Proceedings of the Defense and Security Symposium, Unmanned Ground Vehicle Technology VI (OR54), Orlando FL, 12-16

T-transformation: Traversability Analysis for Navigation on Rugged Terrain

Cang Ye and Johann Borenstein

Advanced Technologies Lab, the University of Michigan
1101 Beal Ave, Ann Arbor, MI, USA 48109-2110

ABSTRACT

In order to maneuver autonomously on rough terrain, a mobile robot must constantly decide whether to traverse or circumnavigate terrain features ahead. This ability is called *Obstacle Negotiation* (ON). A critical aspect of ON is the so-called traversability analysis, which evaluates the level of difficulty associated with the traversal of the terrain. This paper presents a new method for traversability analysis, called *T-transformation*. It is implemented in a local terrain map as follows: (1) For each cell in the local terrain map, a square terrain patch is defined that symmetrically overlays the cell; (2) a plane is fitted to the data points in the terrain patch using a least-square approach and the slope of the least-squares plane and the residual of the fit are computed and used to calculate the *Traversability Index* (TI) for that cell; (3) after each cell is assigned a TI value, the local terrain map is transformed into a traversability map. The traversability map is further transformed into a traversability field histogram where each element represents the overall level of difficulty to move along the corresponding direction. Based on the traversability field histogram our reactive ON system then computes the steering and velocity commands to move the robot toward the intended goal while avoiding areas of poor traversability. The traversability analysis algorithm and the overall ON system were verified by extensive simulation. We verified our method partially through experiments on a Segway Robotics Mobility Platform (RMP), albeit only on flat terrain.

Keywords: traversability analysis, T-transformation, polar traversability index, obstacle negotiation, terrain navigation, elevation map, traversability map, traversability field histogram.

1. INTRODUCTION

Autonomous navigation of mobile robots on rough terrain [1, 2, 3] is of great importance to mobile robotics research. The DARPA Grand Challenge event in March 2004 is one point in case. There are two crucial issues in terrain navigation: terrain mapping and path planning. Terrain mapping aims at building a 3-Dimenstional (3-D) map or a grid-type elevation map of the robot's operating environment, while path planning aims at determining the robot's steering and velocity commands based on the terrain map. This paper is dedicated to path planning. Our method for building the local terrain map is explained in detail in our earlier papers [4, 5]. Path planning for rough terrain navigation first has to analyze the traversal property of the terrain map. This process is called *Terrain Traversabiliy Analysis* (TTA) [6]. It then determines the steering and velocity commands which allow the robot to traverse moderate terrain while avoiding high profiled terrain features and/or obstacles. We call this path planning ability *Obstacle Negotiation* (ON). Since a local terrain map is used, this is a local path planning method.

Much work has been done on TTA. Langer et al [7] computed elevation statistics of the terrain, including the minimum height, the maximum height, and the slope of a terrain patch. He then classified terrain cells as traversable or untraversable by comparing their elevation statistics with threshold values. The limitation with this method is that traversability is expressed in binary form. Gennery [8] proposed a TTA method based on least squares fitting. His method fits a plane to a small terrain patch using a Least Square Error (LSE) approach. The slope of the fitted plane and the residual of the fit are then used to estimate the slope and roughness of the terrain patch, respectively. In order to deal with the inaccuracy of the stereovision data, Gennery assigned a weight to each data point according to its accuracy before searching for the least-squares plane. The algorithm requires the computation of the covariance matrix of each data point and slope calculation is iterative. Therefore, it is computationally expensive. A similar TTA algorithm without iterations was adopted by Singh [6]. In this algorithm, the roll, pitch, and roughness measures are normalized in the range [0, 1], and the lowest value among these three measures determines the overall *goodness* of a cell. Singh's method

then creates a goodness map at each time instant by weighting a sequence of previous goodness maps. Seraji [9] proposed a traversability measure that uses fuzzy logic. In their algorithm, the concept of *Traversability Index* (TI) was used to represent the degree of ease, with which the regional terrain could be navigated, and it was described by a number of fuzzy sets. Three terrain characteristics, namely, roughness, slope, and discontinuity, were extracted from the stereo image and described by fuzzy sets. A fuzzy inference system was then built to map the terrain characteristics to the TIs. The method is fast and allows real-time computation of the TIs. However, human expert knowledge is required to compile and tune the fuzzy rules.

To determine the steering and the velocity commands, behavior-based navigation methods have been employed in a number of research projects [7, 10]. Behavior-based methods decompose a navigation system into a number of task-specific behavior modules, each of which determines its own motion recommendations for the robot. As the motion recommendations from different modules may be in conflict with each other, a behavior arbitrator is used to fuse the motion recommendations and produce the steering and velocity commands for the robot. For instance, the system described in [10] comprises a Traverse-terrain Behavior (TB), an Avoid-obstacle Behavior (AB), a Seek-goal Behavior (SB), and a Behavior Integration Module (BIM). Each module uses fuzzy logic to compute the motion commands in the form of fuzzy set. The BIM infers a weighting factor for each behavior modules and the fuzzy commands recommended by the three modules are weighted and *defuzzified* to produce the motion commands for the robot. This method is similar to the flat ground navigator in [11].

Some researchers employed the so-called "arcs approach" [7, 6, 12, 13]. This method first generates a number of candidate arcs. It then either votes for the arc with the largest clearance [7], or calculates the costs along each arc and selects the arc with the lowest cost [6, 12]. Finally, the robot is steered along the winning arc. In [6] the cost function is the sum of traversability values along each arc. Here the D* algorithm is used to minimize the cost. Gennery's path planner [8] computes the cost of driving through a grid. The cost function comprised two components: the distance traveled and the probability that the slope or roughness may be too large to traverse. A path planner is then employed to find a path that minimizes the total cost. The probability approach takes into account the inaccuracy of stereovision data.

There are some attempts to apply potential field methods to terrain navigation [14]. However, the application of potential field methods to obstacle avoidance may result in oscillatory motion [15] or cause the robot to get trapped in a local minimum. The Vector Field Histogram (VFH) method, developed by one of the co-authors [16] of this paper, overcomes the limitations of potential field methods. The VFH algorithm has been demonstrated to produce non-oscillatory, fast obstacle avoidance for a variety of mobile robots at our lab and at other research labs.

