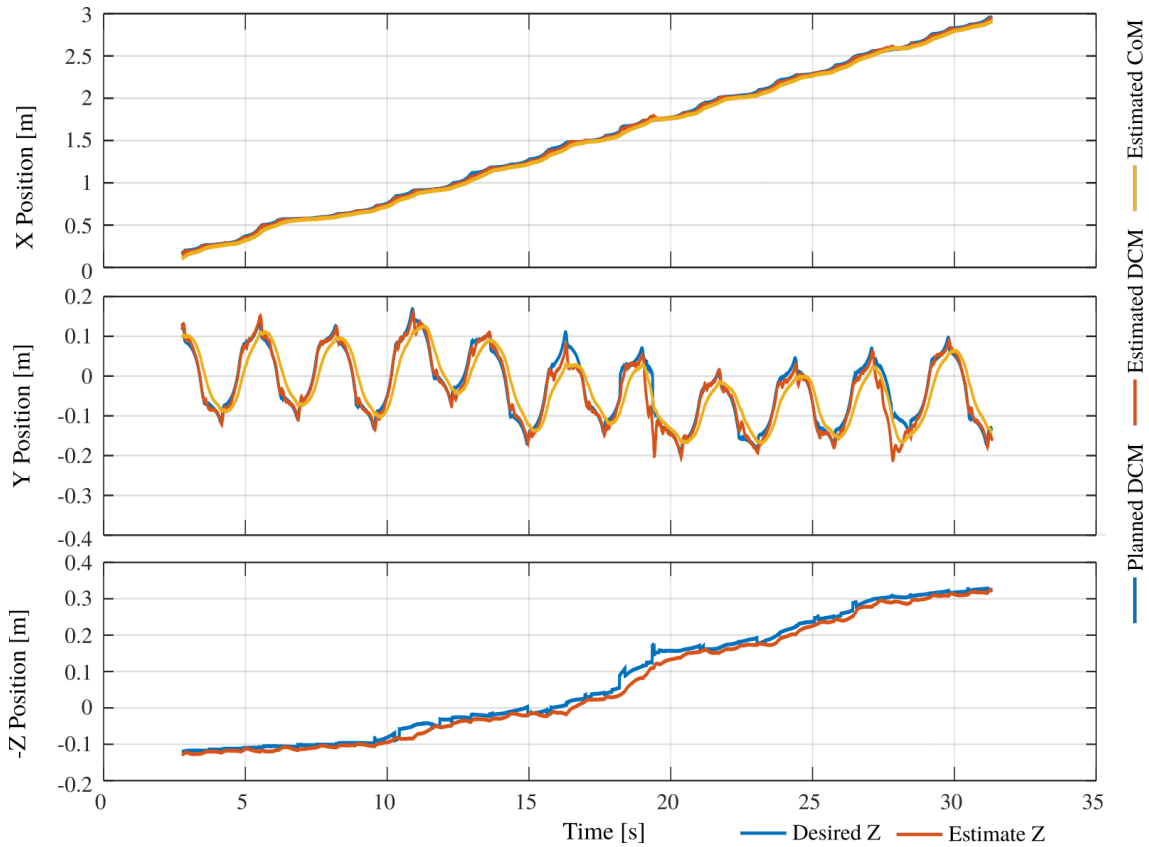


Figure 20. DCM, CoM, and height trajectories vs time. The top figure is the X position of the DCM and CoM, the middle figure is the Y position of the DCM and CoM, and the bottom figure is the CoM desired and estimated height in the Z direction as with respect to time. This data is from the experiment shown in Figure 17.

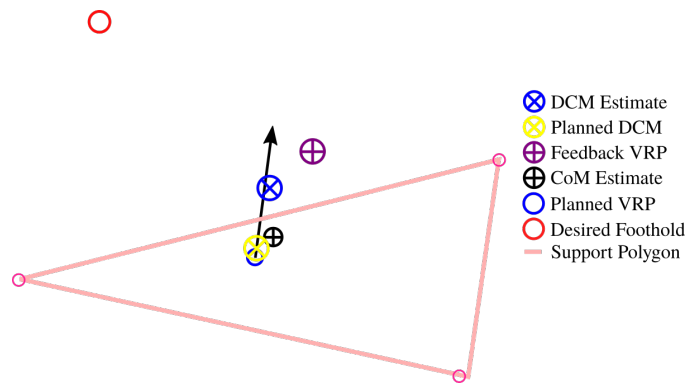


Figure 21. Dynamics that drive early touchdown that occurred in the circled area in Figure 18. As shown, the estimated DCM is considerably outside the support polygon in the direction of the desired DCM direction. This means that earlier touchdown will greatly benefit the stability of the robot.

