# Legged Motion Planning in Complex Three-Dimensional Environments

Andrew Short
*University of Wollongong*, ajs875@uowmail.edu.au

Tirthankar Bandyopadhyay
*CSIRO*

## Abstract

While legged robots are well suited to navigating complex three-dimensional (3-D) environments, their practical applicability is still hampered by the time taken to plan manoeuvres to traverse these challenging environments. We present contact dynamic roadmaps (CDRM), which extend dynamic roadmaps with contact information. The CDRM is precomputed offline to generate a discretized mapping from each leg's workspace to its configuration space, and then adapted online to the environment to rapidly identify collision-free foothold positions. The concept behind this is to perform the expensive foothold candidate generation and collision checking phases offline and store the data for use in the online planner. The CDRM is coupled with a rapidly-exploring random tree planner to generate acyclic full-body motion plans in complex 3-D environments. The performance of the approach is validated and compared in simulation in a wide variety of scenarios that require full-body planning to successfully navigate.

## Disciplines

Engineering | Science and Technology Studies

## Publication Details

Short, A. & Bandyopadhyay, T. (2018). Legged Motion Planning in Complex Three-Dimensional Environments. IEEE Robotics And Automation Letters, 3 (1), 29-36.

# Legged Motion Planning in Complex Three-Dimensional Environments

Andrew Short[1] and Tirthankar Bandyopadhyay[2]

***Abstract*—While legged robots are well suited to navigating complex 3D environments, their practical applicability is still hampered by the time taken to plan manoeuvres to traverse these challenging environments. We present Contact Dynamic Roadmaps (CDRM), which extend Dynamic Roadmaps (DRM) with contact information. The CDRM is pre-computed offline to generate a discretised mapping from each leg's workspace to its configuration space, and then adapted online to the environment to rapidly identify collision-free foothold positions. The concept behind this is to perform the expensive foothold candidate generation and collision checking phases offline and store the data for use in the online planner. The CDRM is coupled with a Rapidly-exploring Random Tree (RRT) planner to generate acyclic full-body motion plans in complex 3D environments. The performance of the approach is validated and compared in simulation in a wide variety of scenarios that require full-body planning to successfully navigate.**

***Index Terms*—Legged Robots; Motion and Path Planning**

## I. Introduction

LEGGED robots enable navigation in challenging environments where other platforms, such as wheeled or tracked vehicles, have had limited success. The flexibility of legged robots come from their high Degrees of Freedom (DOF), at the cost of computational complexity in planning and control.

Full-body planning for such high DOF robots is challenging as the set of feasible configurations that enable stable locomotion lie in a very small subspace of the whole configuration space. There have been numerous approaches toward generating a solution, such as a rigorous complete algorithm [1], or decomposing the problem into a search between underlying lower dimensional manifolds [2]. However, these approaches require significant time in computation and can only be applied to a priori known environments. When the environment changes, the full motion plan has to be recomputed, which makes these approaches unsuitable for many practical applications.

Manuscript received: February, 15, 2017; Revised May, 13, 2017; Accepted June, 11, 2017.

This paper was recommended for publication by Editor Nancy Amato upon evaluation of the Associate Editor and Reviewers' comments. Andrew Short is supported by an Australian Government Research Training Program Scholarship, the University of Wollongong Global Challenges program and the ARC Research Training Centre for Naval Design and Manufacturing (ARC IC140100003).

[1]Andrew Short is with the School of Mechanical, Materials, Mechatronic and Biomedical Engineering, University of Wollongong, Australia `ajs875@uowmail.edu.au`

[2]Tirthankar Bandyopadhyay is with Autonomous Systems, Data61 Commonwealth Scientific and Industrial Research Organisation (CSIRO), Australia `tirtha.bandy@csiro.au`

Digital Object Identifier (DOI): see top of this page.

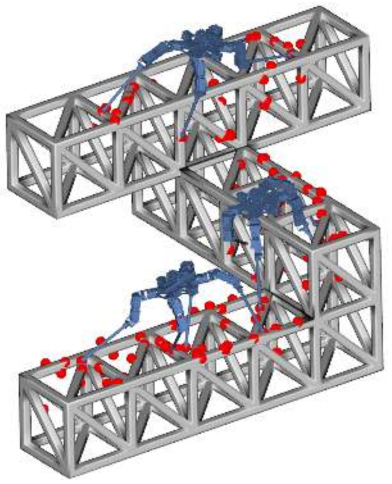


Fig. 1: A quadruped robot navigating a complex 3D structure.

In environments that are unknown a priori or only partially known, a popular approach in legged robot navigation is to generate cyclic gaits and adapt gait parameters (e.g. stride length and stride height) in a reactive manner [3]. This has achieved significant success due to its simplicity and ease of implementation on real systems. However, this only succeeds in situations that have limited complexity and often fails in more challenging environments which require complicated full-body manoeuvres. In this paper, we present a full-body planning algorithm for legged robots capable of navigating such complex environments as shown in Fig. 1.

Similar to earlier approaches [4] we decompose the problem of robot motion planning into body planning and planning of individual legs to enable stability and reachability. The use of the pre-computed Contact Dynamic Roadmap (CDRM) structure to identify footholds for individual legs is the crux of our approach. While previous work solves the problem sequentially, we explicitly determine the existence of valid footholds and stable configurations before accepting a body pose. This precludes conditions where the robot plans would have to be recomputed when the feasibility hypothesis fails.