Inspired by the VFH method, in this paper we propose a new TTA method, called *T-Transformation*, which transforms a local terrain map into a one-dimensional traversability field histogram, based on which we proposed an ON method to determine the robot motion. The paper is organized as follows: In Section II, we provide a brief overview of our ON system. Then, in Sections III and IV we describe our TTA algorithm and the ON algorithm, respectively. In Section V, we present extensive simulation results and some limited experimental results of the overall ON systems. Section VI concludes the paper.

2. OVERVIEW OF THE OBSTACLE NEGOTIATION SYSTEM

Figure 1 shows a block diagram of our ON system. The system consists of four main modules: Terrain Mapping, Terrain Traversability Analysis (TTA), Traversability Field Histogram, and Motion Control. In this section we give a brief overview of the ON system.

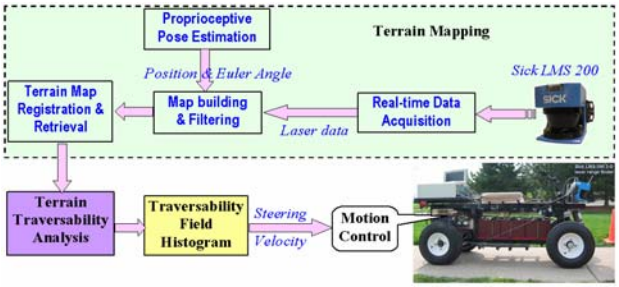


Fig. 1 Diagram of the obstacle negotiation system: the module within the dashed lines represents the Terrain Mapping module

The terrain mapping module (see [4] for the details) produces a terrain map for obstacle negotiation. In our system, it is represented by a 2-dimensional (2-D) grid-type map, in which each cell holds a value representing the height of the terrain at that cell. Such a map is also called a 2½-D map or simply a terrain map.

The task of the TTA module is to analyze the terrain's

overall traversability along each direction. It first analyzes the taversability property of individual cells in the terrain map and generates a new 2-D grid-type map, called the traversability map. Each cell in this map holds a value that expresses the degree of difficulty for the robot to move across that cell. The TTA method then analyzes the overall traversal property of the terrain along each direction and transforms the traversability map into a one dimensional traversability field histogram.

The Traversability Field Histogram (TFH) module is a reactive path planner. It analyzes the traversability field histogram and generates steering and velocity commands to move the robot along the traversable direction. The TTA and TFH are the main subjects of this paper.

3. TERRAIN TRAVERSABILITY ANALYSIS

3.1 Traversability Map

Let us assume we have a terrain map $\mathbf{E} = \{z_{i,j}\}$, where i and j are the row and column cell indices, respectively. The Robot Center Point (RCP) is located at $(x_i, y_j, z_{i,j})$ in the terrain map, where x_i and y_j are the coordinates corresponding to cells (i, j). For simplicity we placed the RCP at the robot's geometric center. We then define a square terrain patch $P = \{z_{k,l} \mid_{k=i-L,\cdots,i+L;\, l=j-L,\cdots,j+L}\}$. P has a side length of $2L+1$ (in terms of the number of cells) and it is centered at cell (i, j). L is an integer and its value is chosen in such a way that the terrain patch completely envelops the robot regardless of the robot's orientation.

We then fit a plane to this terrain patch using the LSE approach and obtain the normal **n** to the fitted plane. The slope of the terrain patch is then estimated by

$$\alpha = \cos^{-1}(\mathbf{n} \cdot \mathbf{b}), \tag{1}$$

where $\mathbf{b} = (0,0,1)^T$; the roughness of the terrain patch is approximated by

$$\sigma = \sqrt{\sum_{i=1}^{i=N} d_i^2}, \tag{2}$$

where d_i is the distance between the i^{th} data and the fitted plane. Finally, the TI of cell (i, j) is calculated by

$$\tau_{i,j} = |F_1\alpha| + |F_2\sigma / N|. \tag{3}$$

F_1 and F_1 are empirically chosen and their value represents the contribution of the terrain slope and roughness to the TI value. In this study we used F_1=300 and F_2=6, which resulted in the slope having a slightly larger contribution than the roughness.

The traversability analysis is performed cell by cell in the local terrain map. Once each cell is assigned a TI value, the terrain map **E** is transformed into a traversability map $\mathbf{T} = \{\tau_{i,j}\}$. We call this process "*Cartesian Traversability*

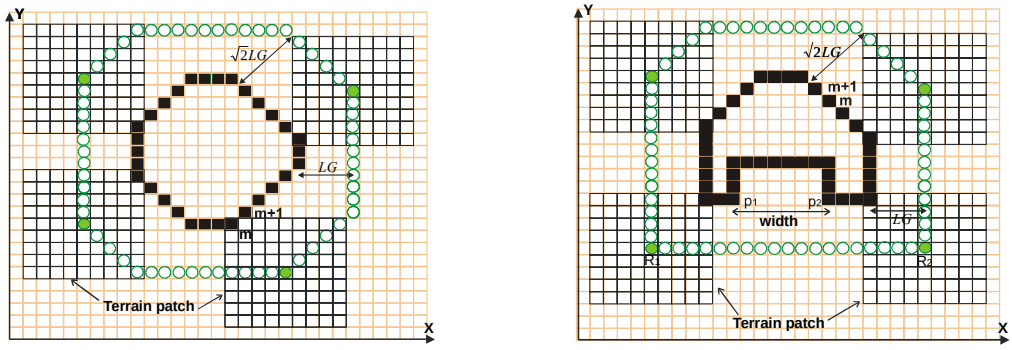
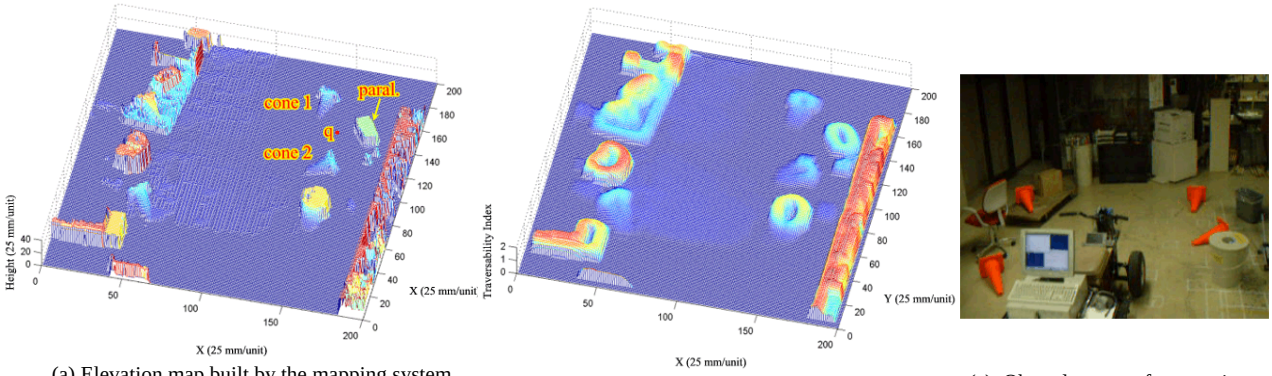