18cm high, and included a 15cm high curve. The robot was told to walk to a goal placed 3.3 m in front of it, and did so over 35.7 s. The robot was able to successfully navigate the concrete, go over the curb, and onto the slope on the other side from the experiments in Figure 15, reaching a goal placed 3.3m in front of it. This took a total of 35.7s, meaning the robot averaged approximately



Figure 22. Image of the rock field set up at Camp Lejeune. The robot used LIDAR data to model the environment using the approach presented in Section 3 and planned the contacts using the A* planner in Section 4, shown on the right in the figure. The robot then successfully walked over the blocks, up the curb, and onto the inclined concrete on the other side.

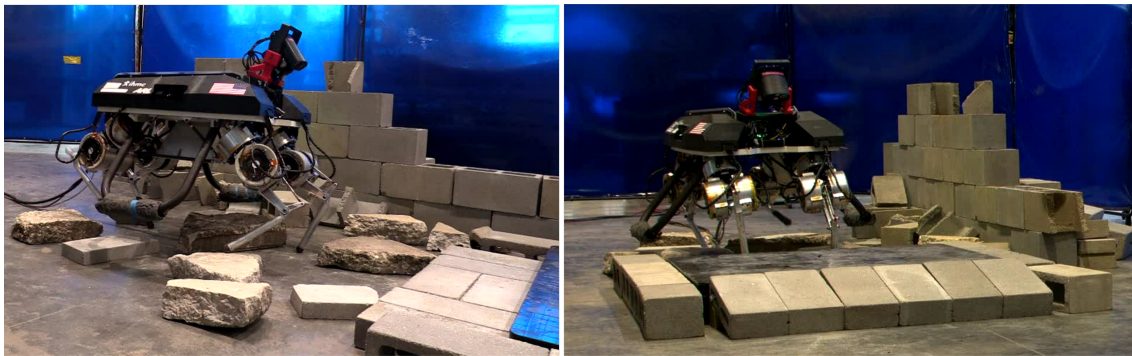


Figure 23. Image of the rock field set up in our laboratory. In this experiment, the robot did blind walking with an amble gait, recomputing its steps online to track a desired forward velocity.

9.5 cm/s. However, because of the chosen leg configuration, it encountered significant difficulty avoiding shin collisions, resulting in LLAMA often scraping its shins on the concrete, creating difficulty in achieving touchdown and imparting significant disturbances to the robot. Additionally, as in the previous experiment, the state estimate encountered considerable vertical drift during execution, accumulating approximately 29 cm in total over the 3.3 m course.

For our final experiment in this work we had the robot blindly walk using a desired forward velocity in our laboratory over similar broken concrete and cinder blocks as at Camp Lejeune, shown in Figure 23. These blocks varied in height from 8cm to 15cm. The same inclined blocks were used from previous experiments. For this, an amble gait was chosen. The robot was able to successfully traverse this terrain at an average velocity of 12 cm/s. For this to be possible, we increased the step height significantly, otherwise the robot would get “stuck” trying to walk forward, repeatedly stepping in the same location because the shins would collide with the obstacle and the robot would slide backwards.

7. Discussion

When comparing the results of our approach to those of other quadrupedal robots in terms of the achieved walking speeds, it is useful first to examine flat-ground performance. Our experimentation and control architecture were primarily focused on the design of a controller that can navigate rough terrain, hence the emphasis placed on these obstacles in our experiments. However, on flat, level ground, we achieved trotting speeds of approximately 45 cm/s in experimentation using this