We use a sampling based approach to generate a motion plan for the robot. Sampling-based planners are an effective tool for high DOF motion planning [5]. For multi-query problems in a fixed environment, Probabilistic Roadmaps (PRMs) [6] are well suited as the roadmap can be pre-computed offline, leading to an online cost of only performing a graph search. However when the environment changes, the roadmap has to be discarded or modified. For mobile robots, dynamic and unknown environments are common and hence single query

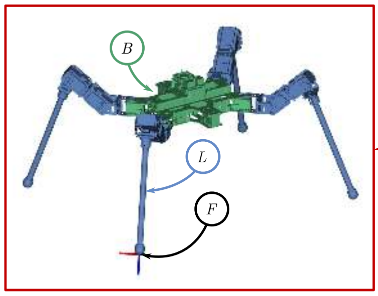


Fig. 2: A quadruped robot labelled with the notation used

planners such as Rapidly exploring Random Trees (RRTs) [7] are generally used. While these planners are suitable for high DOF systems, computational times for full-body planning are still high for online planning.

Dynamic Roadmaps (DRM) [8] are a good compromise, where a workspace to configuration space mapping is generated offline by associating a cell of the workspace to nodes of a PRM in the configuration space. However, existing DRM based planning approaches have only been applied to fixed based platforms. This paper presents modifications required to enable DRMs to be applied to mobile legged robots. Instead of using DRMs for a globally changing environment, we instead transform the origin of the DRM with the position of the mobile robot. This effectively results in the environment, as seen from the origin of the DRM, changing as the robot moves. In contrast to planners such as Obstacle Based PRM [9], environment information isn't used during roadmap construction, so the expensive DRM generation can be done offline once per robot and efficiently re-used in different environments. The PRM connectivity information is updated using the DRM, in effect using pre-computed offline collision checks, enabling fast online planning as shown in recent works [10]–[12].

While promising, basic DRMs cannot be directly applied to grasping, manipulation or navigation of legged platforms, as these tasks require explicit contact with the environment and DRMs treat these contact points as collisions. To specifically address this issue, we introduce Contact DRMs (CDRMs) that in addition to maintaining workspace-configuration space mapping also map potential end-effector or foot-tip positions of the leg or manipulator arm.

## II. Problem Statement

A legged robot $R$ operates in a known workspace $\mathcal{W} = \mathcal{R}^3$, with the region $\mathcal{O} \subseteq \mathcal{W}$ being occupied by obstacles. An example quadruped robot, along with the notation used, is shown in Fig. 2. The robot is composed of a robot body $B$ to which $n$ legs are attached. The body $B$ is free to translate and rotate in $SE(3)$. A full-body robot configuration is denoted by $q_r$ with the robot's configuration space (C-space) $\mathcal{C}$ being the set of all possible configurations. The C-space region invalidated by the presence of obstacles and self-collisions is denoted by $\mathcal{C}_{obs}$ and the obstacle-free region of the configuration space $\mathcal{C}_{free} = \mathcal{C} - \mathcal{C}_{obs}$.

For this paper, we assume that the $n$ legs have the same kinematic structure, but this is not a hard limitation of the approach. Each leg $L$ is a kinematic chain of $m$ DOFs. The configuration of a leg is denoted $q_l$, and the workspace region occupied by the leg is denoted $L(q_l)$. The foot-tip position of a leg at configuration $q_l$ is $F(q_l)$.

A contact configuration is defined as a leg configuration $q_l$ that places the foot tip within a certain Euclidean distance $\epsilon$ of the environment surface $\mathcal{O}$. This defines the contact space $\mathcal{C}_{contact}$ as in Eqn. 1. For a contact configuration to be valid, the remaining links of the leg must not collide with the environment. Collision-free contact configurations are often only a small portion of the leg's C-space, meaning random rejection sampling is not feasible as the probability of samples being valid foothold configurations is very small.

$$\mathcal{C}_{contact} = \{q_l \in \mathcal{C} \mid distance(F(q_l), \mathcal{O}) < \epsilon\} \tag{1}$$

Each valid contact places the end-effector tip link within distance $\epsilon$ of an obstacle surface, without the configuration being invalidated by any of the remaining manipulator links colliding with an obstacle. The value of $\epsilon$ chosen must be sufficiently small that, during execution, the foot tip not being placed exactly on the surface can be compensated for by low level leg controllers, such as impedance controllers.

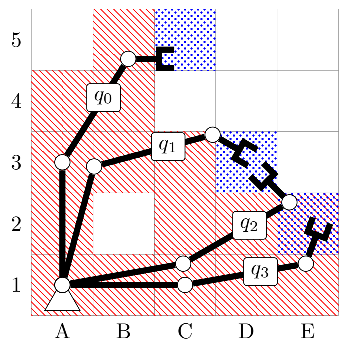
The presented motion planner is required to generate a valid path $\tau : [0, 1] \rightarrow \mathcal{C}_{free}$ from a start body pose $B_s$ to a goal $B_g$ such that $\tau(0) = B_s$ and $\tau(1) = B_g$. For a path to be valid, it must be collision free and also maintain sufficient contacts with the environment to ensure that a stability criteria is continuously satisfied.

There are several stability criteria available for legged robots, such as iterative projection [13] and linear programming with a robustness measure [14]. As the stability checking procedure is not a focus of this paper, a simple approximation is used, similar to [15]. To check if a configuration is valid, the Centre of Mass (COM) of all links of the robot is calculated and projected on the $xy$ plane (assuming that gravity is in the $-z$ direction). A configuration is considered valid if the COM is within the projected $xy$ convex hull of all foot contacts. This approximation is appropriate for flat environments, but a more complete model is required for planning on very steep and irregular terrains [13].

## III. Contact Dynamic Roadmaps

Key to our approach is the Contact Dynamic Roadmap (CDRM) data structure which extends the Dynamic Roadmap (DRM) structure introduced by Leven and Hutchinson [8]. Existing DRM implementations have been demonstrated only on fixed-base manipulators which do not make contact with the environment. Our implementation moves with a mobile robot base, and stores additional contact information to allow making contacts with the environment.

