Perceptive Whole Body Planning for Multi-legged Robots in Confined Spaces

Journal Article

Author(s):
Buchanan, Russell ⓘD; Wellhausen, Lorenz ⓘD; Bjelonic, Marko; Bandyopadhyay, Tirthankar; Kottege, Navinda; Hutter, Marco ⓘD

REGULAR ARTICLE

WILEY

Perceptive whole-body planning for multilegged robots in confined spaces

Russell Buchanan[1] ⬤ | Lorenz Wellhausen[2] | Marko Bjelonic[2] |
Tirthankar Bandyopadhyay[3] | Navinda Kottege[3] | Marco Hutter[2]

[1]Oxford Robotics Institute, University of Oxford, Oxford, United Kingdom

[2]Robotic Systems Lab, ETH Zürich, Zürich, Switzerland

[3]Robotics and Autonomous Systems Group, CSIRO, Pullenvale, QLD, Australia

Correspondence
Russell Buchanan, Oxford Robotics Institute, University of Oxford, Oxford OX1 3PJ, United Kingdom.
Email: russell@robots.ox.ac.uk

Abstract

Legged robots are exceedingly versatile and have the potential to navigate complex, confined spaces due to their many degrees of freedom. As a result of the computational complexity, there exist no online planners for perceptive whole-body locomotion of robots in tight spaces. In this paper, we present a new method for perceptive planning for multilegged robots, which generates body poses, footholds, and swing trajectories for collision avoidance. Measurements from an onboard depth camera are used to create a three-dimensional map of the terrain around the robot. We randomly sample body poses then smooth the resulting trajectory while satisfying several constraints, such as robot kinematics and collision avoidance. Footholds and swing trajectories are computed based on the terrain, and the robot body pose is optimized to ensure stable locomotion while not colliding with the environment. Our method is designed to run online on a real robot and generate trajectories several meters long. We first tested our algorithm in several simulations with varied confined spaces using the quadrupedal robot ANYmal. We also simulated experiments with the hexapod robot Weaver to demonstrate applicability to different legged robot configurations. Then, we demonstrated our whole-body planner in several online experiments both indoors and in realistic scenarios at an emergency rescue training facility. ANYmal, which has a nominal standing height of 80 cm and a width of 59 cm, navigated through several representative disaster areas with openings as small as 60 cm. Three-meter trajectories were replanned with 500 ms update times.

KEYWORDS
emergency response, extreme environments, legged robots, motion planning

1 | INTRODUCTION

There are many situations in which humans must enter confined spaces, such as industrial inspection or in response to emergencies. These environments are dangerous however, and there is a need for new technologies to make working in them safer. Legged robots are uniquely well suited for these spaces as their many degrees of freedom (DOF) enable locomotion over terrain that is hard or even impossible for wheeled or tracked robots. However, there remain significant challenges in robotic legged locomotion in confined

spaces. Planning algorithms must compute both foot trajectories and body poses which ensure stability and avoid collisions with the environment. A robot may need to plan steps over obstacles on the ground while also lowering or rotating its body to pass under an overhanging obstacle, such as in Figure 1. Since prior information is not guaranteed, a robot also needs sensors to map the surrounding terrain.

The principal challenge in planning for legged robots comes from the complexity of the problem and often motivates a trade-off between planning time, detail of the robot model, and planning horizon. Current whole-body planners for multilegged robots have fast update rates but do not consider collisions of the robot's body with terrain (Fankhauser, Bjelonic, Bellicoso, Miki, & Hutter, 2018; Magaña et al., 2019; Mastalli et al., 2017). Sampling-based planners that do consider these collisions take significant time for computation and have not been demonstrated on real robots (Grey, Ames, & Liu, 2017; Kumagai, Morisawa, Nakaoka, & Kanehiro, 2018; Short & Bandyopadhyay, 2018). Geisert, Yates, Orgen, Fernbach, and Havoutis (2019) have shown that the acyclic planner from Tonneau et al. (2018) can be applied to multilegged robots; however, their method relies on inflating the collision model of the robot, which would prevent it from working in confined spaces. Additionally, it takes several seconds to generate a trajectory and has not been demonstrated on a real robot.

In this paper, we attempt to fill the gap in current methods and enable multilegged robots to walk in confined spaces. Our method generates several meter trajectories of the 6DOF pose of the torso and plans the three-dimensional (3D) positions of the feet for each footstep. Our approach is hierarchical, first sampling a feasible trajectory of body poses, which is then smoothed using a gradient descent method similar to Covariant Hamiltonian Optimization for Motion Planning (CHOMP; Zucker et al., 2013). Next, footsteps and swing trajectories are adapted based on the terrain similar to Fankhauser et al. (2018). This way, a robot can crawl under or step over obstacles. The final phase of our planner is the optimization of the body pose to maintain the nominal rotation and height while also ensuring static stability while taking a step. Our method can update fast enough for replanning on a real robot. Our main contributions are highlighted below:

- We present a method for whole-body planning for multilegged robots which generalizes to different platforms and can run onboard a real robot with less than half a second update time.
- Our method is particularly suited for confined spaces as we do not inflate the collision model of the robot. By combining sampling with trajectory optimization, we also avoid many local minima associated with confined spaces.
- Our method is one of the few to be demonstrated on a real robot and, to the best of our knowledge, to first to be deployed in field experiments. We show our robot navigating confined spaces used for training emergency rescuers.

2 | RELATED WORKS

While many DOF enable legged robots to traverse challenging terrain, it also makes whole-body planning computationally demanding. Despite these challenges, there exist many examples of perceptive whole-body planning. In this section, we discuss the current state of the art and identify gaps in capabilities.

2.1 | Trajectory planning

There exists a significant body of work on trajectory planning in 3D. Typically approaches attempt to formulate an optimization problem that balances obstacle avoidance with reaching a goal state. *TrajOpt* (Schulman et al., 2014) is one method which uses a Sequential Quadratic Programming (SQP) solver to compute trajectories. To check for collisions, they decompose the robot geometry and objects in the environment into meshes or convex 3D primitives and compute the closest points on each object to the other. CHOMP (Zucker et al., 2013) is another optimization-based approach which instead uses a Signed Distance Field (SDF) for collision checking. These fields are a discretization of 3D space where the value of each voxel is the distance from that voxel to the nearest object. Figure 2 shows an example from simulated data. Once the SDF is generated, it can be queried for collisions in constant time. In our previous work (Buchanan et al., 2019), we used CHOMP and SDFs to plan a robot's body height and leg span in

FIGURE 1   ANYmal autonomously navigating confined spaces using onboard mapping and no prior knowledge of the environment. The footholds, body poses, and transitions were all planned on the robot [Color figure can be viewed at wileyonlinelibrary.com]
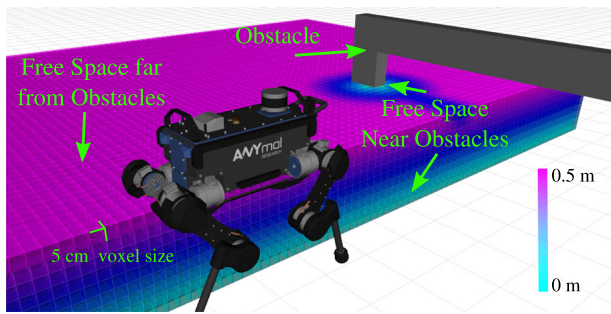
FIGURE 2 We show a 3 m × 3 m × 1 m SDF, centered on the robot, with two cross sections at $z = 0.5$ m and $y = 0$ m so that the robot model and obstacle are visible. The 5 cm voxels are colorized to indicate distance from obstacles. Cyan voxels are closer to the obstacle while purple voxels are further away. SDF, Signed Distance Field [Color figure can be viewed at wileyonlinelibrary.com]

confined spaces. We assumed, however, flat terrain and did not plan for the footholds of the robot nor the robot's 6DOF pose. The work described in this paper significantly extends the capability of our previous planner, and we show in simulations how we enable a robot to walk through previously untraversable terrain.

Other trajectory planners employ random sampling. In this case, rather than explicitly defining an optimization problem, only the constraints and state costs are defined, and robot states are randomly sampled until a valid path is found. The rapidly exploring random trees (RRT) algorithm (Lavalle, 1998) has been used extensively in this regard. Adapted algorithms can gain some of the useful features of optimization problems, such as asymptotic optimality (Karaman & Frazzoli, 2011). Directed methods, such as RRT-Connect (Kuffner & LaValle, 2000), can efficiently find paths between points, but these methods suffer from the curse of dimensionality, and thus, it can be challenging to plan for legged robots online. This has motivated several different approaches and compromises to planning for legged robots.

2.2 | Planning for legged robots

Many whole-body planners for multilegged robots are primarily focused on finding footholds in rough terrain and optimizing body posture for stability. Mastalli et al. (2017) and Fankhauser et al. (2018) both search in an elevation map for footholds on suitable terrain. Magaña et al. (2019) train a Convolutional Neural Network (CNN) to adapt footholds based on the elevation map for quicker reactions to disturbances. These methods have shown quadrupeds, climbing stairs, and walking in rough terrain. However, they do not consider collisions of the body of the robot with the terrain and would not be able to walk in confined spaces.