same controller with no tuning modifications. When compared to other platforms, the IIT HyQ quadruped used a customized control and offboard power and hydraulic pumps, and was able to perform walking trots at 1.6 m/s and flying trots at 2.0 m/s (Guizzo, 2012). Early on, the Boston Dynamics Big Dog was able to achieve trot speeds of 1.8m/s on flat ground and 0.7m/s over loose rock beds, all without sensors in the loop (Playter et al., 2006; Buehler et al., 2006). The ETH ANYmal robot has been demonstrated as capable of achieving trots at 1.2m/s using both MPC and learned gaits (Bellicoso et al., 2018; Hwangbo et al., 2019). Perhaps most impressively, the MIT MiniCheetah has achieved 3.7m/s on a treadmill (Kim et al., 2019). All of these approaches, however, are inherently different from the goals of our work presented here, and aim at achieving robust *blind* walking, as opposed to specifically tackling foothold-driven, rough terrain walking. Additional hardware results include the 1.6 m/s on flat ground reported by the Boston Dynamics’ Spot, and the impressive mobility demonstrated by the Ghost Robotics Vision 60. However, little is known about the actual methods being used in these platforms, making it hard to compare, with most applications and demonstrations focusing on continuous surfaces rather than the rough terrain that is the focus of this work.

Perhaps a more equitable comparison is with approaches that, like ours, have an intrinsic objective of achieving footholds in rough terrain. The coupled planners from Aceituno-Cabezas et al., 2017a and Mastalli et al., 2020 on HyQ achieved similar performance to the results presented in our paper, navigating 20-25 cm gaps, 10 degree slopes, and height changes up to 25 cm. Walking speeds on these terrains were all approximately 10-13 cm/s. (Mastalli et al., 2020) uses a model predictive control-like policy, where the desired CoM trajectories are computed every timestep. Interestingly, despite having shown HyQ is capable of much higher trotting speeds using other controllers, the approach in (Mastalli et al., 2020) was only capable of achieving 31cm/s on flat ground, a speed comparable to our own controller. The decoupled planners that first plan steps and then plan body paths, similar to our approach, have reported slightly higher speeds, but also over qualitatively similar terrain. (Mastalli et al., 2015) demonstrates HyQ navigating terrain similar to (Mastalli et al., 2020), without reporting achieved speeds. ANYmal has been demonstrated navigating terrain of similar difficulty to ours at 30cm/s (Jenelten et al., 2020). Both the HyQ and ANYmal robots rely on the ability to dynamically replan their footholds, which uses a realtime heightmap update at its core. Notably, ANYmal only computes a single step ahead, which loses the ability to plan complex step paths, but can be done extremely efficiently. Similar to our work, these other results do not inherently guarantee kinematic or dynamic achievability by the robot, instead relying on both approximate reachability and careful tuning to achieve this feasibility.

The disparity in speed performance of approaches that include environmental constraints at the planning phase vs approaches that aim for control robustness is, apparently, large. While we are uncertain as to what drives this underlying difference, this is certainly an area worth exploring. It is possible that, by emphasizing safe footholds, our environmentally driven approach sacrifices beneficial foot poses, whereas robust, blind approaches prioritize nominal robot configurations and then respond to errors accordingly. While the argument has been made that the speed advantage shown by some of the robust approaches like in (Buehler et al., 2006) are driven more by natural dynamics using simpler controllers, this has been somewhat negated by the success of the numerically-precise MPC approaches like those presented on the Mini-Cheetah (Kim et al., 2019).

It is very difficult to compare results, however, between different robotic platforms that have inherently different properties, such as comparing the hydraulic BigDog and HyQ robots to the series elastic ANYmal and quasi-direct drive LLAMA robots. Simulation, in addition, often does not capture all the underlying challenges each of these platforms presents, such as structural compliance and joint friction. This situation, then, makes it extremely difficult to benchmark the results illustrated by one approach on one robot to another approach on a different robot. Additionally, often the “art” that goes into legged systems lies in transferring simulated results to hardware, where it is common to discover that the original idea won’t work at all!

A major challenge for generating and executing robust, quadrupedal gaits is the decision of when to make and break contact. When taking very fast steps, such as what has been demonstrated with

the MIT Mini-Cheetah (Kim et al., 2019), any deviation from the desired gait timing can severely affect subsequent steps, as there is so little timing margin. In this application, making and breaking contact is driven almost entirely by the gait clock, particularly for forcing contact despite it not having been physically achieved on time. When walking very slowly, as in a static gait, the opposite is often the case, as initiating contact prematurely can result in rapidly diverging robot dynamics. This problem raises the question of where on the continuum of step durations should the gait switch from being driven by physically-detected contact to being driven by the gait clock? For most of the work presented here, we required contact be detected, meaning the timing was driven by contact rather than the gait clock. Contact detection without force sensors is challenging, however, and is an ongoing area of research, with particularly promise shown from probabilistic blending of gait time and virtual contact force (Bledt et al., 2018). It is unclear, however, whether it would have been better to have contact be set by the gait clock.

