Hierarchical Rough Terrain Motion Planning using an Optimal Sampling-Based Method

Michael Brunner, Bernd Brüggemann, and Dirk Schulz
Fraunhofer Institute for Communication, Information
Processing and Ergonomics FKIE
Wachtberg, Germany
{michael.brunner, bernd.brueggemann,
dirk.schulz}@fkie.fraunhofer.de

***Abstract*— Mobile robots with reconfigurable chassis are able to traverse unstructured outdoor environments with boulders or rubble, and overcome challenging structures in urban environments, like stairs or steps. Autonomously traversing rough terrain and such obstacles while ensuring the safety of the robot is a challenging task in mobile robotics.**

In this paper we introduce a two-phase motion planning algorithm for actively reconfigurable tracked robots. We first use the completeness of a graph search on a regular grid to quickly find an initial path in a low dimensional space, considering only the platform's operating limits instead of the complete state. We then take this initial path to focus the RRT* search in the continuous high-dimensional state space including the actuators of the robot. We do not rely on a detailed structure/terrain classification or use any predefined motion sequences. Hence, our planner can be applied to urban structures, like stairs, as well as rough unstructured environments. Simulation results prove our method to be effective in solving planning queries in such environments.

I. Introduction

Systems with reconfigurable chassis are able to alter their state using actuators to increase their traction, move their center of mass to achieve a more stable state, or to lift themselves over an edge. This ability extends the mobility of robots and allows them to traverse a greater variety environments.

In urban environments, steps or stairs are generally obstacles for fixed-chassis robots, but can be overcome by systems with reconfigurable drives. Unstructured outdoor environments involve obstacles - for instance, coarse debris, rubble, rocks, or steep inclinations - which can only be overcome by robots with reconfigurable drives.

Even for a trained operator, it is a challenging task to safely steer a robot over such obstacles. There are many aspects to consider when driving over obstacles. On unstructured terrain, the robot's stability becomes more important as the danger of falling over is greater compared to flat terrain. Inertia and momentum will be increasingly important if a robot is operated close to its limits. Moreover, due to the uneven terrain the system may react differently to the same commands.

In this paper, we present a two-phase rough terrain motion planning algorithm for tracked actively reconfigurable robots like the Packbot or the Telemax (Fig. 1). We first employ a graph search that quickly provides an initial path using the robot's operating limits instead of the complete robot state. The second phase uses RRT* [1] to find an asymptotically optimal path in the continuous high-dimensional robot state space which includes the actuators. To make this search efficient we focus the search on the segments of the initial path which lead through rough areas. Our method does not rely on terrain classifications or fixed motion sequences. Hence, it can be applied to rough, unstructured outdoor environments as well as obstacles in urban surroundings (e.g., stairs). See Fig. 2 for an overview. We consider our method to be the global planning component within a robotic system. The controller which executes the plan and takes care of localization and obstacle avoidance is beyond the scope of this work.

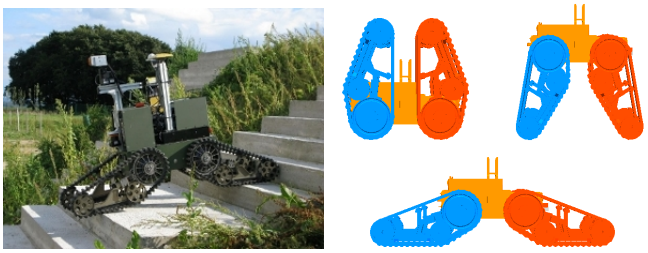

Fig. 1: The Telerob Telemax is $65\,cm$ long, $40\,cm$ wide and weighs about $70\,kg$. It has 4 tracks which can be rotated. Extended, the robot's length is $120\,cm$. The robot is skid-steered and drives up to $1.2\,\frac{m}{s}$. Configurations are: (left) folded (actuators at $-90°$), (right) extended (actuators at $80°$) and (bottom) maximal ground contact (actuators at $15°$).

Many algorithms for traversing rough terrain or climbing structures involve a preceding classification step. For instance [2] and [3] identify staircases, using line detection, to steer the systems during climbing and to fix the robot's heading to the gradient of the stairs. In [4] the classification of the environment is used to encode specialized skills in a *Behavior Map*. This map is utilized by their method to explore rough disaster environments.