Fig. 2 Cartesian Traversability Analysis with L=4, i.e., the terrain patch size is 9 cells x 9 cells: the four solid circles are the resulting cells of the CTA with the four corresponding terrain patches. +—boundary of E-obstacle, ○ and ●—boundary of T-obstacle G—side length of the cell

Analysis" (CTA). We define an obstacle in a terrain map as a cluster of neighboring cells with non-zero elevation value and call this obstacle an E-obstacle (elevation obstacle). The CTA transforms an E-obstacle into a cluster of neighboring cells with non-zero TIs and we call this cluster of cells a T-obstacle. Fig. 2 is an illustration of the CTA on a terrain map with one obstacle. The boundary of the E-obstacle is depicted by solid squares while the boundary of the T-obstacle is outlined by circles. In a grid-type terrain map, an E-obstacle's boundary comprises line segments in X-, Y-, or diagonal direction. The CTA grows the boundary aligned with X- or Y-direction by L cells and it grows the boundary aligned diagonally by $\sqrt{2}L$ cells. This automatically takes into account the robot's dimensions during navigation and it allows us to treat the robot as a point in the path-planning algorithm described in the following section.



(a) Elevation map built by the mapping system (b) Traversability Map (c) Obstacle course for mapping

Fig. 3 Cartesian Traversability Analysis on the elevation map built by the terrain mapping system: terrain patch size = 9 cells × 9 cells

Figure 3a depicts a terrain map built by our Terrain Mapping module from the actual data of a Sick laser rangefinder (LRF). The LRF was mounted in forward-downward looking orientation on the front of a mobile robot. This way the LRF's fanned beam swept across the floor while the robot traversed the obstacle course in Fig. 3c. Fig. 3b shows the traversability map obtained by the CTA algorithm.

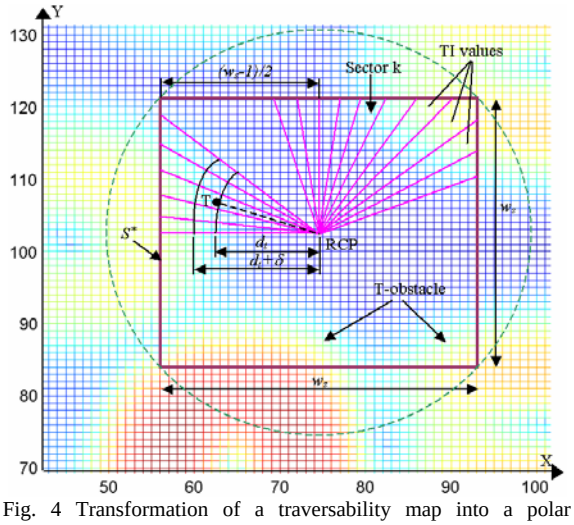


Fig. 4 Transformation of a traversability map into a polar histogram: X- and Y-axes represent the world coordinates.

3.2 Polar Traversability Index

Our ON method determines the robot's next heading and the velocity based on the traversal property of the local terrain map surrounding the robot. Therefore, we need to analyze the overall traversal property of the local terrain along each direction, that is, to do traversability analysis in polar coordinate. We call this "*Polar Traversability Analysis*" (PTA).

The PTA algorithm first forms a square-shaped local terrain map S^*, which overlays the global terrain map **E** like a window. There are $w_s \times w_s$ cells in S^*, and their size is the same as that of cells in **E**. S* centers at the momentary position of the RCP, and we chose w_s=37 in our implementation. As the side length of a grid cell is 250 mm, S^* covers a physical region of 9.2 × 9.2 m (37×250 mm = 9.2 m). Next, the PTA algorithm performs the CTA over the cells in S^* and transforms the local terrain map into a traversability map.

Figure 4 is the 2-D representation of a traversability map, where the TI value is rendered in color. Each cell with nonzero TI values in the traversability map generates an imaginary vector field, which exerts virtual repulsive forces on the robot and repels it away from an untraversable region. We call this field "traversability field." The magnitude of the traversability field produced by cell (i, j) in the traversability map is

$$m_{i,j} = b\,\tau_{i,j}^2 (1 - \frac{d_{i,j}}{d_{\max}}), \tag{4}$$

where $\tau_{i,j}$ is the TI value of cell (i, j), $d_{i,j}$ is the distance between cell (i, j) and the RCP, $d_{\max} = \sqrt{2} w_s / 2$ is the distance between the RCP and the vertex of the S^*, and b is a constant scaling factor. The direction of cell (i, j) to the RCP is calculated by

$$\beta_{i,j} = \tan^{-1} \frac{y_j - y_o}{x_i - x_o}, \tag{5}$$

where x_i and y_i are the coordinates of cell (i, j), and x_0 and y_0 are the present coordinates of the RCP.