When attempting to implement our whole-body control scheme on LLAMA, we ran into significant challenges getting adequate fidelity on the actuator force control. This difficulty can largely be attributed to the actuator stiction, which was measured at approximately 9 Nm per motor. Typically, this resistance does not pose a significant problem, as it can be overcome via a force control loop. However, with no load-cells, our measured force was actually just the measured current, and did not account for this stiction. While we might have addressed this problem via sufficient motor modeling and characterization, that work-around was beyond the scope and timeline of this work.

Additionally, beyond force control, joint stiction had a significant effect on the robot’s overall compliance. Since LLAMA has no contact sensors, either, contact was estimated using the forces from the motors, meaning that a relatively large amount of loading was required on each leg to overcome the motor stiction for contact detection. Limited fidelity in the force control also necessitates the use of an impedance loop to get sufficient tracking, similar to what was used previously on robots like Atlas during the DARPA Robotics Challenge (Koolen et al., 2016; Kuindersma et al., 2016). This work-around has the unfortunate byproduct of further reducing robot compliance. Additionally, it means that traditional methods for controlling quadrupeds such as virtual model control (Pratt et al., 2001) are not possible, as they rely on accurate force tracking, and typically do not provide joint kinematic setpoints for closed-loop impedance control.

We also observed that the kinematic design of robot limbs can greatly influence the effectiveness of various control methodologies. For example, if all trajectories and controls are being performed in joint-space, singularities in the leg workspace are not important, as the forward kinematics are completely identified. This observation enables clever optimization of the kinematics to maximize efficiency, a tactic used heavily in a companion paper on the design of LLAMA. However, if the trajectories and controls are performed in Cartesian or task-space, Jacobians must be used, meaning workspace singularities in the kinematics are critical. One option is to limit the workspace to avoid these singularities. Unfortunately, this solution would result severely limiting the workspace of LLAMA’s front limbs, and cannot be done for terrain such as that in Figure 22. In that case, the robot reached a singular configuration when attempting to step onto a concrete block (Figure 24). Resolution of this singularity was required for the robot to be able to traverse this environment.

One of the major drawbacks of the “scan, plan, then execute” approach we took in this work is that it is inherently more susceptible to decreased robustness coming from state estimation drift and modeling errors. Any accrued state estimation drift can easily cause the robot to step where there is actually no support. This situation can be corrected in one of two ways: Either introduce localization to compensate for drift or dynamically rescan the environment and replan the footholds while walking. We believe this second method is likely the better of the two, as it both updates the model of the world the robot is using to correct errors in the original scan and recomputes footholds online.

It is interesting to note that the experiment using pre-planned footsteps with knowledge of the terrain environment in Figure 22 was only marginally more reliable than blindly walking over similar terrain in the lab as in Figure 23. This could partially be attributed a decrease in reliability coming from the underlying terrain (sand vs. concrete). However, it is likely more due to both dynamic

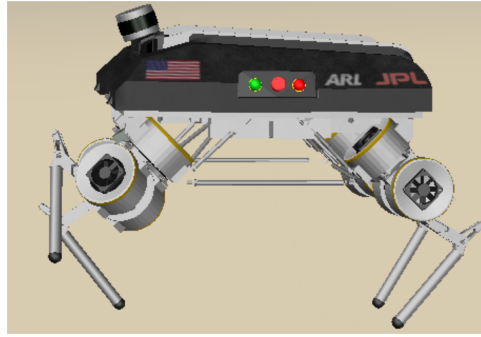


Figure 24. In this configuration, which was found when running at Camp Lejeune in experiment Figure 22, the front left foot Jacobian has a singular value of 0.0197, leaving multiple possible solutions to the inverse kinematics problem.

replanning of the footholds and the underlying foothold selection being more robust. While the A* planner tries to drive the configuration of the robot towards resembling the nominal X-Gait configuration, that is not always achieved, resulting in robot stances that are slightly less stable. Additionally, by dynamically replanning the steps, we allow the robot to use step adjustment, which makes it more robust to tracking error. Lastly, since the upcoming steps are automatically updated in response to step adjustment and tracking error, the upcoming plan is more robust overall.