Rough terrain planning methods often involve a decomposition into sub-tasks and a multi-phase planning scheme. [5] first categorizes the environment into separate terrain classes. Afterwards a high-level planner samples a path across terrain types and then triggers a specialized sub-planner for each

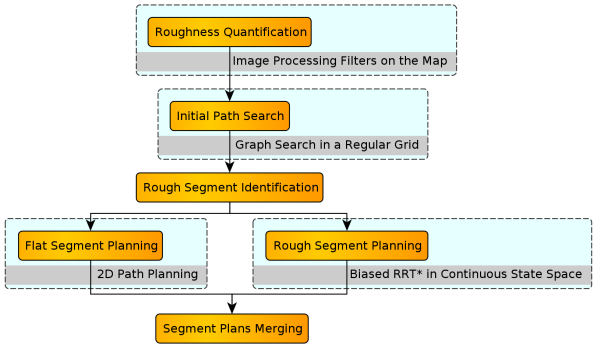

Fig. 2: Method overview: Using filters from image processing we perform a roughness quantification of the map. The initial path is found within a regular grid performing a graph search. Afterwards the rough segments are identified; for flat segments we apply a 2D planning scheme and a biased RRT* algorithm to refine rough path segments. Finally the segment plans are merged to provide the final path.

path segment according to the terrain. [6] introduces a controller for fast quadruped locomotion over rough terrain. The controller decomposes the controlling task into several sub-tasks; they first produce an approximate path and subsequently improve this initial solution to ensure kinematic reachability and a smooth and collision-free trajectory.

Roadmap methods are commonly applied to solve the rough terrain path planning problem [7]–[9]. In [8] a breadth-first search algorithm is used on a 3D lattice to plan stable paths for actively reconfigurable robots. The system's stability guides the search on a triangular mesh of the environment. [9] employs a graph-search to find motion sequences in a discretizied state space. The paths also include motions of controlled balance losing (e.g. minor falls from small edges).

Sampling-based methods have also been used to find paths through rough environments. [10] presents a RRT-based algorithm which finds routes of low mechanical work following valleys and saddle points in continuous cost spaces. Their method also includes a mechanism based on stochastic optimization to filter irrelevant configurations and to control the exploration behavior. In [11] a RRT variant is employed for kinematic path planning for a LittleDog robot. The authors bias the search in the task space and use motion primitives to speed up planning. [12] uses different terrain parameters to guide the RRT expansions and iteratively increases its roughness tolerance if no solution is found.

Our work differs from the works mentioned above as we distinguish between flat and rough regions, but do not rely on a detailed terrain classification or on fixed motion sequences. Therefore, we are not limited to a set of terrain or structure classes imposed by any categorization or a set of fixed motion sequences. As a consequence, our algorithm can be applied to a variety of environments. Further, instead of focusing the detailed motion planning on the entire initial path we concentrate the search on rough segments, avoiding expensive planning in flat areas.

The remainder of this paper is structured as follows: in section II we introduce the roughness quantification used. Section III describes our planning algorithm. Experiments are provided in section IV, and we conclude in V.

II. Map and Terrain Roughness

As in [4], [5], [9] we use a heightmap to represent the environment. Using a map allows us to simplify the perceptual task of 3D navigation in rough terrain and to focus on the motion planning aspect of rough terrain navigation [6]. Building the map is beyond the scope of this paper.

In order to assess the risk at a position within the map, we use techniques from image processing to compute a roughness quantification. We first apply a maximum filter with a squared kernel to the height differences. A distortion of the range of values can be prevented by a threshold, which conveniently can be set to the robot's maximal traversable height. The threshold is also used to scale the values to $[0, 1]$. Subsequently, we apply a two dimensional Gaussian blur to smooth the transitions. The maximum filter prevents isolated peaks to be smoothed by the Gaussian filter. Further, an appropriate kernel size allows us to virtually inflate the hazardous areas which is commonly done in 2D navigation to keep the robot away from obstacles. Using such filters makes the computation simple and highly parallelizable. We use this roughness quantification in both planning phases to adjust the behavior according to the difficulty of the environment.

III. Motion Planning Algorithm

Driving with actively reconfigurable robots on rough terrain introduces a large planning space. Additionally, the robot's safety is not naturally satisfied and must be validated. The robot's actuators must be incorporated into the planning process and the quality of the path must be judged not only by its length or execution time, but also by the robot's safety. First, we employ a preliminary path search to quickly find an environment-guided path to the goal. Subsequently, the path is used to constrain the search space for the RRT* planner. The RRT* planner refines the initial path and determines a path of robot configurations including the actuators.