A DRM can be seen as an extension of PRMs with additional information stored at its edges and vertices that can be updated at runtime. However unlike PRMs, DRMs use a workspace mapping to explicitly maintain and modify the C-space connectivity graph as the environment changes, enabling the robot to quickly adapt the motion plan as compared to recomputing the whole plan.



| Cell | $\phi_v(w)$ | $\phi_c(w)$ |
|---|---|---|
| A1 | $q_0$, $q_1$, $q_2$, $q_3$ | $\emptyset$ |
| A2 | $q_0$, $q_1$ | $\emptyset$ |
| B1 | $q_2$, $q_3$ | $\emptyset$ |
| C5 | $\emptyset$ | $q_0$ |
| D3 | $\emptyset$ | $q_1$, $q_2$ |
| E2 | $q_2$ | $q_3$ |

Fig. 3: Partial CDRM vertex and contact mappings for a 3-DOF manipulator. Cells with colliding configurations are marked red and cells with contacts are dotted blue. A selection of cell mappings are shown in the table.

We generate a CDRM for a single representative leg and then use this in the online planning phase to allow for rapid contact configuration identification. CDRM generation begins by creating a Probabilistic Roadmap (PRM) [6] assuming an obstacle-free workspace. The PRM is a graph $G = (V, E)$ where the vertices are leg configurations and the edges are straight line paths connecting two vertices. This graph approximates the connectivity of $\mathcal{C}$. A dense PRM is generated offline, discarding configurations and edges which result in a self-collision. A weighted Euclidean C-space distance metric is used to connect nearby vertices, but workspace metrics can also be used [11].

The second stage of the offline pre-processing then maps from the leg's workspace to the C-space roadmap. The workspace is decomposed into cells, most commonly using a uniform grid decomposition. These discretised workspace cells are then mapped to the vertices and edges in $G$, resulting in the mappings shown in Eqns. 2 and 3. These mappings allow querying the roadmap vertices and edges which result in the leg intersecting a workspace cell $w$.

$$\phi_v(w) = \{q_l \in V \mid L(q_l) \cap w \neq \emptyset\} \tag{2}$$
$$\phi_e(w) = \{e \in E \mid \exists L(q_l) \cap w \neq \emptyset, q_l \in e\} \tag{3}$$

The CDRM builds on the DRM by adding an additional mapping $\phi_c(w)$, which encodes the configurations which place the foot tip within each cell as shown in Eqn. 4. Fig. 3 shows an example CDRM for a planar manipulator in a 2D workspace. The process for generating the CDRM is discussed in detail in Sec. IV-A.

$$\phi_c(w) = \{q_l \in V | F(q_l) \cap w \neq \emptyset\} \tag{4}$$

During the online planning phase, the environment is known and the cells $w_{obs} = \{w \mid w \cap \mathcal{O} \neq \emptyset\}$ are identified which are occupied by obstacles. The mappings $\phi_v(w)$, $\phi_e(w)$ and $\phi_c(w)$ can then be used to identify configurations which result in both a contact and a collision with the environment. As the mapping is generated offline using a pre-processing phase, this is a fast operation which does not require any additional collision checking.

The motion planner uses this mapping for two purposes once the environment is known. Firstly, it can identify configurations which result in a foothold contact but do not place any other part of the leg in collision with the environment. Secondly, motion planning queries can be performed for single-leg transition motions. When an obstacle is observed, the cells the obstacle occupies can be queried to identify the roadmap portion which becomes invalidated. A graph query is then used on the valid nodes and edges to plan paths. This allows the roadmap connectivity to be maintained in response to obstacles without requiring expensive collision checks for each vertex and edge.

## IV. Full-body Planner

The motion planner uses the Rapidly Exploring Random Tree (RRT) framework [7] and offline pre-computed CDRM to perform full-body legged motion planning queries. We first outline the planner, then Sec. IV-A details the offline generation of the CDRM mappings and Sec. IV-B shows how it is used as part of the online planner.

Our approach first generates body poses $B$, and then connects them to nearby body poses using single leg transition motions. This hierarchical approach trades completeness for computational efficiency, but is still able to plan complex paths as shown in Sec. V. Our approach also differs from approaches such as [16] which first generate foothold configurations before body poses. An outline of the online planning process is:

1) Body poses are sampled and connected to nearby poses in a tree structure. This is continued until the goal is reached, or the allotted planning time has expired.
2) The CDRM is used to identify candidate leg configurations (footholds) for each leg which place the foot tip in contact with the environment, without the rest of the leg colliding with the environment.
3) Inverse Kinematics (IK) are used to generate a single-mode transition configuration for each candidate foothold. It must also be free of collision, and the CDRM is used to ensure that it can be connected to a foothold in the previous body pose.
4) If there is at least one valid foothold available for each leg, a full-body state is created by randomly sampling a foothold for each leg. These are combined to produce a full-body state which is then tested for self-collision and stability.
5) Once the goal is reached, the best full-body state for each body pose is identified using a heuristic cost function, yielding a path composed of discrete full-body configurations.

### A. Offline Contact Dynamic Roadmap Generation

As introduced in Sec. III, a CDRM combines a C-space PRM [6] with workspace mappings. It is generated offline using a time-consuming mapping process and then used online for motion planning queries. We generate $\phi_c(w)$, $\phi_v(w)$ and $\phi_e(w)$ for a single representative leg in a pre-processing phase and save it for later use in motion planning queries.

In general different CDRMs could be created for each leg if they had different structures. The leg is treated as an $m$ DOF manipulator located at the origin of an obstacle-free workspace $\mathcal{W}_l \in \mathcal{R}^3$.

In the first stage, we generate the C-space PRM $G = (V, E)$ approximating the leg's configuration space. We use the Open Motion Planning Library (OMPL) [17] PRM implementation to generate $n$ vertices and connect each of them to the nearest $k$ vertices with a straight line connection to create the roadmap. The environment is assumed to be obstacle free, so the only configurations and edges discarded are those which violate kinematic constraints or result in self-collision.

