Autonomously Traversing Obstacles
Metrics for Path Planning of Reconfigurable Robots on Rough Terrain

Michael Brunner, Bernd Brüggemann and Dirk Schulz
Unmanned Systems Group of the Fraunhofer Institute for Communication, Information Processing and Ergonomics FKIE, Wachtberg, Germany

Keywords: Metric, Traversability, Obstacle, Rough Terrain, Reconfigurable Chassis, Motion Planning, Mobile Robot, Autonomy.

Abstract: The fixed chassis design of commonly employed mobile robots restricts their application to fairly flat environments, as the wheel diameters or the track heights impose hard limits on their mobility. Unstructured outdoor and urban environments alike comprehend many different invincible obstacles for most of those systems, like stairs, boulders or rubble. However, there are mobile robots with reconfigurable chassis providing a higher degree of mobility and enabling them to overcome such obstacles. Yet, current planning algorithms rarely exploit those enhanced capabilities, limiting these systems to the same environments as the fixed chassis robots. This paper focuses on the metrics used by our motion planner. The employment of a two-stage planning approach allows us to use different cost functions for the initial path search and the detailed motion planning step. The purpose of the initial search is to quickly find a fast environment-driven path to the goal. Hence, it uses fast computable heuristics to assess the drivability, i.e. a risk quantification and the utmost operation limits of the robot model. The detailed planning step determines the desired robot configurations. For this purpose, we consider the actuator controls, the system's stability, an estimate of the traction, and the driving speed in addition to the quantities used in the first stage.
We present experiments to illustrate the influence of the safety weights and real world experiments which prove the validity and feasibility of the metrics used by our motion planning algorithm.

1 INTRODUCTION

Fixed chassis robot platforms are commonly used today. These platforms are limited to fairly flat areas, as their design prevents them from traversing structures with high edges or steep inclinations. Unstructured outdoor and urban environments alike include many different invincible obstacles for most of those robotic systems, like stairs, boulders or debris. However, mobile robot with reconfigurable chassis exist. Altering the system configuration enhances the mobility of the robot and enables it to traverse a wide variety of obstacles, which would be insuperable otherwise.

To drive a mobile robot across the above mentioned obstacles is a challenging task even for a trained operator. There are many aspects which have to be considered when driving over obstacles, most of which can be neglected for 2D navigation on flat terrain. Especially, the stability of the robot is essential as the robot falls over more easily. Inertia and momentum are increasingly important when a fast robot is operated close to its limits. Moreover, varying contact points of the mobile system change the system's behavior to actuator and driving commands.

Figure 1: The Telemax robot model, a tracked platform with four actuators, enabling the system to overcome more complex obstacles than many other robot systems.

Introduced planning and controlling algorithms for traversing rough terrain or overcoming obstacles depend heavily on the robot's shape, actuators, and abilities. This may be comprehensible by considering that these systems ought to traverse very rough terrain for which they must operate at their limits. This, in turn, requires to exhaust the often unique capabilities

Brunner M., Brüeggemann B. and Schulz D..
Autonomously Traversing Obstacles - Metrics for Path Planning of Reconfigurable Robots on Rough Terrain.
DOI: 10.5220/0004033400580069
In *Proceedings of the 9th International Conference on Informatics in Control, Automation and Robotics* (ICINCO-2012), pages 58-69
ISBN: 978-989-8565-22-8

given by their specific actuators. However, more general algorithms often apply only to a subset of robots as the designs of mobile robots differ vastly and use specific mechanisms for obstacle traversal.

We developed a two-stage motion planning approach for reconfigurable robots to traverse rough, unstructured outdoor environments as well as to overcome obstacles in urban surroundings. We generate a preliminary path using the platform's operating limits instead of the complete robot state and, subsequently, we apply a detailed motion planning step to refine the initial path in rough regions. Our algorithm is general in the sense that we do not rely on predefined motion sequences or obstacle classifications. Furthermore, our algorithm can be used with different robot models if given a model that can be queried for stability.

The metrics used by our motion planner are at the center of this paper. Many different metrics and cost functions are proposed for path planning in 2D navigation as well as for rough terrain traversal. Some quantities, like time or path length, are applicable to both scenarios. In 2D navigation, aspects like stability, traction or risk are usually neglected because they can safely be assumed. In contrast, quantifying these aspects during planning on rough terrain is essential. In our approach, we use a risk assessment to adjust the robot's actions. If performed solely, actuator movements do not increase the path length, thus, we measure the path execution time to account for these actions. In rough areas we resort to often employed quantities, e.g. stability or traction, and incorporate them as costs into our approach to determine appropriate robot configurations.

The remainder of this paper is structured as follows: section 2 provides an overview of some related work. In section 3 we describe the robot model used for this work. We give a brief scheme of our algorithm in section 4. Section 5 discusses issues of the sensor coverage for obstacle traversal, the necessity of global information, and introduces the risk assessment. Sections 6 and 7 illustrate the metrics used for the initial path search, and the metrics utilized by the detailed planner in hazardous areas, respectively. We provide an evaluation of the safety weights and real world experiments in section 8 and conclude in 9.

2 RELATED WORK

Extensive work has been done on traversability metrics and cost functions for path planning of mobile robots. A binary notion of traversability and the path length are generally accepted as sufficient measures in 2D navigation. However, as the terrain becomes more challenging, more detailed traversability assessments are used and further aspects, like the stability, the amount of turning, or traction are considered during path planning. In this section we outline some of the previous work done in this area of research.

The National Institute of Standards and Technology (NIST) has introduced stepfields as a means of repeatable test methods for robot mobility to capture statistically significant performance information (Jacoff et al., 2008). The NIST also proposed three metrics for stepfields; two concerning the coverability of the terrain, i.e a difficulty measure of the entire region, and one called crossability which depicts the difficulty to move between two specified locations. While the coverability is based on the variations in height difference, the crossability is given by the least cost path in terms of the terrain roughness along the path, the amount of turning and the path length. All metrics are scaled by the robot size and wheel diameter or track height since these parameters influence the ability of the robot to overcome obstacles (Molino et al., 2007). The NIST's crossability metric is based on (Iagnemma and Dubowsky, 2004) where the same quantities are used to include the robot model into the traversability measure. Since we intent to use the traversability measure to guide our search algorithm, a global definition like the coverability is not suitable. Similarly, we consider roughness, the amount of turning and the path length for planning. In contrast, since the robot size and wheel or track height are not fix for robots with reconfigurable chassis, we do not include these parameters. Further, we measure the execution time of a path rather than the path length to account for actuator movements which do not influence the path length, but require time.