In equation (4), $\tau_{i,j}$ is squared such that the impact of a cell with a small/large TI value is diminished/magnified. Since $m_{i,j}$ is proportional to $-d_{i,j}$, cells with non-zero TIs generate larger magnitudes when they are closer to the robot and smaller ones when they are farther away.

In order to evaluate the overall difficulty traversing along a direction, the magnitude of the traversability field produced by all cells in the same direction should be summed up. For this reason, S^* is divided into n sectors, each of which has an angular resolution of α=360°/n. Each sector k, for $k = 0, \cdots, n-1$, corresponds to a discrete angle $\rho = k\alpha$. The cells (i, j) in S^* are assigned to the k^{th} sector according to

$$k = \text{int}(\beta_{i,j} / \alpha) \tag{4}$$

For each sector k, the overall magnitude of the traversability field h_k is then calculated by

$$h_k = \sum_{i,j} m_{i,j} \tag{5}$$

From Eq. (4) & (7), h_k is the distance-weighted sum of the squared TIs of the cells in a specific sector; and it represents the overall difficulty traversing the terrain in the corresponding direction. We call h_k polar traversability index. In our implementation α = 5°, i.e., there are 72 sectors. The polar traversability indices produced by Eq. (7) are discontinuous. Since they are used to determine the robot's steering and velocity, this may result in jerky motion. In order to alleviate this problem, we use the following averaging function to smooth the polar traversability index:

$$h_k' = \frac{\sum_{l=-p}^{p} h_{k-l}}{2p+1} \tag{6}$$

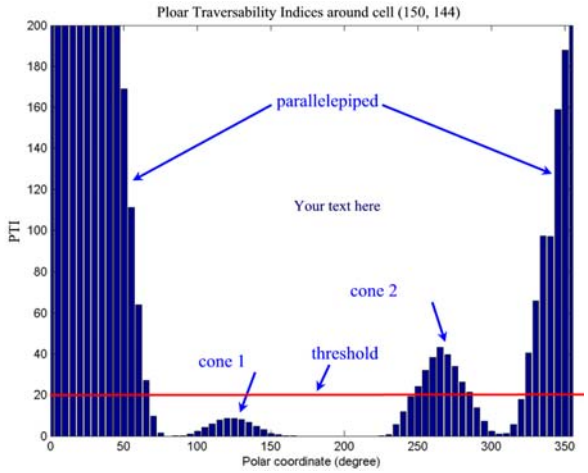


Fig. 5 This traversability field histogram was created when the RCP was at point '*q'* in Fig. 3a

where h_k' is the Smoothed Polar Traversability Index. For simplicity, we call it "*Polar Traversability Index*" (PTI) from now on. We found that p=3 produces satisfactory smoothed result.

We call the above transformation—from the local terrain map to the PTIs— "*Polar Traversability Transformation,*" or "*T-transformation*" for short. The PTI of each sector represents the level of difficulty of moving in the corresponding direction. In this paper, The PTIs of the local terrain at the robot's instant position are displayed in the form of polar histogram. Since each PTI represents the overall

magnitude of the traversability field along the corresponding direction, we call this histogram "traversability field histogram" and denoted it **H**. Figure 5 depicts the **H** for the momentary situation of Fig. 3a, where the RCP is located at cell (150, 144) (labeled 'q' in Fig. 3a). The parallelepiped object produced large PTIs in sectors 1-14 and sectors 64-72 (i.e., -40°–70°) because it is high in elevation. Cone 1 generated small PTIs in sectors 19-30 (i.e., 95°–150°) since its low-profile tip faced the robot, whereas cone 2 produced relatively larger PTIs in sector 47-60 (i.e., 235°–300°) because its base (with its higher elevation) faced the robot. This exemplifies how a PTI correctly reflects the overall traversability of each sector in region S*.

4. OBSTACLE NEGOTIATION USING POLAR TRAVERSABILITY HISTOGRAM

In this paper, we extend the VFH algorithm [16] to rough terrain navigation and propose a new ON method, the Motion-context Guided Traversability Field Histogram (MGTFH) algorithm. The MGTFH method employs the traversability field concept and the robot's motion-context to guide the robot in traversing rough terrain. It should be noted that conventional 2-D obstacle avoidance is a special case under the ON framework, and is handled as such by our MGTFH.

4.1 Steering Control

When the robot travels across feature-rich terrain, the histogram **H** changes from instance to instance. The objective of the ON algorithm is to steer the robot into a direction with lower PTI, while still maintaining a direction that is close to the target direction. As can be seen from Fig. 5, a traversability field histogram typically has "hills" (sectors with high PTIs) and "valleys" (sectors with low PTIs). A "candidate valley" is a cluster of consecutive sectors with PTIs below a certain threshold (e.g., Fig. 4 has two candidate valleys). The candidate valley, which is eventually used to determine the robot's next heading direction is called the "*winning valley*." In the basic VFH algorithm [16], the winning valley is always the one that is closest to the target direction. We call this approach the *"closest-valley-wins*" scheme. However, we have modified this approach in the algorithm proposed in this paper: the winning valley does no longer have to be the one closest to the target direction. As we will show below, the latter approach helps avoid certain trap-situations. With either approach, once a winning valley is identified, the algorithm selects a sector in the winning valley as the robot's next heading direction.

Assuming that the robot's target is (x_t, y_t), the sector number k_t of the target vector is calculated by Eq. (4) and (6) (x_i and y_i are replaced by x_t and y_t in this case). We denote the i^{th} candidate valley by $v_i = [k_i^R, k_i^L]$ where k_i^R and k_i^L represent the sector number of the right and left border of candidate valley v_i, respectively. The width of the valley, s_i, is $s_i = k_i^L - k_i^R + 1$ sectors. A valley is called a "wide valley" if $s \geq s_{\max}$ (in our system $s_{\max} = 12$), otherwise, we call it a "narrow valley." We denote the winning valley by v_w. The robot's next heading direction is then defined as

$$k_h = \begin{cases} k_w^R + \min(s_w, s_{\max})/2 & \text{if } |k_t - k_w^R| \leq |k_t - k_w^L| \\ k_w^L - \min(s_w, s_{\max})/2 & \text{otherwise} \end{cases} \quad (7)$$

where k_w^R and k_w^L represents the sector number of the right and the left border of v_w, respectively. The border determining the next heading direction is called the winning border, and s_w is the width of v_w. If the winning valley is narrow, the robot is directed to steer into in the direction of the center of the valley. If the winning valley is a wide one, then a sector $s_{max}/2$ sectors from the winning border is selected as the new steering direction. Treating the final selection of a steering direction as a spatial problem within the polar histogram eliminates the oscillatory behavior inherent in potential field methods [15] and represents the major distinction between those methods and the VFH method.