The roadmap must be dense enough to approximate the reachability of the leg, as each configuration is a potential foothold to be used by the planner. As obstacles are added to the environment, portions of the roadmap are invalidated and the graph must be robust enough to maintain connectivity. While we use uniform sampling, a possible enhancement to improve the roadmap quality would be to bias sampling to ensure that foot tip positions are distributed across the workspace as in [10].

In the second stage we then decompose $\mathcal{W}_l$ into a number of uniform grid cells $w \in \mathcal{W}_l$ with a specified resolution $r$. We then map cells in this cellular decomposition back to the C-space roadmap. We also augment the DRM by including contact information to create the CDRM.

Calculating the maps $\phi_c(w)$, $\phi_v(w)$ and $\phi_e(w)$ is difficult, so as in [8] we calculate the inverse maps by placing the leg links at the configurations being tested and then checking which workspace cells it occupies. This is done using a voxelisation routine based on a triangle-box intersection algorithm [18] to test if the cell is occupied.

Firstly to calculate the inverse vertex mapping $\phi_v^{-1}(q_l)$ we voxelise each configuration $q_l \in V$ in the PRM. Each cell $w \in \mathcal{W}_l$ occupied by the configuration has $q_l$ added to its list of configurations. In order to calculate the inverse edge mapping $\phi_e^{-1}(q_l)$ each edge $(q_1, q_2) \in E$ is individually voxelised. To voxelise an edge, the edge is recursively bisected and voxelised until no more occupied cells are seen. These occupied cells are then added to the mapping. Finally, for each $q_l \in V$ the leg's forward kinematics are used to determine the foot tip contact position. This is then inserted into the mapping.

The concept behind the CDRM is to perform the expensive foothold candidate generation and collision checking phases offline, and then store the data for use in the online planner. When the environment is known, the pre-computed CDRM can be transformed to the origin of a leg being considered. The CDRM is then collided with the environment to give $w_{obs}$, the cells which collide with this environment. Candidate foothold leg $q_{footholds} \in V$ configurations are those for which the end-effector position, but no other part of the leg, is in collision with the environment. This can be identified from the CDRM mappings: $q_{footholds} = \phi_c(w) - \phi_v(w)$.

The limitation of this approach is that the foothold positions and collision checks will only be accurate to within the mapping resolution $r$. If the mapping is generated at a sufficiently fine resolution, this may be acceptable. One solution would be to adjust the foothold configurations using IK to place them directly on the environment surface. Alternatively, an impedance controller which can compensate for this during motion execution can be used.

### B. *Online Full-body Planning*

The full-body motion planner is based on the Rapidly exploring Random Tree (RRT) framework [7], but could also be applied to other sampling-based planners such as PRMs. Our implementation is single-threaded, but parallelisation could also be used to improve performance. The RRT algorithm is shown in Alg. 1, and our implementation again leverages the Open Motion Planning Library (OMPL) [17]. A tree of body poses is created rooted at the start pose $B_s$. Body poses are sampled, validated, and an attempt is made to connect to the nearest existing pose in the tree.

**Algorithm 1** The full-body RRT algorithm

```
T.INIT($B_s$)                          ▷ Create a tree rooted at $B_s$
while planning time not exceeded do
    $B_{rand} \leftarrow$ SAMPLEBODYPOSE()
    $B_{prev} \leftarrow$ NEARESTVERTEX($T, B_{rand}$)
    $B_{new} \leftarrow$ NEWSTATE($B_{prev}, B_{rand}, \Delta B$)
    if CANCONNECT($B_{prev}, B_{new}$) then
        T.ADDVERTEX($B_{new}$)
        T.ADDEDGE($B_{prev}, B_{new}$)
        if $B_{new} = B_g$ then return success
    end if
end while
```

The SAMPLEBODYPOSE function generates a 6-DOF body pose $B_{rand}$. With some probability (by default 5%) it will yield $B_g$ to guide the search towards the goal. Otherwise, it will randomly sample a collision-free body pose from $\mathcal{W}$. In order for a body pose to be valid, all foot tips must be within a certain distance $\epsilon$ of the environment to reach the surface. If the environment is outside the work envelope of any of the legs, the pose is discarded. This is a simplified analogy of [2] to ensure the environment remains within the reachable workspace of each leg.

The nearest body pose already in the tree $B_{prev}$ is then found using NEARESTVERTEX and extended towards $B_{rand}$ by up to the range parameter $\Delta B$ to create $B_{new}$. The range parameter limits the maximum distance between two successive steps, and must be within the reachability of the legs. The CANCONNECT function checks that $B_{new}$ is valid and can be reached from $B_{prev}$ with a single footstep for each leg. This state and transition validation is the most time consuming part of the planner and consists of:

- checking that the body does not collide with $\mathcal{O}$ between $B_{prev}$ and $B_{new}$ (Sec. IV-B1),
- ensuring that there is at least one valid foothold configuration for each leg at $B_{new}$ (Sec. IV-B2),
- validating that foothold configurations can be reached from the previous body pose $B_{prev}$ (Sec. IV-B3), and
- generating at least one full-body state which is free of self-collision and statically stable (Sec. IV-B4).

These steps are described in detail below:

*1) Body Collision:* For the first stage it is assumed that the body translates linearly from $B_{prev}$ to $B_{new}$. A straight-line collision check is performed between the body and the environment moving between the two poses using recursive bisection [19]. If the body collides at any point the state $B_{new}$ is invalidated. This is only an approximation to the true movement. If required, a final check could be performed after the full-body motion plan is generated to validate this body movement.