Sampling planners have been used successfully for legged robots (Grey et al., 2017; Kumagai et al., 2018; Short & Bandyopadhyay, 2018), where the focus is typically on reducing computational complexity by reducing the sampling space. Grey et al. (2017) do this by simplifying the model of the robot and reducing the space of feasible motions. Short

and Bandyopadhyay (2018) save on computation by sampling a roadmap of feasible configurations for their quadruped offline in the absence of any terrain. Then, when planning for a given environment, the sampling space is significantly reduced, allowing planning in complex environments. Along similar lines, Kumagai et al. (2018) precompute a global footstep path then perform the more time-sensitive body motions online. All of these methods demonstrate complex planning through cluttered and narrow environments but are still very computationally demanding, taking several seconds to generate a plan. It is also notable that none of these methods has been demonstrated on a real robot.

The acyclic contact planner from Tonneau et al. (2018) uses a hierarchical approach and is the most similar method to our own. Similar to our method, they initially sample a trajectory of feasible poses for the base of the robot. Unlike our method, they use an inflated abstraction of the robot model to check for collisions. While this speeds up planning considerably, it is notable that this would not be suitable for planning in confined spaces. Similar to Short and Bandyopadhyay (2018), they sample from a selection of precomputed configurations, selecting configurations that avoid collisions and maximize manipulability. This step is by far the most time-consuming part of their pipeline. In contrast, our method is much faster at selecting footholds because we constrain the contact planning step to a 2D manifold. This simplification makes our method less appropriate for humanoid robots but massively speeds up planning and is quite reasonable for a multilegged robot. Geisert et al. (2019) implemented the planner from Tonneau et al. (2018) in simulation on a quadruped robot. They bias the contact sampling to encourage a suitable Support Polygon (SP), and they use an inflated collision model to prevent the robot from lowering its torso. In contrast, we enforce stability as an optimization constraint and use the exact collision model of the robot's torso.

Like the majority of the methods above, we rely on static stability. If the robot's Center of Mass (COM) lies within the convex hull of foot contacts projected onto the horizontal plane, the robot is assumed stable. As we use multilegged robots with at least four legs, we can exploit this approximation, although the robot must walk more slowly. Other methods incorporate the robot's dynamics directly in the optimization problem. Winkler, Bellicoso, Hutter, and Buchli (2018) solve the full robot posture, including transition motions, by formulating a Nonlinear Programming (NLP) problem while Aceituno-Cabezas et al. (2018) use Mixed Integer Programming. Both methods approximate the robot dynamics as a single mass with inertia and contact points for feet. This modeling enables them to plan for dynamic gaits, which increases robot speed and robustness against disturbances. They do not, however, consider collisions of the body with the terrain and must sacrifice either planning time or planning horizon.

2.3 | Mapping

For a robot to plan and walk through real confined spaces, it must be able to map the terrain and identify free space. OctoMap (Hornung, Wurm, Bennewitz, Stachniss, & Burgard, 2013) has been widely used

in robotics for occupancy mapping; however, it is not particularly efficient for collision checking as data look-ups require tree traversal. Voxblox (Oleynikova, Taylor, Fehr, Siegwart, & Nieto, 2017) builds 3D maps using voxel hashing, which allows it to grow the map only when there is new data. They also compute the SDF directly from the depth measurements. However, they trade accuracy for real-time performance and use large (10–20 cm) resolutions. While this is acceptable for flying robots, which must keep a significant distance from obstacles, it is not suitable for ground robots in confined spaces. Another method often used for ground robots is digital elevation mapping (Herbert, Caillas, Krotkov, Kweon, & Kanade, 1989). These are often called 2.5D maps because a grid is aligned with the robot's walking plane, where each cell records the terrain height. This representation of the environment has been highly successful for legged robots (Belter, Łabęcki, Fankhauser, & Siegwart, 2016; Fankhauser, Bloesch, & Hutter, 2018) because it is very memory efficient and provides a useful approximation of the ground for foot placement. We use multi-elevation maps (Buchanan et al., 2019), which are a compromise between 2.5D and 3D mapping. We cluster depth measurements into two elevations below and above the robot. An SDF can then be computed between these two 2.5D maps.

3 | POSTURE PLANNING

In this section, we will go into the details of our planning method. As shown in Figure 3, we use a hierarchical approach to planning, starting with a goal pose and current robot posture. We first randomly sample the planning space to find a feasible trajectory of discrete body poses. Due to the randomness of this method, this trajectory is highly irregular, therefore we perform smoothing in continuous time. This trajectory can be reused for online planning. For each pose in the trajectory we seek to find four-foot contacts which are selected from the map based on terrain suitability. The body pose is optimized for stability to ensure the footholds are possible to attain, and finally leg swing trajectories are computed to avoid collisions. Table 1 summarizes information for each planning step. In the second column we highlight which parts of the robot

model are considered, the third column details if, and how, collisions are checked, and the last column indicates the sections in this paper describing the planning module.

3.1 | Preliminaries

We define frame I as our fixed inertial frame of reference. The body frame B is attached to the robot's torso and moves with the robot. A pose is described with a 4 × 4 transformation matrix belonging to the *special Euclidean group* SE(3). The robot's body pose relative to the inertial frame is written as T_{IB} which is given by

$$ T_{IB} := \begin{bmatrix} R_{IB} & {}_I t_{IB} \\ 0_{1\times3} & 1 \end{bmatrix} \in SE(3), \tag{1} $$

where $R_{IB} \in SO(3)$ is the rotation matrix that projects the components of a vector from the body frame B to the inertial frame I and ${}_I t_{IB} \in \mathbb{R}^3$ is the position from frame I to B with respect to (w.r.t.) frame I. The robot's feet are modeled as point contacts with position ${}_I t_{IF_i} \in \mathbb{R}^3$ and no rotation. We define the robot's *posture* w.r.t. I, ${}_I P$, as a combination of the body pose and foothold positions in frame I such that

$$ {}_I P = \left\{ T_{IB}, {}_I t_{IF_i} \mid T_{IB} \in SE(3), \quad {}_I t_{IF_i} \in \mathbb{R}^3, \forall i = 0, 1, 2, ..., N \right\}, \tag{2} $$

where N is the number of legs of the robot. Our goal is to compute ${}_I P$ for a legged robot from its start to a goal point while avoiding obstacles.

3.2 | Constraints

To ensure that the final plan is feasible and safe for a robot to carry out we impose several constraints in each phase of planning. In this section, we define and describe these constraints:

1. *Contact Reachability*: The robot must be able to attain every body pose while maintaining contact with the ground. To ensure this,
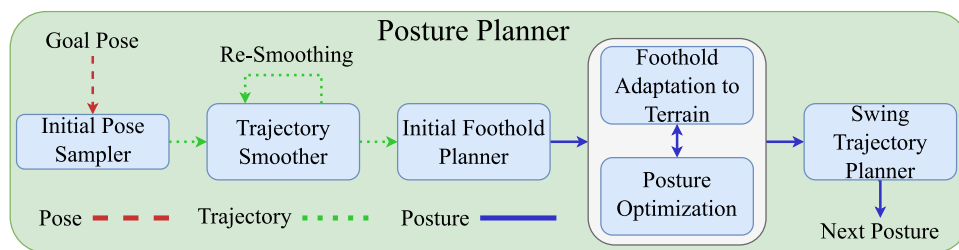

FIGURE 3 Information flow diagram. The planner receives a goal pose defined in (1) and begins to plan a trajectory for the robot. An initial trajectory of poses is sampled from an SDF which is then smoothed. This trajectory is resmoothed for replanning each footstep. Initial footholds are generated for the next pose in the trajectory so that every pose corresponds to a posture as defined in (2). Footholds and body pose are then optimized based on terrain costs and solving an NLP problem, respectively. Finally, the swing trajectory is computed. NLP, Nonlinear Programming; SDF, Signed Distance Field [Color figure can be viewed at wileyonlinelibrary.com]

Planning step	State space	Collision model	Environment representation	Section
Initial sampler	$\boldsymbol{T}_{IB}$	Body	SDF	3.2.1
Trajectory smoother	$\boldsymbol{T}_{IB}$	Body	SDF	3.2.2
Initial footholds	$_i\boldsymbol{P}$	Body and feet	None	3.3
Foothold adaptation	$_i\boldsymbol{t}_{IF_i}$	Feet	Elevation map	3.3
Posture optimization	$_i\boldsymbol{P}$	Body and feet	None	3.4
Swing trajectory planner	$_i\boldsymbol{t}_{IF_i}$	Feet	SDF	3.3

Abbreviation: SDF, Signed Distance Field.

we set a maximum pose height parameter h_{max} which is the maximum allowable height above the terrain elevation.[1]

2. *Sequential Reachability*: It must be possible for the robot to transition between two sequential body poses with a single step. We enforce this by limiting the Euclidean distance between two sequential body poses to be less than a maximum step distance parameter s_{max}.

3. *Collisions*: Body poses must avoid collisions with the terrain. We use an SDF to check several points on the robot's torso for collisions. For the initial sampling, we check if the point is in collision or not. For smoothing, the checkpoint has a radius r_c and a cost function $f_{obstacle}$ such that obstacles inside the radius are a soft constraint for the smoothing process. Unlike Tonneau et al. (2018) and Grey et al. (2017) we do not inflate the body of the robot which is undesirable for navigating confined spaces. The collision modeling is described in more detail in Section 4.2.