In practice, determining the winning valley is not as straightforward as the proceeding sections suggest. Simply selecting the candidate valley closest to the target direction as the winning valley (this is the above-mentioned *closest-valley-wins* scheme) may result in so-called "trap" situations. When the robot is in motion, the PTI of each sector changes all the time. This may switch the winning valley from one candidate valley to another. In our simulations we found that under certain conditions rapidly changing winning valleys may result in cyclic behavior. That is, the robot

drives around in circles despite the existence of obvious exits from the area. To overcome this problem we propose the concept of "motion-context" in this paper. The robot's motion-context at time step *t* is defined as

$$\mu = sign(k_h - k_t), \tag{8}$$

where k_t and k_h are the sector number of the target vector and the robot's heading vector at time step *t-1*, respectively. $\mu = 0$ means the robot is moving straight toward the target while $\mu = -1$ and $\mu = 1$ indicate that the robot is deviating from the target direction *t* because of an avoidance maneuver initiated by a right-turn or left-turn, respectively. The distances between the target vector and the borders of the i^{th} valley are defined as $D_i^L = | k_t - k_i^L |$ and $D_i^R = | k_t - k_i^R |$. Their minimum values are denoted $D_{\min}^L$ and $D_{\min}^R$, respectively.

We use Table I to select the winning valley and the winning border. This decision table gives the robot a tendency to follow the contour of an obstacle with the obstacle in its left. To exit this contour-following mode when the direction to the target is free, an exit condition must be met. Once the exit condition is met, the robot will be forced to move straight toward the target direction and thus reset the motion-context. In order to do this, we declare the target direction to be "free" if each sector of the cluster of sectors $[k_t - s_{\max}/2, k_t + s_{\max}/2]$ lies in a candidate valley, i.e., the PTI of each sector is smaller than the threshold value. The steering control algorithm first checks whether the target direction is free. If so, then the robot's next heading is pointed at the target direction. Otherwise, the scheme in Table I is applied to determine the winning border and thus the steering direction.

TABLE I. DECISION TABLE FOR THE WINNING VALLEY AND BORDER

If μ	μ=1	μ=0	μ=-1
$D_{\min}^L > D_{\min}^R$	k_{j+1}^R	k_j^L	k_j^L
$D_{\min}^L \le D_{\min}^R$	k_j^L	k_j^L	k_j^L

4.2 Speed Control

The robot runs initially at its maximum speed v_{max} (1 m/s in our simulator). It will try to maintain this speed unless forced by the MGTFH algorithm to a lower value, which is determined at each time step as follows:

The PTI in the current direction of travel is denoted as h_c'. $h_c' > 0$ indicates that rough terrain lies ahead of the robot. A large value of h_c' suggests that an obstacle lies ahead. In either case a reduction in speed is in order. In our system speed is reduced inversely proportional to the PTI value in the momentary direction of travel:

$$v' = v_{\max}(1 - \frac{\min(h_c', h_m)}{h_m}) \tag{9}$$

h_m is a constant, which is empirically determined to produce a sufficient reduction in speed. A reduction in speed is also required when the robot approaches the target. In this paper, we further reduce the speed by

$$v'' = \frac{v' \min(d_t, d_m)}{d_m} \tag{10}$$

where d_t is the distance between the target and the RCP, while d_m is a constant (we used 1.5 m in our simulator). Note that v'' is actually the projection of the robot speed V on the X-Y plane. It produces a movement l_{xy} in the X-Y plane along the robot's next heading, and moves the RCP to (x_0', y_0'). We then calculate the robot's speed by

$$V = \frac{v'' l_{xyz}}{l_{xy}}, \tag{11}$$

where l_{xyz} is the 3-D distance between the current RCP position at (x_o, y_o, z_o) and the next RCP position at (x_o', y_o', z_o').

In summary, the MGTFH algorithm functions as follows (for simplicity, we describe the algorithm in the X-Y plane):