*2) Foothold Generation:* The second stage in validating $B_{new}$ is to generate at least one valid foothold configuration for each leg. These configurations must be collision-free and place the foot tip within $\epsilon$ of the environment. If any of the legs don't have a valid foothold configuration, the body pose $B_{new}$ is invalidated; a consequence of this is that all legs will be in contact with the environment at all body poses. Approaches which would allow legs to be free include optimising for minimum number of contacts [20] or using a priority queue for contact selection [2].

To generate candidate footholds the CDRM origin is transformed to each leg's local origin for the body pose $B_{new}$ being validated. A collision check is performed to identify CDRM workspace cells $w_{obs}$ in collision with the environment. Leg configurations $q_{obs} \in w_{obs}$ which collide with the environment are identified. The leg configurations which create a contact with the environment, but not a collision, $q_{footholds}$, are then queried from the CDRM. This is in contrast to approaches such as [21] which require footholds to be generated in a pre-processing phase.

*3) Transition Validation:* The planner is structured to move each leg to a new position using a pre-defined creep gait order and then translate the robot body. Therefore, each generated foothold configuration must be valid and reachable from both the current body pose $B_{new}$ and the previous body pose $B_{prev}$. For each of the candidate foothold configurations, closed-form Inverse Kinematics (IK) are used to generate a configuration which reaches the same foothold in $B_{prev}$ as in $B_{new}$ (a transition foothold). If the IK solver cannot find a solution or the IK-generated configuration collides with the environment, the foothold is not reachable from the previous body pose and is discarded.

When the body is at $B_{prev}$, each IK-generated transition foothold must be reachable from a foothold that was used at $B_{prev}$ through the CDRM C-space roadmap. The closest $k$ configurations in the roadmap are found using a nearest neighbours data structure and a straight line connection is attempted to each of these configurations. A connected components data structure is then used within the roadmap to identify if the IK-generated solution can be connected to any footholds at the preceding body pose $B$. Vertices which are invalidated by the environment as identified in Sec. IV-B2 are not used. Once one of the closest $k$ configurations has been found by the search, the foothold is known to be reachable by a collision-free motion from a previous foothold. If no previous footholds are found, the foothold is discarded.

*4) Full-body State Generation:* The body pose $B_{new}$ now has, for each leg, at least one valid and reachable foothold configuration. The final stage of body pose validation generates full-body states which combine the body pose with a configuration for all legs. The planner generates these states by randomly sampling from the available footholds for each single leg, and then checking whether the full state is free of self-collision and satisfies the stability criteria. Up to $n$ sampling attempts are used to generate up to $m$ valid full-body states. If no valid states can be generated, the body pose is discarded.

Once a candidate set of full-body states has been generated, the best one is selected using a heuristic cost function. For the examples in this paper, the full-body state closest to a reference configuration was selected. Alternative heuristic functions could be used such as maximising the stability robustness of the selected full-body configuration [2]. Generating multiple candidate states and selecting the best using a heuristic function was found to significantly improve path quality compared to selecting the first valid state found.

Once $B_{new}$ has a stable full-body state, the same foothold positions must be checked for stability and self-collision at the previous body pose $B_{prev}$. IK is used to generated the full-body state with the previous body pose, and the same full-body state is validated for self-collision and stability.

It is assumed that if the body satisfies the stability criteria at both $B_{new}$ and $B_{prev}$ with the same foot tip positions, the intermediate body motion is also statically stable, as the COM would remain over the support polygon formed by the legs. If this condition is required to be explicitly validated the path can be interpolated and static stability checked along the entire path. Due to proximity it is also assumed that the legs are collision free for body translation from $B_{new}$ to $B_{prev}$.

Finally, any footholds that are not used as an element of a valid full-body state are discarded, as they are in inaccessible regions. This prevents subsequently generated states from attempting to connect to these footholds.

## V. RESULTS

The planner was tested using the quadruped shown in Fig. 2. Each of the four legs is kinematically identical, approximately 0.45m long and has three DOFs. Tests were run using a Core i7 4700M CPU and 16GB RAM.

We first detail the generation of the CDRM in Sec. V-A, including the key parameters chosen. Planning results are then shown on planar terrains, individual challenging features and extended paths in complex scenarios which combine multiple of these features. We then compare our planner to a basic RRT implementation and a state of the art full-body legged motion planner.

To aid visualisation, only a few representative full-body configurations for each path is shown, with red dots used to show all footholds. The maximum allowed planning time was 120s and the average time over 10 tests was taken. Dual-axis plots are used to show the results with planning times on the left axis and success rates on the right axis. Data points where no solution was reached due to timeout are marked with 0% success rate but not plotted in time axis. OMPL [17] path simplification using short-cutting techniques was applied. Since this post-processing only smoothed an existing valid path, the time taken was not included in the results.

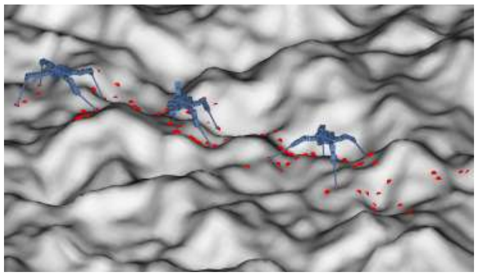


Fig. 4: Terrain with roughness of ± 0.45m

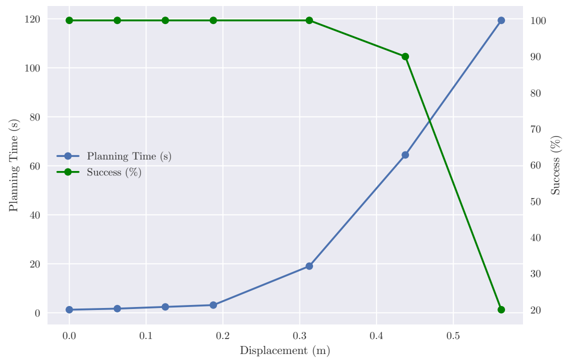


Fig. 5: Planning times and success rate vs. terrain roughness for a 4m path

### A. CDRM Generation

We generated a number of CDRMs with 10000 candidate foothold configurations connected to their nearest 10 neighbours. We varied the discretisation resolution and measured the generation time, query time and CDRM size. Generation time and size varied proportionally to the cube of the resolution, while the query time also depended on the resolution of the environment model. The size could be reduced by using DRM compression techniques as in [8].