A. Finding an Initial Path

To find the initial path we use the roughness quantification to guide the robot, i.e. to avoid hazardous areas and to prefer low risk routes. The consideration of the complete state especially the actuators is not necessary in flat regions, whereas it is essential in rough regions to increase the robot's safety and to ensure successful traversal. At the beginning we do not know through which parts of the environment the path will lead and if considering the complete state is really necessary. Therefore, we use the operating limit of the mobile robot and set the actuators aside. The maximal height difference traversable by the robot constitutes the operating limit. This way we obtain the least restrictive limit.

We build a motion graph which represents the ability of the mobile robot to traverse the environment (Fig. 3). The motion graph is based on the operating limits of the robot. The transition costs are given by the time t required to traverse

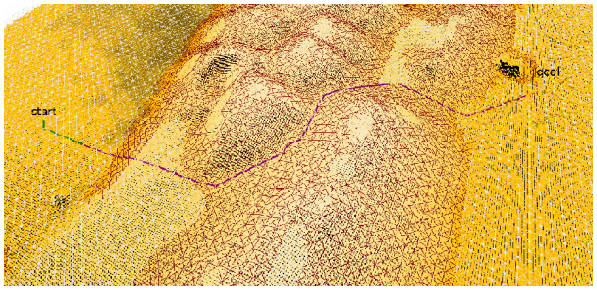

Fig. 3: The motion graph of a map encodes the traversability of the terrain. Areas with small height differences are white and areas with higher differences are red. The preliminary path splitted into segments through flat regions (green) and segments through rough regions (purple).

a graph edge e. Hereby, we reduce the permissible velocity according to the terrain roughness:

$$t = \frac{d}{\max\left(v_{\min}, (1 - w_1 \cdot r) \cdot v_{\max}\right)},$$

where d is the length of edge e and r the maximal roughness of the vertices of e. $v_{\min}$ and $v_{\max}$ are the minimal and maximal forward velocity of the robot. A safety weight w_1 adjusts the importance of safety: low safety weights diminish the influence of the risk, hence lead to possibly shorter, but riskier paths. On the contrary, high values increasingly force the robot to take low risk paths. With those edge weights we find a path performing a usual graph-search, e.g. A*-search or Dijkstra-search. A more detailed description of the metrics used by our algorithm is found in [13], including a discussion of the safety weights.

B. Identifying Rough Path Segments

Similar to 2D navigation, the robot's stability can be assumed in flat areas since any robot configuration may be applied with no or little risk. Hence, the planning problem can be considerably simplified. Avoiding unnecessary planning in a high-dimensional space for easily accessible parts of the environment speeds up planning. However, rough regions require a detailed planning of the robot's configurations including the actuators and the validation of the robot's safety to ensure a successful task completion.

While constructing the motion graph, we distinguish between flat and rough areas. Flat areas correspond to moderate height differences; rough areas involve challenging height differences. We use this distinction to split up the initial path into flat and rough path segments (Fig. 3) and to determine whether an expensive detailed motion planning is necessary.

C. Refining Rough Path Segments

Considering the entire robot state including the actuators allows us in this phase to assess the robot's configuration more comprehensively in terms of safety and time. The state of a reconfigurable robot may consist of the 6D pose and the n actuator values $(x, y, z, \theta, \psi, \phi, a_1, \ldots, a_n)$ of which the (x, y) position, the orientation θ and the actuators are controllable. Sampling-based methods are usually employed to search such high-dimensional spaces. Therefore, we refine the rough segments of the initial path using a modified version of the RRT* algorithm. The expansion of the RRT* algorithm is biased using the initial path to guide the search and make it more efficient. Erratic actuator movements are prevented through a simple sampling heuristic for actuator values.