While our presented control approach does use some elements of model predictive control (MPC), it is not the same as other more classical MPC approaches for trajectory tracking (Kim et al., 2019; Villarreal et al., 2019; Neunert et al., 2018). There have been a number of mixed results as to the effectiveness of MPC for legged platforms, with the more effective approaches tending to be on faster-moving robots, where the robot is capable of responding to the modified dynamic plan. We believe that, because of the much greater control authority presented with quadrupeds from both their larger support polygon and faster contact changes, it is likely that MPC is a more effective approach here than traditional slow moving humanoids, which are severely limited in their control authority. These approaches have been used with great success in blind-walking controllers, but have, for whatever reason, had difficulty translating to architectures designed around rough terrain (see the discussion at the beginning of this section). This is certainly an area to explore more in-depth in the future.

It is also likely that planning over the compressed search space of planar regions is an unnecessary optimization for quadrupeds. This approach was inspired by our work on humanoid robots (see (Griffin et al., 2019)), where the planar regions offer both a compressed and easy-to-use convex model of the environment. While this is ideal for humanoid robots, where an entire contact polygon must be considered when planning footholds, it is likely unnecessary for quadrupeds. Instead, traditional height maps with local evaluations of valid footholds at each voxel are likely sufficient due to the robot having point feet, and have been extensively used previously (Fankhauser et al., 2014; Mastalli et al., 2015; Fankhauser et al., 2018). This represents less computation, resulting in a faster perception pipeline, and could facilitate dynamic replanning for a more robust quadruped gait in complex environments. It is also readily compatible with our presented planning frameworks in Section 4.

8. Conclusion

In this paper we presented a control and planning architecture and demonstrated it on the experimental quadrupedal robot platform, LLAMA. We used our experience with humanoid and bipedal robots to design and implement some critical components, such as the perception processing and momentum-based control. We proposed a novel extension of the DCM planner to quadruped

robots, while leveraging many of the same whole-body momentum control techniques used in bipeds. We also introduced a number of quadruped-specific techniques. These innovations include two methods for desired foothold calculations based on a concept called the X-Gait, a new approach for step adjustment for balance regulation, and a fusion between kinematic and torque based contact detection.

While the duration of the project limited the effort for validating and refining the architecture proposed in this paper, we demonstrate several experiments demonstrating the potential efficacy of our approach. These experiments were conducted both in our laboratory and at the RCTA Capstone Event at the Camp Lejeune Marine Corp base in North Carolina, and demonstrated the robot's ability to traverse rough terrain both indoors and outdoors using our controller approach.

Perhaps most significantly, we were able to gain general lessons on the control of quadrupedal robots and present these insights in this work. Major take-aways include the significance of control consideration on the kinematic design of articulated robots and the importance of low-friction, high-quality force-control quality when attempting to implement compliant control without any closed-loop force sensing. Additionally, we discuss differences in system requirements between bipedal and quadrupedal robots, including the requirements of environmental representations and the impacts of the much faster gait-cycles present in quadrupeds.

We plan to improve the perception-processing pipeline to enable dynamic replanning of the desired footholds and providing robustness to state estimation drift and dynamic environments. Additionally, we intend to evaluate our current stepping strategy for recovering balance using real-time perception data and investigate its impact on a planned sequence of footsteps. Finally, we plan to explore more in depth use of model predictive control its effectiveness when trying to achieve specific footholds in complex environments to drive higher speed locomotion that still has the same environmental awareness.

Acknowledgments

The work described in this publication was carried out at the Florida Institute for Human and Machine Cognition (IHMC), in collaboration with Autonomous Systems Division, U.S Army Research Laboratory; Jet Propulsion Laboratory, California Institute of Technology; Autonomous Systems, General Dynamics Land Systems; Applied Physics Laboratory (APL), John Hopkin's University. Special thanks are due to Mark Gonzalez and Dilip Patel from General Dynamics and John Nicholson from Florida State University.