In the area of planetary rovers most of the proposed traversability concepts are heavily based on terrain, robot and dynamic models to capture the terrain-vehicle interactions in detail (Howard and Kelly, 2007). Mechanical models are used to identify the cohesion and internal friction angles of the terrain in order to quantify traversability (Iagnemma et al., 2004). The Traversability Index is a fuzzy rule-based measure quantifying the slope of the terrain and the roughness with respect to the size and density of rocks within the camera frame (Seraji, 1999). It was extended in (Howard et al., 2001), additionally incorporating the terrain discontinuity (i.e. cliffs) and hardness as it affects traction. In our approach we also incorporate an estimate of the vehicle traction, however, it is less model based.

Many different stability margins were proposed for measuring the stability of a mobile robot. A

comparative overview can be found in (Garcia et al., 2002). A common way to measure the static stability of a system is to project the center of mass onto the supporting polygon. Other stability margins are more accurate as they consider the height of the center of mass. In (Miro et al., 2010) the Force-Angle Stability Margin (FASM) is incorporated as slowness value into the Fast Marching Method (FMM) to favor more stable paths. As we do not contemplate any forces at this point we use the Normalized Energy Stability Margin (NESM) (Hirose et al., 2001), also employed in (Magid et al., 2008).

Others classify the environment using fuzzy rules and Markov Random Fields to generate so called behavior maps which encode preconditions and costs for a skill-based traversal algorithm (Dornhege and Kleiner, 2007). Or they label the terrain through geometric heuristic rules as flat, vertical, stairs or unknown along with associated costs (penalizing steep slopes and rough terrain) to apply specific motion primitive planners for each class (Rusu et al., 2009). However, we do not want to base our motion planning algorithm on a structure classification scheme as this will limit the algorithm to the set of defined structures. Hence, our metrics solely use general properties, rather than semantic structure labels.

3 ROBOT PLATFORM

We use a Telerob Telemax robot model in our research, see figure 1. The robot is $60\,cm$ long, $40\,cm$ wide, and weighs about $70\,kg$. It has 4 tracks which can be rotated 170° from entirely hinged all the way down lifting the robot about $45\,cm$ up. Completely stretched the robot has a length of $120\,cm$. The robot is equipped with a skid-drive, and its maximal translational speed is $1.2\,\frac{m}{s}$.

4 BRIEF ALGORITHM OVERVIEW

In this section we give a short overview of our algorithm. We employ a two-stage planning algorithm for reconfigurable robots to traverse rough environments and to overcome obstacles in urban areas. Given a map we first compute the risk distribution (section 5) and build a motion graph according to the robot's operating limits (section 6). We perform a Dijkstra search to find the initial path in this motion graph. Afterwards, we identify the path segments leading through rough terrain and construct a state graph for a tube-like area around each rough path segment (section 7). Using a Dijkstra search we find sequences of robot states including the desired actuator positions. Finally, applying a default configuration for segments in flat areas, all segment paths are combined to provide the final path.

5 MAP AND RISK DISTRIBUTION

Whether a given structure is traversable or not, is not easily determined. In 2D navigation, this is usually handled with a simple threshold on the height differences; everything above this threshold is invincible. When aiming at overcoming structures, this question becomes very hard to answer. For 2D navigation a 2D laser range finder is sufficient to gather the necessary information about the surroundings. In contrast, using a 3D sensor for traversing obstacles and navigating through rough environments is not enough, due to the still very limited sensor coverage.

First, installing sensors on a mobile robot introduces a very narrow view; second, determining what will come after successfully climbing onto an obstacle is very difficult when using common sensors; and third, while traversing an obstacle the robot's pose will often orient the sensors in a way that they are unable to cover the environment. For example, consider the traversal of a flight of stairs; the very narrow view makes it hard to recognize the stairs especially all the way up, and while on the stairs and close to the top the sensors cover very few of the ground. This means, deciding on the traversability of an obstacle based on local sensor information is very hard. Therefore, we came to the conclusion that traversing rough terrain and obstacles cannot be addressed without global information like a map. Means of how to deduce the traversability of a greater area from local sensor information is beyond the scope of this work.

Further, discovering that an obstacle is actually invincible during traversal is an unpleasant situation. In this case the robot may be on top of an obstacle and be required to retreat backwards or, worse, since obstacles are generally not traversable from every angle, the robot may be stuck and, thus, unable to move safely off the obstacle. A map may help reducing such incidents but certainly cannot account for all situations, as a map usually is not detailed enough and physical aspects, like friction coefficients, are generally unavailable.

Also related to the first point, especially for rough environments it cannot be assumed that a path to a desired goal exists at all. Moreover, a map allows to as-

sess the risk of a path and, thus, enables to determine whether driving through a hazardous area is worth the risk or circumventing the obstacle with reasonable additional costs is more appropriate.

In addition, a map may be used to improve localization by comparing the robot's actual pose with the expected pose deduced from the map. However, the localization and an enhanced controller are subject to future work.

Using a map for motion planning in rugged environments introduces another problem. The validity of the planning is closely related to the level of detail of the map. However, large detailed maps are rarely available. This can be addressed by detailed patches for rough regions and a greater coarse map.

5.1 Risk Distribution

In our approach we use a heightmap to represent the environment because it is simple to use and sufficient for our application. In order to assess the difficulty of a position within the map, we use techniques from image processing to compute a risk distribution of the map. As indicator how difficult an area of the map is, we use the height differences between cells. For a given map cell $c(x,y)$ we determine the maximal height difference within a window w of size $k_x \times k_y$ around the cell, i.e., we apply a maximum filter

$$c(x,y) = \max_{(i_1,j_1),(i_2,j_2)\in w_{x,y}} \{|c(i_1,j_1) - c(i_2,j_2)|\}$$

with

$$w_{x,y} = \{(i,j) : |x-i| \leq k_x \wedge |y-j| \leq k_y\}.$$

However, we prevent a distortion of the range of values through a threshold h_{max}, which conveniently can be set to the robot model's maximal traversable height. Further, the values are scaled to $[0,1]$ using h_{max}. Subsequently, we perform a Gaussian blur on this grid of height differences to smooth the transitions and, more importantly to propagate the risk, i.e. inflate large heights and sharp edges. The Gaussian kernel for a two dimensional blur is defined as

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{\sigma^2}}.$$

Given kernel sizes of k_x and k_y for both dimensions, this leads to a matrix $G \in M(k_x+1, k_y+1)$ with elements

$$G(i,j) = \frac{1}{2\pi\sigma^2} e^{-\frac{\left|i-\frac{k_x}{2}\right|^2 + \left|j-\frac{k_y}{2}\right|^2}{\sigma^2}},$$

where $k_x/2$ and $k_y/2$ indicate integer divisions. For each cell in the map we convolve this kernel matrix with the window of the same size around the cell. As the height differences directly correspond to the risk for the system, we call this smoothed grid of maximal height difference a risk distribution. Figure 2 shows the final risk distribution of some maps.