We selected a resolution of 0.01m as it provided ample accuracy for the robotic system in question, without the CDRM becoming too large in terms of memory usage. As each leg is 0.45m long, the resolution is significantly smaller than the reachability. During online execution, the position error is likely to exceed 0.01m. For the CDRM parameters selected, the offline generation time was 28 minutes and the resulting CDRM was 2.4GB. The online query time during planning was 0.04s to evaluate 10000 foothold positions for all legs. Performing the same process online without the CDRM took approximately 2.1s, demonstrating the efficiency of our approach to rapidly evaluate foothold configurations.

### B. Terrain

A planar terrain of 5m × 5m was created using 12800 triangles. Each vertex was vertically displaced using a Perlin noise texture [22] to approximate surfaces of different roughness as shown in Fig. 4. The maximum vertical displacement was varied from 0 (a flat surface) to ± 0.5m (a very rough surface exceeding the reachability of the legs). Planning times and success rates for a path 4m in length are shown in Fig. 5 for several roughness values. As the roughness increases, planning time also increases, but remains under 5s up to a roughness of ± 0.2m.

The planning time for smooth ground is in the order of only a few seconds, allowing paths in simple environments to be planned very quickly. However, a gait-based planner would be more suited to these problems and would produce a faster and smoother path. As the terrain becomes rougher our planner maintains a high success rate until the terrain roughness exceeds the reachability of the robot. As the terrain becomes more complex, full-body planning is increasingly applicable as explored further in Sec. V-C.

Online planning requires generating motion plans within the time it would take to execute them. To test if online planning was possible, a path of one body length was planned on a terrain with a high roughness of ± 0.45m, which is of the order of the robot height. A plan to traverse a single body length was found in 1.5s and the execution of this on a physical platform would take significantly longer.

### C. Features

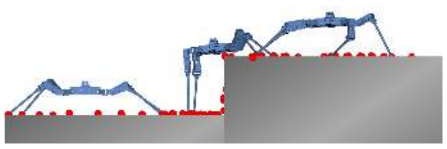
A second set of experiments was run where a path was planned over a single complex terrain feature to test if the planner can deal with varied complex features while maintaining low planning times and using the robot's full reachability. A 3m path was planned over: a step of varying height (Fig. 6), a gap of varying width (Fig. 7) and an overhanging bar at a varying height (Fig. 8). Feature dimensions were varied to simulate various difficulty levels.

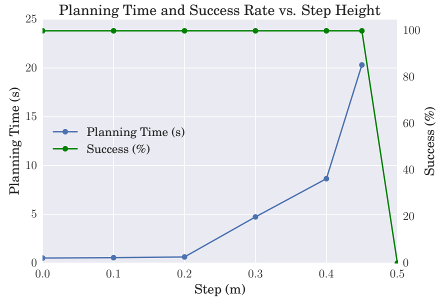
*a) Step (Fig. 6):* The step height was varied between 0 and 0.5m, with the planning time remaining within 2s for a height of up to 0.2m and under 10s for 0.4m. The success rate was 100% until the step height reached 0.45m, which is the limit of the robot leg's reachability, demonstrating that the full workspace of the leg was used.

*b) Gap (Fig. 7):* Planning time remained within 2.5s as the gap size increased to 0.6m. Planning was also successful until the gap reached 0.7m and the gap could no longer be traversed. Again, this is at the limits of the robot's reachability. As can be seen in Fig. 7, the simplified stability criteria allows the robot to place footholds inside the gap itself, allowing it to cross such wide gaps and the planner took advantage of this.

*c) Overhang (Fig. 8):* An overhanging bar placed in a flat environment which the robot must crawl under or climb over. The planning time remains within ∼10s if the bar is $\leq$ 0.1m or $\geq$ 0.2m high. A peak of 27s is observed when the bar is set at 0.15m as the bar directly obstructs the robot body. As the bar is raised, the robot transitions from climbing over the bar to crawling underneath.

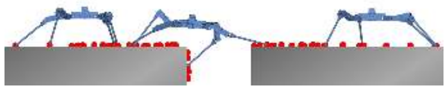
These three scenarios demonstrate that the planner is flexible enough to deal with a number of representative terrain features while maintaining low planning times. Failure rates only become high as the environmental complexity approaches the limits of the robot's reachability. This illustrates that the density of the CDRM is sufficient to exploit the extent of the robot's dexterity, while still remaining quick to search.