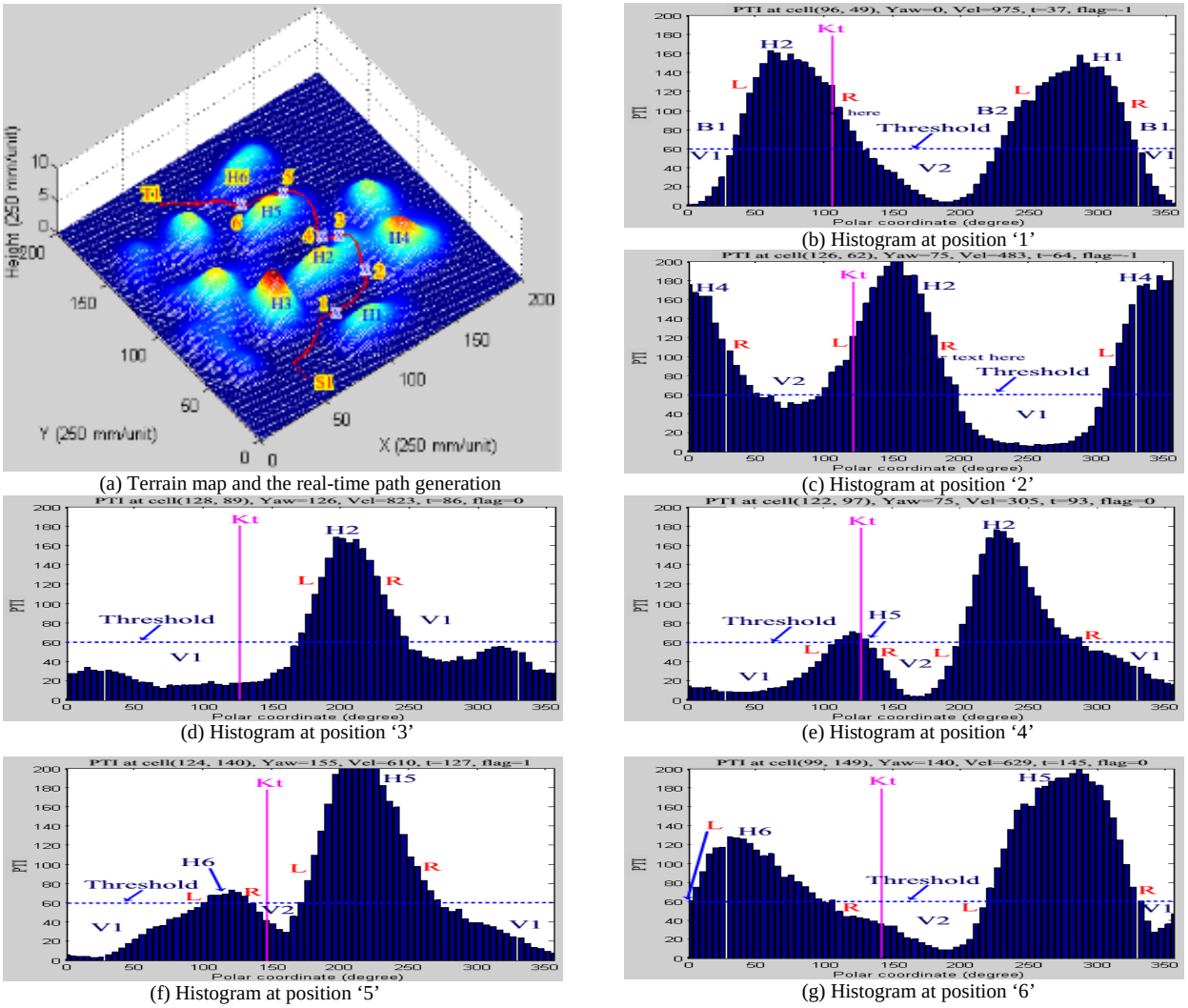
1) Start the robot with $v'' = v_{max}$ and $\mu=0$.
2) If the target is reached, stop the robot; otherwise go to Step 3.
3) Calculate the polar histogram. If the target direction is free, direct the robot straight toward the target with v" given by Eq. (12). Otherwise, go to Step 4.
4) Apply decision Table I to determine the winning valley and the winning border and thus the steering direction. Move the robot along that direction at speed v" given by Eq. (12). Go to 2.

5. SIMULATION AND EXPERIMENTAL RESULTS

5.1 Navigation on simulated rough terrain

To test our ON method in the widest possible range of environments, we developed a simulator able to generate arbitrary terrain maps and run a simulated robot through those environments. Figure 6a shows a typical run over simulated rough terrain. The run started at 'S_1', terminated at target 'T_1', and was generated in real-time.

Figures 6b-g show snapshots of the polar histograms for points 1-6 on the robot's path. The robot first steered to the passage between hills 'H1' and 'H7' and moved to point '1'. The polar histogram at this point is displayed in Fig. 6b. Since the motion-



(a) Terrain map and the real-time path generation

(b) Histogram at position '1'

(c) Histogram at position '2'

(d) Histogram at position '3'

(e) Histogram at position '4'

(f) Histogram at position '5'

(g) Histogram at position '6'

Fig. 5 A navigation task from 'S1' to 'T1': Kt – target sector; L/R – left / right border of a candidate valley; flag – motion-context, t – time step; Vel – velocity of the robot in 3D space; Yaw – yaw (heading) angle.

context was –1, the left border of valley ‘V1’ was used to determine the heading (0° in this case). The robot then went through the passage between hills ‘H1’ and ‘H2’ while maintaining the motion-context, and moved toward the moderate slope between hills ‘H2’ and ‘H4’, before reaching position ‘2’. At this point, the polar histogram is depicted in Fig. 6c. Since the motion-context is –1, the left border of valley ‘V2’ won and was used to calculate the next heading, in this case, 75°. Following hill ‘H2’, the robot climbed up the slope and arrived at point ‘3’. At this point, the target direction is free, i.e., the cluster of sectors $[k_t - s_{\max}/2, k_t + s_{\max}/2]$ lies in the wide valley ‘V1’ as depicted in Fig. 5d. Since the exit condition was met (note that at this position, the motion-context was zero), the robot left hill ‘H2’ and moved straight toward the target. It maintained this heading and the zero motion-context until it reached the divergence point at position ‘4’, where hill ‘H5’ entered region S^* and thus created a peak (labeled ‘H5’ in Fig. 6e) in the histogram. According to Table I, the MGTFH algorithm selected the left border of valley ‘V1’ and produced the next heading of 75°. The robot then steered around the base of hill ‘H5’ and until it faced the narrow passage between hills ‘H5’ and ‘H6’. Because this passage only produced a narrow valley ‘V2’ (in Fig. 6f), the target direction is not free. The robot continued to follow hill ‘H5’ until it reached point ‘5’, where its motion-context switched to 1 (Fig. 6f). Due to the narrow valley ‘V2’, the heading always points to the center of ‘V2’. This way, the robot moved in the middle of the passage between ‘H5’ and ‘H6’. During this course, the width of valley ‘V2’ gradually increased. When the robot arrived at point ‘6’, valley ‘V2’ became a wide valley and the target direction was free. The robot then moved straight toward the target and came to a full stop at ‘T1’. When we ran the same simulation using the *closest-valley-wins* scheme, the robot was trapped among hills ‘H1’, ‘H2’, ‘H3’ and ‘H4’.

We carried out numerous runs on widely differing real and simulated terrain maps with the proposed algorithm. In all cases, the robot reached its targets with finite-length paths and without being trapped.