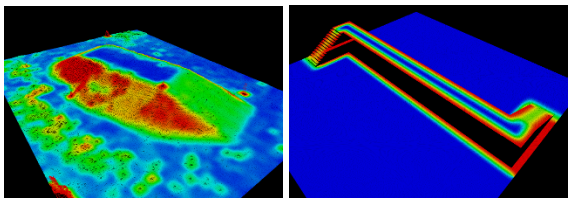

Figure 2: Risk distributions of two different maps; left of a real map build by a laser range finder and right for an artificial urban map. The risk of a region is based on the height differences in this area. The roughness quantification is used to control the planning process. The colors are indicating the degree of risk, ranging from blue for flat regions over green and yellow to red, high risk areas.

Note, comparing, for instance, a flight of stairs to a ramp of the same gradient angle, the stairs provides less contact points for traversal with a tracked robot, and turning is much more severe on these discrete contacts. Hence, a flight of stairs should have a higher risk compared to a ramp of the same gradient angle. Using the introduced formulation allows us to distinguish between those two structures in exactly this way. Moreover, using an appropriate window size allows us to virtually inflate the hazardous areas, which is commonly done in 2D navigation to keep the robot away from obstacles. In contrast, high risk areas are avoided by the robot but if required, do not prohibit traversal. Another benefit is that the computation is simple and highly parallelizable.

We use this risk distribution in both metrics, for the initial path search as well as for the detailed motion planning, to adjust the behavior according to the difficulty of the environment.

6 METRICS FOR THE INITIAL PATH SEARCH

Driving with reconfigurable robots on rough terrain and over obstacles introduces a large search space and more constraints compared to 2D navigation must be satisfied. Additionally, 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 but also by the robot's stability and traction, the amount of turning and time required for actuator movements. We, therefore, employ a first path search to quickly find an environment-driven and fast path to the desired goal location. The path is subsequently used to

restrict the search space for the second planning phase which determines the final path consisting of the robot configurations including the actuator commands.

Our initial path search employs the introduced risk distribution. This roughness quantification is used to steer the robot away from hazardous areas and to prefer less risky routes. Usually, an environment consists of fairly flat areas as well as rugged and challenging parts. In flat areas, the consideration of the robot's complete configuration including actuator values is not necessary. In contrast, it is essential in rough areas to increase robot safety and ensure successful traversal. At this planning phase, we do not know through which parts of the environment the path will lead, therefore, we omit the complete state during this planning stage. We rather stick to the utmost operating limits of the mobile base neglecting the actuators. These operation limits do include the maximal roll and pitch angles before tipover as well as the maximal traversable height. We identify the least restrictive operating limits through the most stable configuration on flat ground. However, using a greater set of configurations generally improves the robot states in terms of stability as more configurations can compensate for a greater variety of situations.

We build a motion graph $G_m = (V_m, E_m)$ which represents the ability of the mobile robot to traverse the environment. The vertices $v_i \in V_m$ model positions p_i of a dense[1] regular grid. A vertex v is added to V_m if the maximal risk value r within the robot's footprint does not exceed some threshold. If the transition from a position p_i to a neighboring position p_j does not violate the robot's limits in terms of inclination and height differences, the edge $e_{ij} = (v_i, v_j)$ is included in E_m. Examples of motion graphs are given in figure 3.

Using the risk distribution, we are able to define the required time $t_v(ij)$ to move from a position p_i to a neighboring position p_j as a simple function of the risk. Let d_{ij} be the distance between p_i and p_j, and let $r_{ij} = \max(r_i, r_j)$ be the higher risk of the involved positions. Then t_v is defined as

$$t_v(i,j) = \frac{d_{ij}}{\max\left(v_{min}, (1 - r_{ij} \cdot w_{\text{is}}) \cdot v_{max}\right)}, \quad (1)$$

where v_{min} and v_{max} are the minimal and maximal forward velocity, respectively. $w_{\text{is}} \in [0,1]$ is the safety weight for the initial path search; low values will diminish the influence of the risk distribution and, hence, lead to possibly shorter, yet more risky paths.

[1]The resolution of the grid is usually half the robot size to avoid the requirement of intermediate validity test. Also, situations in which solutions are lost due to the discretization are few.

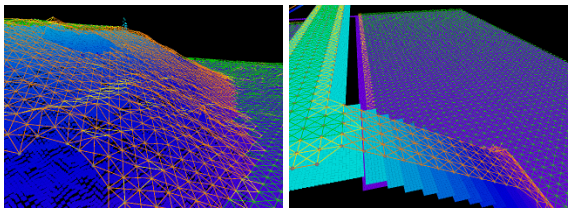

Figure 3: Examples of motion graphs for two different maps; left for a map of a training hill and right for an artificial urban map. The motion graphs encode whether the given robot model is able to traverse the terrain. Areas with risk values and slopes below a convenient level are green, areas with higher risk levels and convenient slopes are yellow, and areas with higher risk values and higher slopes are orange.

Whereas, high values increasingly force the robot to take low risk paths. Note, the less risky the area, the faster the robot can drive and the shorter the time. By using a graph based formulation and encoding the costs in the edge weights, we are able to perform a Dijkstra search.

The main purpose of this planning phase is to find a path to the goal which can be used to restrict the search space for the subsequent detailed motion planner. Therefore, we differentiate between areas with risk values and slopes below a convenient level, areas with higher risk levels and convenient slopes, and areas with higher risk values and higher slopes (see figure 3).

We split the path into segments leading through flat areas with low risk or no environmental slopes and segments through rough regions with high risks and slopes. See figure 4 for examples. For flat segments the stability of the robot system can be safely assumed as done in 2D navigation. Further, any robot configuration may be applied with no or little risk (given the pose is stable in itself). Therefore, we do not perform a detailed motion planning of the robot's actuator configurations for flat segments. However, rough regions require an additional planning of the robot's actuator controls and the consideration of the stability to ensure safety and task completion. This planning phase and the metrics used are described in more detail in section 7. In regions with higher risk values but still convenient slopes we reduce the velocity of the robot.

Distinguishing between flat and rough areas during this planning stage allows us to speed up planning since we are avoiding unnecessary planning in a high dimensional space for easily accessible parts of the environment. However, we ensure the stability and safety of the robot through a detailed planning step for rough path segments. Disregarding the robot state and using the operating limits simplifies the traversability assessment, hence, the initial path search is fast.