4. *Stability*: Postures should be statically stable. To ensure this, we consider the SP, which is defined by the convex hull of the feet positions projected onto a plane whose normal aligns with gravity. If the COM of the robot, projected onto this plane, lies within the polygon, the posture can be said to be stable. We check for this in the posture optimization step, and details can be found in Section 3.4.

In Figure 4 we illustrate how the first three constraints can be violated. The initial pose sampler and the trajectory smoother consider constraints 1–3 from above while the *stability* constraint is handled by the final pose optimization which takes place after foothold selection. Since the optimization occurs in the final step, the robot's body pose may have to change to ensure stability, and this could result in a collision with the terrain. This limitation is a result of using a hierarchical planner: lower levels of planning may not be able to consider all of the constraints. In practice, we have found the few body collisions to be minor and have never interfered with the robot's mission.

---

[1]This is similar to Tonneau et al. (2018) but only considers height rather than an approximation of the workspace of each leg. Our constraint eliminates the need to generate a reachable workspace for every robot and we have found it to be sufficient for multilegged robots.

3.2.1 | Initial pose sampling

We initially sample a trajectory of discrete body poses from the robot's state to the goal. The sample space is a 3D box which is large enough to contain the two endpoints of the trajectory with additional space in the positive and negative X and Y directions so that paths can be found around obstacles. We use the RRT-Connect algorithm (Kuffner & LaValle, 2000) which generates two RRTs, one rooted at the start and the other at the goal. Poses are randomly sampled and added to each tree using a greedy heuristic until the trees connect. We reject poses that violate constraints 1–3 and use the SDF from the map to avoid collisions as a hard constraint.

3.2.2 | Trajectory smoothing

Our method for smoothing the body pose trajectory is based on the local optimization method CHOMP (Zucker et al., 2013). To use this planner, we reformulate our trajectory in continuous time as

$$\xi(t) = [x(t), y(t), z(t), \phi(t), \theta(t), \psi(t)], \tag{3}$$

where x, y, z represent the position and $\phi(t), \theta(t), \psi(t)$ the rotations in Euler angles of the robot body at time t. The robot's maximum absolute pitch and roll are limited to 45° which allows us to use Euler angle rotations without danger of singularities. We use the functional gradient descent update rule given by

$$\xi_{i+1} = \xi_i - \frac{1}{\eta}\boldsymbol{A}^{-1}\bar{\nabla}\mathcal{U}[\xi_i], \tag{4}$$

where the norm $\boldsymbol{A}$ is formed by multiplication of differencing matrices and acts as a smoothing operator. The learning rate η regulates the speed of convergence to a solution for each iteration i. $\bar{\nabla}\mathcal{U}$ is a functional gradient that operates on the trajectory configuration function $\xi(t)$ defined in (3). This functional gradient is the sum of two gradients $\bar{\nabla}\mathcal{F}_{smooth}$ and $\bar{\nabla}\mathcal{F}_{obstacle}$.

The smoothness gradient $\bar{\nabla}\mathcal{F}_{smooth}$ is calculated as the negative second-order derivative of $\xi(t)$ and ensures that generated trajectories are kept smooth. The obstacle gradient $\bar{\nabla}\mathcal{F}_{obstacle}$ is the sum of costs $c(x)$ for each collision checkpoint. The cost for a single point is calculated by
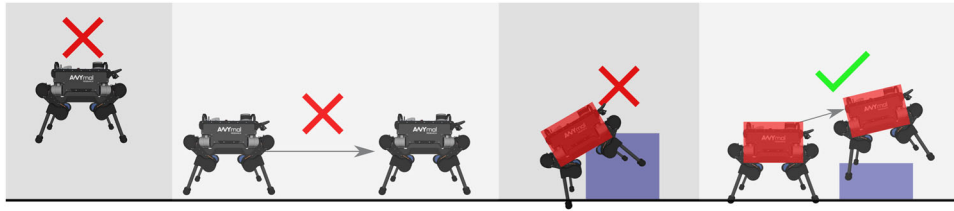
FIGURE 4 Depiction the first three planning constraints. *From left to right*: (1) For contact reachability, the body pose cannot be too high above the terrain. (2) To maintain sequential reachability, two consecutive poses cannot be too far apart. (3) Body poses must not place the torso inside an obstacle. (4) The last robot transition satisfies constraints 1–3 [Color figure can be viewed at wileyonlinelibrary.com]

$$c(x) = \begin{cases} -x + \frac{1}{2}r_c & \text{for } x \leq 0, \\ \frac{1}{2r_c}(x - r_c)^2 & \text{for } 0 < x \leq r_c, \\ 0 & \text{for } r_c \leq x, \end{cases} \quad (5)$$

where x is the collision distance queried from the SDF and r_c the collision check radius. The obstacle gradient is projected along the gradient of the SDF so that the trajectory is pushed away from all obstacles.

We perform the gradient update step (4) until no collision checkpoint centers are inside obstacles (although collisions may exist inside r_c). To enforce *Contact Reachability*, we use a series of limits on $\xi(t)$. If the trajectory violates these limits, we calculate the violation trajectory ξ_v, that is, the set of vectors representing the amount of violation for each pose in the trajectory. We then calculate a new trajectory without violations $\hat{\xi} = \xi + A^{-1}\xi_v$. This is explained in detail in Zucker et al. (2013).

3.3 | Foothold planning

In the previous sections, we were concerned with planning a trajectory of body poses, here we discuss how for each pose in the trajectory, the foothold $_I t_{IF}$ is computed. In contrast to body poses, we only plan for the next desired foothold rather than the full trajectory from start to goal pose. As the robot moves, more of the terrain will become visible and mapped. By planning a trajectory of body poses, we can look ahead and replan paths around obstacles. Since the robot moves relatively slowly, it would be unnecessary to plan more than a few footsteps, so we save on processing by planning only one step in advance.

The robot has predefined nominal feet positions, $\hat{t}_{F_i}$. We use a fixed, statically stable gait: right-hind (RH), right-fore (RF), left-hind (LH), and left-fore (LF). With these parameters and a given body pose in the trajectory, we can compute the next ideal foothold geometrically. We then search in the elevation map around this nominal foothold and

evaluate foothold scores based on the terrain. The nominal foothold is updated with the foothold with the best score that is also kinematically feasible. The swing trajectory is initialized with knot points that are then adapted to avoid collisions. More about this process can be read in Fankhauser et al. (2018) as we use the same method.

3.4 | Pose optimization

The nominal next body pose $\hat{T}_{IB}$ comes from the trajectory of body poses computed in Section 3.2.2. From this pose, the terrain-aware next footholds $_I t_{IF_i}$ were selected from the elevation map, and now a final optimization step is done to adjust the body pose to ensure the *Stability* constraint is not violated. We formulate this as an NLP problem, which we solve with an SQP solver. This method is sensitive to initialization, so to avoid falling into a local minimum, we first make a guess based on geometric calculations. We initialize the nominal body rotation $\hat{R}_{IB}$ and height component of nominal position $_I \hat{t}_{IB}$. The footholds $_I t_{IF_i}$ are fixed. We compute the planar projection of the centroid of the SP $\bar{t}_{SP}$ in the XY plane and set this as the XY components of the base pose.

Next, we seek to minimize the following optimization objective while considering the constraints:

$$\underset{_I t_{IB}, R_{IB}}{\text{minimize}} \quad f(x) = \mathcal{R}(_I t_{IB}, R_{IB}) \text{ Reachability}$$
$$+ C(_I t_{IB}, R_{IB}) \text{ Center of Mass} \quad (6)$$
$$+ \mathcal{H}(_I t_{IB}) \text{ Height},$$

$$\text{such that} \quad A_{SP} t_{COM}(_I t_{IB}, R_{IB}) \leq b_{SP}, \quad (7)$$
$$h_{\min} \leq h_i(_I t_{IB}, R_{IB}) \leq h_{\max,i} \; \forall \; i \in N.$$

where t_{IB} and R_{IB} are the final body position and rotation, respectively. Table 2 gives the equations and a description for each

TABLE 2 Optimization objective costs

Cost	Equation	Description
Reachability $\mathcal{R}$	$\sum_{n=1}^{N} \| _I\hat{t}_{F_i} - _I t_{F_i}(_I t_{IB}, R_{IB}) \|_2^2$	Cost on difference between default foot positions $_I\hat{t}_{F_i}$ and foot positions resulting from the optimized pose
Center of Mass C	$w_{COM} \| _I\bar{t}_{SP} - _I\bar{t}_{COM}(_I t_{IB}, R_{IB}) \|_2^2$	Cost on difference between SP centroid and COM planar projections
Height $\mathcal{H}$	$\|S_z^T(_I\hat{t}_{IB} - _I t_{IB})\|_2^2$	Cost on difference between desired height and optimized pose height

Abbreviations: COM, Center of Mass; SP, Support Polygon.

term in (6). The first component $\mathcal{R}(_I t_{IB}, R_{IB})$ is a reachability cost which penalizes body poses $(_I t_{IB}, R_{IB})$ that result in footholds t_{IF_i} which are far from the nominal footholds $\hat{t}_{IF_i}$. The next term, $C(_I t_{IB}, R_{IB})$, places a cost on instability by penalizing the difference between the SP centroid planar projection $\bar{t}_{SP}$ and the projection of the robot's COM onto the XY plane $\bar{t}_{COM}$. w_{COM} is a weight on the cost and acts as a safety margin. $\mathcal{H}(_I t_{IB})$ penalizes a difference between optimized height and desired height. S_z is selection matrix $[0\ 0\ 1]$ so that only the height component of the position difference is considered.