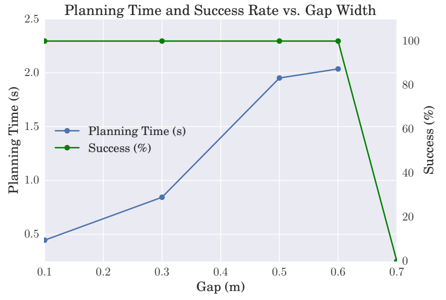

(a) Climbing a 0.4m step



(b)

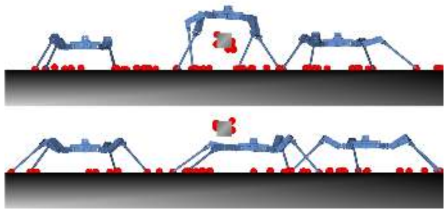
Fig. 6: Step and planning results

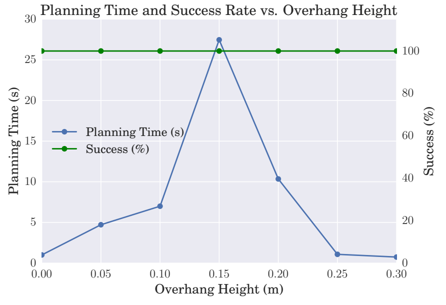

(a) Crossing a 0.5m gap



(b)

Fig. 7: Gap and planning results


(a) Crawling (i) over a 0.15m and (ii) under a 0.25m overhang



(b)

Fig. 8: Overhang and planning results

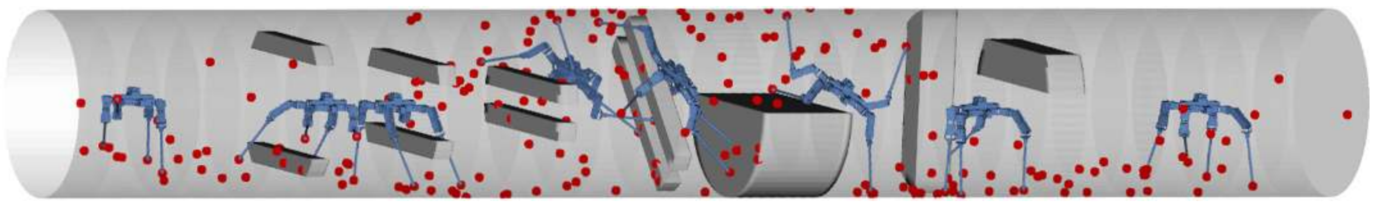


Fig. 9: Moving through the inside of an obstructed pipe

| | Truss (Fig. 1) | Pipe (Fig. 9) |
|---|---|---|
| Planning time (s) | 76.9 | 86.3 |
| Success (%) | 80 | 90 |
| Foothold generation (s) | 67.5 | 67.5 |
| State generation (s) | 1.45 | 1.92 |
| Transition validation (s) | 7.84 | 16.7 |

TABLE I: Planning times and success rates for the scenarios

## D. Scenarios

Full-body motion planning enables traversal of complex 3D structures, where cyclic gait-based planners would not succeed. Two sample scenarios shown in Figs. 1 and 9 show such complex scenarios which are composed of many individually challenging features, such as narrow passages and steps. We present the planning times and success rates for these two example problems in Table I.

*a) Truss:* The scenario in Fig. 1 shows a robot climbing a complex 3D truss structure. The planned path is approximately 4m long and took 76.9s to plan with a success rate of 80%. As the planning time was approaching the limit of 120s, the success rate was not 100%. This occasional failure in complex environments is mitigated by the ability to re-plan due to the relatively short planning time.

*b) Pipe:* A 7m path was planned through a pipe with several obstructions. This scenario is inspired by a robot with magnetic feet being used to perform an inspection task. Results were similar to the previous scenario, with a planning time of 86.3s, and success rate of 90%. The footholds being placed on the ceiling is due to the simplified stability criteria and would need to be adjusted for real-world usage.

These scenarios show the ability to plan extended motion paths which traverse a series of complex features. Most of the planning time is spent evaluating footholds, demonstrating the benefits of using the CDRM to optimise this.

## E. Comparisons

To validate our approach, we compared our CDRM planner with two other full-body planners (a) a naive 18-DOF (12 DOF leg configuration + 6 DOF body pose) RRT planner, and (b) the state of the art Reachability Based PRM (RB-PRM) planner [4]. All three approaches were run on the robot kinematic model shown in Fig. 2.

We evaluated the RRT planner on a planar terrain where samples that did not place all feet on the ground were rejected. This took approximately 180s to generate a single valid sample, illustrating this approach is not feasible for planning with contacts as the probability of selecting valid contact configurations is small.

Next, we ran the RB-PRM planner from the open-source Humanoid Path Planner package [23] on the same robot for two scenarios: rubble (Fig. 10) and the truss (Fig. 1). The crux of [4] is that if each robot leg is operating within its reachability limits, it is assumed a stable full-body configuration can

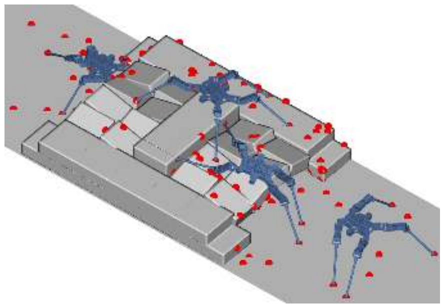


Fig. 10: Crossing over rubble-strewn ground (from [23])

be generated. For the rubble scenario shown in Fig. 10, our planner took on average 44s to generate a motion plan, while the RB-PRM planner took between 10s and 15s which was comparable to the 7s reported in [4] using the HyQ platform. We then tested both approaches on the truss shown in Fig. 1. Our planner generated paths in 77s, but RB-PRM failed to plan the vertical transitions, likely as the reachability assumption did not hold. These results show that on planar terrains our planner is within the same order of magnitude performance as [4], while enabling planning in complex environments such as Fig. 1 where computing a guide path and subsequently generating contacts may fail.

## VI. Conclusion

We have presented CDRMs which extend DRMs with contact information. These can be used to rapidly identify footholds for legged motion planning. Pre-computing foothold and collision information ahead of time allows valid footholds for a body pose to be identified with minimal collision checks, by directly querying the CDRM. This allows using a sampling-based planner to generate full-body plans which satisfy leg contact and stability criteria constraints in complex 3D environments with high success rates. This differs from approaches which serially decouple planning such as [4] in that a failure in a later planning stage can be recovered from. Our planner was demonstrated on individual terrain features as well as complex 3D environments.