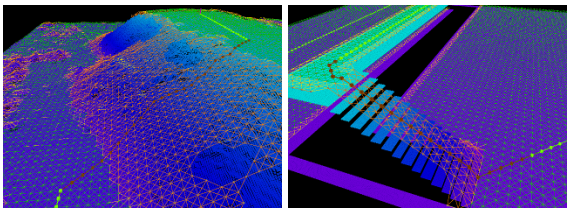

Figure 4: Path segments of the initial paths for two planning queries on different maps. Segments through flat regions which do not require further planning are shown in light green, segments through rough regions will be refined in the second planning phase and are shown in dark red.

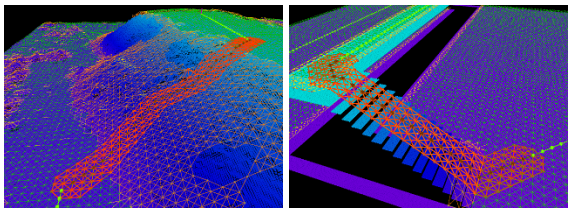

Figure 5: Two examples of tubes (light red) around the initial paths in rough areas. The tubes are used to restrict the search space of the detailed motion planning step.

7 METRICS FOR PLANNING IN ROUGH AREAS

Compared to flat environments, driving on rough terrain is more challenging and exposes the mobile robot to a greater risk. Therefore, using simply the operating limits of the robot is not sufficient. We need to consider the complete robot configuration including actuator values to estimate the robot's stability and traction instead. The configuration of a reconfigurable robot may look like

$$(x, y, z, \theta, \psi, \phi, v, \omega, \dot{v}, \dot{\omega}, a_1, \ldots, a_n),$$

where the first part describes the 6D pose of the robot. The translational and rotational velocities are v and ω, and the corresponding accelerations are depicted by $\dot{v}$ and $\dot{\omega}$. a_i are the control values of n actuators. The stability or the contact points with the environment may also be included in this state. Reducing the state vector to the controllable part leads to

$$(x, y, \theta, v, \omega, \dot{v}, \dot{\omega}, a_1, \ldots, a_n).$$

The controllable states still build a large and high dimensional search space, which cannot be searched exhaustively. Sample-based methods are usually employed to handle high dimensional search spaces. However, sample-based methods provide the first sampled solution, i.e. not necessarily a optimal or even good. Further, due to their random nature, those methods struggle to find a solution if only a few exist at all, i.e. they suffer from the narrow passage problem. Addressing the narrow passage problem and applying sample-based methods to optimal control is an active field of research. Due to the fundamental issues, we consider sample-based methods as not suited for solving the problem of traversing rough terrain and obstacles.

Consequently, we introduce a graph-based approach using the initial path as basis to restrict the space of robot states to a tube around this path (see figure 5). We, therefore, assume that the best path with respect to the complete robot configurations complies with or is close to the initial path. This assumption mostly holds. It would be violated primarily if time consuming actuator movements are necessary by which no distance is gained. This rarely happens since, first, we advocate simultaneous execution of movements (as described later) and, second, the initial path prefers less risky routes which generally require less actuator movements. Finally, the best path considering the robot's entire configuration must also be a fast path.

By using a subsequent planner to refine the path in rough areas, we are able to apply another cost function. This, in turn, allows us to increase the importance of the robot's safety. The detailed motion planning accounts for the system's stability and traction and for the time consumed by rotation and actuator movements in order to prevent unnecessary actions. Since the robot's speed is very low when traversing hazardous areas, we neglect forces and dynamic stability for now. The terms of the cost function for this planning phase can be divided into a safety term and a time term.

First, we define the state graph $G_s = (V_s, E_s)$ which models a discrete subspace $X_s \subset X$ of the state space. Each vertex $v_i \in V_s$ corresponds to a state $x_i \in X_s$. Each edge $e_{ij} \in E_s$ models a valid transition from state x_i to x_j. The validity of a transition is subject to the movement constraints of the robot model. The edge weights are given by the cost function derived in this section. In the following we will use the state space notation to introduce the metrics.

7.1 System Safety Metric

The safety of the system is affected by several factors. We incorporate the risk assessment, the system stability and an estimate of the traction into the safety metric. To quantify these values we approximate the robot's footprint by the best fitting plane using a least-squares method.

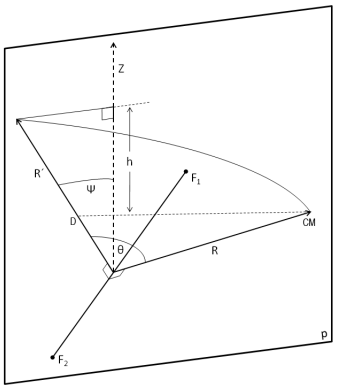

Figure 6: Illustration of the Energy Stability Margin. F_1 and F_2 represent a border of the supporting polygon, i.e. a rotation axis within the plane p. R is the vector from the border to the center of mass (CM). Θ depicts the angle between R and the vertical plane and ψ the inclination of the rotation axis with respect to the horizontal plane. R' is obtained by rotating R around the rotation axis until it is contained in plane p. D is computed through $D = |R|(1 - cos(\Theta))$ and $h = |R|(1 - cos(\Theta)) \cdot cos(\psi)$ provides the energy stability level. Finally, the NESM is defined as $s = min_i(h_i)$ with h_i being the normalized energy level of the ith edge of the supporting polygon.

7.1.1 System Stability

We assess the static stability of the robot using the Normalized Energy Stability Margin (NESM), see (Garcia et al., 2002) for a comparative overview of several stability margins. In contrast to the commonly used projection of the center of mass onto the supporting polygon, the NESM directly provides a notion of quality.

We will give a short overview of the Normalized Energy Stability Margin, for a detailed discussion see (Hirose et al., 2001). The NESM basically indicates the amount of energy which must be applied to tip the robot over the "weakest" edge of the supporting polygon. It is derived from the Energy Stability Margin (ESM) (Messuri, 1985) which is illustrated in figure 6.