The constraint (7) enforces static stability where A_{SP} and b_{SP} are the linear constraints which limit the COM to lie inside the SP. We also encode the *reachability constraint* by constraining the body height from a given pose $h_i(t_{IB}, R_{IB})$ to lie between minimum and maximum values h_{min} and h_{max}.

4 | MAPPING AND COLLISION CHECKING

Our goal is to employ the posture planner online with a robot that is also mapping its surroundings. Not only do we need a method for modeling the environment and robot, but also a method to check for collisions. This section covers these concepts briefly as many of the methods we use here are similar to Buchanan et al. (2019) and Fankhauser et al. (2018).

4.1 | Multilevel elevation mapping

The robot's map consists of a 2D grid defined centered on the robot's body which follows it as it moves. On this grid two elevation levels are defined, one which represents the terrain below the robot and is denoted *floor* and another for obstacles above denoted *ceiling*. The elevations are computed with distance measurements from the robot's onboard sensor to nearby obstacles and terrain. For point measurement, the mean and variance $[\hat{h}, \hat{\sigma}_h^2]$ of the height measurement are updated in each cell by means of the Kalman filter

$$\hat{h}^+ = \frac{\sigma_p^2 \hat{h}^- + \hat{\sigma}_p^{2-} h_p}{\sigma_p^2 + \hat{\sigma}_h^{2-}}, \qquad \hat{\sigma}_h^{2+} = \frac{\hat{\sigma}_h^{2-} \sigma_p^2}{\sigma_p^2 + \hat{\sigma}_h^{2-}}, \qquad (8)$$

where − and + superscripts indicate the filter states before and after a measurement, respectively. The subscript p indicates the unfiltered sensor measurement variance which comes from empirical models, such as Fankhauser et al. (2015). Before fusion into the Kalman filter, points are separated into an elevation $E \in \{floor, ceiling\}$ with means and variances $[\hat{h}_f, \hat{\sigma}_f^2]$ and $[\hat{h}_c, \hat{\sigma}_c^2]$. This clustering is done by calculating the probability P of a new point observation $\hat{h}$ belonging to an elevation. The probability distribution of points around an elevation is modeled as $\mathcal{N}(\mu_E, \sigma_E)$ so that the likelihood function is given by

$$P(\hat{h}_p | E) = \frac{1}{\sqrt{2\pi\sigma_E^2}} e^{-\frac{(\hat{h}_p - \mu_E)^2}{2\sigma_E^2}}. \qquad (9)$$

In Figure 5 we show how this mapping method works in practice in narrow spaces.

4.2 | Collision checking in SDFs

To plan collision-free trajectories in 3D space, we convert our elevation maps to SDFs. Each voxel in an SDF contains the signed distance from that voxel to the nearest obstacle boundary where free space has positive values and space inside obstacles are negative (see Figure 2). Elevation mapping is done with cell sizes 2 cm across and the SDF is generated with 2 cm voxel size of at 1 Hz which is faster than Voxblox (Oleynikova et al., 2017) for the same voxel size.

We query the SDF at various checkpoints which are fixed on the robot. We select checkpoint locations depending on the robot's model composed of geometric primitives and specific placement strategies for boxes and cylinders. For example, in Figure 6, we show a box with points placed along each of the edges. There are gaps on the faces of the box where it is possible that an obstacle is not detected. Because our mapping representation includes all obstacles in either a *floor* or *ceiling* elevation, any obstacle which might pass in the face of the box would also collide with the top or bottom edge. This way, we can spend less computation checking the entire surface of the robot while still having reasonable assurance of detecting all collisions. We compute these checkpoints in the robot's body frame at start-up to save on processing during run time.
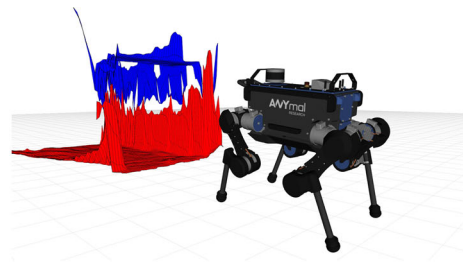
(a)

(b)

FIGURE 5 The left image shows ANYmal inspecting a narrow space and the right image is a rendering of the resulting elevation map. The red mesh represents the *floor* elevation and the blue mesh represents the *ceiling* [Color figure can be viewed at wileyonlinelibrary.com]

FIGURE 6 Diagram of geometric primitives and allocation of collision checkpoints. For boxes we place checkpoints along every edge (left), while for cylinders we also place points around edges as well as along columns at 90° intervals. The checkpoints are placed close enough such that their radii overlap [Color figure can be viewed at wileyonlinelibrary.com]
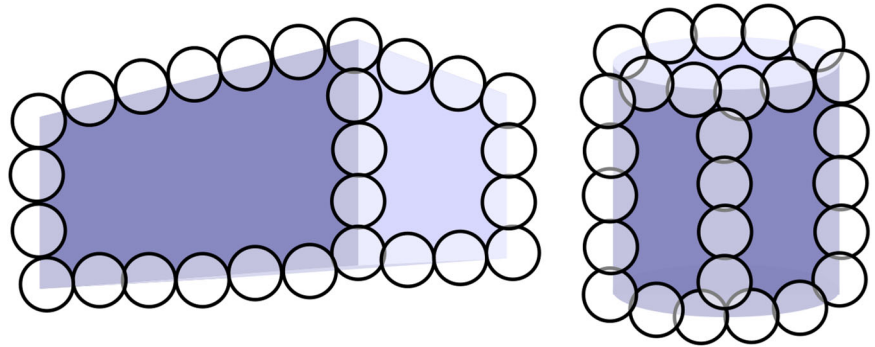
5 | IMPLEMENTATION

To perform experiments, we implemented our planner on the quadrupedal robot ANYmal (Hutter et al., 2016). We use an Intel Realsense D435 camera at a roll angle of 90° to perceive terrain above and below the robot. Pointclouds from the Realsense are processed on a dedicated PC with Intel i7-7600U processor at 3.5 GHz. The elevation maps are encoded using the GridMap (Fankhauser & Hutter, 2016) data structure and are passed to the second PC (which has similar specs) at 2 Hz over a wired Local Area Network (LAN) connection. Once a map is received, the SDF is computed in a separate thread using the approach presented in Felzenszwalb and Huttenlocher (2012) and saved in memory where it can be queried as needed.

Our planner starts when a new goal is received. In our experiments, the goal was always 3 m ahead of the robot. For the initial pose sampling, we use the RRT-Connect implementation in the Open Motion Planning Library (Şucan, Moll, & Kavraki, 2012). Searching is limited to 1 s, but in practice, it takes around 100 ms to find an exact solution. Table 6 in Section 7 shows the best, average, and worst-case times for each step in the planner. Once a successful pose trajectory has been found, we proceed to the smoothing step. In later planning iterations, we reuse the smoothed trajectory instead of sampling again, although we repeat the smoothing step. Resmoothing the previous trajectory instead of sampling each time speeds up replanning in case a dynamic obstacle enters the path, or a previously occluded obstacle enters view. If the smoothing fails, we re-initialize from new random samples. Once the footsteps have been selected and the final pose optimized, the motion is passed to the robot's controller as a Free Gait (Fankhauser et al., 2016) motion. ANYmal's whole-body controller then converts this motion into actuator commands, and the robot can carry out the motion.

6 | EXPERIMENTS

We performed three sets of experiments. The first was a series of simulated environments which explored the versatility of the planner. The next two sets of experiments were deployed on a real robot to verify our method's applicability. Online experiments were first performed in the lab using an adjustable door and then later in the field at a training facility for emergency responders. In lab experiments, we tested some of the same environments as in simulation. The purpose of the field experiments was to test the robustness of our method to realistic conditions.

6.1 | Simulated experiments

We created five different environments in simulation: *Low Gap*, *Rotated Gap*, *Low Gap with Step*, *Thin Gap*, and *Random Obstacles*. In each case we tested 10 trials with different dimensions or number of obstacles. Figure 7 shows the robot traversing three of these environments. It is worth noting that the Occupational Safety and Health Administration (OSHA) in the United States defines entry portals <24 in. (61 cm) in the smallest dimension as "Restricted" since any smaller would be impossible for a rescuer to enter with breathing equipment (OSHA, 2011).