ORCID

Robert Griffin  <https://orcid.org/0000-0002-1128-346X>
 Sylvain Bertrand  <https://orcid.org/0000-0003-4405-2665>
 Jay Jasper  <https://orcid.org/0000-0002-1216-936X>
 Sisir Karumanchi  <https://orcid.org/0000-0002-0685-4125>
 Rachel Hegeman  <https://orcid.org/0000-0003-2869-8773>
 Jerry Pratt  <https://orcid.org/0000-0001-8414-5220>

References

- Acetuno-Cabezas, B., Dai, H., Cappelletto, J., Grieco, J. C., and Fernández-López, G. (2017a). A mixed-integer convex optimization framework for robust multilegged robot locomotion planning over challenging terrain. In *2017 IEEE/RSJ 17th International Conference on Intelligent Robots and Systems (IROS)*, pages 4467–4472. IEEE.
- Acetuno-Cabezas, B., Mastalli, C., Dai, H., Focchi, M., Radulescu, A., Caldwell, D. G., Cappelletto, J., Grieco, J. C., Fernández-López, G., and Semini, C. (2017b). Simultaneous contact, gait, and motion

- planning for robust multilegged locomotion via mixed-integer convex optimization. *IEEE Robotics and Automation Letters*, 3(3):2531–2538.
- Ajalloeian, M., Pouya, S., Sproewitz, A., and Ijspeert, A. J. (2013). Central pattern generators augmented with virtual model control for quadruped rough terrain locomotion. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 3321–3328. IEEE.
- Barasuol, V., Buchli, J., Semini, C., Frigerio, M., De Pieri, E. R., and Caldwell, D. G. (2013). A reactive controller framework for quadrupedal locomotion on challenging terrain. In *2013 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2554–2561. IEEE.
- Bazeille, S., Barasuol, V., Focchi, M., Havoutis, I., Frigerio, M., Buchli, J., Caldwell, D. G., and Semini, C. (2014). Quadruped robot trotting over irregular terrain assisted by stereo-vision. *Intelligent Service Robotics*, 7(2):67–77.
- Bellicoso, C. D., Gehring, C., Hwangbo, J., Fankhauser, P., and Hutter, M. (2016). Perception-less terrain adaptation through whole body control and hierarchical optimization. In *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, pages 558–564. IEEE.
- Bellicoso, C. D., Jenelten, F., Gehring, C., and Hutter, M. (2018). Dynamic locomotion through online nonlinear motion optimization for quadrupedal robots. *IEEE Robotics and Automation Letters*, 3(3):2261–2268.
- Bertrand, S., Lee, I., Mishra, B., Calvert, D., Pratt, J., and Griffin, R. (2020). Detecting usable planar regions for legged robot locomotion. In *2020 IEEE/RSJ 20th International Conference on Intelligent Robots and Systems (IROS)*. IEEE.
- Bledt, G., Wensing, P. M., Ingersoll, S., and Kim, S. (2018). Contact model fusion for event-based locomotion in unstructured terrains. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 4399–4406. IEEE.
- Buehler, M., Grimminger, F., Campbell, D., and Raibert, M. (2006). Biologically inspired robots at boston dynamics. In *Bionic 2006 Industrie-Kongress*.
- Craig, J. J. et al. (1989). Introduction to robotics.
- Edelsbrunner, H. and Mücke, E. P. (1994). Three-dimensional alpha shapes. *ACM Transactions on Graphics (TOG)*, 13(1):43–72.
- Englsberger, J., Koolen, T., Bertrand, S., Pratt, J., Ott, C., and Albu-Schäffer, A. (2014). Trajectory generation for continuous leg forces during double support and heel-to-toe shift based on divergent component of motion. In *2014 IEEE/RSJ 14th International Conference on Intelligent Robots and Systems (IROS)*, pages 4022–4029. IEEE.
- Fankhauser, P., Bjelonic, M., Bellicoso, C. D., Miki, T., and Hutter, M. (2018). Robust rough-terrain locomotion with a quadrupedal robot. In *2018 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1–8. IEEE.
- Fankhauser, P., Bloesch, M., Gehring, C., Hutter, M., and Siegwart, R. (2014). Robot-centric elevation mapping with uncertainty estimates. In *Mobile Service Robotics*, pages 433–440. World Scientific.
- Geisert, M., Yates, T., Orgen, A., Fernbach, P., and Havoutis, I. (2019). Contact planning for the anymal quadruped robot using an acyclic reachability-based planner. In *Annual Conference Towards Autonomous Robotic Systems*, pages 275–287. Springer.
- Griffin, R. J., Wiedebach, G., Bertrand, S., Leonessa, A., and Pratt, J. (2017). Walking Stabilization Using Step Timing and Location Adjustment on the Humanoid Robot, Atlas. In *2017 IEEE/RSJ 17th International Conference on Intelligent Robots and Systems (IROS)*, pages 667–673. IEEE.
- Griffin, R. J., Wiedebach, G., McCrory, S., Bertrand, S., Lee, I., and Pratt, J. (2019). Footstep planning for autonomous walking over rough terrain. In *2019 IEEE-RAS 19th International Conference on Humanoid Robots (Humanoids)*, pages 9–16. IEEE.
- Guizzo, E. (2012). Italian quadruped robot goes for a walk.
- Gutmann, J.-S., Fukuchi, M., and Fujita, M. (2008). 3d perception and environment map generation for humanoid robot navigation. *The International Journal of Robotics Research*, 27(10):1117–1134.
- Havoutis, I., Ortiz, J., Bazeille, S., Barasuol, V., Semini, C., and Caldwell, D. G. (2013). Onboard perception-based trotting and crawling with the hydraulic quadruped robot (HyQ). In *2013 IEEE/RSJ 13th International Conference on Intelligent Robots and Systems (IROS)*, pages 6052–6057. IEEE.
- Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., and Burgard, W. (2013). Octomap: An efficient probabilistic 3d mapping framework based on octrees. *Autonomous robots*, 34(3):189–206.
- Hwangbo, J., Lee, J., Dosovitskiy, A., Bellicoso, D., Tsounis, V., Koltun, V., and Hutter, M. (2019). Learning agile and dynamic motor skills for legged robots. *Science Robotics*, 4(26).