1) Cost Function: Our cost function of the RRT* algorithm is described in detail in [13]. We will provide a short overview in this paper. To account for varying distances between random configurations we interpolate motions and compute the costs of the resulting sequences. The cost function is more accurate compared to the initial path search because it captures more aspects of the robot state. The cost c between two configurations breaks down into a safety c_{safety} and a time c_{time} component:

$$c = w_2 \cdot c_{\text{safety}} + (1 - w_2) \cdot c_{\text{time}}.$$

To quantify the safety, we examine the stability of the robot S using a stability margin and the center of mass, the traction T by considering the actuator angles to the surface and the environmental risk r from the roughness quantification. We take the average values of the involved configurations.

$$c_{\text{safety}} = \frac{r + \frac{1}{2}(S + T)}{2}.$$

The time term includes the time required for translation t_v and rotation t_ω as well as for actuator movements t_a. We use a triangle inequality to favor simultaneous execution.

$$c_{\text{time}} = \frac{t_v^2 + t_\omega^2 + t_a^2}{t_v + t_\omega + t_a}.$$

The safety weight w_2 allows to adjust the robots behavior, resulting in faster paths with less actuator movements or safer configurations through better suited actuator positions. However, adjusting the actuators to the current situation requires more time which increases the execution time of the path [13].

2) Path Bias: The first planning phase provides an approximate solution. We initialize the RRT*-search with this solution and further use it to focus the expansion of the algorithm. To concentrate the RRT*-expansion on a search space around the initial path we sample a position from a Gaussian distribution centered at the initial path. We first select a position p on the path uniformly at random and then sample from a two dimensional Gaussian distribution around p. The standard deviation of the Gaussian distribution determines the level of focus. In our setup we use $\sigma = 0.65\,m$, i.e. the robot length. Focusing the RRT*-search on the area of the initial path is based on a similar approach in [14]. It reduces the number of iterations required to achieve better solutions compared to a non biased search. See Fig. 4 for an analysis.

3) Configuration Rejection: If reaching a new configuration involves more costs than the current best path, the new configuration will not contribute to a better solution. Therefore, we reject configurations which are guaranteed not to be part of the final solution [14]. This prevents

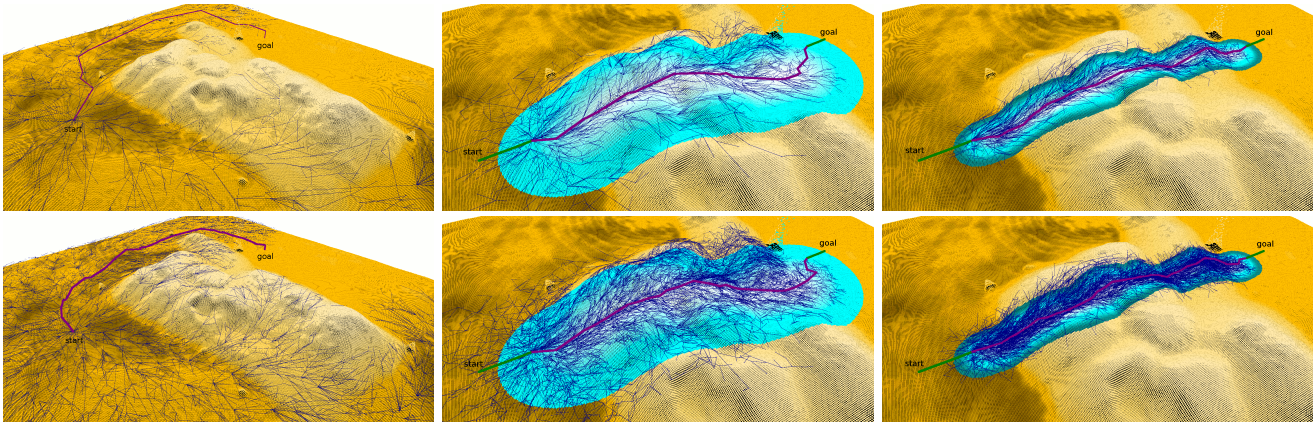

Fig. 4: The images show exemplary planning queries using no path bias (left), a path bias of $\sigma = 2.0\,m$ (middle) and a path bias of $\sigma = 0.65\,m$ (right). The top row shows the planning trees after 2500 iterations, the bottom row after 10000 iterations. To determine the final path costs after 10000 iterations, we performed 10 runs with each setup. Using no path bias the average cost was 25.01 (stdev: 5.19). For a path bias of $\sigma = 2.0\,m$ the average cost was 15.83 (stdev: 0.35) and for $\sigma = 0.65\,m$ the average cost was 14.47 (stdev: 0.12). Planning took $5\,s$, $47\,s$ and $100\,s$, respectively. The increasing time requirements are due to a greater number of rewiring options. Because we visualize only 2 dimensions of a 7D state vector, the displayed planning trees do not show the RRT* characteristic structure in every detail. The path bias increases the sampling density in the area around the initial path and allows the algorithm to find a better path within fewer iterations.

the planning tree from becoming cluttered with obstructive configurations which in turn increases efficiency.