1. *Low Gap*: In this experiment we progressively lowered the door in increments of 5 cm, each time having ANYmal pass through to the other side. We did this repeatedly until we reached a gap only 60 cm high and 1 m wide. We found the robot could pass under gaps of 70 cm and above with 100% success, 65 cm with 70% success, and 60 cm with 40%.
2. *Rotated Gap*: In the next environment, we created an opening in the shape of a 1-m high and 1-m wide isosceles right triangle. The purpose of this was to test navigating in differently shaped confined spaces. When initially tested in simulation, we did not randomly sample the initial trajectory and used a straight-line initialization instead. We found it was rare for CHOMP to solve a trajectory even when employing the Hamiltonian Monte Carlo method as in Zucker et al. (2013). Once we introduced an entirely randomly sampled trajectory as initialization, CHOMP converged much faster to better trajectories. In the simulations, the robot could pass through this gap with 100% success in 10 trials.
3. *Low Gap with Step*: This environment was created by lowering an overhanging obstacle to 70 cm, and raising a step in increments of 5 cm. In simulation the robot could step over 5 cm with 90% success and 60% for 10 cm.
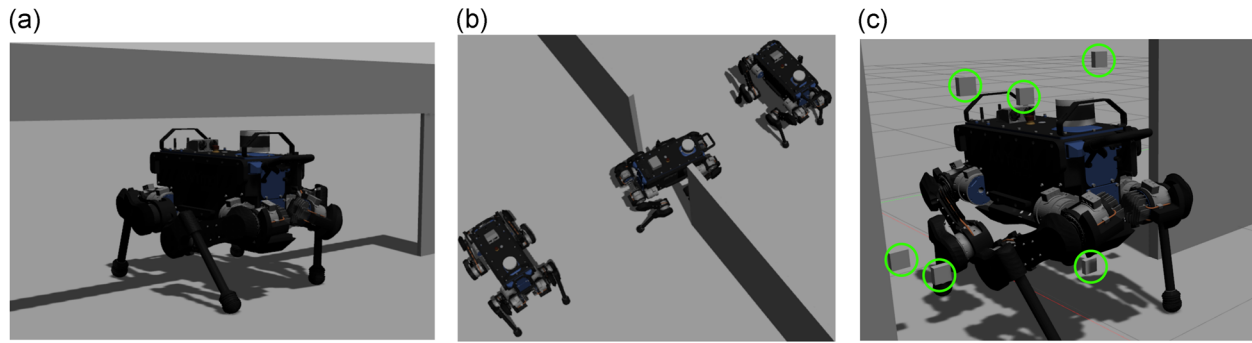
(a) (b) (c)

FIGURE 7 Images of robot navigating simulated environments. *Left*: Low Gap environment (60 cm). *Middle*: The Thin Gap environment (70 cm). *Right*: The Random environment (seven blocks). Here the randomly placed blocks are circled in green. One of the seven blocks is obscured by the robot [Color figure can be viewed at wileyonlinelibrary.com]

4. *Thin Gap*: For this environment we created a wall with a gap of variable width to the robot's left. We set a goal on the other side of the wall with the same orientation as the starting pose. To succeed, the robot needed to rotate counterclockwise 90°, pass straight through the gap then rotate back to the starting yaw rotation. Attempts to run our planner without the initial sampling would always fail as CHOMP could not converge to such a large rotation. We did this with increasingly narrow width and found the robot, which is 59 cm wide from knee to knee when standing, could pass through a gap 80 cm width with 100%, 75–65 cm 80% of the time, and 60 cm 20%.

5. *Random Obstacles*: Here we created a 1 m × 1 m gap in a wall in front of the robot. We then randomly spawned 2 cm × 5 cm × 5 cm blocks in the gap which were fixed in the air. This created randomized narrow spaces which could include obstacles above or below the robot. Sometimes the robot had to crawl under, step over or rotate its torso around obstacles. We performed 10 trials with 3 blocks, 5, 7, and 9 and found success rates of 70%, 50%, 40%, and 10%, respectively.

6.2 | Indoor experiments

We performed indoor experiments using a custom adjustable doorway. The space can be made narrower by sliding the plywood door downwards along aluminum profiles. The door can also be rotated, creating an angled gap and a metal bar can be raised to create a step. Figure 8 shows ANYmal navigating the door in each of the indoor experiments we performed.

To provide additional quantitative analysis, we recorded the physical dimensions of each door and examined the normalized deformation. This metric is useful for comparing different robots, which may have different physical constraints on minimum size. For example, in our first experiment, we progressively lowered the door, forcing ANYmal to crawl lower and lower. The lowest possible space that could be navigated was 60 cm, which corresponds to a non-normalized percent deformation of 25% as ANYmal has a nominal height of 80 cm. In Buchanan et al. (2019) we used the following

equation to normalize deformation percentages between different robots:

$$\text{deformation} = 100\% * \left(1 - \frac{\text{measured-limit}}{\text{nominal-limit}}\right), \quad (10)$$

which gives a normalized deformation of 43% as ANYmal has a minimum possible height of 34 cm, which is the height of the torso. This is near the 35% minimum deformation shown in Buchanan et al. with the hexapod robot Weaver. The disparity is likely due to different nominal configurations of the legs. The hexapod's legs spread up and outwards from the body in an insectoid fashion. ANYmal's legs were kept in a mammalian configuration during these experiments such that the feet were close under its body. While ANYmal can walk with an insectoid configuration, there is no automatic way in the controller to switch.

With these metrics, we replicated, indoors, three of the simulated experiments: *Low Gap*, *Rotated Gap*, and *Low Gap with Step*. In each case, we arranged the door to create the desired gap, then placed the robot in front and commanded a goal on the other side. All planning and perception were done on the robot without any prior knowledge. We recorded the robot kinematics to observe how it executed the plans.

6.2.1 | Low Gap

On the real robot, we found similar success rates as in simulation. Failures were always either caused by collision with the door or singularities in computing the inverse kinematics. As the robot lowers its torso significantly, the hip joints become very close to the feet. This reduces the manipulability of the legs and can lead to failure computing the inverse kinematics during a leg swing. Collisions with the door are likely due to the hierarchical nature of the planner. In the final optimization step, it is possible that the robot pose changes slightly. This can lead to a body pose which is in collision with the terrain. Additional collisions are due to parts of the robot we do not model, such as the leg below the hip joint.

The top plot of Figure 8 shows the plotted position and rotation over time as the robot crawled through the doorway. The torso of
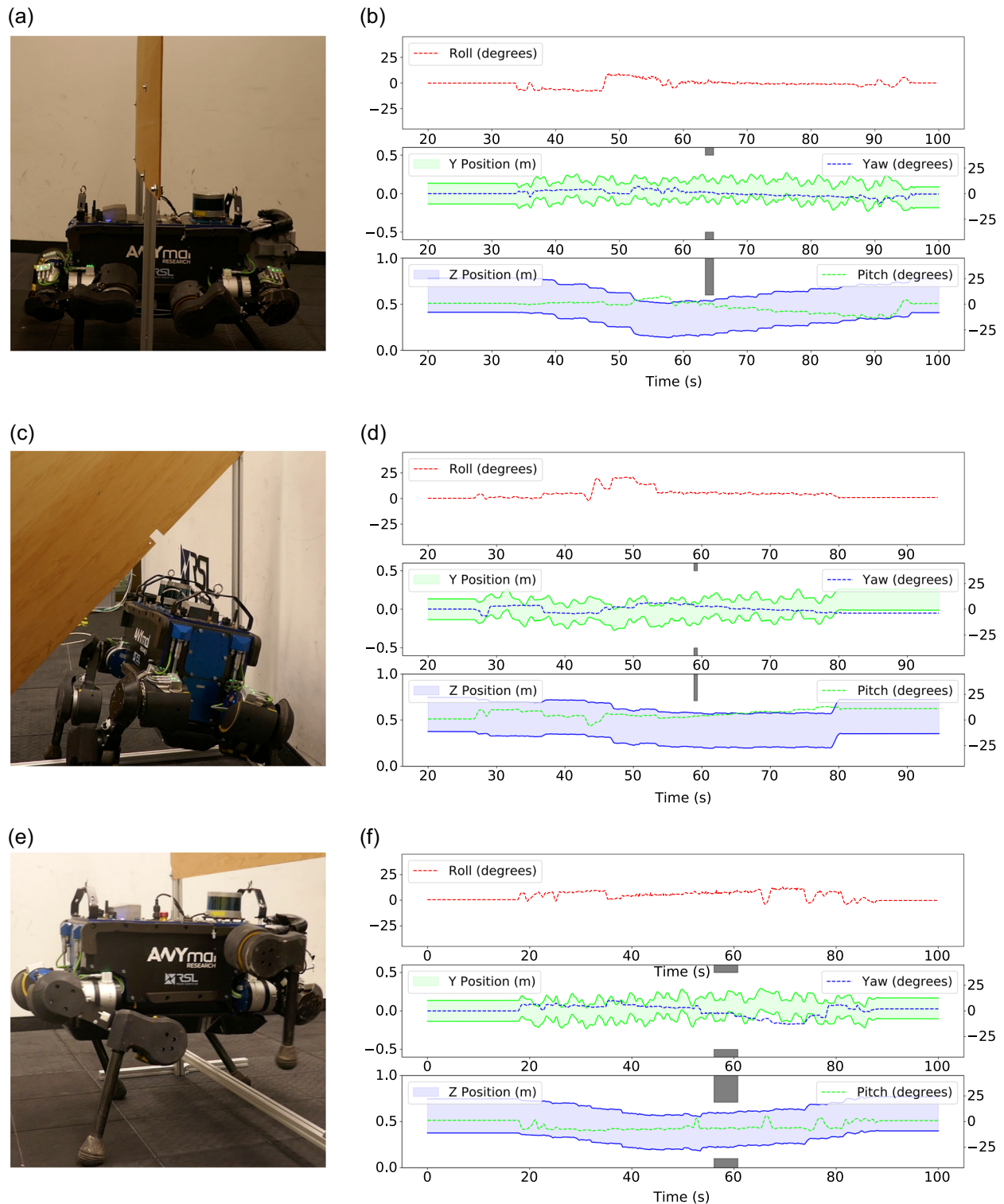
FIGURE 8 (Top) *Low Gap* (60 cm), (Middle) *Rotated Gap*, and (Bottom) *Low Gap with Step* (10 cm). On the left are images of ANYmal completing each experiment, and on the right are corresponding plots of odometry. In the plots on the right, position is represented with solid, filled regions while rotation is represented with dashed lines. Within each plot, the top subplot shows roll, the middle subplot shows *Y* position and yaw, and the bottom subplot shows *Z* position and pitch. Shaded, colored regions represent the collision region of the widest part of ANYmal's torso. The dark gray regions are indicative of obstacles. Their heights are based on physical dimensions while their width represents the time between the front of the robot entering the confined space and the rear leaving [Color figure can be viewed at wileyonlinelibrary.com]