- Jenelten, F., Miki, T., Vijayan, A. E., Bjelonic, M., and Hutter, M. (2020). Perceptive locomotion in rough terrain – online foothold optimization.
- Kalakrishnan, M., Buchli, J., Pastor, P., Mistry, M., and Schaal, S. (2011). Learning, planning, and control for quadruped locomotion over challenging terrain. *The International Journal of Robotics Research*, 30(2):236–258.
- Kalakrishnan, M., Buchli, J., Pastor, P., and Schaal, S. (2009). Learning locomotion over rough terrain using terrain templates. In *2009 IEEE/RSJ 9th International Conference on Intelligent Robots and Systems (IROS)*, pages 167–172. IEEE.
- Katz, B., Di Carlo, J., and Kim, S. (2019). Mini cheetah: A platform for pushing the limits of dynamic quadruped control. In *2019 International Conference on Robotics and Automation (ICRA)*, pages 6295–6301. IEEE.
- Kim, D., Carballo, D., Di Carlo, J., Katz, B., Bledt, G., Lim, B., and Kim, S. (2020). Vision aided dynamic exploration of unstructured terrain with a small-scale quadruped robot. In *2020 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2464–2470. IEEE.
- Kim, D., Di Carlo, J., Katz, B., Bledt, G., and Kim, S. (2019). Highly dynamic quadruped locomotion via whole-body impulse control and model predictive control. *arXiv preprint arXiv:1909.06586*.
- Kolter, J. Z., Kim, Y., and Ng, A. Y. (2009). Stereo vision and terrain modeling for quadruped robots. In *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1557–1564. IEEE.
- Kolter, J. Z., Rodgers, M. P., and Ng, A. Y. (2008). A control architecture for quadruped locomotion over rough terrain. In *2008 IEEE International Conference on Robotics and Automation (ICRA)*, pages 811–818. IEEE.
- Koolen, T., Bertrand, S., Thomas, G., De Boer, T., Wu, T., Smith, J., Engelsberger, J., and Pratt, J. (2016). Design of a momentum-based control framework and application to the humanoid robot atlas. *International Journal of Humanoid Robotics*, 13(01):1650007.
- Kuindersma, S., Deits, R., Fallin, M., Valenzuela, A., Dai, H., Permenter, F., Koolen, T., Marion, P., and Tedrake, R. (2016). Optimization-Based Locomotion Planning, Estimation and Control Design for Atlas. *Autonomous Robots*, 40(3):429–455.
- Lien, J.-M. and Amato, N. M. (2006). Approximate convex decomposition of polygons. *Computational Geometry*, 35(1-2):100–123.
- Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G., and Semini, C. (2020). Motion planning for quadrupedal locomotion: coupled planning, terrain mapping and whole-body control. *IEEE Transactions on Robotics*.
- Mastalli, C., Havoutis, I., Winkler, A. W., Caldwell, D. G., and Semini, C. (2015). On-line and on-board planning and perception for quadrupedal locomotion. In *2015 IEEE International Conference on Technologies for Practical Robot Applications (TePRA)*, pages 1–7. IEEE.
- Neunert, M., Stäubli, M., Gifftthaler, M., Bellicoso, C. D., Carius, J., Gehring, C., Hutter, M., and Buchli, J. (2018). Whole-body nonlinear model predictive control through contacts for quadrupeds. *IEEE Robotics and Automation Letters*, 3(3):1458–1465.
- Playter, R., Buehler, M., and Raibert, M. (2006). Bigdog. In *Unmanned Systems Technology VIII*, volume 6230, page 62302O. International Society for Optics and Photonics.
- Pongas, D., Mistry, M., and Schaal, S. (2007). A robust quadruped walking gait for traversing rough terrain. In *2007 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1474–1479. IEEE.
- Pratt, J., Chew, C.-M., Torres, A., Dilworth, P., and Pratt, G. (2001). Virtual model control: An intuitive approach for bipedal locomotion. *The International Journal of Robotics Research*, 20(2):129–143.
- Raibert, M., Blankespoor, K., Nelson, G., and Playter, R. (2008). Bigdog, the rough-terrain quadruped robot. *IFAC Proceedings Volumes*, 41(2):10822–10825.
- Rebula, J. R., Neuhaus, P. D., Bonnlander, B. V., Johnson, M. J., and Pratt, J. E. (2007). A controller for the littledog quadruped walking on rough terrain. In *2007 IEEE International Conference on Robotics and Automation (ICRA)*, pages 1467–1473. IEEE.
- Shkolnik, A., Levashov, M., Manchester, I. R., and Tedrake, R. (2011). Bounding on rough terrain with the littledog robot. *The International Journal of Robotics Research*, 30(2):192–215.
- Takenaka, T., Matsumoto, T., and Yoshiike, T. (2009). Real time motion generation and control for biped robot -1st report: Walking gait pattern generation. In *2009 IEEE/RSJ 9th International Conference on Intelligent Robots and Systems (IROS)*, pages 1084–1091. IEEE.