4) Actuator Sampling: Some robots have several actuators which can be controlled separately. To prevent erratic actuator movements by such models we employ a simple sampling heuristic. We determined a set of categories of actuator configurations and sample a category according to its utility. The categories are: all actuators equal, front as well as rear actuators equal, left as well as right actuators equal, and all actuators different. For a given category we sample the corresponding number of actuator values to determine the final configuration: one for the first, two for the second and third, and four values for the last category. The two-fold sampling smooths actuator motions and decreases path costs (Fig. 5).

IV. Experimental Results

In this section we provide a comparison of our RRT* planning algorithm with the roadmap planner we introduced in [13]. The major difference between the two algorithms is that the sampling-based algorithm searchs a considerable larger continuous state space. Therefore, it solves a significantly harder planning problem than the roadmap planner which searches a limited and discretized state space.

The roadmap planner was tested in two scenarios which we will use to compare the two algorithms. The environments are shown in Fig. 6 and Fig. 7. Both maps were recorded using a laser scanner. Their sizes are $36.4 \times 30.45\,m$ and $43.95 \times 32.95\,m$, respectively. The cell size of the maps is $0.05\,m$.

The setup of the roadmap planner is as follows: The resolution of the motion graph is $0.3\,m$ (half the robot length). We consider eight orientations in each position ($45°$ steps). The actuator values are bound to $[-45°, 45°]$ in steps of $15°$ and both front respectively both rear actuators are required to be the same. The safety weights are set to $w_1 = 0.75$ and $w_2 = 0.5$. We choose a folded configuration with all actuators at $-45°$ as default, which is applied in flat areas. The maximal ground contact is achieved with all actuators at $15°$.

The setup of the RRT* planner is as follows: We keep the same motion graph resolution and bounds for the actuators and use the same safety weights as for the roadmap planner. To determine the set of neighbors we use the adaptive ball formulation [1] with an initial radius of $r = 0.65\,m$. To create a path bias we set $\sigma = 0.65\,m$ for the Gaussian distribution. The probabilities of the actuator categories are $p = 0.20$ for all actuators equal, $p = 0.65$ for front as well as rear actuators equal, $p = 0.10$ for left as well as right actuators equal and $p = 0.05$ for all actuators different. The number of iterations is 10000 for each rough segment.

To compare both algorithms we gave the same planning query to the roadmap planner and to the RRT* planner. We measured the path cost and the planning time. To illustrate the planning results Fig. 6 and Fig. 7 show the paths and the actuator joints of the two methods. The roadmap planner is a deterministic algorithm, thus, we performed a single query to determine the path cost and planning time. For the random RRT*-planner we performed 10 times the same planning query and computed the average cost and planning time. We use a computer with a 3.33GHz Intel Xeon CPU and 12GB memory. If a path contains several rough segments, we will parallelize the planning queries for the rough segments.

As expected the roadmap planner is faster than the RRT* planner. However, the RRT* planner also finds a valid and stable path to the goal with little more cost. To find a path over the hill of rubble (Fig. 6) the roadmap planner required about $7\,s$. The cost of the path is 6.56. In comparison,