ANYmal is not a single rectangle; there are handles at the front and back, making it thinner in the middle. Thus, the planner initially had ANYmal roll its body and pitch forwards to fit the front handle under the door. Then the rotations leveled out as the center of ANYmal's body passed through the doorway before pitching up to get the back handle through. This demonstrates the 6DOF planning we do for the body and allows the robot to minimize the amount of time spent crouched down.

6.2.2 | Rotated Gap

In experiments with the real robot, the planner succeeded in five out of seven trials, and we show the results from one of these trials in the middle plot of Figure 8. The planner optimized a combination of roll, pitch, and yaw to orient the body through the free space. This demonstrates how the optimization keeps the COM inside the SP as a more substantial roll would be less stable. Because we include this in our objective (6), the planner instead finds an optimal combination of rotations so that none is too extreme.

The planner guided ANYmal's position to the right to avoid the narrower part of the triangle on the left. This way, once the front of ANYmal passes through the gap, large rotations are no longer needed, and the body straightens out. We also tried to reduce the angle of the doorway to make the space even more narrow, but this quickly led to higher rates of failure. The main reason for failure was the top of the robot colliding with the door. It was found that our mapping method does not work well with angled walls. There are inherently gaps in the map where *floor* changes to *ceiling*, and on an angled wall, these unmapped spaces can be significant and are located directly in the center of the door. This gap was much more pronounced in the real experiments as opposed to simulation due to higher sensor noise. The result was the planner guiding the robot very close to the door and sometimes directly into collision.

6.2.3 | Low Gap with Step

The final indoor experiment we performed was the *Low Gap with Step*. ANYmal could step over 5 cm with 100% success which dropped to 50% at 10 cm. From the resulting plot at the bottom of Figure 8, we can see how the robot lowers its body to be below 70 cm. To overcome the higher step, the robot pitches upwards before the gap, which is the opposite behavior to the *Low Gap* experiment. This is because we maximize reachability allowing the front legs to swing up and over the step. Once the body has passed through the gap, the robot slowly pitches forwards and raises its body. The two spikes in pitch and roll after the doorway correspond to the two back legs stepping over the obstacle.

This lab experiment was the most challenging. The robot typically failed due to singularities, which is similar to the *Low Gap* experiment at 60 cm. In such a confined space, the manipulability of the legs is very low. We increased the default foot positions in the X and Y directions to increase the distance between the feet and hips, but it could not eliminate the problem.

6.3 | Weaver experiment: Rotated Gap

The robot would not have been able to walk through any of these confined spaces without the contributions outlined in this paper. To demonstrate this, we repeated the *Rotated Gap* experiment in simulation on the robot Weaver (Bjelonic et al., 2018). With our previous planner (Buchanan et al., 2019), the robot could not find a path and continuously collided with the obstacles. This is because it did not use an initially sampled trajectory to guide the torso, which resulted in the smoothing process diverging or becoming stuck in a local minimum. Additionally, without planning for the torso's roll and with the large collision abstraction of the whole body, it made it more difficult to find a path. We show an image from these simulations in Figure 9. It would have also been impossible for Weaver to complete the *Low Gap with Step* experiment as previously we did not plan footsteps. This shows the versatility of our method as we have demonstrated the out planner on two types of multilegged robots.

6.4 | Field experiments: Wangen an der Aare Training Village

The Swiss Federal Office for Defence Procurement *armasuisse* organizes the Advanced Robotic Capabilities for Hazardous Environments (ARCHE) event. The event takes place at the *Übungsdorf* (Training Village) in the town of Wangen an der Aare in Switzerland. This facility is a training site for the rescue and ordnance disposal units of the Swiss military.

We visited the Training Village in July 2019, and over the course of a week, found several confined spaces to test our planner. We show here our results from ANYmal navigating three different confined spaces that firefighters could be expected to crawl through as part of their training. In each case, we placed ANYmal in front of the opening and only commanded the robot to go forwards. All perception and planning were done onboard the robot, which explored completely autonomously and untethered.

6.4.1 | Rectangular Gap

The first confined space we tested in the Training Village was a rectangular opening 80 cm high and 70 cm wide, which leads into the side of a burned concrete building. It also included a 10-cm step, which required ANYmal to climb up into the small space. The top of Figure 10 shows ANYmal climbing in this space. To allow ANYmal to enter, we opened a small metal door and cleared some vegetation. As ANYmal is nominally 80 cm high, it would typically barely fit inside
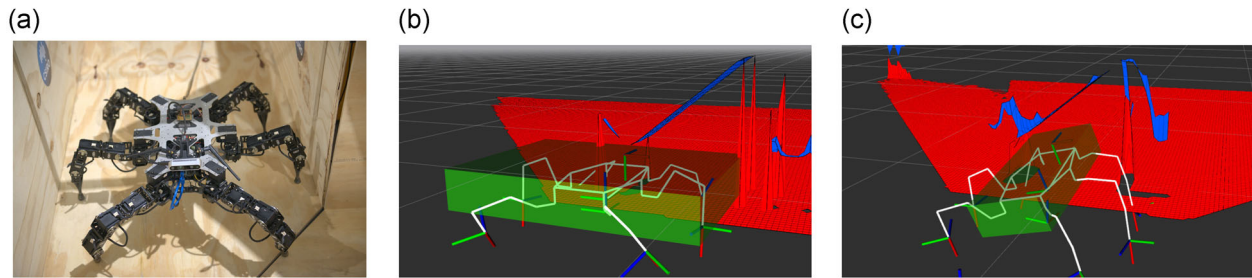
FIGURE 9 (Left) The hexapod robot Weaver, (Middle) *Rotated Gap* experiment using our previous work, and (Right) The same environment using the methods in this paper. We did not plan footsteps for Weaver as the controller did not support this feature. In white, we show the skeleton of the robot's model and in green we show the collision model. The red mesh represents the *floor* elevation and the blue mesh represents the *ceiling* [Color figure can be viewed at wileyonlinelibrary.com]

this space. However, our planner had the robot lower its torso so that it could enter the house and map the interior.

Also in Figure 10, we show the position and rotation of the robot's base. ANYmal did not collide with the walls and was able to enter this space twice. In each case, however, the mapping failed for a part of the ceiling inside the building. The result was that the planner raised the body slightly too early and came very close to the wall. This problem is likely due to a difference in lighting between outside (bright sunlight) and inside (unlit darkness) the building. We have found that a significant contrast in lighting can negatively affect the Realsense camera.

6.4.2 | Crumbling Wall

The next task we attempted was to have ANYmal walk along a Crumbling Wall. As shown in the middle of Figure 10, there are several large slabs of concrete that lean against the wall creating triangular holes. The ground was flat, and the area was less confined than the *Rectangular Gap*; however, large pieces of concrete blocked the robot's path, which forced it to walk around the obstacles. The first obstacle on ANYmal's left was barely in the field of view of the camera, and so was not mapped very effectively, which led to the robot walking very close. The overhanging obstacle is quite high and close to the wall, so it was not a significant issue for ANYmal to fit inside.

6.4.3 | Collapsed Building

The final experiment we attempted was to enter a pile of rubble created by a building collapse, as shown at the bottom of Figure 10. This space was highly constrained due to the ledge on the robot's left and the large curved piece of concrete on the right. There was also a 5-cm depression in the floor. As shown from the odometry in Figure 10, when it reached the depression, it rotated significantly to reach the feet down. It rotated again to reach the feet back up while crawling under the curved concrete. Inside, it came very near to the ceiling, scraping against it several times.

Because of the very confined space, our planner initially failed to find a body trajectory until we reduced the collision radius r_c

from 5 to 2 cm. This change made it possible to plan trajectories, but this likely resulted in the robot scraping against the concrete. The robot was able to continue walking but, because of the statically stable gait, if the contact had been worse, the robot would have fallen over.

7 | RESULTS AND ANALYSIS

Our experiments represent a wide variety of narrow spaces a robot could be expected to walk through. We tested our method in five different simulated environments including with randomized obstacles. We then replicated three of these experiments in trials on the real robot. A summary of the simulation only experiments is given in Table 4, while experiments performed both in simulation and in the lab are summarized in Table 5. Success rates on the real robot matched the simulated results very closely, although the *Rotated Gap* had more failures on the real robot due to problems with mapping). In Table 3 we show the minimum and maximum values of the robot torso height and rotation. For height, we give the measurement of the very top of the robot's torso. Finally, we also performed three different field experiments[2] in which ANYmal was successfully able to crawl inside confined spaces at a training facility for firefighters.