- Vernaza, P., Likhachev, M., Bhattacharya, S., Chitta, S., Kushleyev, A., and Lee, D. D. (2009). Search-based planning for a legged robot over rough terrain. In *2009 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2380–2387. IEEE.
- Villarreal, O., Barasuol, V., Wensing, P., and Semini, C. (2019). Mpc-based controller with terrain insight for dynamic legged locomotion. *arXiv preprint arXiv:1909.13842*.
- Winkler, A., Havoutis, I., Bazeille, S., Ortiz, J., Focchi, M., Dillmann, R., Caldwell, D., and Semini, C. (2014). Path planning with force-based foothold adaptation and virtual model control for torque controlled quadruped robots. In *2014 IEEE International Conference on Robotics and Automation (ICRA)*, pages 6476–6482. IEEE.
- Winkler, A. W., Mastalli, C., Havoutis, I., Focchi, M., Caldwell, D. G., and Semini, C. (2015). Planning and execution of dynamic whole-body locomotion for a hydraulic quadruped on challenging terrain. In *2015 IEEE International Conference on Robotics and Automation (ICRA)*, pages 5148–5154. IEEE.
- Zucker, M., Ratliff, N., Stolle, M., Chestnutt, J., Bagnell, J. A., Atkeson, C. G., and Kuffner, J. (2011). Optimization and learning for rough terrain legged locomotion. *The International Journal of Robotics Research*, 30(2):175–191.

How to cite this article: Griffin, R., McCrory, S., Bertrand, S., Calvert, D., Lee, I., Neuhaus, P., Stephen, D., Jasper, J., Karumanchi, S., Kourchians, A., Emanuel, B., Holmes, E., Hegeman, R., Pusey, J., & Pratt J. (2022). Quadrupedal Walking over Complex Terrain with a Quasi-Direct Drive Actuated Robot. *Field Robotics*, 2, 356–384.

Publisher's Note: Field Robotics does not accept any legal responsibility for errors, omissions or claims and does not provide any warranty, express or implied, with respect to information published in this article.