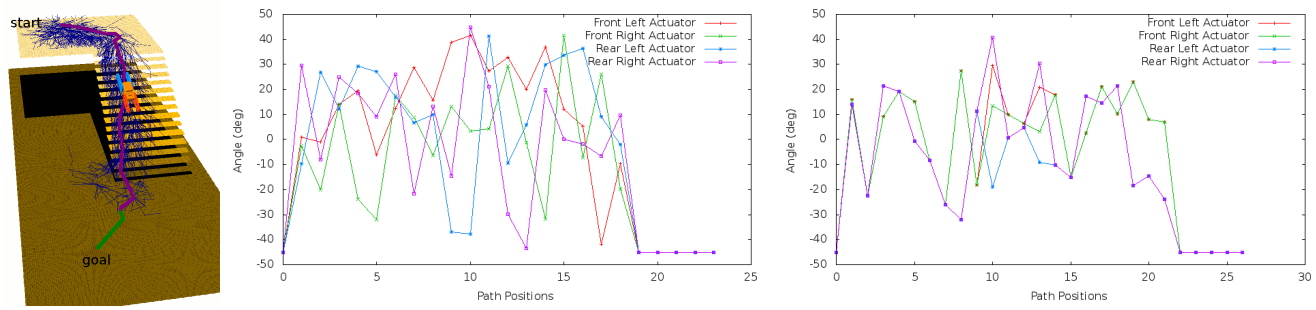

Fig. 5: The left image shows the map with the planning query and a exemplary solution. The plots show the actuator values of a sample solution using no heuristic (middle) and using the two-fold actuator sampling (right). We used the following category probabilities: all actuators equal ($p = 0.20$), front as well as rear actuators equal ($p = 0.65$), left as well as right actuators equal ($p = 0.10$), and all actuators different ($p = 0.05$). We performed 10 queries for each setting. The average cost for the two-fold sampling was 6.18 with a standard deviation of 0.14. The average cost for the non-guided sampling was 6.53 with a standard deviation of 0.12. Planning took about $85\,s$ on average for 2500 iterations. The two-fold actuator sampling generates smoother motions and decreases path costs.

the RRT* planner has an average path cost of 7.35 with a standard deviation of 0.09. The planning time was about $124\,s$ on average. For the second scenario (Fig. 7), the roadmap planner needed about $11\,s$ to find a path of cost 11.53. The RRT*-planner required about $173\,s$ to generate a plan. The average cost is 12.57 with a standard deviation of 0.12. Since our cost function produces values between 0 and 1 and given the path length of the queries, the differences in path quality between the roadmap planner and the RRT* planner are tolerable considering the following advantages of the sampling-based approach.

Our RRT* algorithm has several general advantages over the roadmap planner. The latter must plan on a considerable smaller and discretized state space to be tractable. In contrast, the RRT* planner considers a continuous robot state space. Hence, a solution is not limited to grid positions, which is beneficial especially if the environment involves inclinations and structures that are not aligned with the grid (compare paths in Fig. 7 a and c). Also, planning in continuous state space allows more configurations to be used. Hence, it has the potential to overcome more and harder to traverse obstacles. Finally, the sampling-based approach scales better to large configuration spaces.

V. Conclusion and Future Work

We present a two-phase motion planning algorithm for tracked actively reconfigurable robots to traverse rough terrain. A graph search quickly determines an initial path using the robot's operating limits. Subsequently we use a biased RRT* search to refine rough segments of the initial path in the continuous state space which includes the actuators. Since we do not rely on terrain classifications or motion sequences, our algorithm can be applied to a wide variety of environments.

Implementing the RRT*-based planner on a real robot is the main focus in the future. A more comprehensive identification of the robot's contact points with the environment and overcoming more challenging obstacles with sharp edges as well as control space planning for kinematically feasible paths are interesting extensions of this work.

References

[1] S. Karaman and E. Frazzoli, "Sampling-based Algorithms for Optimal Motion Planning," *Int. J. of Robotics Research*, vol. 30, no. 7, pp. 846–894, June 2011.

[2] A. I. Mourikis, N. Trawny, S. I. Roumeliotis, D. M. Helmick, and L. Matthies, "Autonomous Stair Climbing for Tracked Vehicles," *Int. J. of Robotics Research*, vol. 26, no. 7, pp. 737–758, 2007.

[3] E. Mihankhah, A. Kalantari, E. Aboosaeedan, H. Taghirad, and S. A. A. Moosavian, "Autonomous Staircase Detection and Stair Climbing for a Tracked Mobile Robot Using Fuzzy Controller," in *IEEE Int. Conf. on Robotics and Biomimetics*, 2009.

[4] C. Dornhege and A. Kleiner, "Behavior maps for online planning of obstacle negotiation and climbing on rough terrain," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2007.

[5] R. B. Rusu, A. Sundaresan, B. Morisset, K. Hauser, M. Agrawal, J.-C. Latombe, and M. Beetz, "Leaving Flatland: Efficient Real-Time Three-Dimensional Perception and Motion Planning," *J. of Field Robotics*, vol. 26, pp. 841–862, 2009.