We additionally recorded the timing of each step in our planner during the lab experiments. Table 6 shows the best, average, and worst time that it took for each of the planning steps to be executed. These times were recorded on ANYmal directly as it walked through each confined space. Initial sampling and smoothing are by far the most time-consuming steps in the planning pipeline; however, timing typically improves as the robot walks since we reuse the trajectory and repeat the smoothing. Total planner time excludes SDF generation, which is computed in a separate thread. The average planning time for all experiments was <500 ms.

---

[2]A video showing ANYmal performing all of the online experiments discussed in this paper is available at https://youtu.be/C2e0JTdwid0 and is included in the online version of this article. See the appendix.
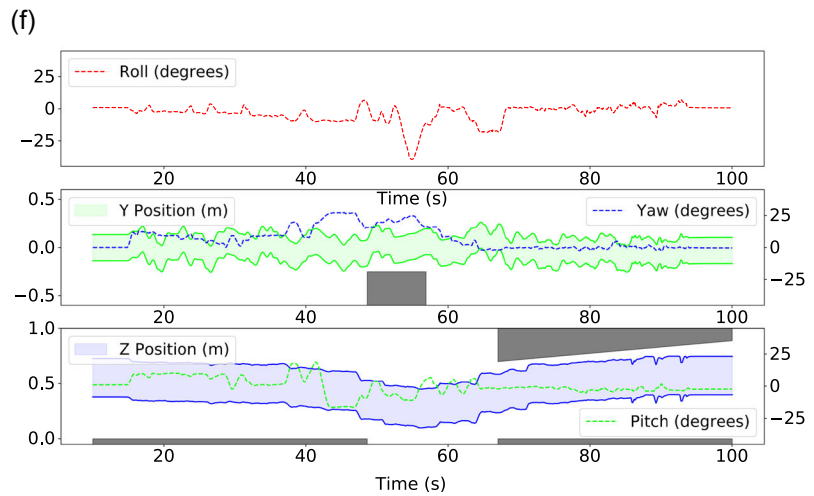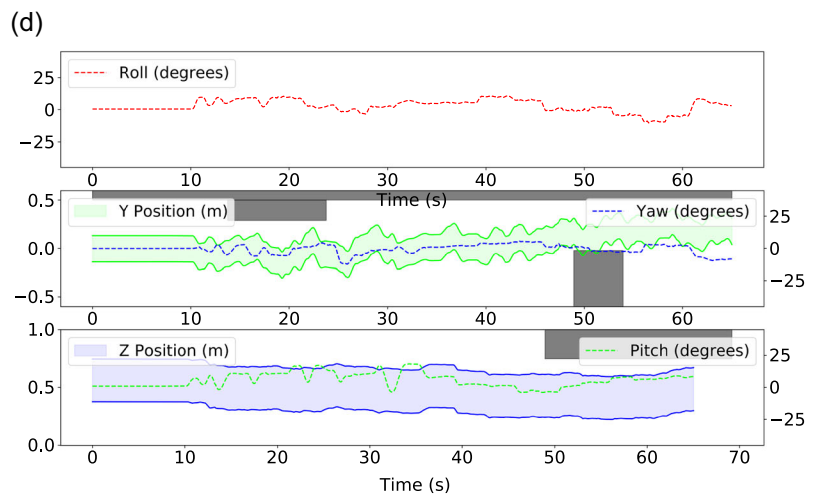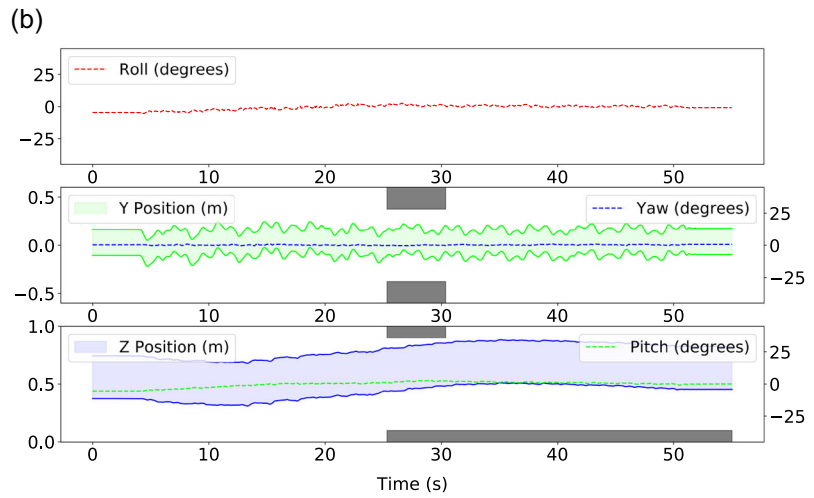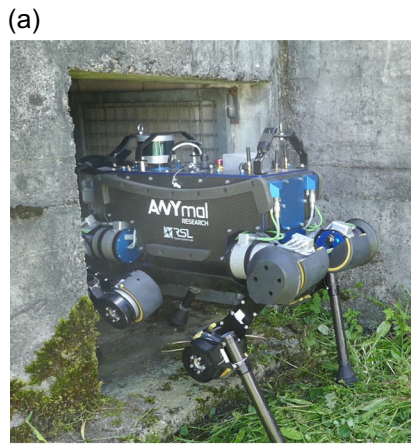
FIGURE 10 (Top) *Rectangular Gap*, (Middle) *Crumbling Wall*, and (Bottom) *Collapsed Building*. On the left are images of ANYmal completing each experiment, and on the right are corresponding plots of odometry. In the plots on the right, position is represented with solid, filled regions while rotation is represented with dashed lines. Within each plot, the top subplot shows roll, the middle subplot shows Y position and yaw, and the bottom subplot shows Z position and pitch. Shaded, colored regions represent the collision region of the widest part of ANYmal's torso. The dark gray regions are indicative of obstacles. Their heights are based on physical dimensions while their width represents the time between the front of the robot entering the confined space and the rear leaving [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 3 Min and max posture adaptations achieved by robot in each real experiment

	Height (cm)		Roll (deg.)		Pitch (deg.)		Yaw (deg.)	
	Min	Max	Min	Max	Min	Max	Min	Max
Low Gap (60 cm)	50.9	78.3	−8.12	9.05	−13.57	7.87	−9.8	7.24
Rotated Gap	56.4	74.6	−2.54	20.52	−5.63	13.34	−6.70	6.33
Gap with Step (10 cm)	55.2	77.4	−4.94	12.12	−9.14	5.91	−12.97	10.24
Rectangular Gap	68.0	88.3	−5.28	2.80	−5.40	2.87	−0.46	1.40
Crumbling Wall	59.2	74.8	−10.47	10.56	−3.80	18.35	12.26	5.66
Collapsed Building	46.8	76.9	−40.01	6.30	−16.43	19.02	−2.77	27.25

TABLE 4 Successful trials for simulated only experiments

	Thin Gap (80 cm)	Thin Gap (75–65 cm)	Thin Gap (60 cm)	Random (3 blocks)	Random (5 blocks)	Random (7 blocks)	Random (9 blocks)
Simulation	10/10	8/10	2/10	7/10	5/10	4/10	1/10

TABLE 5 Successful trials for simulated and lab experiments

	Low Gap (>70 cm)	Low Gap (65 cm)	Low Gap (60 cm)	Rotated Gap	Step (5 cm)	Step (10 cm)
Simulation	10/10	7/10	4/10	10/10	9/10	6/10
Real Robot	3/3	5/7	2/4	5/7	4/4	2/4

TABLE 6 Times for each lab experiment as run on ANYmal's hardware

	Low Gap (65 cm)	Rotated Gap	Gap with Step (10 cm)
SDF generation	410–**600**–970	380–**578**–1335	419–**524**–761
Initial sampling	6E − 04–**1**–78	7E − 04–**132**–1095	8E − 04–**83**–1060
Smoothing	2–**24**–298	2–**168**–3163	4–**84**–1503
Foothold and posture optimization	0.1–**0.5**–1	0.1–**0.5**–4	0.1–**0.4**–1
Swing trajectory	0.03–**0.3**–1	0.02–**0.4**–2	0.2–**0.4**–1
Total planner	4–**125**–885	4–**411**–4773	5–**261**–1641

Note: All times are given in milliseconds and organized as: Best–**Average**–Worst.

8 | CONCLUSION

Confined spaces are extremely hazardous for humans, while legged robots have the capabilities to be deployed instead. For legged robots to enter these spaces, they need the ability to perceive and adapt their posture in complex 3D environments. This is especially true for confined spaces in which a robot must crawl under obstacles or rotate its body in a narrow space. The objective of this study was to create a fast and reliable planner that allows legged robots to navigate confined spaces. We do this with a hierarchical planner that first samples a random trajectory of body poses, which is then smoothed using the local optimization

algorithm CHOMP. Nominal foot positions are updated based on terrain scores obtained from elevation mapping. Finally, a joint posture optimization is done to ensure the robot can safely carry out the planned trajectory.