The planner has a number of limitations: it is most applicable in environments with complex obstacles. For more open, planar and continuous environments a decoupled planner such as [4] or a gait-based planner may produce better results. The generated footholds are also limited to by the resolution and quality of the CDRM.

Future developments to this work include using post-processing to improve path quality. We also hope to integrate a more complete stability model such as [14]. We are implementing the planner on a physical platform, for which a more robust stability criteria is the key improvement required.

## Acknowledgment

The authors thank Steve Tonneau for his kind assistance using the Humanoid Path Planner RB-PRM package.

## References

[1] T. Bretl, “Motion planning of multi-limbed robots subject to equilibrium constraints: The free-climbing robot problem,” *The International Journal of Robotics Research*, vol. 25, no. 4, pp. 317–342, 2006.

[2] S. Tonneau, N. Mansard, C. Park, D. Manocha, F. Multon, and J. Pettré, “A reachability-based planner for sequences of acyclic contacts in cluttered environments,” in *International Symposium on Robotics Research (ISSR 2015)*, 2015.

[3] S. Kajita, F. Kanehiro, K. Kaneko, K. Fujiwara, K. Harada, K. Yokoi, and H. Hirukawa, “Biped walking pattern generation by using preview control of zero-moment point,” in *Robotics and Automation, 2003. Proceedings. ICRA’03. IEEE International Conference on*, vol. 2. IEEE, 2003, pp. 1620–1626.

[4] S. Tonneau, A. D. Prete, J. Pettré, C. Park, D. Manocha, and N. Mansard, “An efficient acyclic contact planner for multiped robots,” 2016.

[5] M. Elbanhawi and M. Simic, “Sampling-based robot motion planning: A review,” *IEEE Access*, vol. 2, pp. 56–77, 2014.

[6] L. E. Kavraki, P. Svestka, J.-C. Latombe, and M. H. Overmars, “Probabilistic roadmaps for path planning in high-dimensional configuration spaces,” *IEEE transactions on Robotics and Automation*, vol. 12, no. 4, pp. 566–580, 1996.

[7] S. M. LaValle, “Rapidly-exploring random trees: A new tool for path planning,” 1998.

[8] P. Leven and S. Hutchinson, “A framework for real-time path planning in changing environments,” *The International Journal of Robotics Research*, vol. 21, no. 12, pp. 999–1030, 2002.

[9] N. M. Amato, O. B. Bayazit, L. K. Dale, C. Jones, and D. Vallejo, “OBPRM: An obstacle-based PRM for 3d workspaces,” in *Proceedings of the Third Workshop on the Algorithmic Foundations of Robotics on Robotics: The Algorithmic Perspective*, 1998, pp. 155–168.

[10] M. Kallman and M. Mataric, “Motion planning using dynamic roadmaps,” in *Robotics and Automation, 2004. Proceedings. ICRA’04. 2004 IEEE International Conference on*, vol. 5. IEEE, 2004, pp. 4399–4404.

[11] T. Kunz, U. Reiser, M. Stilman, and A. Verl, “Real-time path planning for a robot arm in changing environments,” in *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 2010, pp. 5906–5911.

[12] S. Murray, W. Floyd-Jones, Y. Qi, D. Sorin, and G. Konidaris, “Robot motion planning on a chip,” in *Robotics: Science and Systems*, 2016.

[13] T. Bretl and S. Lall, “Testing static equilibrium for legged robots,” *IEEE Transactions on Robotics*, vol. 24, no. 4, pp. 794–807, 2008.

[14] A. Del Prete, S. Tonneau, and N. Mansard, “Fast algorithms to test robust static equilibrium for legged robots,” in *Robotics and Automation (ICRA), 2016 IEEE International Conference on*. IEEE, 2016, pp. 1601–1607.

[15] D. Belter and P. Skrzypczyński, “Integrated motion planning for a hexapod robot walking on rough terrain,” *IFAC Proceedings Volumes*, vol. 44, no. 1, pp. 6918–6923, 2011.

[16] K. Hauser, T. Bretl, J.-C. Latombe, K. Harada, and B. Wilcox, “Motion planning for legged robots on varied terrain,” *The International Journal of Robotics Research*, vol. 27, no. 11-12, pp. 1325–1349, 2008.

[17] I. A. Şucan, M. Moll, and L. E. Kavraki, “The Open Motion Planning Library,” *IEEE Robotics & Automation Magazine*, vol. 19, no. 4, pp. 72–82, December 2012, http://ompl.kavrakilab.org.

[18] T. Akenine-Möller, “Fast 3d triangle-box overlap testing,” in *ACN Siggraph 2005 Courses*. ACM, 2005, p. 8.

[19] R. Geraerts and M. H. Overmars, “A comparative study of probabilistic roadmap planners,” in *Algorithmic Foundations of Robotics V*. Springer, 2004, pp. 43–57.

[20] K. Bouyarmane, A. Escande, F. Lamiraux, and A. Kheddar, “Potential field guide for humanoid multicontacts acyclic motion planning,” in *Robotics and Automation (ICRA), 2009 IEEE International Conference on*. IEEE, 2009, pp. 1165–1170.

[21] P. Vernaza, M. Likhachev, S. Bhattacharya, S. Chitta, A. Kushleyev, and D. D. Lee, “Search-based planning for a legged robot over rough terrain,” in *Robotics and Automation, 2009. ICRA’09. IEEE International Conference on*. IEEE, 2009.

[22] K. Perlin, “An image synthesizer,” *ACM Siggraph Computer Graphics*, vol. 19, no. 3, pp. 287–296, 1985.

[23] “Humanoid path planner - RBPRM module,” https://github.com/humanoid-path-planner/hpp-rbprm, commit 5a3715d.