[6] M. Kalakrishnan, J. Buchli, P. Pastor, M. Mistry, and S. Schaal, "Learning, Planning, and Control for Quadruped Locomotion over Challenging Terrain," *The Int. J. of Robotics Research*, vol. 30, pp. 236–258, 2010.

[7] M. Rufli, D. Ferguson, and R. Siegwart, "Smooth Path Planning in Constrained Environments," in *IEEE Int. Conf. Robot. and Autom.*, 2009.

[8] J. Miro, G. Dumonteil, C. Beck, and G. Dissanayake, "A kyno-dynamic metric to plan stable paths over uneven terrain," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2010.

[9] E. Magid, T. Tusubouchi, E. Koyanagi, T. Yoshida, and S. Tadokoro, "Controlled Balance Losing in Random Step Environment for Path Planning of a Teleoperated Crawler-Type Vehicle," *J. of Field Robotics*, vol. 28, pp. 932–949, 2011.

[10] L. Jaillet, J. Cortes, and T. Simeon, "Sampling-Based Path Planning on Configuration-Space Costmaps," *IEEE Transactions on Robotics*, vol. 26, pp. 35–646, 2010.

[11] A. Shkolnik, M. Levashov, I. R. Manchester, and R. Tedrake, "Bounding on rough terrain with the LittleDog robot," *Int. J. of Robotics Research*, vol. 30, no. 2, pp. 192–215, 2011.

[12] A. Ettlin and H. Bleuler, "Rough-Terrain Robot Motion Planning based on Obstacleness," in *9th Int. Conf. Control, Autom., Robot. Vis.*, 2006.

[13] M. Brunner, B. Brueggemann, and D. Schulz, "Autonomously Traversing Obstacles: Metrics for Path Planning of Reconfigurable Robots on Rough Terrain," in *Int. Conf. on Informatics in Control, Automation and Robotics*, 2012, Best Paper Award.

[14] B. Akgun and M. Stilman, "Sampling heuristics for optimal motion planning in high dimensions," in *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2011, pp. 2640–2645.