We performed various experiments in the lab to test the limits of our planner. First, in simulation, we compared our method with our previous work and demonstrated how Weaver, a hexapod could traverse new spaces more complex spaces. We then deployed our planner on a real quadrupedal robot. ANYmal, which is ordinarily 80 cm tall, was able to crawl through a gap only 60 cm high, which is defined by OSHA as "Restricted" and would be considered too dangerous for humans. We also demonstrated how the robot rotates its

body in irregularly shaped gaps and can step over obstacles. We conducted several field experiments at a training facility for search and rescuers. Our planner enabled the robot to crawl inside of realistic small spaces where firefighters train. Our method is one of the few capable of planning for the whole body of the robot online while avoiding body collisions with the environment and is the only one to be demonstrated in realistic field tests. Our work is a valuable step towards enabling robots to operate in all kinds of environments outside the lab.

9 | FUTURE WORK

Some important lessons were learned from the deployment of our planner on a real robot. While our method provides a useful solution to whole-body planning for multilegged robots, it relies significantly on a statically stable gait. If there are errors in mapping or planning, the robot can collide with the terrain as it did in the *Collapsed Building*. However, if we were to incorporate dynamics into our optimization, our planner would be more robust to these disturbances. This would, of course, lead to more computational requirements and require planning times fast enough to react.

A truly 3D mapping approach that does not cluster into floor and ceiling could reduce map artifacts and, therefore, lower failure rates. This would lead to improvements in walking around sharp objects or through angled gaps, such as the *Rotated Gap*. Future work in this area should focus on finding the right trade-off between map detail and computation complexity.

ORCID

Russell Buchanan http://orcid.org/0000-0001-9172-5856

REFERENCES

Aceituno-Cabezas, B., Mastalli, C., Dai, H., Focchi, M., Radulescu, A., Caldwell, D. G., … Semini, C. (2018). Simultaneous contact, gait, and motion planning for robust multilegged locomotion via mixed-integer convex optimization. *IEEE Robotics and Automation Letters*, 3(3), 2531–2538.

Belter, D., Łabecki, P., Fankhauser, P., & Siegwart, R. (2016). RGB-D terrain perception and dense mapping for legged robots. *International Journal of Applied Mathematics and Computer Science*, 26(1), 81–97.

Bjelonic, M., Kottege, N., Homberger, T., Borges, P., Beckerle, P., & Chli, M. (2018). Weaver: Hexapod robot for autonomous navigation on unstructured terrain. *Journal of Field Robotics*, 35(7), 1063–1079.

Buchanan, R., Bandyopadhyay, T., Bjelonic, M., Wellhausen, L., Hutter, M., & Kottege, N. (2019). Walking posture adaptation for legged robot navigation in confined spaces. *IEEE Robotics and Automation Letters*, 4(2), 2148–2155.

Fankhauser, P., Bellicoso, C. D., Gehring, C., Dubé, R., Gawel, A., & Hutter, M. (2016). Free gait—An architecture for the versatile control of legged robots. In *IEEE-RAS International Conference on Humanoid Robots*. (pp. 1052–1105). Cancun, Mexico.

Fankhauser, P., Bjelonic, M., Bellicoso, C. D., Miki, T., & Hutter, M. (2018). Robust rough-terrain locomotion with a quadrupedal robot. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 5761–5768). Brisbane, QLD, Australia.

Fankhauser, P., Bloesch, M., & Hutter, M. (2018). Probabilistic terrain mapping for mobile robots with uncertain localization. *IEEE Robotics and Automation Letters*, 3(4), 3019–3026.

Fankhauser, P., Bloesch, M., Rodriguez, D., Kaestner, R., Hutter, M., & Siegwart, R. (2015). Kinect v2 for mobile robot navigation: Evaluation and modeling. In *International Conference on Advanced Robotics (ICAR)* (pp. 388–394). Istanbul, Turkey.

Fankhauser, P., & Hutter, M. (2016). A universal grid map library: Implementation and use case for rough terrain navigation. In A. Koubaa (Ed.), *Robot operating system (ROS)—The complete reference* (Vol. 1). Cham, Switzerland: Springer.

Felzenszwalb, P. F., & Huttenlocher, D. P. (2012). Distance transforms of sampled functions. *Theory of Computing*, 8(1), 415–428.

Geisert, M., Yates, T., Orgen, A., Fernbach, P., & Havoutis, I. (2019). Contact planning for the ANYmal quadruped robot using an acyclic reachability-based planner. In K. Althoefer, J. Konstantinova, K. Zhang (Eds.), *Towards Autonomous Robotic Systems Conference (TAROS)* (pp. 275–287). Cham, Switzerland: Springer.

Grey, M. X., Ames, A. D., & Liu, C. K. (2017). Footstep and motion planning in semi-unstructured environments using randomized possibility graphs. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 4747–4753). Singapore.

Herbert, M., Caillas, C., Krotkov, E., Kweon, I. S., & Kanade, T. (1989). Terrain mapping for a roving planetary explorer. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 997–1002). Scottsdale, AZ.

Hornung, A., Wurm, K. M., Bennewitz, M., Stachniss, C., & Burgard, W. (2013). OctoMap: An efficient probabilistic 3D mapping framework based on octrees. *Autonomous Robots*, 34(3), 189–206.

Hutter, M., Gehring, C., Jud, D., Lauber, A., Bellicoso, C. D., Tsounis, V., … Hoepinger, M. (2016). ANYmal—A highly mobile and dynamic quadrupedal robot. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* (pp. 38–44). Daejeon, South Korea.

Karaman, S., & Frazzoli, E. (2011). Sampling-based algorithms for optimal motion planning. *The International Journal of Robotics Research*, 30(7), 846–894.

Kuffner, J. J., & LaValle, S. M. (2000). RRT-connect: An efficient approach to single-query path planning. In *IEEE International Conference on Robotics and Automation (ICRA)* (Vol. 2, pp. 995–1001). San Francisco, CA.

Kumagai, I., Morisawa, M., Nakaoka, S., & Kanehiro, F. (2018). Efficient locomotion planning for a humanoid robot with whole-body collision avoidance guided by footsteps and centroidal sway motion. In *IEEE-RAS International Conference on Humanoid Robots (Humanoids)* (pp. 251–256). Beijing, China.

Lavalle, S. M. (1998). *Rapidly-exploring random trees: A new tool for path planning* (Technical Reports No. 98-11). Ames, IA: Computer Science Department, Iowa State University.

Magaña, O. A. V., Barasuol, V., Camurri, M., Franceschi, L., Focchi, M., Pontil, M., … Semini, C. (2019). Fast and continuous foothold

adaptation for dynamic locomotion through CNNs. *IEEE Robotics and Automation Letters, 4*(2), 2140–2147.

Mastalli, C., Focchi, M., Havoutis, I., Radulescu, A., Calinon, S., Buchli, J., ... Semini, C. (2017). Trajectory and foothold optimization using low-dimensional models for rough terrain locomotion. In *IEEE International Conference on Robotics and Automation (ICRA)* (pp. 1096–1103). Singapore, Singapore.

Oleynikova, H., Taylor, Z., Fehr, M., Siegwart, R., & Nieto, J. (2017). Voxblox: Incremental 3D Euclidean signed distance fields for on-board MAV planning. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. (pp. 1366–1373). Vancouver, BC, Canada.

OSHA. (2011). *Occupational safety and health standards: General environmental controls: Permit-required confined spaces* (Standard No. 1910.146). Retrieved from https://www.osha.gov/laws-regs/regulations/standard number/1910/1910.146 (Accessed on 2019-07-08).

Schulman, J., Duan, Y., Ho, J., Lee, A., Awwal, I., Bradlow, H., ... Abbeel, P. (2014). Motion planning with sequential convex optimization and convex collision checking. *The International Journal of Robotics Research, 33*(9), 1251–1270.

Short, A., & Bandyopadhyay, T. (2018). Legged motion planning in complex three-dimensional environments. *IEEE Robotics and Automation Letters, 3*(1), 29–36.

Şucan, I. A., Moll, M., & Kavraki, L. E. (2012). The open motion planning library. *IEEE Robotics & Automation Magazine, 19*(4), 72–82. Retrieved from http://ompl.kavrakilab.org

Tonneau, S., Prete, A. D., Pettré, J., Park, C., Manocha, D., & Mansard, N. (2018). An efficient acyclic contact planner for multiped robots. *IEEE Transactions on Robotics, 34*(3), 586–601.

Winkler, A. W., Bellicoso, C. D., Hutter, M., & Buchli, J. (2018). Gait and trajectory optimization for legged systems through phase-based end-effector parameterization. *IEEE Robotics and Automation Letters, 3*(3), 1560–1567.

Zucker, M., Ratliff, N., Dragan, A., Pivtoraiko, M., Klingensmith, M., Dellin, C., ... Srinivasa, S. (2013). CHOMP: Covariant Hamiltonian optimization for motion planning. *The International Journal of Robotics Research, 32*(9–10), 1164–1193.

SUPPORTING INFORMATION

Additional supporting information may be found online in the Supporting Information section.

APPENDIX A: INDEX TO MULTIMEDIA EXTENSIONS

Table A1 describes the video included in the online version of this article.

TABLE A1 Multimedia extensions

Extension	Media type	Description
1	Video	Video summary of paper and demonstrations of field experiments