The rotation axis is given by F_1 and F_2 as a border of the supporting polygon within the plane p. R is the vector from the border to the center of mass (CM). The angle between R and the vertical plane is given by Θ, and ψ depicts the inclination of the rotation axis with respect to the horizontal plane. R' is obtained by rotating R around the rotation axis until it is contained in plane p. D is computed through $D = |R|(1 - cos(\Theta))$. Finally,

$$h = |R|(1 - cos(\Theta)) \cdot cos(\psi) \tag{2}$$

provides the energy stability level for the given rotation axis. The NESM is now defined as

$$s = min_i(h_i), \tag{3}$$

where h_i is the normalized energy level with respect to the ith boundary of the supporting polygon. Our stability cost is the stability of a robot state x, i.e.

$$S = 1 - \xi_s \cdot s(x), \tag{4}$$

where $\xi_s = \frac{1}{s_{max}}$ is a normalization term using the maximal stability value of a given robot model to scale the cost to $[0,1]$.

The computation of the stability of the system depends on the accuracy of the center of mass. Therefore, we compute the distributed center of mass, as

$$CM = \sum_{i=1}^{n} c_i \cdot m_i, \tag{5}$$

where CM is the position of the center of mass of the system. c_i and m_i are the centers of mass and the masses of the n body parts, i.e. the chassis and the actuators. Thereby, we can determine the center of mass with respect to the actuator positions which increases the accuracy of the position of the center of mass.

Due to the minimum function in the stability margin the stability value may stay the same for several actuator positions. In order to reach a more stable configuration the robot might have to perform a sequence of actuator movements which do not have an immediate gain in stability. Therefore, we need another quantity which helps to pass through such sequences. We decided to use an estimate of the robot's traction.

7.1.2 Traction Estimate

The traction of the robot is increasingly important when traversing rough terrain or obstacles. However, we do not want to use any information about the surface properties because maps with information accurate enough to aid planning are very hard to obtain. Therefore, we estimate the actuators' ground contact as an indicator of the traction. The ground contact of an actuator a_k is defined as its angle to the surface. Then, the traction cost T of a state $x \in X_s$ is given as the average over those n angles.

$$T = \xi_t \cdot \frac{1}{n} \sum_{i=1}^{n} \psi(a_k), \tag{6}$$

where $\psi(a_k)$ is a function providing the angle to the surface of an actuator a_k. $\xi_t = \frac{1}{\pi/2}$ normalizes the cost to $[0,1]$. Hence, the smaller the angle, the greater the estimated traction, the safer the robot state in terms of

traction. Note, measuring the ground contact of the actuators in terms of their angle to the surface provides a qualitative measure of the traction. It can be viewed as an indicator for traction as the traction between two objects generally increases with the contact area between those objects. Also, this estimate serves our purpose of lowering the robot's actuators on rough terrain.

The combination of the stability cost, the traction cost and the risk value of the roughness quantification constitutes the safety term c_{safety} of our cost function. Given two states $x_i, x_j \in X_s$ the safety cost is defined as

$$c_{\text{safety}}(i,j) = \frac{r_{ij} + \frac{1}{2}(S_{ij} + T_{ij})}{2}, \tag{7}$$

where $r_{ij} = \max(r_i, r_j)$ is the higher risk of both states. Similar, $S_{ij} = \max(S_i, S_j)$ is the greater stability cost of both involved states, and $T_{ij} = \max(T_i, T_j)$ is the maximum traction cost.

7.2 Execution Time Metric

Besides a save path we also want to find a fast path to the goal. In addition, we consider the time required for rotation and actuator movements which the setup of the initial path search does not facilitate. For the initial path search we defined the translational velocity as a function of the maximum velocity and the roughness. Since we now have a better understanding of the risk of a robot state, we use the safety cost c_{safety} of the previous section to adjust the velocity.

$$t_v(i,j) = \xi_v \cdot \frac{d_{ij}}{\max\left(v_{min}, (1 - w \cdot c_{\text{safety}}(i,j)) \cdot v_{max}\right)}, \tag{8}$$

where d_{ij} is the distance between x_i and x_j, and v_{min} and v_{max} are the minimal and maximal forward velocity, respectively. $w \in [0,1]$ is the safety weight of the second planning phase and controls the impact of the state safety on the planning. This term is also normalized to $[0,1]$ using $\xi_v = \frac{1}{\max_{i,j} t_v}$.

The maximal physically possible rotational velocity depends on the ground contact of the robot's actuators. For instance, if the actuators of the Telemax model are completely stretched with a total length of $1.2\,m$, rotating is practically impossible. We performed several experiments to determine the rotational velocities given the actuator positions and composed a look-up table (for continuous values this would be a function). Please note, the quantities are sufficient for planning at this level of detail, even though they are experimentally established on flat ground and different surface frictions are neglected. The essential information these values provide is that the maximum rotational velocity varies with the robot configuration, and some configurations are more qualified for turning than others. Using this information the time required for turning from state x_i to state x_j is given by

$$t_\omega = t_\omega(i,j) = \xi_\omega \cdot \frac{|\theta_i - \theta_j|}{\frac{1}{2}(\omega(a_i) + \omega(a_j))}, \tag{9}$$

where θ_i and θ_j are the orientations of the poses corresponding to x_i and x_j, respectively. $\omega(\cdot)$ provides the rotational velocity of an actuator configuration, i.e. in our case the value of the look-up table. Further, $a_i = (a_1(i), \ldots, a_n(i))$ is the vector of actuator controls of x_i; a_j is defined accordingly. $\xi_\omega = \frac{1}{\max_{i,j} t_\omega}$ normalizes the time values to $[0,1]$.

Similar to the previous formulas, we define the cost of the actuator movements as the normalized movement time.

$$t_a = t_a(i,j) = \xi_a \cdot \frac{|a_i - a_j|}{\frac{1}{2}(v_a(i) + v_a(j))}, \tag{10}$$

where a_i and a_j are defined as before and $v_a(\cdot)$ is the actuator velocity. Again, $\xi_a = \frac{1}{\max_{i,j} t_a}$ scales the values to $[0,1]$.

To save execution time we would like the system to perform actions simultaneously. Therefore, the formulation of the time value of our cost function favors simultaneous execution by implementing the triangle inequality. We omitted the state variables i and j for simplicity.

$$c_{\text{time}}(i,j) = \frac{t_v^2 + t_\omega^2 + (1 - w \cdot c_{\text{safety}})^2 \cdot t_a^2}{t_v + t_\omega + (1 - w \cdot c_{\text{safety}}) \cdot t_a}. \tag{11}$$

The time value measuring the actuator movements is scaled by the inverse safety cost since in hazardous areas and especially in unstable states, we want the system to apply actuator adjustments which improve the robot state even if they are time consuming. Note, this cost function is only applied in rough regions and may not be applicable in flat areas. In flat areas actuator movements are generally unnecessary and solely introduce costs, thus, should not be favored.