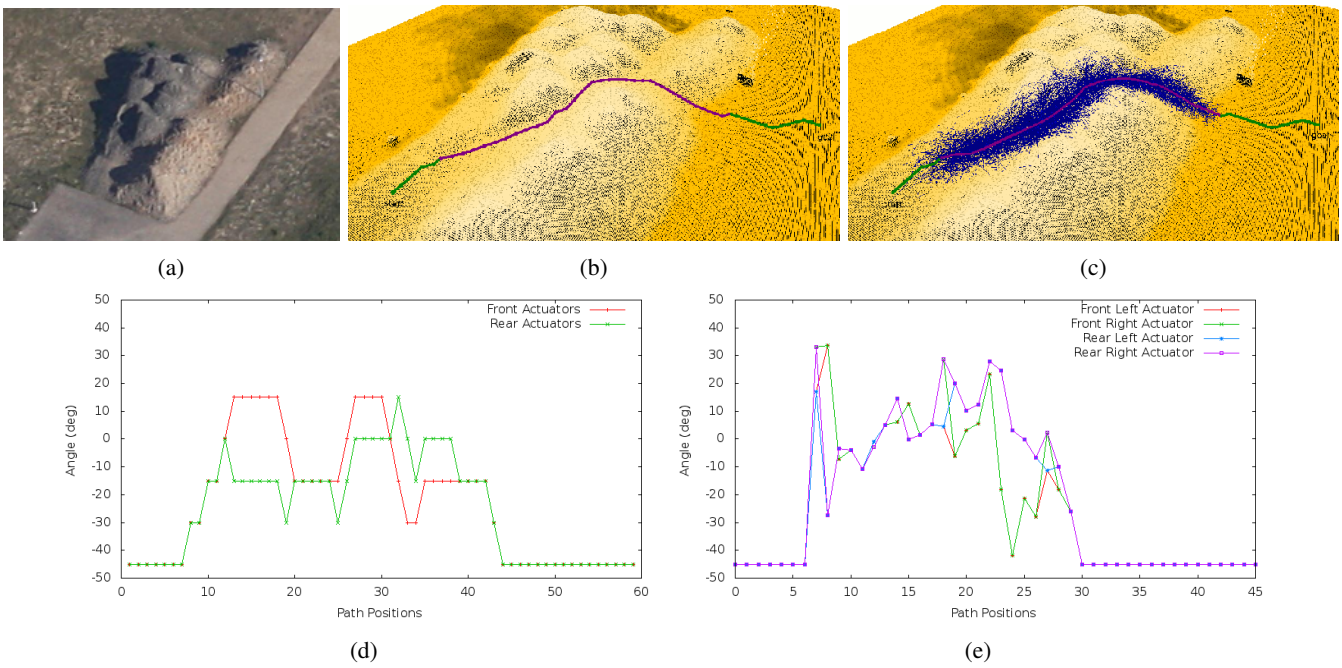

Fig. 6: Results for a hill of rubble. (a) A Google Maps image of the testing environment. (b) The path produced by the roadmap planner. The path cost is 6.56 and the algorithm required about 7 s for planning. (c) A sample path and the planning tree of the RRT* planner after 10000 iterations. The average cost of the path over 10 planning queries is 7.35 (stdev: 0.09) and planning took about 124 s. (d) The actuator values of the roadmap planner path. (e) The actuator values of the RRT* planner path.

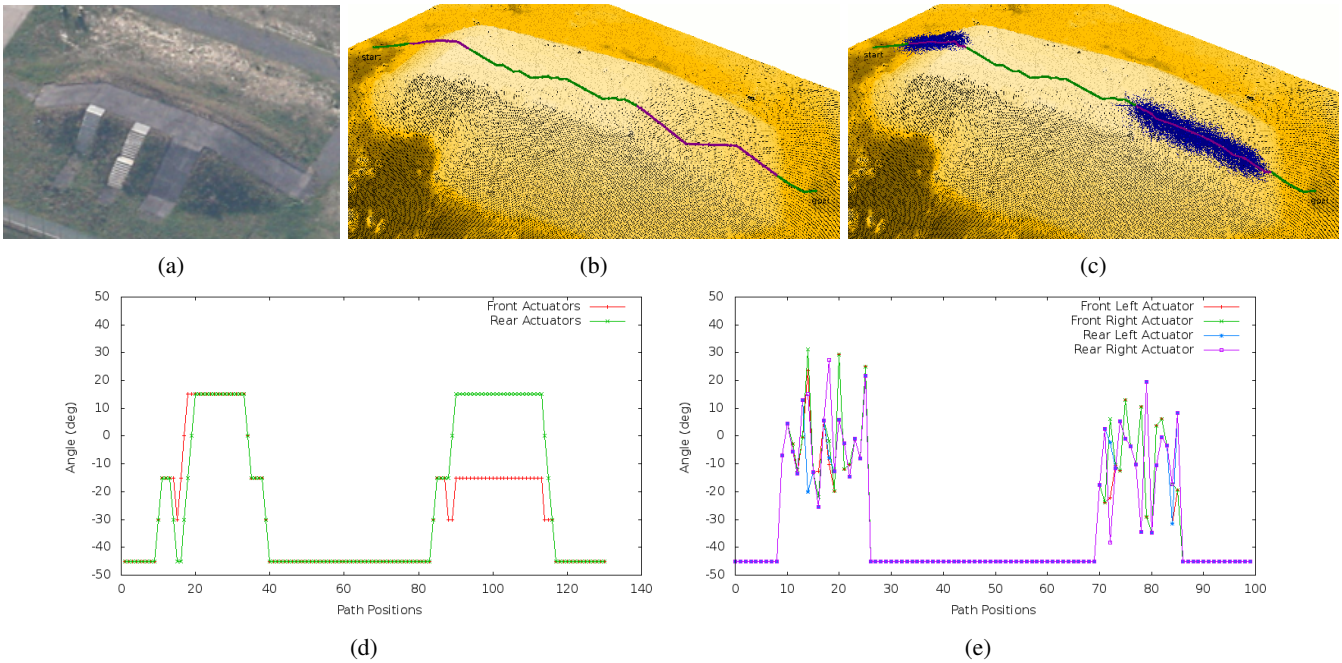

Fig. 7: Results for our testing hill. (a) A Google Maps image of a the testing hill. (b) The path of the roadmap planner with a cost of 11.53. The planning required about 11 s. (c) A sample path and the planning tree of the RRT* planner after 10000 iterations. The average path cost over 10 planning queries is 12.57 (stdev: 0.12) and planning took 173 s. (d) The actuator values of the roadmap planner path. (e) The actuator joint values of the path produced by the RRT* planner.