One way of assessing the performance of a path planner for rough terrain is to observe the robot’s roll and pitch angles. A good path should be short but without large roll and pitch angles. In our simulation, we assume that the two rear wheels always remain in contact with the ground since the robot’s center of mass lies in the rear, and we computed the robot's roll and pitch angles at each time step. Our simulation results revealed that the ON algorithm successfully guided the robot in traversing moderately rough terrain. Taking the navigation task in Fig. 6a as an example, the roll and pitch angles of the robot using the ON algorithm are within [-6, 6.5] and [-8, 9], respectively.

5.2 Experiment on the Segway Robotic Mobility Platform

In this section, we present tests of our ON algorithm with the Segway Robotic Mobility Platform (RMP) as depicted in Figure 6. In these experiments we tested only one special case of the general 3-D obstacle negotiation ability of our method, namely, motion on flat terrain.

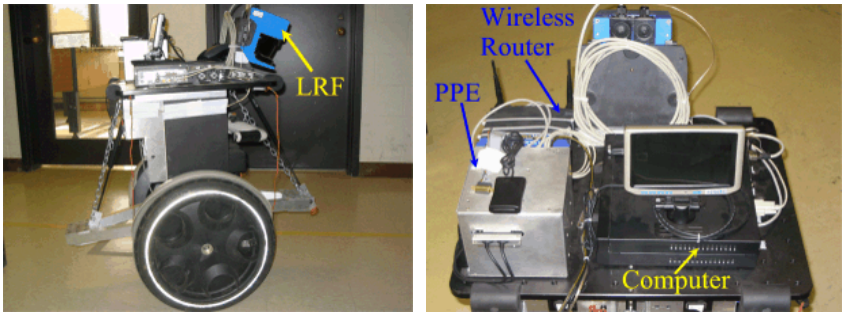
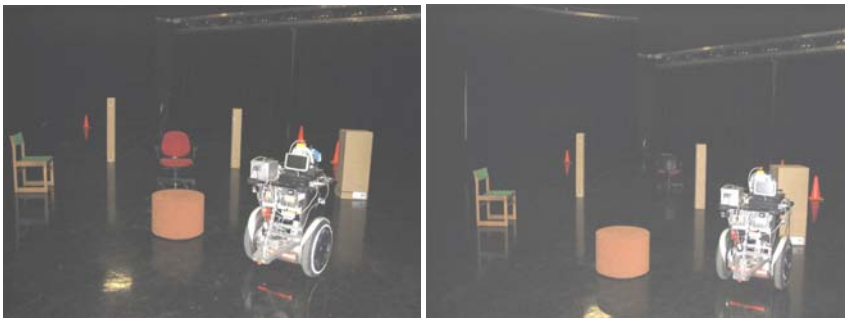


Fig. 6. The Segway RMP: The main components for terrain mapping are the PPE and the LRF which are connected to the on-board computer. The wireless router is used for the access of the off-board computer to the mapping system.

The Segway RMP is the robotic version of the well-known Segway Human Transport (HT) platform. Both versions of the Segway are dynamically self-balancing platforms. In order to move forward or backward, the robot leans its body forward or backward. The robot’s controller reacts to this disturbance by accelerating forward or backward, in order to bring its wheels under the shifted center of gravity, to maintain balance. Since much of the drive control involves leaning forward or backward, the platform and thus the sensor have non-zero and varying pitch angles even on flat ground. This varying pitching motion in fact scans the laser beam in elevation and provides the LRF a vertically wider field of view. However, it requires a very accurate

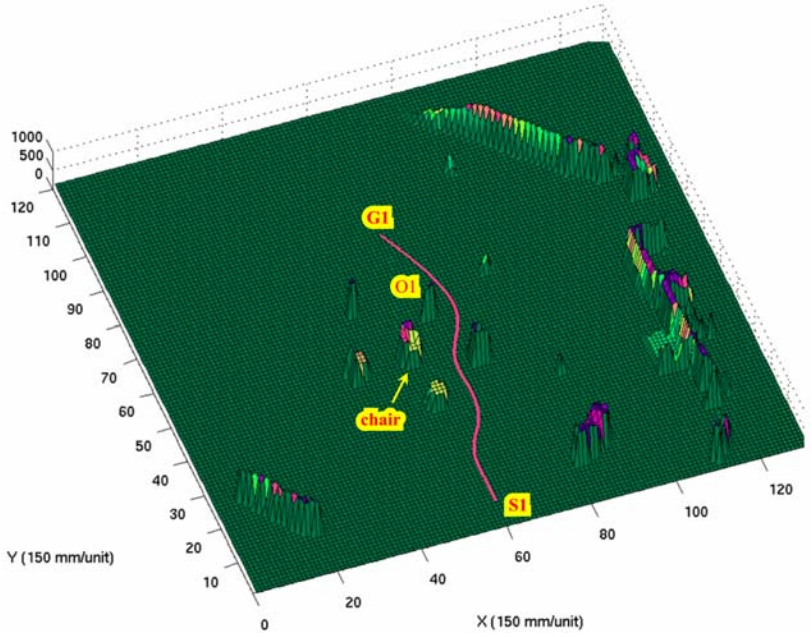


(a) Obstacle course 1 (b) Obstacle course 2

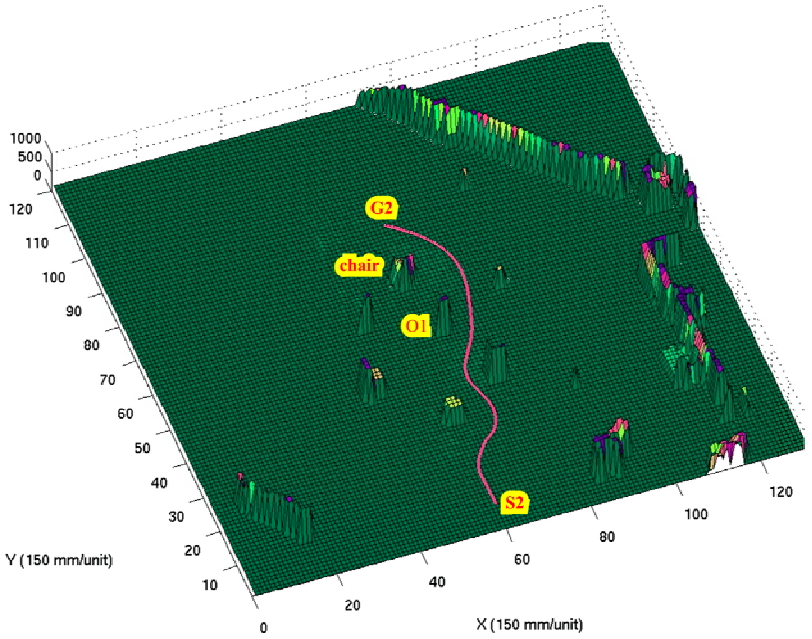
Fig. 7. Obstacle courses used in experiment

pose estimation system for terrain mapping. In our case this function is provided by our in-house developed FLEXnav Proprioceptive Positioning (PPE) system [17]. One particularly useful feature of the Segway is that its footprint is roughly circular and it can turn on the spot. This feature simplifies path planning and obstacle negotiation.