The cost function used to determine the edge weights consists of the safety metric and the time measure. The trade-off between safety and speed can be adjusted by $w \in [0,1]$ for different mission goals. Given two neighboring states $x_i, x_j \in X_s$ the cost of the transition is defined by

$$c(i,j) = w \cdot c_{\text{safety}}(i,j) + (1 - w) \cdot c_{\text{time}}(i,j). \tag{12}$$

Note, since all quantities are normalized, the value of the cost function is also bound to $[0,1]$. It is important to state here, that even though we introduced

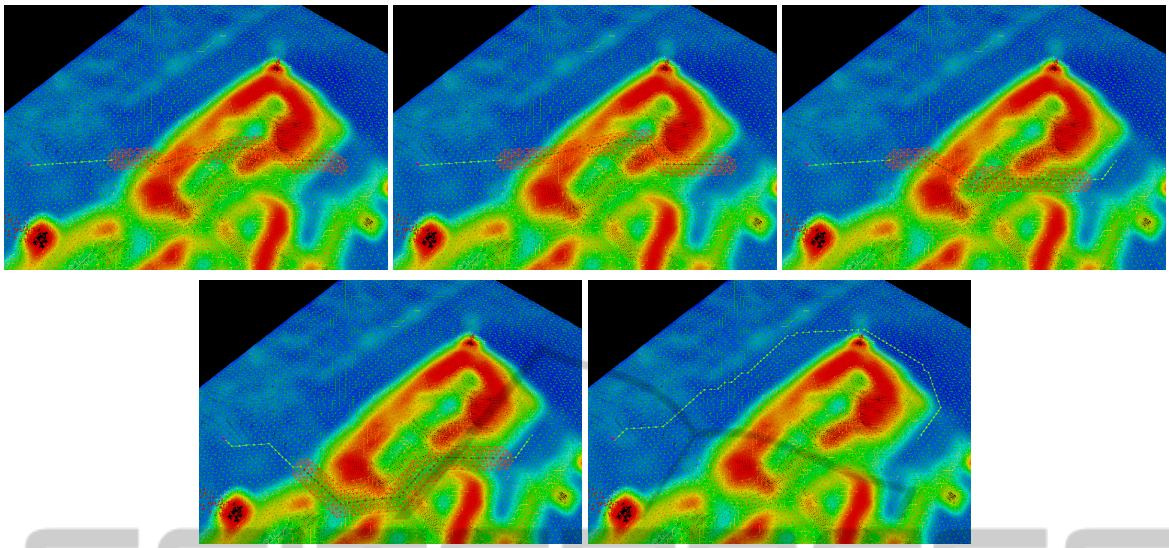

Figure 7: Influence of the safety weight of the initial path search. It determines the primary direction of the final path.

several normalization terms, the only arbitrary values are the minimal velocity v_{min} and the safety weights w_{is} and w. All other values are identified by the robot model and its abilities. Using a graph-based model with the defined edge weights allows us to perform a Dijkstra search on the state graph G_s to find the most stable, yet, fastest path to the goal considering the complete robot state.

8 EXPERIMENTAL RESULTS

In this section we present several experiments. First, we illustrate the influence of both safety weights on the initial path search w_{is} and the detailed motion planning step w. Second, we preformed tests on real-world maps with a real robot to prove that our metrics and plans are valid and executable. To the best of our knowledge, there are no other approaches which aim at overcoming arbitrary obstacles with similar reconfigurable robots. Hence, we do not provide a comparison with other approaches. However, it would be possible to compare single aspects of our metrics but we consider this unfit to provide any insight of how well our metrics are suited as a whole for obstacle traversal compared to others.

8.1 Impact of the Safety Weights

The safety weight of the initial path search w_{is} determines the main direction of the final path. Small weights will lead to short paths, higher values will cause low risk paths. Figure 7 shows the different paths obtained for a given start and goal position and the safety weights 0.0, 0.25, 0.5, 0.75 and 1.0. As one can see for a weight of 0.0 the path leads to a direct path to the goal position. Increasing the safety weight forces the algorithm to face the direction of the steepest gradient when climbing inclinations and to avoid more and more high risk areas (red). Eventually, the path avoids even less risky regions (green) and circumvents the hill.

While the safety weight of the initial path search determines the primary course of the final path, the safety weight of the detailed motion planning step w influences the driving velocity and the applied actuator configurations. With an growing importance of the robot safety the translational velocity decreases and the changing actuator positions becomes cheaper. Also, the time required to move the actuators amplifies the execution time. Therefore, with an increasing safety weight the execution time rises and the safety cost declines. This trend is shown in figures 8 and 9. For these experiments we restricted the detailed motion planning to the initial path in order to prevent the path adjustments within the tube to disturb the results and build a common basis for the comparison.

8.2 Real-world Experiments

We performed tests with the Telerob Telemax model to prove that the plans proposed by our motion planner are valid and executable by a real robot. The environments are shown in figures 10 and 12. Both maps were recorded using a laser range finder and were subsequently filled and smoothed to facilitate planning. Their sizes are $36.4 \times 30.45\,m$ and $43.95 \times 32.95\,m$ respectively. For those tests we used the following

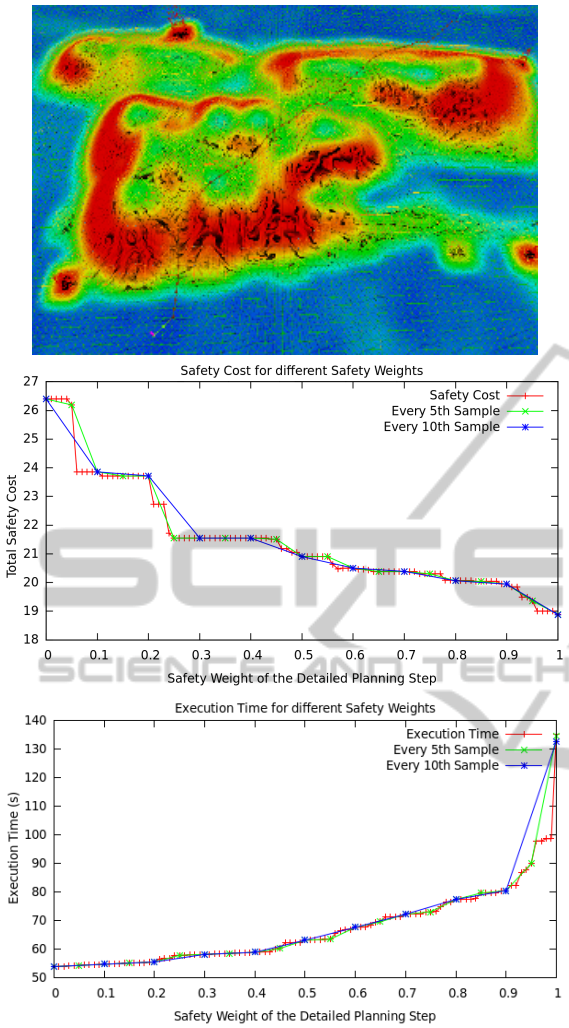

Figure 8: Influence of the safety weight of the detailed motion planning step on the execution time and safety of the path. The top image shows the first path used for evaluation.

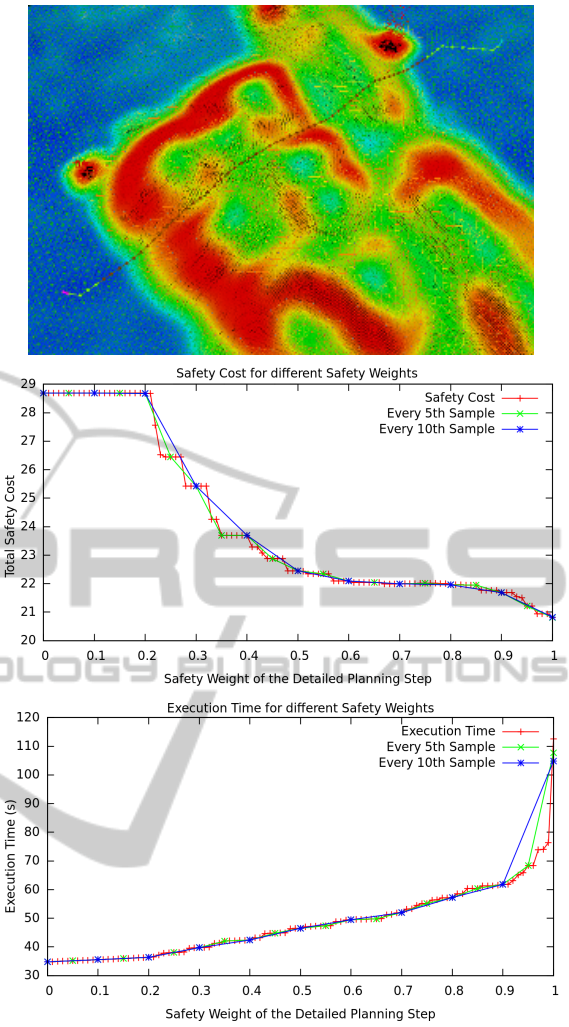

Figure 9: Influence of the safety weight of the detailed motion planning step on the execution time and safety of the path. The top image shows the second path used for evaluation.

values: The robot's maximal traversable height is $h_{max} = 0.5\,m$. In high risk areas it is allowed to drive $v_{min} = 0.2\,\frac{m}{s}$ and on flat ground $v_{max} = 1.2\,\frac{m}{s}$. The discretization of the maps was $5\,cm$ and the window size of the filters is 20 by 20 cells. The resolution of the motion graph was $30\,cm$ (half the robot length) and we considered 8 orientations in each point (corresponding to a resolution of 45°). By taking half the robot length we do not need to perform intermediate validity tests between two neighboring positions. With respect to the actuator values, both front and both rear actuators were required to be the same. Further, the positions were limited to $[-45°, 45°]$ in steps of 15°. The safety weights were set to $w_{is} = 0.75$ and $w = 0.5$.

On the two outdoor environments, we performed several planning queries of which we present two, one for each map. Figures 11 and 13 show the planned paths and pictures of the execution by our robot. In the first scenario (figure 11) the robot had to cross the hill of rubble through the dips, avoiding the high risk elevations. The robot was able to follow the proposed path while the planned actuator configurations prevented the robot from falling over. Problems during the execution where related to the small-grained material of the rubble which caused the robot to slip casually. Equally, our robot traversed the hill of the second scenario given the motion plan shown in figure 13. The localization was solely based on GPS. In combination with the inaccuracy of the map made it difficult for the controller to determine which part of the plan must be executed.

However, in general, the robot was able to exe-

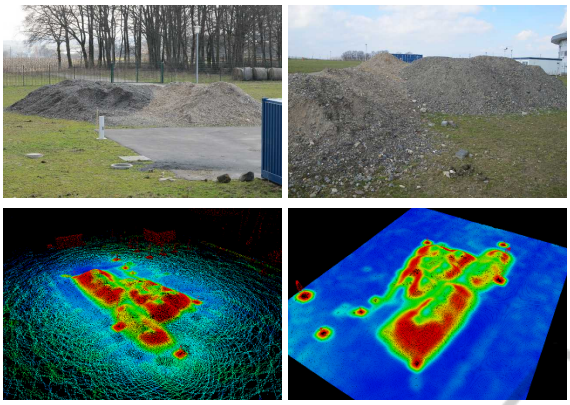

Figure 10: First real-world scenario. The top row shows two images of the hill of rubble. The bottom row depicts the point cloud and the filled, smoothed map. The size of the map is $36.4 \times 30.45\,m$. The colors indicate the risk of traversal, ranging from blue for flat regions over green and yellow to red, high risk areas.

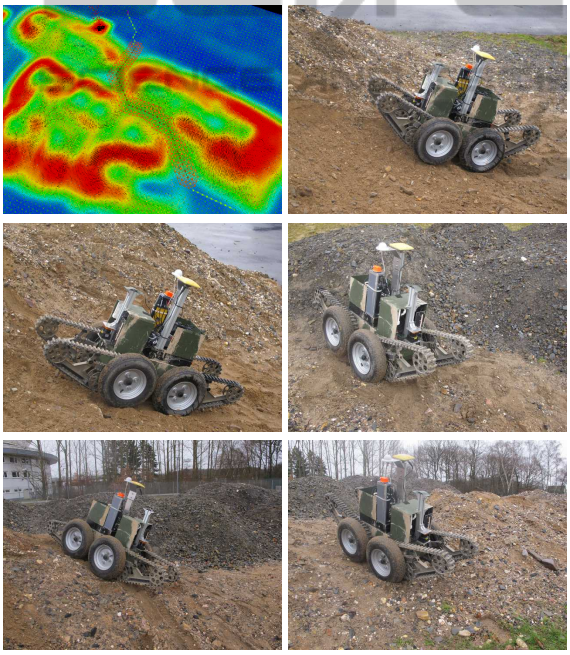

Figure 11: Results of the first real-world experiments. The first image depicts the plan given to the robot. The other pictures show the robot during the execution. If the robot safety permits, configurations are sometimes applied in advanced or while in motion to avoid stop-and-go movement.

cute all plans of our motion planning algorithm and successfully traversed the obstacles. Further, the proposed configurations proved to be suited to ensure the safety of the robot. Problems during the execution were related to terrain parameters or to inaccuracy of the sensor data.