Figures 7a & 7b depicts two obstacle courses we used to run the Segway RMP. In both experiments, the starting point of the robot is (0, 0) while the target point is (0, 12 m). The corresponding results are depicted in Figs. 8a and 8b, respectively. In the experiment depicted in Fig. 8a, the ON system steered the robot along the collision-free path from the start point *S1* to the target *G1*. After the robot avoided obstacle *O1,* the target direction is free. The robot moved straight forward toward the target and finally stop at the target. In the experiment in Fig. 8b, the chair was move to a new position, such that it blocked the robot's way to the target after the robot avoided obstacle *O1*. The robot steered away



(a) Terrain map and the robot trajectory of Fig. 7a



(b) Terrain map and the robot trajectory of Fig. 7b

Fig. 8. Terrain maps and the robot's trajectories

from the chair and deviate from the target first. However, the robot finally came to a full stop at the target *G2*.

6. DISCUSSION AND CONCLUSIONS

We presented a novel method for traversability analysis and obstacle negotiation for mobile robot navigation on rough terrain. The traversability analysis method estimates the slope and roughness of each terrain patch and assigns a traversability index to the corresponding cell, and thus transforms the local terrain map into a traversability map. It then estimates the overall traversability of the terrain along each direction and transforms the traversability map into a 1-dimensional traversability field histogram, from which the velocity and the steering commands are determined.

To overcome trap situations and potential fluctuation in heading with the closest-valley-wins scheme, we proposed the motion-context based algorithm. The simulation runs and the experimental results with the Segway RMP demonstrated that the proposed algorithm is able to navigate robots on rough terrain and flat ground.

ACKNOWLEDGMENTS

This work was funded by the U.S. Department of Energy under Award No. DE-FG04-86NE3796, by DARPA under Award. No. F007571, and under a grant from the University of Michigan's Automotive Research Center (ARC), funded by TACOM.

References

T1 M. Maurette, "Mars rover autonomous navigation," Autonomous Robots, vol. 14, no. 2-3, pp. 199-208, 2003.

2 A. Kelly and A. Stenz, "Rough terrain autonomous mobility–part 1: a theoretical analysis of requirements," Autonomous Robots, vol. 5, no. 2, pp. 129-161, 1998.

3 A. Kelly and A. Stentz, "Rough terrain autonomous mobility–part 2: an active vision, predictive control approach," Autonomous Robots, vol. 5, no. 2, pp. 163-198, 1998.

4 C. Ye and J. Borenstein, "A new terrain mapping method for mobile robots obstacle negotiation," *Proc. the UGV Technology Conference at the 2003 SPIE AeroSense Symposium*, Orlando, FL, April 21-25, 2003.

5 C. Ye and J. Borenstein, "A novel filter for terrain mapping with laser rangefinders," accepted for publication in the *IEEE Transactions on Robotics and Automation*, 2003.

6 S. Singh, et al, "Recent progress in local and global traversability for planetary rovers," *Proc. IEEE International Conference on Robotics and Automation*, 2000, pp. 1194-1200.

7 D. Langer, et al, "A Behavior-based system for off-road navigation," *IEEE Transactions on Robotics and Automation*, vol. 10, no. 6, pp. 776-783, 1994.

8 D. B. Gennery, "Traversability analysis and path planning for a planetary rover," *Autonomous Robots*, vol. 6, no. 2, pp. 131-146, 1999.

9 H. Seraji, "New traversability indices and traversability grid for integrated sensor/map-based navigation," *Journal of Robotic Systems*, vol. 20, no. 3, p 121-134, 2003.

10 H. Seraji and A. Howard, "Behavior-based robot navigation on challenging terrain: a fuzzy logic approach," *IEEE Transactions on Robotics and Automation*, vol. 18, no. 3, pp. 308-321, 2002.

11 C. Ye and D. W. Wang, "A novel navigation method for autonomous mobile robots," *Journal of Intelligent and Robotic Systems*, vol. 32, no. 4, pp. 361-388, 2001.

12 S. Lacroix, et al, "Autonomous Rover Navigation on Unknown Terrains: Functions and Integration," *International Journal of Robotics Research*, vol. 21, no. 10-11, pp. 917-942, 2002.

13 R. Simmons, et al, "Experience with rover navigation for lunar-like terrain," *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems*, 1995, pp. 441-446.

14 H. Haddad, et al, "Reactive navigation in outdoor environments using potential fields," *Proc. IEEE International Conference on Robotics and Automation*, 1998, pp. 1232-1237.

15 Y. Koren and J. Borenstein, "Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation," *Proc. IEEE International Conference on Robotics and Automation*, 1991, pp. 1398-1404.

16 J. Borenstein and Y. Koren, "The Vector Field Histogram–Fast Obstacle-Avoidance for Mobile Robots," *IEEE Transactions on Robotics and Automation*, vol. 7, no. 3, pp. 278-288, 1991.

17 L. Ojeda, M. Raju and J. Borenstein, "FLEXnav: A Fuzzy Logic Expert Dead-reckoning System for the Segway RMP." *Proc. of the Defense and Security Symposium (was Aerosense) Unmanned Ground Vehicle Technology VI (OR54)*, Orlando FL, April 12-16 , 2004.