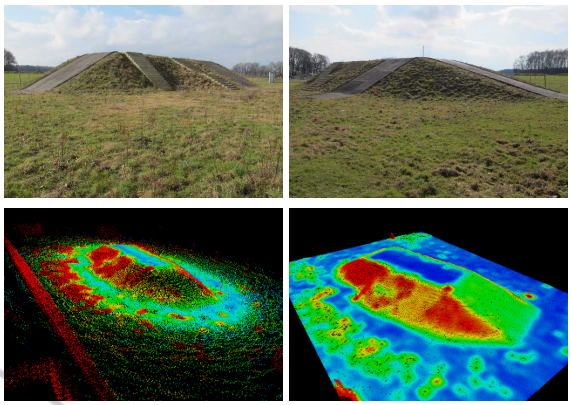

Figure 12: Second real-world scenario. The top row shows two images of the testing hill. In the bottom row the point cloud and the filled, smoothed map are depicted. The size of the map is $43.95 \times 32.95\,m$. The colors indicate the risk of traversal, ranging from blue for flat regions over green and yellow to red, high risk areas.

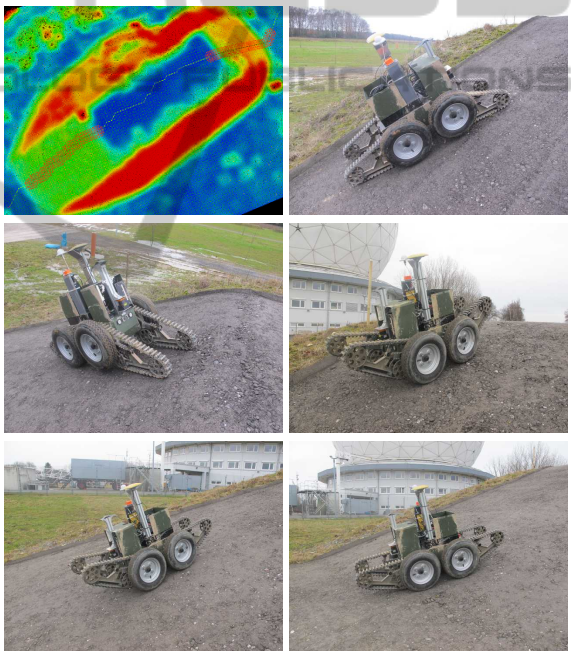

Figure 13: Results of the second real-world experiments. The first image depicts the plan given to the robot. The other pictures show the robot during the execution. If the robot safety permits, configurations are sometimes applied in advanced or while in motion to avoid stop-and-go movement.

9 CONCLUSIONS AND FUTURE WORK

In this paper we presented different metrics used by our motion planning algorithm for robots with reconfigurable chassis to find paths through rough terrain

and over obstacles. By introducing a two-phase planner we are able to use two different cost functions, one to find a low-risk path to the goal and another suited to achieve safe robot configurations along the path. We introduced a roughness quantification which, based on filtered height differences, provides an estimate of the risk the robot is exposed to. The presented cost functions include costs of risk adjusted execution times and safety parameters, like the stability of the system and an traction estimate. Using these metrics we provided several experiments which prove the validity of our measures.

As stated in the experiment section, the algorithm works on a subset of the robot state space X. Improving the state space representation and enhancing the search method to facilitate a more comprehensive search for robot configurations is at the core of our future work. Also, we look at the possibilities to overcome extreme obstacles up to about $40\,cm$ in height as this is the limit given by the manufacturer. This might require new metrics and a more sophisticated controller to cope with arising challenges.

REFERENCES

Dornhege, C. and Kleiner, A. (2007). Behavior maps for online planning of obstacle negotiation and climbing on rough terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Garcia, E., Estremera, J., and de Santos, P. G. (2002). A Comparative Study of Stability Margins for Walking Machines. *Robotica*, 20:595–606.

Hirose, S., Tsukagoshi, H., and Yoneda, K. (2001). Normalized Energy Stability Margin and its Contour of Walking Vehicles on Rough Terrain. In *IEEE International Conference on Robotics & Automation (ICRA)*.

Howard, A., Seraji, H., and Tunstel, E. (2001). A rule-based fuzzy traversability index for mobile robot navigation. In *IEEE International Conference on Robotics and Automation (ICRA)*.

Howard, T. M. and Kelly, A. (2007). Optimal Rough Terrain Trajectory Generation for Wheeled Mobile Robots. *International Journal of Robotics Research*, 26(2):141–166.

Iagnemma, K. and Dubowsky, S. (2004). *Mobile Robots in Rough Terrain - Estimation, Motion Planning, and Control with Application to Planetary Rovers*, chapter Rough Terrain Motion Planning, pages 51–79. Springer Tracts in Advanced Robotics.

Iagnemma, K., Kang, S., Shibly, H., and Dubowsky, S. (2004). Online terrain parameter estimation for wheeled mobile robots with application to planetary rovers. *IEEE Transactions on Robotics*, 20:921 – 927.

Jacoff, A. S., Downs, A. J., Virts, A. M., and Messina, E. R. (2008). Stepfield Pallets: Repeatable Terrain for Evaluating Robot Mobility. In *Performance Metrics for Intelligent Systems (PerMIS) Workshop*.

Magid, E., Ozawa, K., Tsubouchi, T., Koyanagi, E., and Yoshida, T. (2008). Rescue Robot Navigation: Static Stability Estimation in Random Step Environment. In Carpin, S., Noda, I., Pagello, E., Reggiani, M., and von Stryk, O., editors, *Simulation, Modeling, and Programming for Autonomous Robots*, volume 5325 of *Lecture Notes in Computer Science*, pages 305–316. Springer Berlin / Heidelberg.

Messuri, D. A. (1985). *Optimization of the locomotion of a legged vehicle with respect to maneuverability*. PhD thesis, Ohio State University.

Miro, J., Dumonteil, G., Beck, C., and Dissanayake, G. (2010). A kyno-dynamic metric to plan stable paths over uneven terrain. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

Molino, V., Madhavan, R., Messina, E., Downs, A., Balakirsky, S., and Jacoff, A. (2007). Traversability metrics for rough terrain applied to repeatable test methods. In *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*.

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

Seraji, H. (1999). Traversability index: a new concept for planetary rovers. In *IEEE International Conference on Robotics and Automation (ICRA)*.